



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Departamento de Sistemas Informáticos y Computación

IARFID master's thesis

Decision engine software for Computer-Interpretable guidelines

Author: **Alfonso María Pérez González**

Director: **Alfons Juan-Císcar**

Co-director: **Juan Miguel García Gómez**

Co-director: **Salvador Tortajada Velert**

September 2014

ACKNOWLEDGEMENTS

I would like to thank my tutors Alfons Juan, Juan Miguel García and Salvador Tortajada for their guidance and help along all the thesis. I would also like to thank all the IBIME group for this great years, it's been a pleasure working with you. I will miss the brunches with you in the research institute's terrace. I would also like to thank Montse Robles for giving me the opportunity of working with these great people and introducing me in the research world.

Finally, I am thankful to my parents, family, and girlfriend for their encouragement, understanding and support.

GLOSSARY

ACC: Accuracy

AUC: Area Under the Curve

BAR: Balance Accurate Rate

CDSS: Computer Decision Support System

CIG: Computer Interpretable Guideline

DCIS: Ductal Carcinoma in Situ

HER: Electronic Health Record

IDC: Invasive Ductal Carcinoma

QALY: Quality-Adjusted Life Year

RCT: Randomized Controlled Trial

SEN: Sensitivity

SNB: Sentinel Node Biopsy

SPE: Specificity

XMI: XML Metadata Interchange standard

XML: eXtensible Markup Language

1 INDEX

Acknowledgements	2
Glossary.....	3
1 Abstract.....	6
2 Introduction	7
2.1 Objectives.....	8
2.2 Contributions.....	8
3 State of the art.....	10
3.1 Clinical guidelines.....	10
3.2 Data mediator.....	11
3.3 Decision software.....	12
4 Methods – The decision framework	14
4.1 Bayesian methodology.....	14
4.1.1 Bayesian decision background.....	14
4.1.2 Bayesian sequential analysis	15
4.1.3 Including observed facts obtained during the decision steps.....	16
4.2 Criterion model and data flow.....	16
4.3 Bayesian Decision Library.....	17
4.3.1 Bayesian library development.....	18
4.4 The mediator library (Input/output data access).....	18
4.4.1 CDS-DATOR Methods.....	19
4.4.2 Data loaders.....	23
4.4.3 Mapping data sources	26
4.4.4 Procedure to use the CDS-DATOR in a CDSS implementation	26
4.5 Adapter library	26
4.6 Probabilistic models.....	27
4.7 Diagram Data Model.....	27
4.8 Graphical User Interface	29
4.8.1 Sirius	29
4.8.2 Project Explorer	30
4.8.3 Graph Editor Viewer.....	30
4.8.4 Graph Properties Viewer.....	31
4.8.5 Utilities Table viewer	33
4.8.6 Tree Model Viewer.....	33
5 Results	35

5.1	CDS-DATOR.....	35
5.1.1	PROforma case study.....	35
5.1.2	Curiam BT case study.....	40
5.2	Naïve Bayes classifier.....	42
5.2.1	Methods.....	42
5.2.2	Inclusion in Criterion.....	44
5.3	Criterion Case study: Ductal Carcinoma In Situ	46
5.3.1	Problem definition	46
5.3.2	Identify the decision spaces and its actions.....	46
5.3.3	Identify the uncertain events of nature.....	46
5.3.4	Create the decision flow	47
5.3.5	Metric to measure benefit and the utility functions.....	47
5.3.6	Estimate the probabilities	48
5.3.7	Solve the decision problem for each step.....	48
6	Discussion and Conclusions.....	51
6.1	CDS-DATOR.....	51
6.2	Naïve Bayes models.....	52
6.3	Criterion.....	52
6.4	Objectives achieved.....	53
6.5	Further work.....	53
7	Bibliography	54
8	Appendix.....	59
8.1	Input/Output schemas of the Bayesian decision library.....	59
8.1.1	Input schema	59
8.1.2	Output schema	62
8.2	Naïve Bayes Web Interface.....	63
8.3	Data mediator.....	64
8.3.1	PROforma case study.....	64
8.3.2	Curiam case study	66
8.4	DCIS case study	67
8.4.1	Diagram.....	67
8.4.2	Naïve Bayes model mapping file.....	68

1 ABSTRACT

The complexity in the decision-making process is usually caused by the large amount of knowledge of all the possible outcomes. This lack of knowledge, or uncertainty, is what can change drastically the benefits of taking one path of decision instead of another. In the medical environment, clinical guidelines can help in the decision-making process. However most of the systems that currently execute computer-interpretable guidelines model the guidelines leaving the uncertainty outside the decision process.

This work presents a decision support system (Criterion) that permits to develop guidelines in which the decision making is done in an environment of uncertainty. The proposed system gives the user the capability to manage the decision sequence, the utility of the consequences and the uncertainty of the uncertain events, providing a solution for the optimum decision sequence that maximizes the utility or minimizes the cost. The resolution algorithm is based in the Bayesian decision theory and the utility theory. Moreover, Criterion permits to model the uncertainty of the decisions as probabilistic models, which in combination with the utilities creates an optimal and personalized decision process.

During the development of Criterion, we realized that the technologies for storing clinical data as well as the systems making use of it are wide and heterogeneous, which may complicate the exchange and reuse of such data. Current Clinical Decision Support Systems (CDSSs) include modules that deal with this problem by acting as mediators between the data sources and the CDSS. The main problem that appears in these implementations is that the mediators are usually hard-coded within the systems and support one or two of the most usual types of data sources (e.g. XML documents, relational databases, or web services).

Therefore, in combination with Criterion, we propose a lightweight and modular data mediator to facilitate the data access to different data source technologies. The automatic mediation between the data entities used by the CDSS and the data entities of the sources is defined by a mapping scheme and data loaders. Data loaders are provided for different technologies such as web-services, relational-DBs, and XML documents (e.g., standardized clinical documents), leaving the possibility to extend them to other sources. The system is conceived as a non-invasive data wrapper that helps CDSS developers to include data access functionality with minimal effort.

A case study was analyzed to test the complete software functionality, the treatment given to the patients with ductal carcinoma in situ (DCIS, a type of breast cancer). Moreover, the mediator system was evaluated independently, by mapping data sources to two different CDSS: the PROforma computer-interpretable guideline (CIG) engine, and the CDSS for brain tumor diagnosis Curiam BT.

2 INTRODUCTION

Taking informed decisions usually implies a complicated analysis of the situation and needs an exhaustive design process. This process can be much more complex when the decisions are taken in the medical environment (e.g. what is the best treatment for this particular patient: a lumpectomy or a mastectomy? Should we give the patient radiotherapy?). To ease the process of taking decisions based on clinical evidence, the medical community created the clinical practice guidelines (CPGs).

Clinical practice guidelines¹ are a powerful method of standardization and improvement of the clinical care. They are usually defined as a set of plans, sorted in schematic layers that manage patients that have a specific clinical condition.

Nowadays, it is recognized that clinical guidelines based on medical evidence help in the quality and consistency of the medical care [13, 19]. Moreover, it has been proved that clinical guidelines can reduce the costs of such care in specific situations [29]. On the other hand, it has also been proved the problems that the physicians have accessing and using such clinical guidelines [36]. To better understand the clinical guidelines and why they can be so important in the medical practice, we have to know first what the main goals of the clinical guidelines are. All the clinical guidelines try to fulfill the following features:

- Describe the appropriate medical treatments based on medical evidence and group consensus
- Reduce the unjustified variations that can appear in the medical practice
- Provide a rational reference based on the best clinical practices
- Provide a focus for a continuous education
- Promote the efficient use of resources
- Be a focus for the quality controls, even the audits
- Show the possible deficiencies that can exist in the literature and suggest the correspondent future research

Computer-interpretable guidelines (CIGs) code the recommendations based on clinical trials and are able to offer new recommendations dynamically adjusted to the patients. This type of guidelines offer several advantages when compared to static paper guidelines. CIGs usually offer direct links to the references cited, providing the clinicians with the needed background and allow visualizing the clinical guideline in real time. CIGs can help to reveal errors or inconsistencies in the guideline content or codification through the use of an automatic validation process. Moreover, CIGs can help describing in a more precise way the possible states of a patient and can automatically propose an adequate solution for each case.

Although more and more clinical guidelines are being translated to CIGs, most of the guidelines that currently exist are only written in paper and are inaccessible to the clinicians that needed them the most [22]. Moreover, even when the guidelines exist in electronic format and are available through the web, the clinicians usually do not have the time and resources to use them due to the information overload that they face every day [20].

¹ From [12]: Clinical practice guidelines are a “systematically developed statements to assist practitioners and patient decisions about appropriate health care for specific circumstances”

To help with the needs of these professionals and guarantee the quality of their medical care, sophisticated software tools are needed. Due to limitations in our technology, nowadays is not feasible the analysis of textual guidelines without a predefined structure, so there is an urgent necessity to increase the diffusion and utilization of computer interpretable guidelines that can be read and executed automatically.

Criterion is presented as a software that facilitates the process of creating and executing CIGs, allowing to define clinical guidelines as paths of decisions.

The probabilities of the uncertain events specified in uncertain events and outcomes of the CPGs are usually calculated for average groups of people (e.g. the diabetes guideline have different values for the thresholds depending on the country it has been developed) which could lead to potential problems when treating patients that are out of the average range. To solve this, the system presented here also allows to personalize the decisions through the use of patient specific probabilistic models.

Thanks to the mediator module, we can communicate the clinical guidelines with probabilistic models or even complete CDSs that can provide extra information (e.g. personalized probabilities based on the patient's electronic health record) to the decision process. This transforms the decision process from something static to something flexible and open to change without having to manually modify the guideline.

In the following sections we are going to explain the creation and use of the Criterion decision software. Section 3 describes the state of the art for the current systems analyzed. Section 4 describes the methodology and frameworks used to create the system. In Section 5 several case studies are presented to proof all our methodologies. Discussion and conclusions are presented in Section 6.

2.1 OBJECTIVES

The main objectives of this work are:

1. Provide a transparent software mediator to access data independently of the source of such data and that is also easy to adapt to the system that wants to use it.
2. Provide a methodology to change the data sources without having to change the guideline definition or structure
3. Provide a methodology to allow the definition of clinical practice guidelines as Bayesian decision trees
4. Provide a methodology to manage probability and uncertainty inside the decision trees
5. Integrate probabilistic models in the decision-making process of the Bayesian decision tree
6. Design a software that implements the previous objectives and try a "real world" problem

2.2 CONTRIBUTIONS

The main contributions of the methodologies and the system presented in this work are:

- The mediator methodology has been presented in the 2014 - Arctic Conference on Dual-Model based Clinical Decision Support and Knowledge Management [40]

- The Naïve Bayes probabilistic models were used in combination with a decision support system in the HEDECAMA project (reference IPT-2011-1126-900000) and were presented in the international congress of the European Society for Radiotherapy and Oncology (ESTRO33) [42] and the Annual Meeting Scientific Program Committee of the American Society for Radiation Oncology (ASTRO) [31]
- The Criterion software has been registered at the UPV at it has been started using in real clinical environments

3 STATE OF THE ART

3.1 CLINICAL GUIDELINES

Several methodologies to create and execute computer interpretable guidelines have been developed by the medical informatics community in the last decades. A classification of these methodologies can be made if we analyze how they store and manage the clinical knowledge. There are four main categories: rule-based models, task-network models, decision tree models and document oriented models. The main problem with these methodologies is that they usually lack of uncertainty management, losing all the extra knowledge and not adding it to the decision process.

Regarding rule based models, there is a standard defined by the HL7 team (Health Level 7)² to model the medical knowledge as set of rules known as Arden Syntax [1]. Arden models the medical knowledge as independent knowledge modules known as medical logic modules (MLMs). One of the main characteristics that defines this syntax is that it allows to separate the medical logic that defines the guideline (codified with the Arden syntax) from the components dependent of the organizations that define such guideline. Still, the rule based implementations present several disadvantages such as:

- Not including an explicit capability to represent the generic medical logic of the guideline
- Not making semantic differences among the different types of knowledge presented
- Lacking the capability of representing and reusing in an easy way the guidelines and their components
- Not being compatible with guidelines that develop over extended periods of time, like the ones defined for chronic patients

Task-network models have been used during the last two decades to create complex medical care systems that can be executed along extended periods of time. Examples of such systems and guideline languages are: EON[57] and their continuation in SAGE[56], Asbru[35] inside the Asgaard[50] project, PROforma[15], GLIF3[7], HELEN[52], GLARE[55] and Prodigy[24]. All these models present the common characteristic of representing the decision process as if-then-else decisions which usually prevents or makes more difficult to represent and manage the uncertainty.

The decision tree methodology models the medical knowledge as hierarchical decision trees that contain the decision in node form. Some of these systems like ONCODOC [49] or EsPer [10] also fail to represent the uncertainty that always accompanies the decision process. However among these models exist one algorithm called ALCHEMIST [48] that is able to manage the uncertainty by defining in the decision tree model uncertain event nodes and utility measures. This tree model can be executed by the algorithm that will look for the absent data needed for the complete execution of the tree. When finished, the model presents a sensitivity analysis for the variables used in the tree and allows the user to change the tree information to adapt it to specific patients.

Moreover, other approaches exist regarding the uncertainty management such as the Markov Decision Process (MDP) [3] or the Partially Observable Markov Decision Process (POMDP) [21]. These models can also be used in the decision making process, being the main difference

² <http://www.hl7.org/>

presented against the decision trees the capability of representing infinite loops. Thus allowing the Markov processes to represent reoccurring events like in the ones happening in a chronic treatment. Although this is an advantage in comparison to decision trees, having an infinite horizon is not a reasonable assumption to model clinical guidelines.

Finally, from the document point of view, it also exists a standard approved by the American Society for Testing and Materials³ known as Guideline Elements Modeling (GEM) [14]. This modeling allows to structure the text document that contains the guideline as a document that contains well-structured Extensible Markup Language (XML). However, GEM doesn't provide an easy support to the tools that can interpret the result because it does not include a formal language that provides a clear computational model nor the capability to represent the uncertainty associated with the decisions.

3.2 DATA MEDIATOR

When connecting decision systems, we usually face an integration problem due to the wide range of data sources available at the moment. This data integration problem can be solved using a mediator system that allows the data to be parsed from the sources to the systems that need such data. To date, several mediators with different architectures that provide data access to specific CDSSs exist in the literature. Following, we analyze how these systems deal with the data integration problem from the data source point of view. Six solutions, which to our knowledge are the most relevant and up-to-date, were considered: MEIDA [18], KDOM [39], SAPHIRE [30], Kazemzadeh et al. [25], Komulainen et al. [27], and Shaker et al. [51].

The MEIDA system [18] receives a query from the user regarding clinical parameters, it sends the query to a Data Access Module (DAM). This module queries the local database through a query that is properly formatted with all the constraints needed and returns the response to the CDSS. It is assumed that the DAM is always part of some other mediator to the local clinical database, so a data wrapper for the DAM needs to be developed. In their case of study, the mediator IDAN [6] is used. IDAN is a mediator of temporal-abstraction queries to clinical databases. The goal of the IDAN mediator is to answer complex temporal queries regarding both raw data and its abstractions.

The KDOM [39] architecture deals with the data sources through a SQL generator. This module generates queries that are used to retrieve the data from the target electronic medical record and passes the data back to the system.

SAPHIRE [30] is capable of executing clinical guidelines defined in the GLIF language [8]. Inside the guideline definition, the author define the data sources so GLIF can access them during the guideline execution. These sources can be web services, IHE XSD schemas, or HL7 CDA documents and are used to gather all of the patient's data.

In the framework presented in Kazemzadeh et al. [25] the patient data and a mined knowledge previously preprocessed are encoded into HL7 CDA and PMML standards, respectively. Afterwards, the HL7 CDA is accessed through XPath⁴ expressions, and the knowledge accessed from the PMML documents is applied to this data. The result of this process is encoded as an HL7 CDA that will be used by the CDSS.

³ <http://www.astm.org/>

⁴ <http://www.w3.org/TR/xpath/>

The system in Komulainen et al. [27] is defined as a client-server architecture. The client side utilizes web services that use SOAP messages⁵ over *http* to send a query for the data needed. The server side is composed of a decision support service. This service receives the client query, combines a series of scripts that process the information requested, creates another query to a database, and returns the output back to the client side so that it can be shown to the user.

Data access in the Shaker et al. system [51] is obtained through an API. The system is internally defined as a bi-directional data mapping that is defined in a two-tier model. First, a URL containing supported query parameters is passed to a meta-wrapper. The meta-wrapper analyzes the URL and decides the data source to be targeted. Afterwards, the data requested is passed to a source-specific wrapper that produces a valid XML document containing the query results. This resulting document is processed by the meta-wrapper, which returns the data in the mediated schema format.

In summary, all of these systems access their data through one or more combinations of the following methods: SQL queries (created with different types of constrains), XPath routes (to query documents such as HL7 CDA or PMML) and web services. In section 4.4 we will see how the proposed mediator can provide access to all of these sources and how we can integrate the mediator to different CDSSs.

3.3 DECISION SOFTWARE

Up to our knowledge, there are at least four developed software packages for the resolution of decision problems: OpenDecisionMaker, TreeAge, SMLTree and the PROforma composer.

- **OpenDecisionMaker** is an open source project with GNU General Public License v3.0 that is available at Sourceforge⁶. It enables you to find the best alternative for a defined goal with the Analytic Hierarchy Process method. It offers a sensitivity analysis by simulation and provides a way to report the results in a pdf document. This software however is not based on Bayesian decision theory and thus it is sensitive to possible inconsistent solutions. Moreover, compared to our approach, this software does not take into account uncertainty management.
- **TreeAge** is a software developed by TreeAge Software, Inc.⁷. This software allows the user to build graphical decision tree models to analyze any type of decision problem, and analyze the model to evaluate each strategy and choose the best one. The advantage with respect to our software is that it offers the possibility to build Markov models, patient-level simulation models and time-to-event (DES) models. Finally, they can apply cost-effectiveness analysis, sensitivity analysis, and Monte Carlo simulation as well as reporting tools. The main disadvantage of this software is that it requires the user to have a deep understanding of decision theory, Markov modelling, and utility theory. Indeed, it has a graphical user interface that is very similar to an integrated development environment like Eclipse or NetBeans, and its manual has more than 600 pages for the user to read. Thus, this software is more focused on decision-theory expert users than on final users like physicians, financials, or businessmen, which are the target users of our software.

⁵ <http://www.w3.org/TR/soap/>

⁶ opendecisionmak.sourceforge.net

⁷ www.treeage.com

- **SMLTree** is a software used for drawing, calculating, and analyzing decision trees for general purpose decision problems, particularly for medical decision problems. It is focused for final users and presumably two hours of training should be enough for the user to work with it. It helps the user to frame a problem in a decision-analysis format, providing the ability to draw, graph, and perform sensitivity analysis of the decision problem. However, this software was developed as a scientific prototype and it is not easy to obtain it yet.
- **The PROforma composer** is a software developed for the creation and execution of PROforma guidelines. It permits defining PROforma guidelines as a set of data and tasks. The different type of tasks that the user can create are: plans, the main blocks of the guide. They can contain other tasks inside them; decisions, represent the step in the guideline where a choice is made; actions, represent the procedure to follow; enquiries, the nodes where the data bindings are defined. As previously commented, the problem with this system is the same problem that occurs with its language definition, it doesn't allow to define uncertainty. Hence, the user cannot introduce probabilities into the decisions because they are modelled as if-then-else nodes. This limits the software capability to model the complete medical knowledge that the guidelines should contain.

4 METHODS – THE DECISION FRAMEWORK

4.1 BAYESIAN METHODOLOGY

4.1.1 Bayesian decision background

To formalize a decision problem, the first step consists in the creation of the alternatives set. This is the called decision space and is denoted by D . This set must be exhaustive, meaning that it should contain all the reasonable possibilities that could occur in the context of the problem. Furthermore, the alternatives modeled inside D must be mutually exclusive, so if one action from D is chosen, the rest must be discarded.

If we have all the information available regarding a decision, make a decision is a straightforward process. The main problem comes with the lack of information. Deciding which decision take when we facing an uncertain environment is a complex process. How can it be determined the best decision if we do not have all the information relevant to our problem? We introduce this uncertainty in the decision process modeled as uncertain events, so for each one of the possible decisions inside the decision space D we have to consider a set of uncertain events that determine the possible consequences of taking such decision.

Considering that the problem that we want to model has a finite number of alternatives and uncertain events, it can be modelled as a decision tree (Figure 1) as follows:

- $D = \{d_1, d_2, \dots, d_i\}$ is the decision space. Decisions are represented as squares, and the alternatives for such decisions are represented as the output links of the square.
- $\theta_i = \{\theta_{i1}, \dots, \theta_{ik}\}$ is the set of the k uncertain events which occurrence affects to the result of taking the decision d_i . Uncertain events are represented as circles.
- When we take the decision d_i and θ_{ik} happens, the consequence c_{ik} is obtained.

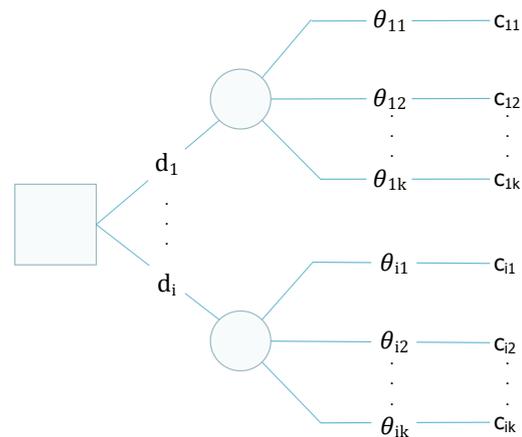


Figure 1: Bayesian decision tree

The measure to know how good or bad is a decision is calculated through the utility function (u). The utility function quantifies the consequences c_{ik} by giving a numeric value to the preferences of the decision-maker between all the possible consequences of his decision.

- The expected utility is the mean utility of the decision d_i and is calculated by the Equation 1
- The decision selected will be the one that maximizes the expected utility

$$u^*(d_i) = \sum_k u(c_{ik}) p(\theta_k | d_i)$$

Equation 1: Expected Utility

4.1.2 Bayesian sequential analysis

Being $D = \{d_1, d_2, \dots, d_T\}$ an unordered set of possible decisions in a complex decision process discretized in steps from 1 to T . Each decision in D may appear throughout the process. Each time t of the decision process, a decision d_t from D must be solved. In every decision d_i , it is possible to choose from a set of alternative actions a_{ij} , where $j \in \{1, \dots, d_i\}$.

Let a_{t-1} be the ordered vector of alternative actions selected in decision steps from 1 to $t-1$. After an action has been taken a set of possible uncertain events may occur. We define θ_{ijk} to be each of the uncertain events associated to each alternative action a_{ij} .

When the decision is at time step $t+1$, the previous events have been observed so they become facts. Hence, o_t is the fact observed at moment t . For instance, at the moment of deciding an adjuvant therapy for breast cancer we may already know the results of a sentinel node biopsy, which at the moment of surgery it was not known and thus it was an uncertain event. Hence, we see that an uncertain event can turn into an observed fact as the decision process goes on. Finally, Θ_t is the set of facts observed from 1 to t .

Given the Bayesian decision theory, the optimal decision is to choose the alternative action a_t^* that maximizes the expected utility at time t conditioned to the previously observed facts Θ_{t-1} and the already selected actions a_{t-1} in previous decisions, as expressed in equation

$$a_t^* | \Theta_{t-1}, a_{t-1} = \arg \max_j u(a_{tj} | \Theta_{t-1}, a_{t-1})$$

Where the utility of the decision given the previous history of facts and decision is defined as

$$u(a_t^* | \Theta_{t-1}, a_{t-1}) = \max_j u(a_{tj} | \Theta_{t-1}, a_{t-1})$$

And where the utility of the intermediate, non-final alternative actions must be recursively estimated from the conditional distributions of the states of nature and the expected utility of posterior decisions.

$$\mathbb{E}[u(a_{tj} | \Theta_{t-1}, a_{t-1})] = \sum_k p(\theta_{tjk} | \Theta_{t-1}) u(a_{t+1}^* | \Theta_{t-1}, \theta_{tjk}, a_{t-1}, a_{tj})$$

And where the utility of the final alternative actions is directly estimated from the conditional distributions of the states of nature and the utility of the outcomes

$$\mathbb{E}[u(a_{Tj} | \Theta_{T-1}, a_{T-1})] = \sum_k p(\theta_{Tjk} | \Theta_{T-1}) u(a_{Tj}, a_{T-1} | \theta_{Tjk}, \Theta_{T-1})$$

Where $u(a_{Tj}, a_{T-1} | \theta_{Tjk}, \Theta_{T-1})$ is the utility of the actions until the moment t given the states of nature.

In this moment, we constraint the utility of sequential actions given a set of states of nature to the sum of the utilities of each action conditioned to its associated state of nature. This constraint makes feasible recursive computational solution efficient keeping the model more flexible than Markov decision processes.

$$u(a_{tj}, a_{t-1} | \theta_{tjk}, \Theta_{t-1}) = u(a_{tj} | \theta_{tjk}, \Theta_{t-1}) + \sum_{a' \in a_{t-1}} u(a' | \Theta_{t-1})$$

4.1.3 Including observed facts obtained during the decision steps

As a result of an action carried out during any decision step d_t of a clinical guideline, new observations o_t from the patient may be available. These observations would modify the knowledge of the uncertain states of nature associated to the next decision steps. It is easy to incorporate these new observations to the vector of observed facts of our model. Hence, we define $\theta'_t = [\theta_t, o_t]$ which conditions the posterior uncertain states of nature, $p(\theta_t | \theta'_{t-1}) = p(\theta_t | \theta_{t-1}, o_{t-1})$ and the utilities of posterior actions $u(d_{tj} | \theta_{tjk}, \theta_{t-1}, o_{t-1})$.

4.2 CRITERION MODEL AND DATA FLOW

The Criterion system has been created following a meta-modeling approach using the latest open source frameworks available in the state of the art. Among other technologies, it uses the Eclipse framework⁸. Eclipse is an integrated development environment (IDE) for the Java language. The different modules that compose Criterion have been developed as independent software components (also known as plug-ins) of the Eclipse IDE, allowing us to take advantage of the Eclipse environment.

The core of the Criterion is defined in the Bayesian decision library (section 4.3). To provide more functionality to Criterion, a mediator library (section 4.4) is used to allow the software access to different data sources. The connection between the core and the GUI is done through the adapter library defined in section 4.5. The decision trees created can be automatically adapted to the data of the patient through the inclusion of probabilistic models, which is defined in section 4.6. Finally, the graphical user interface (GUI) which is integrated in the Eclipse IDE, is defined by the model, edit and editor plug-ins (see sections 4.7 and 4.8). The communication between these modules can be seen in Figure 2.

⁸ <http://www.eclipse.org>

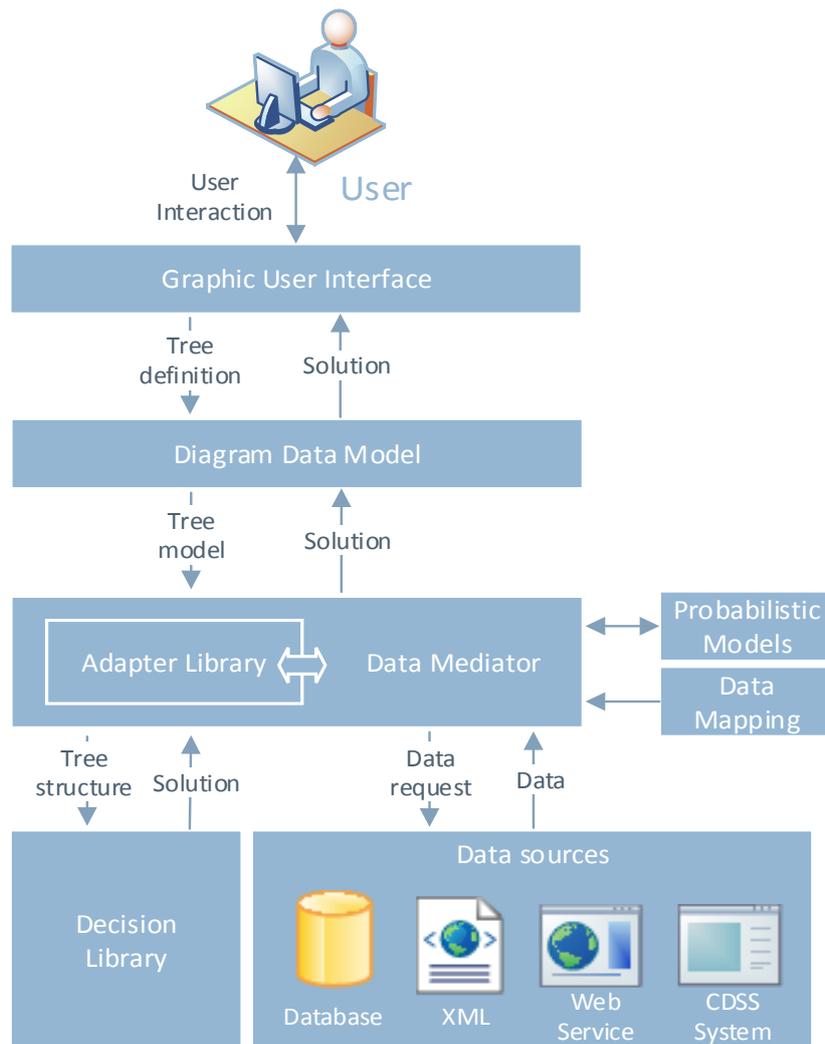


Figure 2: Data flow between the modules of Criterion. The user interacts with the GUI and creates a decision tree. This tree is internally stored as a diagram model. When the user decides to calculate the optimum decision path the tree model is transformed by the adapter library into a tree structure recognized by the decision library. The data needed for the tree execution as well as the probabilistic models defined are loaded by the data mediator and passed along to the decision library in combination with the tree structure. When the decision library solves the decision tree returns a solution that is passed back to the user.

4.3 BAYESIAN DECISION LIBRARY

The purpose behind the Criterion development was to provide the clinical users with a tool that allowed them to transform a clinical practice guideline into a decision problem. Also, to differentiate the system proposed from the rest and to aid in the decision making process, we introduced uncertainty and utilities management. This last step is not straightforward due to the difficult to define the utility functions in clinical domains.

The decision library included in Criterion and presented in this section makes use of the Sequential Bayesian Decision [5]. The proposed methodology allows us to obtain the best sequence of decisions (in each step of the decision process or in general) and to quantify the expected utility, providing also a way to personalize the probabilities for each patient.

4.3.1 Bayesian library development

The Bayesian decision library has been created based on the Bayesian decision theory presented earlier. It is composed by a recursive algorithm based on a cycling sequences of [decision step, alternative action, state of nature] to simulate a Depth-first search (DFS) of the decision tree. This library provides a computational framework over which we can construct a procedural methodology to calculate optimal decisions in clinical guidelines depending on the specific patients' conditions, and constitutes the decision engine of the Criterion software.

Figure 3 presents a simplification of the schema used in the implementation of this library. The *decision* and *uncertain event* model the nodes that are represented in the tree. The *alternative* store the utility function used when calculating the optimum path and the *uncertain events* store the probabilities of going through such path.

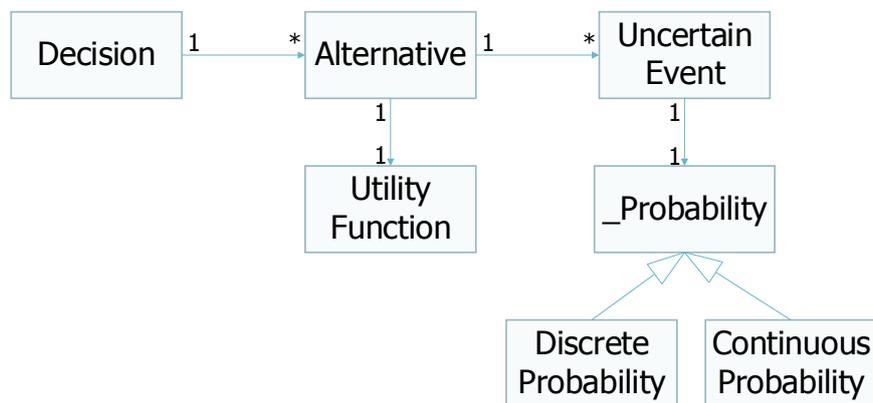


Figure 3: Simplified schema of the Bayesian decision library.

The main functionalities presented by this library are:

- Modelling the decision trees in a computer interpretable form: the library loads the input tree and calculates the solution automatically, based in the algorithm presented previously.
- Structure of the tree based in a XML schema: the library allows the input tree to be defined in XML format. This functionality improves the library functionality because the trees can be validated through the XML schema and also eases the library usability because any XML defining tools can be used to create trees.
- Solution presented as the path followed to take the decision: the library can print out the nodes that compose the optimum solution path and its utilities.

4.3.1.1 Library input/output

To test the library more efficiently and to facilitate its reuse and connectivity in other projects, XML schemas defining the accepted inputs and the created outputs were created. In appendix 8.1 the complete definition of the schemas can be revised.

4.4 THE MEDIATOR LIBRARY (INPUT/OUTPUT DATA ACCESS)

Traditionally, the data integration into CDSSs has been solved by people entering data to the CDSS manually, having to re-code it from the sources to the system. In a scenario like that, an automatic mediator that parses data from the sources to the existing systems could be useful. Such mediators exist, but they are usually included in the implementation of the CDSS and are focused on a specific type of data source (e.g., XML documents, relational databases, or web

services). Although this solution solves the data integration problem, it raises two important questions: What do we have to do if we need to connect other data sources? What do we have to do to connect other CDSSs to the data sources that we currently have? The first question could be solved with an intermediate system that allows the linking of new data sources, thus providing standardization and facilitating the process of creating new mappings. The second question could be solved with an intermediate system that is independent of a system that serves and does not require the modification of the CDSS to be connected. The solution presented here includes these features and has emerged to fulfill the need for lightweight and extensible methods to allow CDSSs such as clinical guidelines, gathering patient data.

The scope of this library is to facilitate data access functionality to CDSS developers and systems. We propose a solution in the form of a data wrapper/mediator architecture [41]. This type of architecture facilitates data access to different data sources exploiting the knowledge encoded in them and creates information for the higher layer of systems [58]. Additionally, this wrapper can be used to bridge the gap between CDSSs and semantic mediators, and between these two types of systems and their data sources.

We agree on the separation between a canonical data model and the access to it from a CDSS. The Clinical Decision Support Data Mediator (CDS-DATOR) presented here is based on mappings between the entities used by the CDSS (e.g. a node in a guideline) and the data entities that are present in the data sources (e.g., a value in an XML document). These sources can be based on databases, web service messaging standards, and electronic health records, but they can also be extended to any other source if needed. Proofs of concept for the PROforma technology [16] (language and engine) and Curiam BT CDSS [45] are presented in section 5.1.2. Moreover, a full integration of this mediator with the Criterion system is presented in section 4.7.

4.4.1 CDS-DATOR Methods

A generic mediator system should not be restricted to one kind of execution engine or data source. The method presented here is conceived to allow the integration of different CDSSs with different kinds of data sources. To facilitate the design, the mediator focuses on the CDSS data access layer. Hence, in order to have a non-invasive tool or module that provides CDSSs with data access capabilities, the architecture is conceived as a wrapper of the CDSS data layer. Thus, when the CDSS requires a data item or a set of data, our mediator will transparently provide it. Consequently, the CDSS will have separate definitions for the knowledge base (e.g., a CIG or the formula of a predictive model) and for its data access. This access is defined by the XML schema of our method, which maps the CDSS data requirements to specific data loaders as we will describe below.

Figure 4 shows three use cases in which the mediator can be used in combination with a CDSS and, if needed, a semantic integrator. Regardless of the path chosen, CDS-DATOR reads the data mapping from a mapping file and stores the information retrieved from the data sources in an internal cache from where the data can be read afterwards. Path 'a') presents the simplest use case. In path 'a') a CDSS connects with the CDS-DATOR asking for data; the mediator looks in the mapping file and retrieves the data from the specified sources. In path 'b'), a semantic mediator is used between the mediator and the data sources. The CDSS makes the data request to the mediator. CDS-DATOR transparently calls a semantic mediator and returns the data to the CDSS. The data provided to the CDSS is not raw data as in path 'a'); it has been processed by a semantic mediator and can present a structure and meaning. Path 'c') presents the most complex use case. The functionality is the same as path 'b'), but it also allows the semantic mediator to access multiple data sources through the CDS-DATOR.

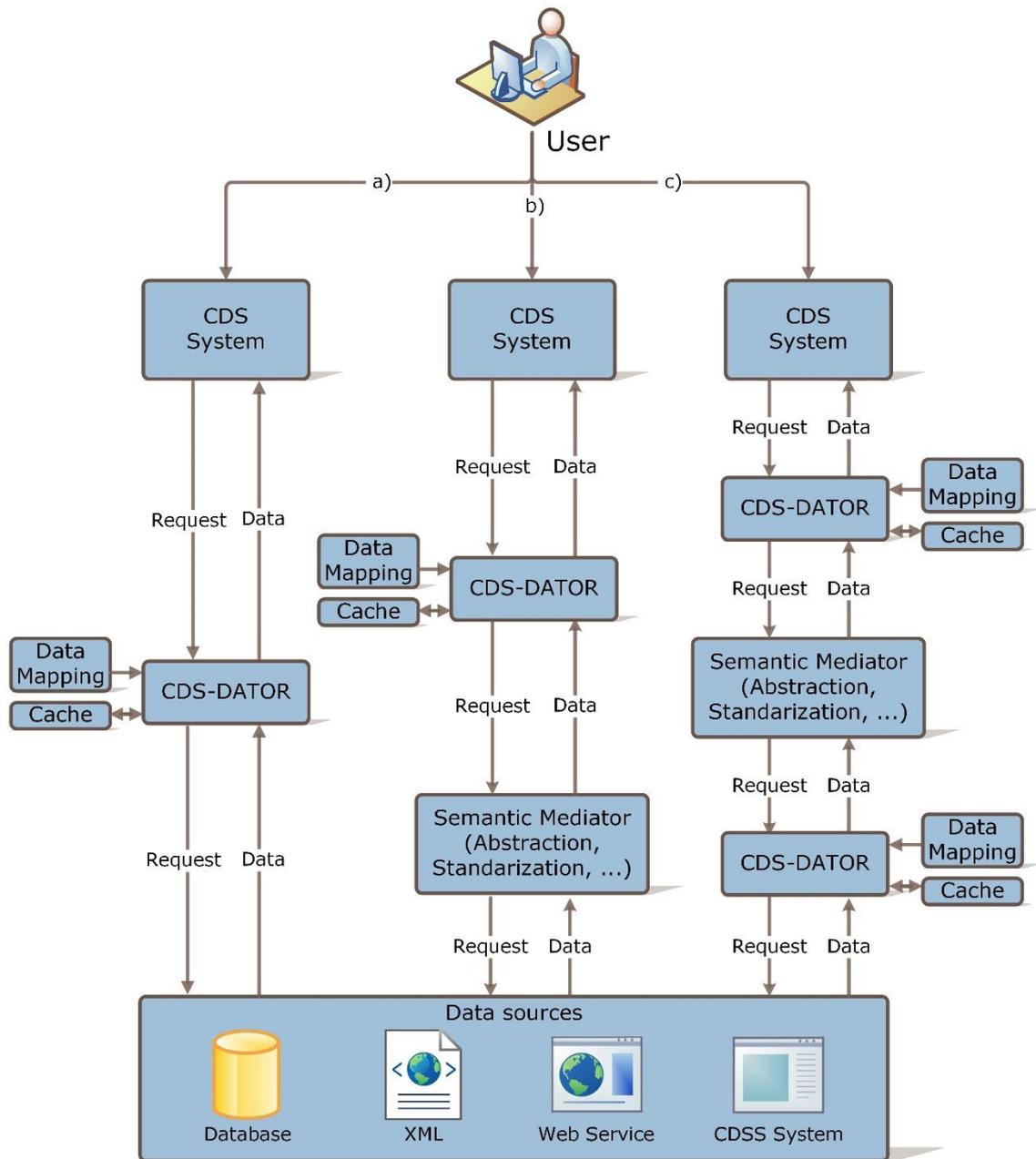


Figure 4: CDS-DATOR can be used as a data mediator between the data sources and the systems that make use of them (path 'a' or 'b') and between the CDSS and its semantic mediator (path 'c'). CDS-DATOR will deal with the semantic mediator as another data source of the CDSS.

The requirements of the system were modeled in UML. The set of classes and interfaces that make up the core of the system to configure and execute a data mediating scenario are described below. This domain architecture⁹ is defined in XML schema. The XML schema elements have a direct Java class representation with the logic required to perform the data gathering (which is automatically parsed using JAXB¹⁰). Therefore, instances of this XML schema

⁹ In this context, the term architecture corresponds to the definition of software architecture of Paul Clemens et al. [17]. "It is the set of structures needed to reason about the system, which comprise software elements, relations among them and properties of both".

¹⁰ <http://jaxb.java.net/>

define the data mappings of a specific CIG or predictive model that has patient data at a specified stage in its execution workflow.

To facilitate the system modularity, especially with regard to the interchangeability and incorporation of new data sources, the architecture is separated into two modules: one related to the CDSS execution engine and the other to the data source features. The engine module conforms the generic elements of the mediation (used as-is), and the data source module can take many forms (following the provided interfaces and adapted to specific data source technologies). Figure 5 shows the definition of the architecture of the first module.

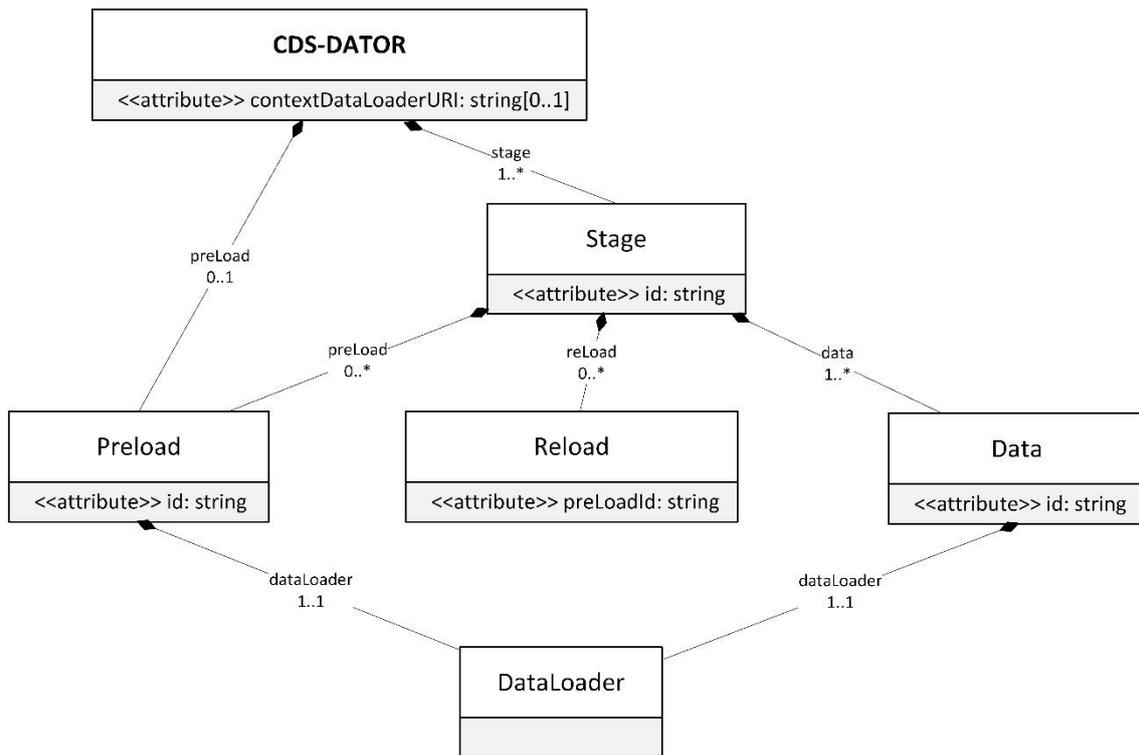


Figure 5: XML schema of the generic elements, which is represented in UML.

The elements that compose the engine module, as shown in Figure 5, are:

- **CDS-DATOR**: the main container of the mediator. It may comprise several stages since different CDSSs can have different ways of accessing working memory, there should be a way to define which structure/class should be used to load the data. The attribute *contextDataLoaderUri* specifies the URI of that class and allows the CDS-DATOR to access the internal data of the CDSS.
- **Stage**: a specific step in the CDSS inference or decision assistance. The mapping definitions of the data required in a stage are included in each stage.
- **Data**: it models a data item in the CDSS (input data) and binds it to a data source. This element contains the *DataLoader* that retrieves the value from the specified data source when the CDSS enters the corresponding stage.
- **DataLoader**: the component in charge of loading the data from a specific data source into the associated data element. It is provided as a generic interface with a generic load method. Therefore, depending in the corresponding technology any implementing class can define the required parameters and functionalities to load data on.

- PreLoad: permits loading specific data into the mediator cache that can be accessed later on. It can be used at the beginning of the CDSS workflow or at any stage (e.g., to load a patient EHR at the beginning of a CIG and consequently access it locally). As the *Data* element, it contains a *DataLoader* to accomplish this task.
- ReLoad: permits a previously pre-loaded data item to be reloaded when it is out of date (e.g., when the aforementioned EHR may have been externally updated).

The data source module is composed by specific interfaces called *DataLoaders*. *DataLoaders* can be defined according to two methods (generic or specific), as described in section 4.4.2. However, both methods can make use of a set of value types for their parameters. To allow the data loaders to use any type of parameter, a generic value type called *ValueOption* is defined. All the data types will inherit it. This permits the separation of the data loaders from the data types and allows new data types to be easily added or removed.

Figure 6 shows the functionality of the different sub-types of *ValueOption*. The *SimpleValue* permits the value to be specified directly in the XML instance with the tag *value*. A *URL* value indicates that the data needs to be searched in a URL address, such as a web service or a file. A *ContextValue* corresponds to data that already exists in the engine cache. Finally, in a *PreLoadValue*, the value of the parameter will be taken from the corresponding data that is stored in the cache of the mediator whose ID corresponds to the one specified.

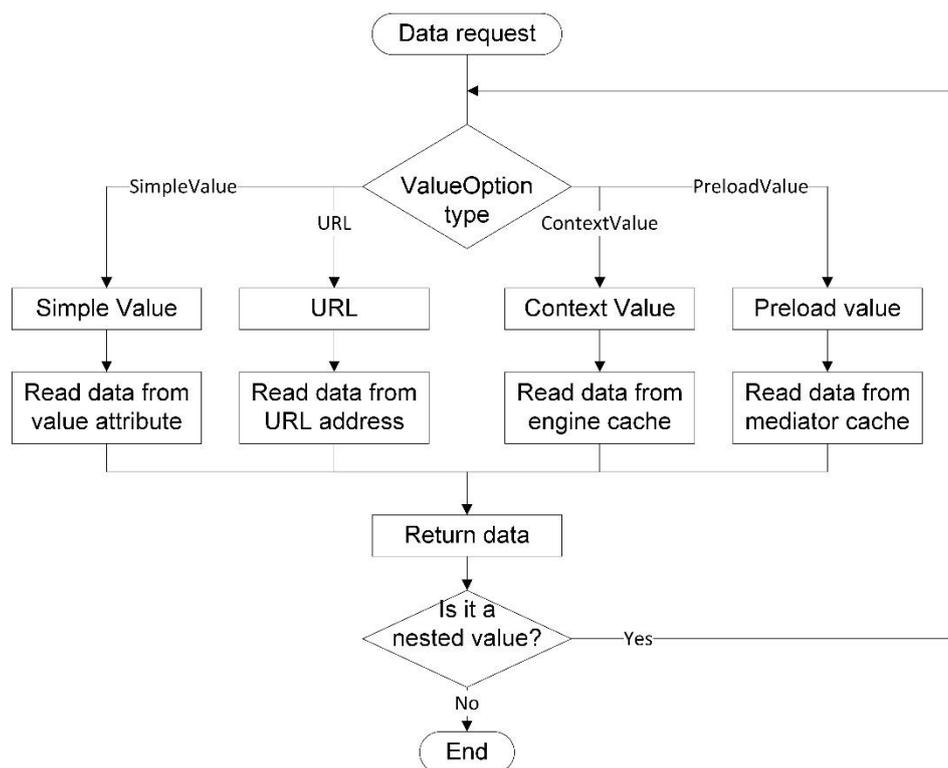


Figure 6: Workflow of the value retrieval

Furthermore, the *ValueOption* values can be nested. This means, for example, that in the path that defines a *SimpleValue* could appear a *ContextValue* (e.g. an *XPath* that specifies the route to some data can contain the patient's ID that is in the memory in the execution engine). In section 4.4.2 examples of this will be presented in combination with other ways of data loading.

There is a distinction between the mediator's cache and the working memory of the execution engine. The mediator's cache can maintain temporal data used by the method (such as data pre-

loaded by the *PreLoad*). On the other hand, the working memory belongs to the implementation of the execution engine (e.g., the set of variables in the context of a CIG). Hence, a task that needs to be done when building a new wrapper for a CDSS is building this memory bridge to allow the mediator to write and read in the working memory of the execution engine.

4.4.2 Data loaders

As mentioned above, the mechanism for loading data from data sources is defined in the *DataLoader* class. Provided as an interface, implementations are required for each different type of data source or technology. We propose two methods for developing specific loaders: 1) to follow the provided 'generic data loader' XML Schema configuration; or 2) to define new configurations for specific data loaders. These approaches are described below.

4.4.2.1 Generic Loader

The *GenericLoader* is composed of a set of elements that can be used to easily define a new *DataLoader*. Figure 7 shows the schema for this method.

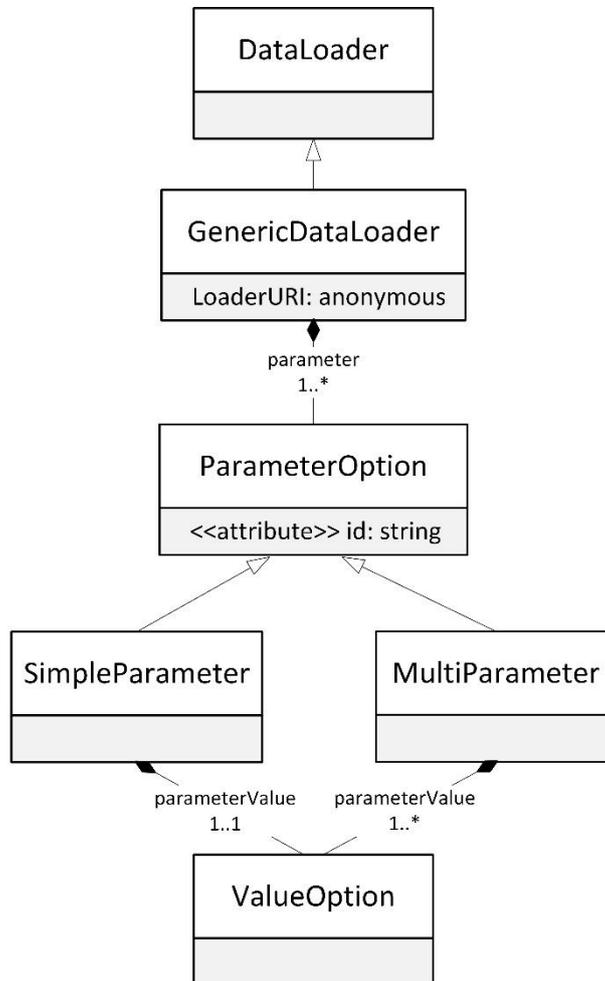


Figure 7: Generic data loader. UML representation of a XSD schema.

When a generic *DataLoader* is developed, a *loaderURI* that points to the loader class must be specified. This URI is called when the specified data is needed, and the class receives the parameters specified in the XML instance. Any number of parameters (*ParameterOption* class) can be specified and each parameter can have one (*SimpleParameter*) or several (*MultiParameter*) values. These parameters look up the data using the *ValueOption* interface

that we defined above. Additionally, a *ContextValue* value is defined inside the *SimpleValue* XPath route. This allows the user to dynamically define routes based on data that are known in execution time. A brief example of an XML instance using the *GenericLoader* can be seen in Code 1.

```
<dataLoader xsi:type="GenericDataLoader">
  <loaderURI>es.upv.ibime.bmg.cdsdc.xpathloader.XPathDataLoader</loaderURI>
  <parameter xsi:type="SimpleParameter" id="url">
    <parameterValue xsi:type="URL">C:/patientData.xml</>
  </parameter>
  <parameter xsi:type="SimpleParameter" id="xpath">
    <parameterValue
xsi:type="SimpleValue">//x:observation/x:code[@code='<query
xsi:type="ContextValue">patientID</query>']/../../x:value/@value</>
    </parameter>
  <parameter xsi:type="SimpleParameter" id="namespaces">
    <parameterValue xsi:type="SimpleValue">x=urn:h17-org:v3</>
  </parameter>
</dataLoader>
```

Code 1: Example of the Generic loader instance. The mediator looks for a file named *patientData.xml* in the url provided. Afterwards, it searches for the value of the data using the XPath route and the Namespace. Before resolving the XPath route, the value defined inside the 'ContextValue' is retrieved and used in the route definition.

The generic loader is useful for developing new data loaders; however, in order to facilitate the definition of XML instances of the most common sources, specific data loaders can also be used. The following subsections describe three specific data loaders that we have included in the mediator to deal with XML, Web Services, and relational databases.

4.4.2.2 XPathLoader

The *XPathDataLoader* permits data retrieval from an XML file in an easy and compact way. The definition of its schema is shown in Figure 8.

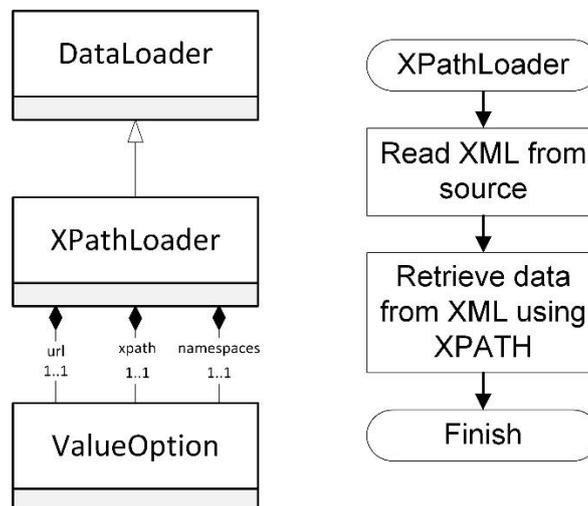


Figure 8: (Left) XPath data loader UML of an XSD schema, (Right) workflow of the XPath data loader

The *XPathLoader* uses three parameters to define the data source in the XML instance: *url*, *xpath* and *namespace*. The *url* contains the route where the document is located. The *xpath* corresponds to the XPath query where the data is defined, and finally the *namespace* corresponds to the namespace used in the XPath query to avoid ambiguities in the data definitions.

The Code 2 example has the same functionality as Code 1. However, the creation of the XML instance is simplified because only the mandatory elements are required.

```

<dataLoader xsi:type="xpath:XPathLoader">
  <xpath:url xsi:type="URL">file://C:/patientData.xml</>
  <xpath:xpath xsi:type="SimpleValue">//x:observation/x:code[@code='<query
xsi:type="ContextValue">patientID</query>']/../x:value/@value</>
  <xpath:namespaces xsi:type="SimpleValue">x=urn:h17-org:v3</>
</dataLoader>

```

Code 2: Example of the Xpath loader. CDS-DATOR will look for a file named patientData.xml in the provided url. Afterwards, it will search for the value of the data using the xpath route and the namespace.

4.4.2.3 WebServiceLoader

The *WebServiceLoader* permits data to be retrieved from web services based on SOAP or REST specifications. To use a REST service, we need to define the web service URL, the expected data type of the return value, and the parameter or parameters sent to the web service. In the case of a SOAP web service, we need to define the web service URL, the name of the method to call, and the parameter or parameters that will be passed to this method. As in the previous loaders, we can use any of the parameter types provided with the mediator as parameters (such as a set of variables in the context of a CIG). The schema of the *WebServiceLoader* as well as its workflow is shown in Figure 9. A usage example of this loader is presented in section 5.1.1.3.

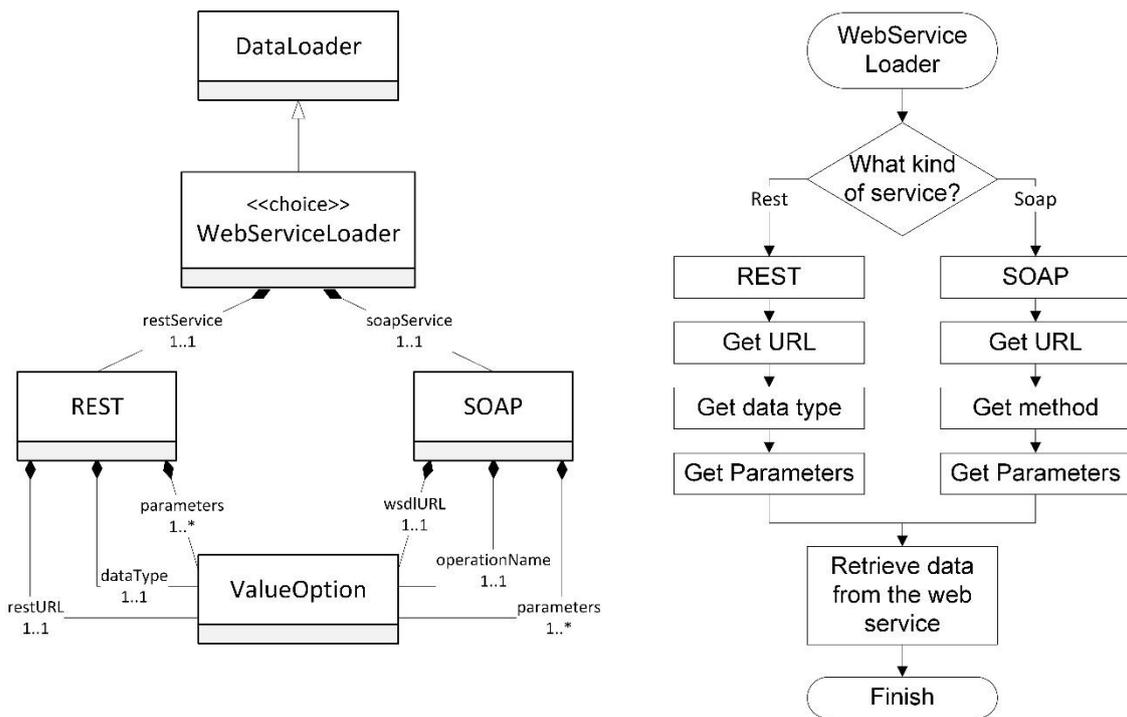


Figure 9: (Left) WebService data loader UML of XSD schema, (Right) workflow of the WebService data loader

4.4.2.4 SQLLoader

The *SQLLoader* permits the definition of SQL queries to relational databases. Figure 10 shows the schema and workflow of this loader. The loader requires two parameters: the connection URL and the SQL query. The parameter *connection* contains a URL that specifies the database connection parameters such as the address, user, and password. The parameter *sql* will contain the SQL query needed to retrieve the data required for the database. Also, as shown in section 4.4.2.1, the output of one loader can be the input of another loader, providing the SQL loader with the capability of dynamic query definitions allowing both the connection and the query to

be obtained from different sources. A usage example of this loader is presented in section 5.1.2.2.

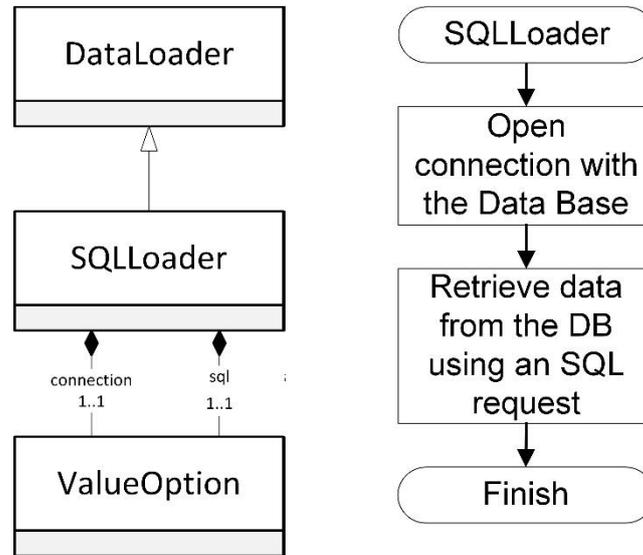


Figure 10: (Left) SQL data loader UML of XSD schema, (Right) workflow of the SQL data loader

4.4.3 Mapping data sources

Defining a new mapping between an execution engine and the data sources is a straightforward procedure. The steps to define a new mediation are the following:

- Identify the data required to be mapped. From the data required by the CDSS, we need to identify the data items and, if applicable, the stage of the CDSS inference in which it is required.
- Identify the data sources to be used. We need to identify the data source for each of the data items from step 1. Depending on the source, we will define the corresponding data loader associated to the corresponding technology as well as its parameters.
- Define the mapping between the sources and the data required by the CDSS. Finally, once the involved data and sources are identified, the final step is to write the XML CDS-DATOR mappings. At each stage in which a data item is needed, we will define its corresponding mapping.

4.4.4 Procedure to use the CDS-DATOR in a CDSS implementation

Defining a connection between a CDSS and the CDS-DATOR implies building a bridge class to connect the memory of the mediator with the memory of the CDSS. The steps to define a new connection are as follows:

- The data retrieval manager/module of the CDSS has to be found
- A wrapper of this data manager has to be developed, allowing the CDSS to connect with the CDS-DATOR
- Everytime a new data item is required by the CDSS, the previously defined wrapper will be called, starting the entire process of data retrieval described above

4.5 ADAPTER LIBRARY

As stated earlier, the Bayesian library was designed originally to test the Bayesian uncertain theory developed for introduce uncertainty in decision trees. Afterwards, a graphical user

interface was developed, but the original Bayesian library was too complex to represent all the knowledge in the interface so a simplification to present the data to the user needed to be done. The model schemas for the Bayesian library and the GUI can be seen in Figure 3 (section 4.3) and Figure 12 (section 4.7) respectively. Although similar, these schemas presented some expected differences due to technical requirements, so an adapter library was created (see Figure 11).

This library does not only manage the data interchange between the Bayesian library and the GUI, it also provides an accessing point to create and adapt more data managers, like the probabilistic models of section 4.6 and the data mediator defined in section 4.4.

Thanks to this decoupled design, the user interface and the Bayesian library are independent from each other, allowing us to make changes independently. The adapter library takes care of all the code transformations needed so if one library changes, the other does not need to be notified of such change.

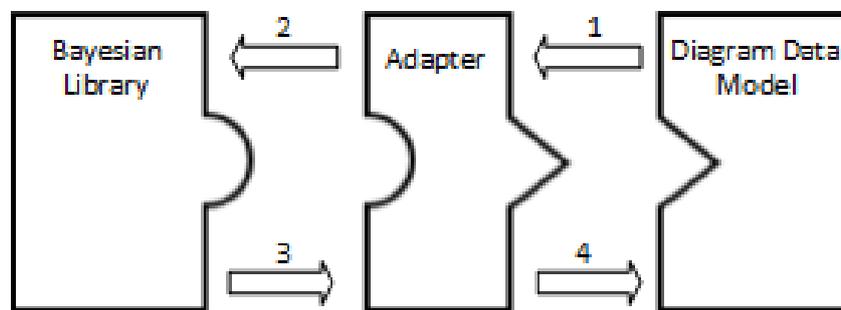


Figure 11: Adapter functionality schema. The diagram data models passes the data contained in it (1) and the adapter transforms it to the decision library data model (2). When a solution is calculated for the Bayesian library, the data is returned to the diagram through the adapter (3, 4).

4.6 PROBABILISTIC MODELS

The power of the decision process presented Criterion relies on the probabilistic management of the uncertain events and the utilities of the decisions taken. The utilities are set in the diagram and usually are fixed unless new medical evidence is discovered. On the other hand, the probabilities of the uncertain events don't need to necessarily be static neither equal to all the patients. A fixed set of probabilities based on the medical evidence can be established, but a dynamic set of probabilities adapted to the patient's need would provide a better and more informed decision.

That is why Criterion can make use of probabilistic models that will set the probabilities of the uncertain events. These models can be trained with the medical evidence available, creating models that adapt to the data of the patient and could produce different paths for different patients.

4.7 DIAGRAM DATA MODEL

The diagram data model used relies on the eclipse modeling framework (EMF) tools (i.e., the Ecore Tools and the extended editing framework), that provides to the developers an architecture to define widgets and data binding.

The core of the diagram data model is specified following the XML Metadata Interchange standard¹¹ (XMI). The XMI is a standard for exchanging metadata information via XML. The main usage of the XMI format is the information interchange of UML models. This provides the core of the application with the interoperability needed to use other EMF-based tools, and allows us to serialize and deserialize the information following the XMI interchange standard.

Models inside the XMI file can be specified using annotated languages such as Java, UML or XML and other specific tools. For the development of Criterion we used the EMF modeling tools. These tools simplify the creation of the XMI file providing a series of editors that create an Ecore file inside of which are defined the classes and the relations that are going to be used by the system. The Ecore file is internally specified following the XMI standard. Figure 12 presents the simplified diagram that is modeled inside the Ecore file:

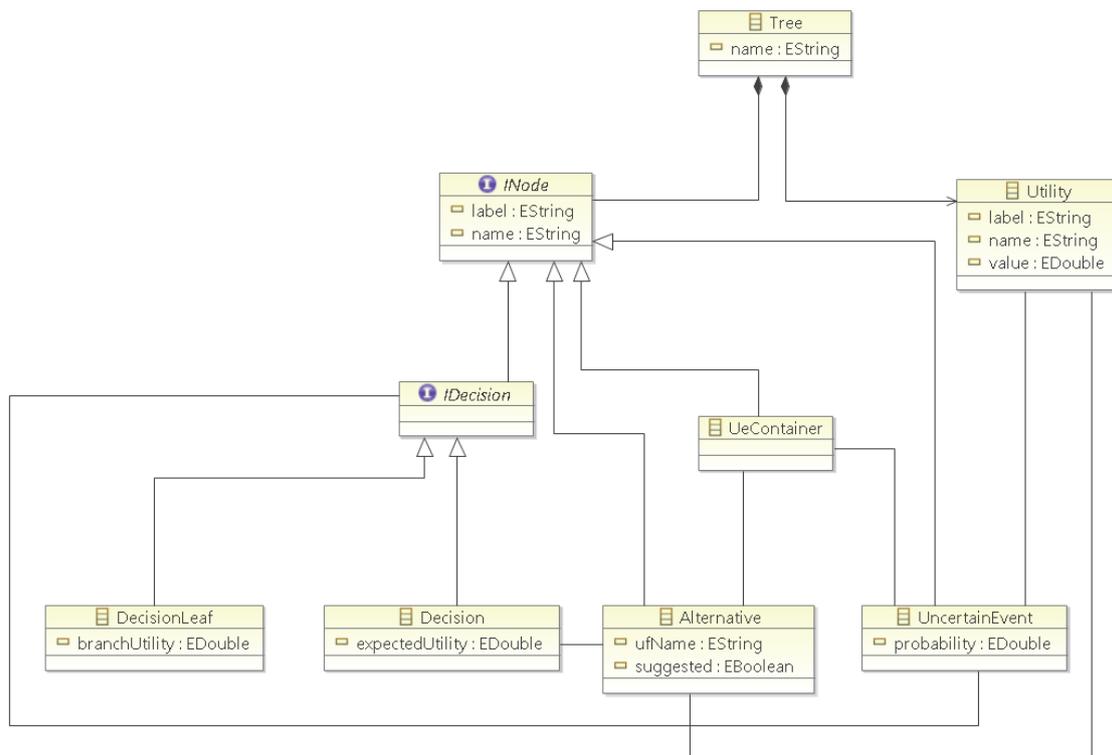


Figure 12: Simplified UML schema representing the classes and relations of Criterion

Based in this model specification, the generator model tool provided by the EMF allows to generate Java code at runtime. This generated code is comprised by a set of classes that define the specified model and a set of adapter classes that enable viewing and editing the data inside the model.

As can be seen in Figure 12, the schema definition created for the user interface is similar to the schema of the Bayesian library present in section 4.3. This was done intentionally to simplify the information interchange between the user interface and the mathematical library. However, the complexity behind the Bayesian library did not allow a one-to-one match and an adapter library had to be created, as it can be seen in section 4.4.

¹¹ <http://www.omg.org/spec/XMI/>

4.8 GRAPHICAL USER INTERFACE

The graphical user interface creation follows a meta-model methodology. The manually programmed code has been reduced to the minimum possible, relying all the code generation on the generation schemas. This programming style is capable of generate thousands of lines of code automatically, reducing greatly the development time and easing the maintenance of the software. The next subsections present the data editors created to allow the user define and execute Bayesian decision trees.

4.8.1 Sirius

The user interface has been developed used the Sirius framework¹², a project inside the Edipse framework that allows the users to create, edit and visualize EMF models. The Sirius library eases the process of binding the data model with the graphical interface through a series plugins that save lots of coding time.

From the Sirius wiki¹³: “Sirius enables the specification of a modeling workbench in terms of graphical, table or tree editors with validation rules and actions using declarative descriptions. All shape characteristics and behaviors can be easily configured with a minimum technical knowledge. This description is dynamically interpreted to materialize the workbench within the Eclipse IDE”.

This dynamic evaluation of the schema created, allows to define the diagram structure to be built and test at the same time. Figure 13 shows how the diagram model is created. After the creation and validation of the model, the user will only see the editor showed in the right.

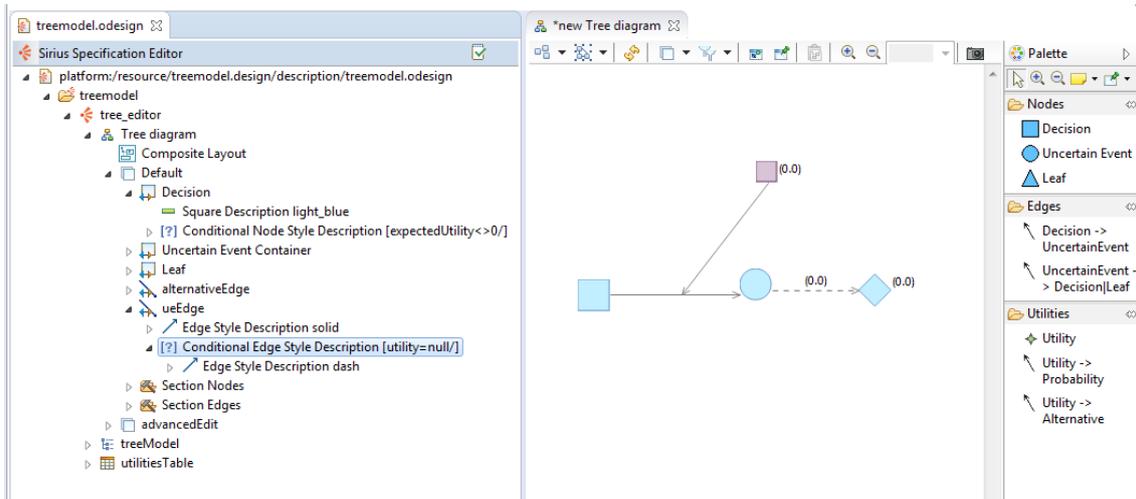


Figure 13: Sirius editing interface. Left, the diagram model definition tools. Right, the test diagram used to see the changes in real time. In this example a special dashed style has been created to show that an edge don't have a utility assigned. The style can be seen at the same time in the editor at the right.

Furthermore, Sirius does not only provide a graphical diagram editor, it also permits connecting the diagram model with Java code, allowing to make calls from the diagram to predefined methods or classes. Besides the model editor, Sirius provides different viewers to present the data to the user. Figure 13 (right) shows the diagram viewer. The diagram viewer has functionalities such as sorting the diagram elements (automatically or following a layout), present the data in layers (allowing to show/hide different layers of information), filtering the

¹² <http://www.eclipse.org/sirius/overview.html>

¹³ <http://wiki.eclipse.org/Sirius>

elements (a filter can be defined to show/hide only specific elements) and exporting the diagram as an image. These features and more are explained in section 4.8.3.

4.8.2 Project Explorer

The project explorer view allows the user to create or delete projects inside of which the diagram models are defined. Figure 14 shows the project explorer view for a simple tutorial created. The diagram information and data bindings are stored in the file *.aird and the file storing the model information is the *.decision. Inside the *aird* file different views to edit the data can be created. Figure 14 shows the three views that the user can create to show the data available in the data model.

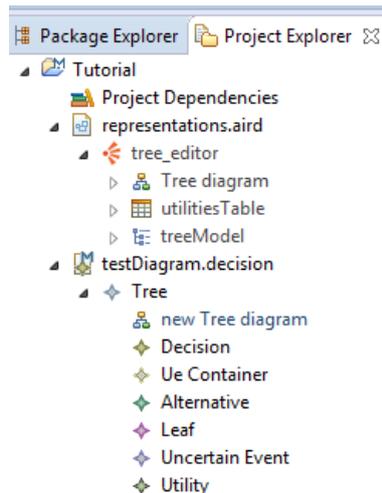


Figure 14: Criterion project explorer view

Furthermore, Criterion makes use of all the Eclipse functionalities so if the user wants, the projects created can be shared, stored or even managed through a version control system such as Subversion¹⁴ or Git¹⁵.

4.8.3 Graph Editor Viewer

The graphical editor is the main editing window used when creating the decision tree and it displays almost all the diagram information. It is composed by two different parts, the graph viewer and the edition palette. Figure 15 presents part of the graph created in section 5.1.

¹⁴ <https://subversion.apache.org/>

¹⁵ <http://git-scm.com/>

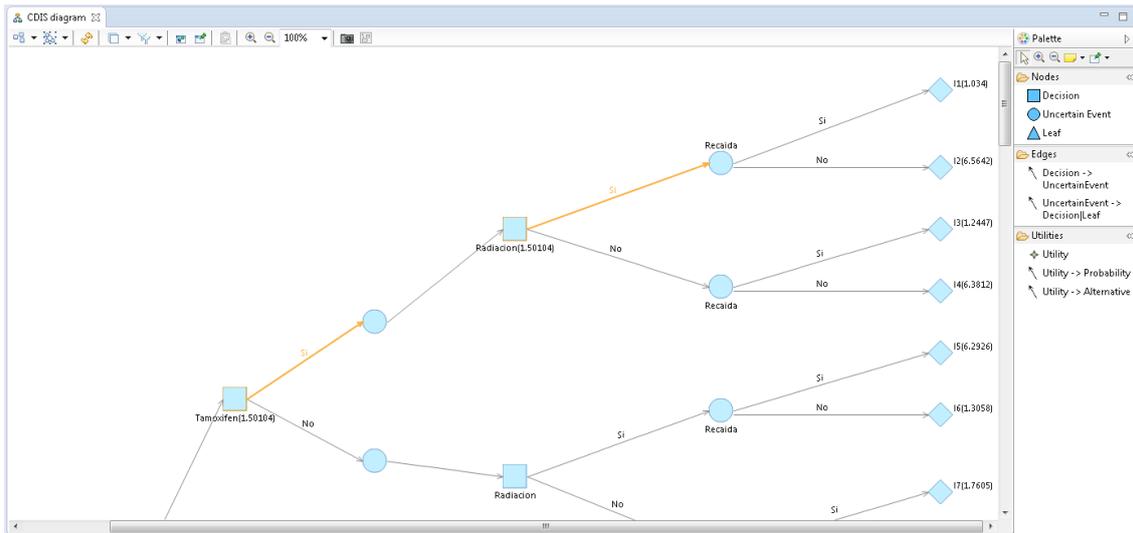


Figure 15: Graph editor. Almost all the view is occupied by the graph viewer which allows to move, select, edit and connect nodes. The left part of this editor presents the palette, where the user selects the node or connection that wants to create.

The graph editor also allows to show/hide diagram information, so the user can focus in editing only selected parts of the graph. At the top of the editor there is a tool bar that offers the user some special features. This toolbar changes if the user selects a graph node or not, presenting specific toolbars for diagram editing or node editing. Figure 16 shows the diagram toolbar which is active by default. This toolbar presents features like: arrange all nodes (1), select all nodes (2), refresh the diagram information (3), show/hide diagram layers (4), create/apply filters to the graph (5), show/hide specific diagram elements (6), zooming (7), and export the complete diagram as an image (8).



Figure 16: Diagram toolbar showed when the user does not have a node selected.

Figure 17 shows the node toolbar. This toolbar appears when selecting a node or a set of nodes. It offers functionalities such as: arrange (1), align (2), pin/unpin (3), copy layout (4), show/hide node (5), show/hide node label (6), delete node (7), several font styles (8), several node color and connector styles (9), and auto size (10).



Figure 17: Node toolbar showed when the user selects a graph node.

4.8.4 Graph Properties Viewer

Although the graph editor shows enough information to create the diagram, sometimes the user need to introduce more specific information in a node. Having all this information in the graph editor would complicate excessively the diagram edition show a properties view was created to deal with such task. In the properties view, the properties of the node currently selected in the diagram editor are showed.

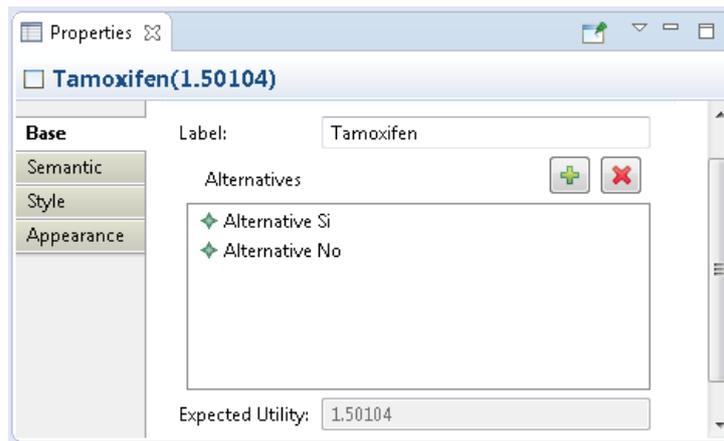


Figure 18: View presenting the properties of a decision node.

The properties editor presented in Criterion is showed in Figure 18. This editor presents all the node information to the user in a friendly and easy to use way. We achieved this by using the Extended Editing Framework which provides the functionalities to define data model editors and bind them with the diagram editor.

4.8.4.1 Extended Editing Framework.

Figure 19 shows how the data is presented to the user when using the standard editor provided by Eclipse to display data. Internal properties that the user should not edit such as the *Name* (the internal id of the node) are presented. Also the medical evidence data which is composed by a list of medical resources is not displayed entirely, and it is difficult to edit not displaying any edit buttons and only answering for double clicks.

Base	Property	Value
	Utility u111	
	Alternative	Alternative Si
	Evidence	Swedish Study (http://www.ncbi.nlm.nih.gov/pubmed/18250350)
	Label	u111
	Name	_ck2UoPDpEeOgB5t8hN2jQ
	Uncertain Event	Uncertain Event Si
	Value	-2.3701

Figure 19: Utility node properties presented using the standard eclipse editor.

To solve this issue we created the node data editors using the extended editing framework. Based in two mapping files, one for define the data bindings of the model properties with the components showed in the interface, and the other to define the code generation process, we are able to present the user the data input interface showed in Figure 20. The main differences

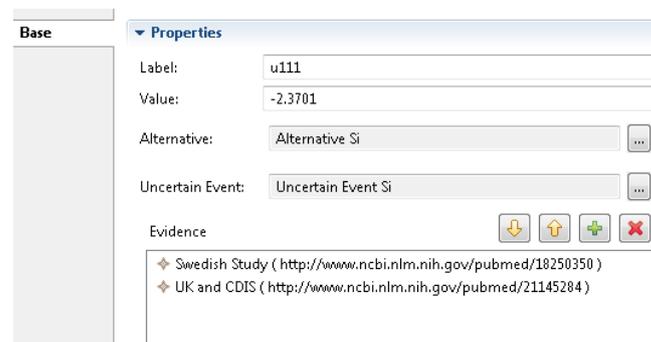


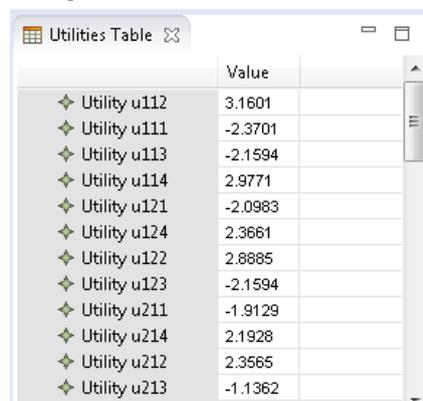
Figure 20: Utility node properties presented using the extended editing framework.

between Figure 19 and Figure 20 are a cleaner presentation of the data and the possibility to edit all the data with only the left click of the mouse (an important feature when dealing with inexperienced users). Furthermore, new functionalities have been included, among others, allowing the user to navigate to the URL introduced in the medical evidence field.

The inclusion of this editing framework does not only improves the data presentation. Defining the data properties bindings in independent files provides our project modularity and reusability, allowing other developers to introduce new properties or changes in the editors without having to modify any other part of the project.

4.8.5 Utilities Table viewer

Inside the utilities nodes, we have to introduce the value that we want to use in the calculations of the best decision path. We dealing with small trees, we can edit these properties directly in the graph editor or in the nodes properties. However, when we have to manage the utilities of a big tree (like the one presented in section 5.1) the task of defining or updating all the utilities can be tedious. The utilities table viewer (Figure 21) has been created to solve this issue, it presents all the utilities defined in the graph in one place and allows the user to edit them without having to navigate the diagram or the tree definition.



	Value
Utility u112	3.1601
Utility u111	-2.3701
Utility u113	-2.1594
Utility u114	2.9771
Utility u121	-2.0983
Utility u124	2.3661
Utility u122	2.8885
Utility u123	-2.1594
Utility u211	-1.9129
Utility u214	2.1928
Utility u212	2.3565
Utility u213	-1.1362

Figure 21: Utilities table viewer. Presents all the utilities of the graph in a table.

4.8.6 Tree Model Viewer

The tree model viewer has been created to give the user a compact view of the tree diagram that is defined. The nodes of these tree view are the nodes defined in the diagram and the father/child relationships are defined through the connections created in the graph editor.

Moreover, when the user clicks in the nodes, the properties view reacts showing the properties of the node selected as if the node has been selected in the diagram editor.

The collapsible tree model viewer has been proved a good ally when dealing with big trees, allowing the user to keep track on the nodes and its relations although they are not show in the main diagram editor.

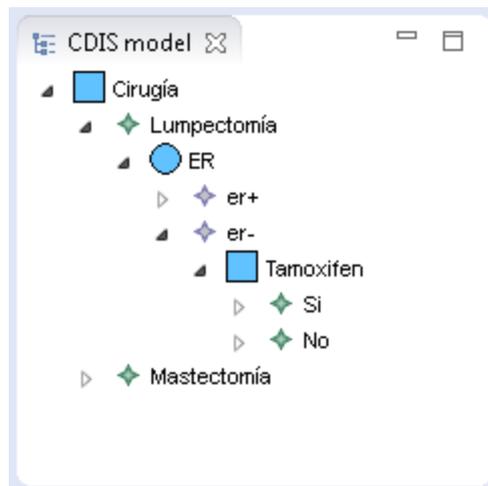


Figure 22: Tree model view. The tree nodes and relations are presented in a compact and collapsible form.

5 RESULTS

In the next sections we are going to analyze the case studies carried out to test the methodologies proposed.

5.1 CDS-DATOR

In this section, we present the results of two case studies that describe the steps followed to develop the wrapper, a simple test case, and finally, the results of the integration. The first one integrates CDS-DATOR with the PROforma CIG engine. The second one allows the Curiam CDSS engine to query a database and retrieve data to classify a case.

5.1.1 PROforma case study

For the purpose of evaluating the effectiveness of the system in CIGs, we have developed a wrapper for the PROforma [16] CIG engine. Such execution engine is the responsible for executing the PROforma guidelines. Since it manages the inputs and outputs of the guideline, it will be the component wrapped by the mediator.

The PROforma language models the clinical knowledge contained in a guideline as a set of tasks and data items. The task abstraction can be subdivided into four different types: plans, decisions, enquiries, and actions. A plan is a container task; it can contain any type and number of tasks, even other plans. A decision task implies a selection between options presented during the execution of the guideline. An enquiry is a request for information at a specific point of the guideline and an action is a procedure which has to be carried out.

The PROforma execution engine enacts the CIG written in the PROforma language. A parser instances the required classes and the structures that are defined in the CIG before it is started. The execution of the guideline can be automatic or step by step, (with the term step meaning the progress between tasks inside the guideline). When the guideline needs external data, the engine has to make a request through an enquiry. Assuming that a guideline needs a large amount of data (e.g., the EHR of the patient when the guideline starts), the time spent manually entering this data may be substantial. CDS-DATOR can overcome this issue by providing the guideline with direct access to the required data on the EHR system.

5.1.1.1 *Integration with PROforma*

PROforma could be extended to allow similar data access capabilities by means of extending its own language; however, this would require changing the original language definition, since it is tied to updates and changes in the language. Our solution does not require any modification in the original PROforma language or code. Hence, the system developed over the PROforma version 1.6 maintains compatibility with the current 1.7 version.

Basically, the CDS-DATOR PROforma wrapper consists in adding event listeners to the nodes where external data is required, i.e., those specified in the CDS-DATOR XML definition (whose name must be equivalent to the node name in the PROforma guideline definition). Thus, the wrapper mainly consists in an extension of the PROforma execution engine entry class (PFEEngine class), extending the PROforma task listener (TaskAdapter) and assigning the corresponding events to the corresponding nodes at startup (Figure 23). Thus, when a data item is required in the enquiry nodes, our event calls the CDS-DATOR procedures to retrieve data from the corresponding data source using the corresponding parameters. When this execution finishes, the execution line will continue in the task listener, which will add the retrieved data to the

PROforma engine. In our case study, the wrapper required two classes and less than 200 lines of code in total. This demonstrates the minor effort that may be required to integrate the CDS-DATOR to provide data access to a CDSS.

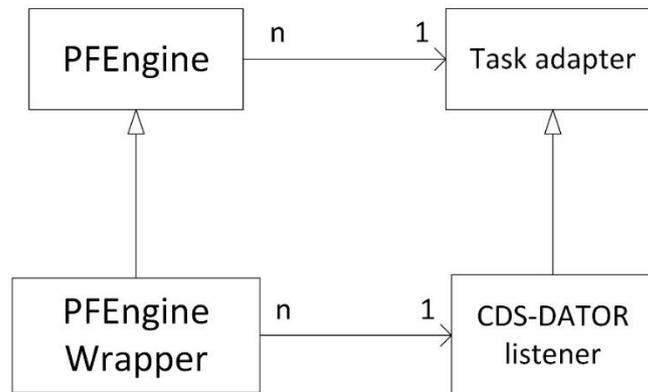


Figure 23: The PFEEngine and TaskAdapter classes belong to the PROforma engine. Extending them in this way allows the software to register new listeners in the engine without having to modify the code inside the engine.

5.1.1.2 Description of the case study guideline

We tested the integration of CDS-DATOR with PROforma using a clinical guideline for the diagnosis plan of patients with chronic obstructive lung disease (COPD); the guideline was written in the PROforma language. The evaluated guideline (Figure 24) was a subset of the complete guideline for the care of patients with COPD and chronic heart failure published in [32]. The tested guideline classified the COPD stage of the patient according to the clinical data and the results of a spirometry test. In order to fully explore the capabilities of CDS-DATOR, we introduced a minor modification in the guideline to retrieve two of the required values from an external CDSS as we describe further on.

The evaluated guideline included: three enquiry nodes, where the data is requested (rhombus); two decision nodes, where a choice between nodes has to be taken (circles); and five action nodes, where the clinical procedures are specified (squares).

The first enquiry node required the following patient data: *age*, *gender*, *chronic_cough*, *chronic_respiratory_failure*, *chronic_sputum_production*, *dyspnea* and *risk_factors*. We considered these data to come from the medical history of the patient. Afterwards, the decision node *compatible_clinic_decision* analyzed the data and recommended one of its options, which may or may not be in favour of continuing the clinical tests.

In our case study, if the node *spirometry_enquiry* is reached, the guideline requested the current measurements of the patient: *FEV1*, *FVC*, the ratio of these (*FEV1/FVC*) identified as *FEV1_FVC_postbronchodilator*, and the current patient *weight*. *FVC* corresponds to the forced vital capacity, which is defined as the maximum air volume that can be exhaled by the patient from a maximally forced expiratory effort. *FEV1* corresponds to the air volume exhaled during the first second of a forced expiratory effort, which is carried out before a maximally forced inspiration. The value of the ratio of these measurements must be compared with a reference value on a healthy person with similar characteristics in order to diagnose COPD. To accomplish this task, the enquiry *spirometry_results_enquiry* makes a request to an external CDSS which takes the data of the patient and internally retrieves the reference value, providing the data *FEV1_prediction* as a result. This result is required for further diagnosis. This calculus is carried

out according to the norms of the Spanish Society of Pneumology and Thoracic Surgery (SEPAR) [38]. Finally, based on decision rules over the collected data, the decision node *spirometry_results_decision* will determine the COPD stage.

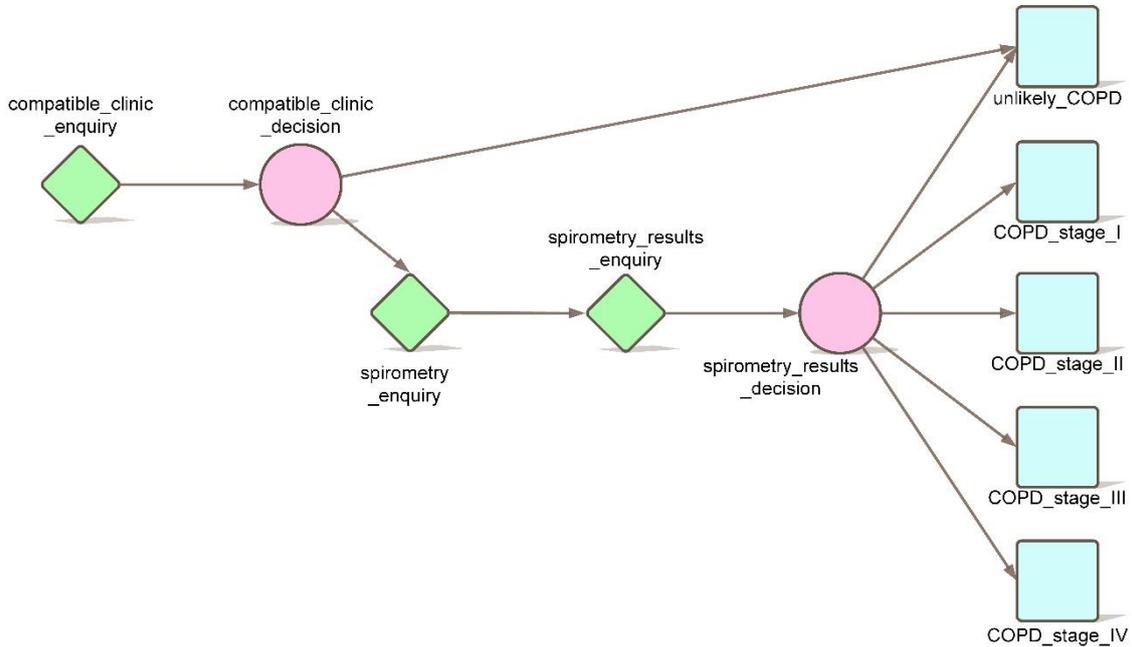


Figure 24: Graphical representation of the PROforma guideline for COPD diagnosis used to test the CDS-DATOR.

5.1.1.3 Proof of concept

In our case study the guideline described above was executed in the wrapped PROforma engine using the CDS-DATOR to retrieve the data required throughout the process of the CIG. In order to test the access to different data source technologies we defined accesses to the following: (1) an HL7-CDA document that contained the patient's EHR, which was retrieved from a HIS web service; and (2) a web service that publishes a secondary CDSS, which obtained the *FEV1_prediction*. Figure 25 describes the data requests carried out by the guideline using the mediator. The first enquiry node was related to (1) and requests data from the patient's history, which is located in the patient's EHR. The second enquiry requests data from new tests or physician observations of the patient. In this case study, we also obtained this data from the retrieved EHR; however, a new data loader could be developed to ask the user for this data by means of a specific graphical user interface, or just by querying a database which has just stored the measurements from the corresponding devices. Finally, the third enquiry node (which is related to (2)) requested its data from the CDSS web service by sending the corresponding input variables. These mappings were defined in the CDS-DATOR XML mapping file (see section 8.3.1)

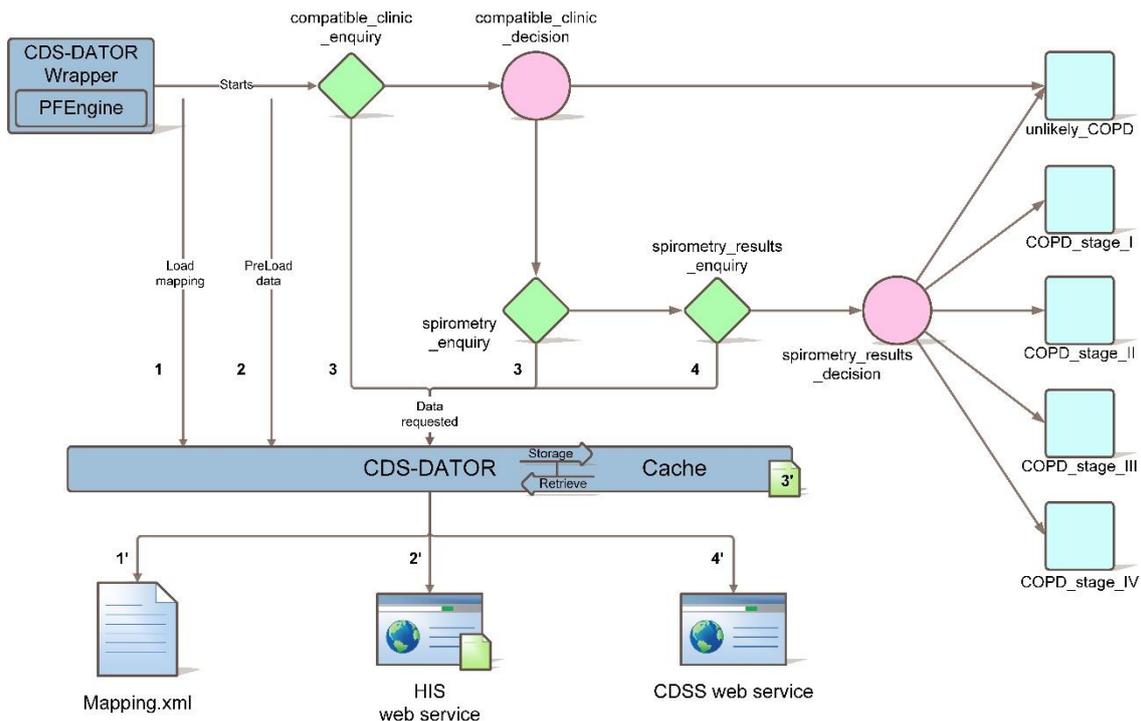


Figure 25: Just after the start of the CIG execution, the mapping file is loaded from the wrapped PFEngine (1). This file (1') tells the CDS-DATOR what data is needed during the guideline execution and where to find it. According to the mapping file, a Preload of the patient's data is made (2) sending a petition to a web service (2') and storing the results in the CDS-DATOR cache. The enquiry nodes ask the mediator for data. The petitions marked as (3) are made to the patient's file stored in cache (3') and the petition (4) is made to a CDSS web service (4'). Finally, once the necessary data is obtained, the guideline makes a decision and suggests a diagnosis.

In this proof of concept, we decided to use an initial PreLoad to store the patient's EHR in the CDS-DATOR cache (Figure 25 (2)). This permits further queries over it, eliminating having to consecutively access the original external data source. The health record retrieved from the web (Figure 25 (2')) was an HL7-CDA document based on the template defined by Sáez et al [43]. Code 3 shows an extract from the HL7-CDA document containing a variable of the patient.

```

<entry>
  <observation classCode="OBS">
    <code xsi:type="CE" displayName="chronic_cough" codeSystemName="Codification
System SNOMED-CT" codeSystem="2.16.840.1.113883.6.96" code="68154008"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    <statusCode code="completed"/>
    <effectiveTime value="198901010000"/>
    <value xsi:type="BL" value="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"/>
  </observation>
</entry>

```

Code 3: Example of a data XML file. The value in this example tells the CDSS that the patient has a chronic cough.

In order to facilitate understanding of the behavior of the proof of concept, we describe a subset of the mapping file. Code 4 shows the lines of the XML mapping instance that were used to retrieve the patient's EHR at the beginning of the execution of the engine (Figure 25 (1)). A SOAP web service was called. A data identification (*patientEHR*) is given to the data retrieved so that the resultant document could be stored in the system's cache to be accessible during the guideline execution. When we wanted to access the data of the document, we had to use this

data ID from a *PreLoad*. Also, the *ContextValue* parameter provided a dynamic way of specifying the patient's ID enabling the reuse of this mapping with other patients in the HIS.

```
<preload id="patientEHR">
  <dataLoader xsi:type="web:WebServiceLoader">
    <web:soapService>
      <web:wSDLURL
xsi:type="URL">http://localhost:9901/PatientHistory?wsdl</>
      <web:operationName xsi:type="SimpleValue">getRecord</>
      <web:parameters xsi:type="ContextValue">patientID</>
    </web:soapService>
  </dataLoader>
</preload>
```

Code 4: This code is used to retrieve the patient's EHR at the beginning of the guideline execution. The 'ContextValue' parameter looks for the patient id in the CIG working memory. The mediator uses SOAP messages to send the petition to the web service and to retrieve the data of the patient.

The following lines in Code 5 show how a data item used in the guideline was retrieved (Figure 25 (3)). In this particular case, the data item was retrieved from the document that was previously stored in cache (Figure 25 (3')). Therefore, the type of the source was a *PreLoadValue* whose identification is the ID that we previously gave to the document (in this case, *patientEHR*). After that, it was only necessary to define the Xpath route where the data item was located, as explained in section 4.4.2.2. In this case, we want to know if the patient had a chronic cough using the corresponding SNOMED-CT code (as indicated in Code 3, this value is true). When the value was retrieved, it was passed to the node *compatible_clinic_enquiry* and the next required data item was checked.

```
<stage id="compatible_clinic_enquiry">
  <data id="chronic_cough">
    <dataLoader xsi:type="xpath:XPathLoader">
      <xpath:source xsi:type="PreLoadValue">patientEHR</>
      <xpath:xpath
xsi:type="SimpleValue">//x:observation/x:code[@code='68154008']/../x:value/@value</>
      <xpath:namespaces xsi:type="SimpleValue">x=urn:h17-org:v3</>
    </dataLoader>
  </data>
</stage>
```

Code 5: This code is used to retrieve information about the patient's chronic cough. The xpath route was applied to the XML file stored in cache and the value of the chronic cough variable was retrieved.

The last example, Code 6 shows how a petition was made to the CDSS, which was an instance of the Curiam CDSS classification engine [46] (Figure 25 (4')). In this case, the CDSS input/output was a REST web service. We communicated the parameters needed to do the calculations and retrieved the data requested in the guideline node.

```

<stage id="spirometry_results_enquiry">
  <data id="FEV1_prediction">
    <dataLoader xsi:type="web:WebServiceLoader">
      <web:restService>
        <web:restURL
xsi:type="URL">http://localhost:7080/es.upv.ibime/rest/fev1Pred</>
        <web:datatype xsi:type="SimpleValue">text/plain</>
        <web:parameters xsi:type="ContextValue">FEV1</>
        <web:parameters xsi:type="ContextValue">FVC</>
        <web:parameters xsi:type="ContextValue">Age</>
        <web:parameters xsi:type="ContextValue">Gender</>
      </web:restService>
    </dataLoader>
  </data>
</stage>

```

Code 6: This code shows how the data is retrieved from a CDSS that is using a web service to communicate. A REST petition with the parameters 'FEV1', 'FVC', and 'Age' is made and the result is the data 'FEV1_prediction', which is required from the node 'spirometry_results_enquiry'.

The complete XML mapping instance of the test where all the data mappings are defined is presented in section 8.3.1.

5.1.2 Curiam BT case study

The second case study shows how to provide data access to the Curiam BT CDSS [45, 47] using CDS-DATOR. Curiam BT is a CDSS for brain tumor diagnosis based on the analysis of 1.5 and 3.0T Single Voxel (SV) Proton Magnetic Resonance Spectroscopy (1H MRS) data and it relies on the generic Curiam CDSS [44].

The Curiam CDSS framework provides the tools to build specific CDSS through a generic user interface and logical software components. The Curiam eases the inclusion of new predictive models using a generic classification framework [46]. Additionally, Curiam provides a generic relational database model which permits simultaneously including data for different domains. Hence, the purpose for this case study is to provide Curiam BT access to MRS spectra stored in such database.

5.1.2.1 Integration with Curiam BT

The Curiam framework which supports Curiam BT was designed following a three layer architecture. These layers separate the components dedicated to the data and configuration, application logic and graphical user interface. The Curiam data layer is divided in two main components, the data access component (or data framework) and the configuration component. For this case study we will wrap the Curiam data framework with our mediator. Figure 26 shows how the mediator is adapted to the Curiam data source manager.

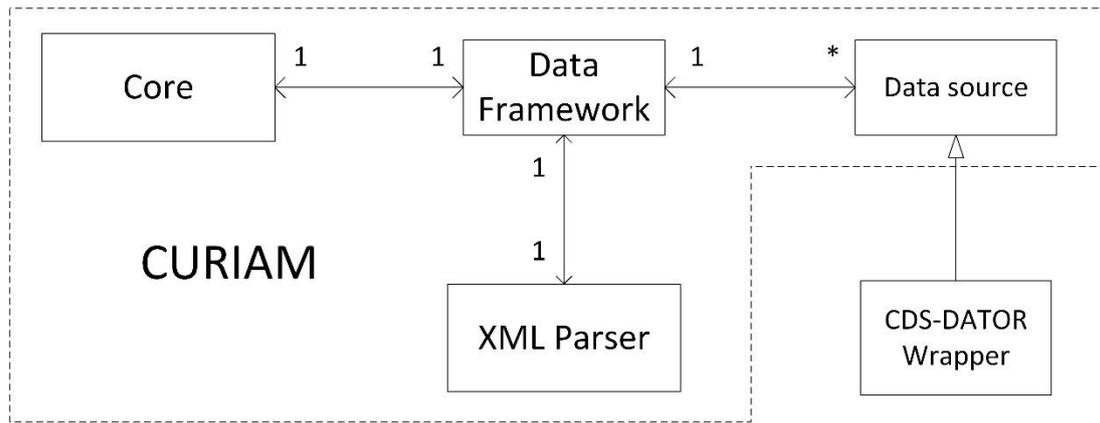


Figure 26: Extension of the data source manager used in Curiam. A wrapper extending the data source provides Curiam with the mediator functionality.

5.1.2.2 Proof of concept

The data expected by Curiam BT is a raw MRS short time echo (STE) signal alone or in combination with a raw long time echo (LTE) signal, according to the predictive model to use as described in [47].

Data signals are stored as 512 points arrays in the generic database that is composed by two tables, the *patients* table and the *variables* table. The *patients* table has two fields: the *patient id* as the principal key, and the *variable* name. The *variables* table is composed by an *id* and a *data* fields. The *id* of the *variables* table is the composition of the *id/variable* fields of the patient table, so each patient has a unique set of variables.

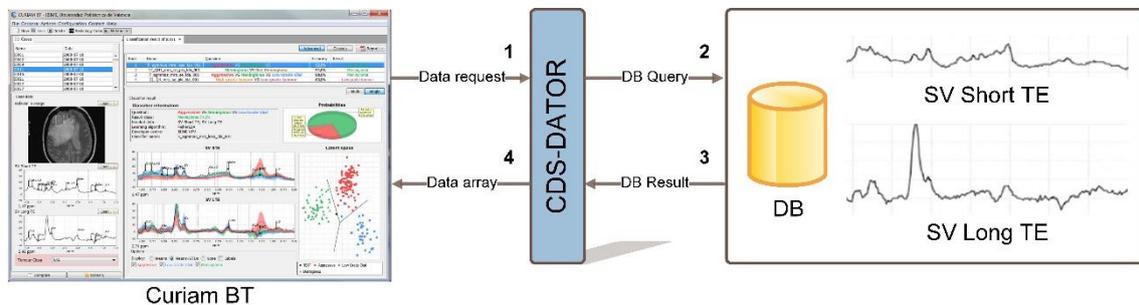


Figure 27: Data request (1) and result (4) of the connection of Curiam BT with the CDS-DATOR.

In Figure 27 we can see the route that follows the data in our case study. When a patient is selected in Curiam BT, their data are requested by the data manager. Two queries that include the patient ID, are passed to the mediator (1). According to the mapping file, the mediator will retrieve the data from the database using the SQL Loader (2, 3). CDS-DATOR returns the data requested to Curiam BT (4) and Curiam BT uses the data retrieved by the mediator to create an XML and access it from the logic layer. Afterwards, the CDSS presents an output based in the data retrieved and the classification models used. The complete XML mapping instance of the case study where all the data mappings are defined is presented in Appendix 8.3.2.

In this case study a database containing the MRS cases is used. Although it is not generalized, some MR scanners manufacturers permit including the MRS in the DICOM medical image

standard¹⁶, which may also be accessed by the mediator. The mediator allows defining web service petitions in SOAP with an unspecified number of parameters, hence a SOAP request message with the search criteria defined as an XML could be created and passed to the DICOM service. The DICOM service that receives the petition generates the results as a soap response in XML, which are received by the mediator.

5.2 NAÏVE BAYES CLASSIFIER

In this case study we explain how to include the Naïve Bayes classifier in the Criterion framework. The purpose of the classifier is to estimate the probabilities of breast cancer relapse after the patient has been treated. Developing a descriptive model of the relations between the independent variables and the relapse or determining the independent variables with more influence on the outcome is considered out of the scope of this development. The models are developed using a machine learning approach, which allows us to use the simple but powerful Naïve Bayes (NB) learning algorithm. NB is considered as a benchmark algorithm [28] and has been successfully applied to many medical domains [26].

5.2.1 Methods

The data used in the models is a dataset (N=84) of independent clinical variables such as patient age, histology of the tumour (*in situ* or invasive), type of breast tumour (ductal or lobular), positive estrogen receptors (yes/no), positive progesterone receptors (yes/no), positive HER2 (yes/no), affected axillary lymph (yes/no), or stage (I, IIA, IIB, IIIA, IIIB, IIIC, or IV); and treatment variables including surgery conducted to the patient (yes/no), radiotherapy (yes/no), chemotherapy (yes/no), or hormonal treatment (yes/no); the dependent variables were the loco-regional relapse (yes/no) and the local relapse (yes/no) of breast cancer after the patient has been treated.

As previously stated, we faced the model classification problem from a machine learning approach. The machine learning methodology is composed by two main steps, the training and the evaluation of the models.

5.2.1.1 Model training

The small set of cases and the missing values difficult the use of more sensitive predictive models such as logistic regression or artificial neural networks. The Naïve Bayes approach presents a robust solution when facing missing and noise data, it is simple to implement and requires little data for training [2], making it a clear election for this classification problem.

The NB algorithm is used as a classification model that assumes independence among the input variables. NB estimates the posterior probability of the output class given the input data based on the Bayes' rule:

$$P(C|\mathbf{x}) = \frac{p(\mathbf{x}|C)P(C)}{p(\mathbf{x})}$$

Equation 2: Bayes rule

¹⁶ <http://medical.nema.org/standard.html>

In Equation 2, C is the class to be classified (i.e. “relapse” or “no relapse”), \mathbf{x} is the input vector, $p(\mathbf{x}|C)$ is the class-conditional probability, $P(C)$ is the prior probability of the class estimated as the observed prevalence, and $p(\mathbf{x})$ is the marginal probability used as a normalization factor. The major assumption of this model is that the class-conditional probability can be factorized into an independent probability for each input variable, as shown in Equation 3:

$$p(\mathbf{x}|C) = \prod_i p(x_i|C)$$

Equation 3: Independent class conditional probability.

This assumption permits us to model each variable conditioned to each class, and to assume that each one follows an appropriate distribution. Thus, the patient’s age is modelled as a continuous Gaussian distribution, the categorical variables are modelled as a multinomial distribution, and the binary variables are modelled as a Bernoulli distribution. The training of the models is focused on adjusting the distributions’ parameters for each variable as well as the prevalence of the classes. Unfortunately there are up to nine times more cases of “no relapse” than cases of “relapse”, that is considered a high imbalance on the classified classes. To overcome this issue we created and tested two different models. The first one uses the original prior probabilities of each class while the second one assumes that each class is equally likely, setting a uniform prior probability for the classes.

5.2.1.2 Model evaluation and results

Due to the small set of cases available (only 84), a 6-fold cross validation was used. The models were trained and then evaluated with an independent test set, and this process was repeated 6 times. Also, the performance of the model was averaged to obtain an evaluation estimate.

The metrics used in the performance evaluation of the models were the accuracy with a 95% confidence interval (ACC), the sensitivity (SEN) and specificity (SPE), the balanced accuracy rate (BAR), and the area under the ROC curve (AUC). This evaluation is based in the work of Brodersen et al. [9].

The results of the classification models are shown in Table 1. Considering the relapse as the positive class, these results show that the uniform prior assumption has higher sensitivity and BAR. The empirical priors show a better accuracy, but the null sensitivity tells us that there is a high imbalance between the classified classes. The models classified using a uniform prior present an improvement in sensitivity and specificity thus, showing a higher balance accurate rate.

Relapse	Prior	ACC(95%CI)	SEN	SPE	BAR	AUC
Loco-regional	Empirical	0.90 (0.82, 0.95)	0.00	1.00	0.50 (0.49, 0.66)	0.68
Local	Empirical	0.94 (0.86, 0.98)	0.00	1.00	0.50 (0.49, 0.75)	0.87
Loco-regional	Uniform	0.76 (0.66, 0.84)	0.62	0.78	0.70 (0.53, 0.82)	0.71
Local	Uniform	0.81 (0.71, 0.89)	1.00	0.80	0.90 (0.63, 0.92)	0.93

Table 1: Results of the Naïve Bayes models for risk of relapse prediction.

Figure 28 also shows that the behaviour of these NB models is significantly better ($\alpha = 0.05$) than random guess in terms of BAR.

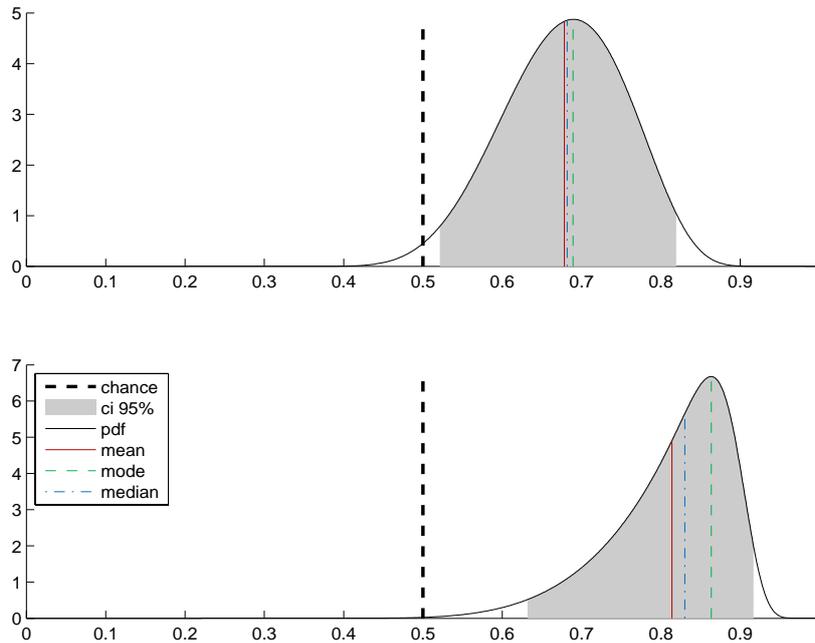


Figure 28: Posterior BAR of the NB model for loco-regional relapse (top) and local relapse (bottom). X-axis is the BAR. Y-axis is the value of the BAR posterior probability density function. The confidence interval at 95% (shaded grey) shows that the BAR is over the limit of guessing by chance (dashed black line)

5.2.2 Inclusion in Criterion

After define, train and evaluate the models we have to add them to the decision software. In Criterion a model can be added through the diagram editor. The user needs to specify the uncertain node where to add the model as seen in Figure 29.

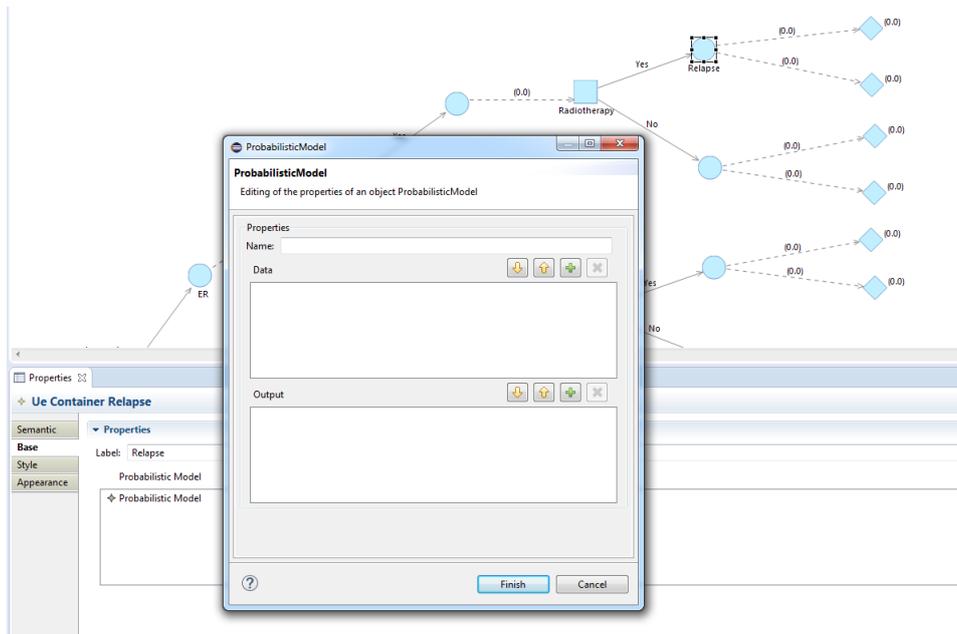


Figure 29: Adding a probabilistic model to the node "Relapse". The model is added through the edition of the node properties.

After selecting the node, the user needs to specify the data required for the models so the mediator knows where to store the data retrieved from the data sources. That is done through the model definition window (see Figure 30). The possible outputs of the model are defined in the same way.

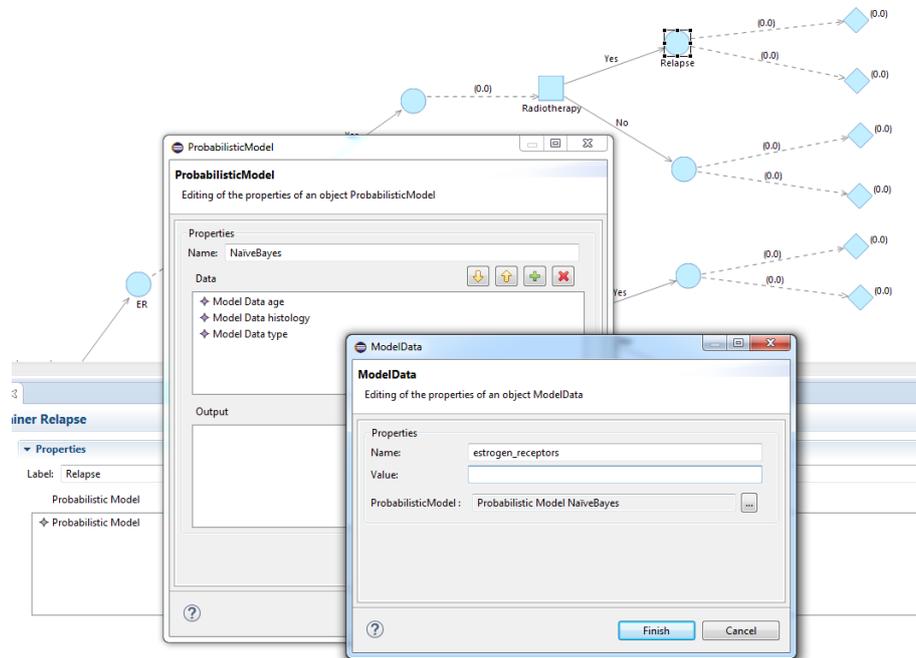


Figure 30: Adding the data needed for the probabilistic model. The model outputs (defined below) are added in the same way.

In Figure 31 can be seen the last step of the model definition which consists on linking the model outputs with the possible outcomes of the uncertain events. If the user doesn't specify that, the system do not know where to map the different outputs of the model.

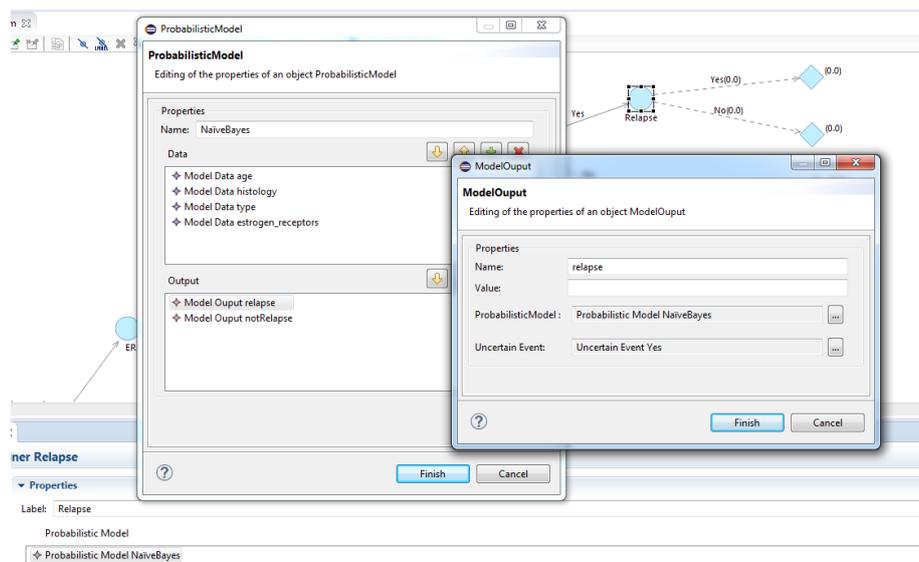


Figure 31: Linking the relapse output of the Naive Bayes model created with the uncertain event of relapse.

Once everything is linked, the probabilistic model is going to be used transparently user, adapting to the patient depending on the data inputs that receives from the mediator.

5.3 CRITERION CASE STUDY: DUCTAL CARCINOMA IN SITU

The DCIS is considered an early form of non-invasive breast cancer (stage 0) and implies that the abnormal cell are still inside the milk ducts of the breast. In the last decades, the development of several mammographic programs to narrow down the treatments of breast cancer increased considerably the diagnostic of DCIS. In United States the cases grown from a 1.87 per 100.000 in 1973 to 32.5 in 2004 [23, 54]. Nowadays, the DCIS treatment considers several different treatments that can vary from conservative surgery with or without radiotherapy, hormonal treatment, sentinel node biopsy or the mastectomy with or without sentinel node biopsy. To create this case study a simplification of the NCCN 2014¹⁷ guideline including only the DCIS was used.

5.3.1 Problem definition

The first steps consist on transforming the clinical practice guideline in a computer interpretable guideline that follows the methodology defined by the Bayesian library (section 4.3). To start, we have to address several issues usually presented in CPGs like ignoring the uncertain states and their possible outcomes, and not defining explicitly the utility of the actions. To overcome these issues we have to follow a methodological procedure to specify clinical guidelines:

1. Identify in the CPG the decision space for each step and its possible alternatives
2. Identify the uncertain events of nature where patients/environment can be
3. Create the decision flow that will define the clinical guideline
4. Define the metric to measure the benefit of the decision process and use it in the utility function for the consequences of each action
5. Estimate the likelihood of the states of nature given the observations
6. Solve the decision problem of each step t selecting the alternative action that maximize the expected utility function

5.3.2 Identify the decision spaces and its actions

Table 2 presents the set of decisions and actions found when treating a DCIS cancer. The data presented on the table is based on a simplification of the NCCN 2014 breast cancer guideline.

Decisions	Actions
Surgery	Lumpectomy, Mastectomy
Tamoxifen prescription	Yes, No
Radiotherapy	Yes, No

Table 2: Set of decisions and its possible actions for a simplifications of the DCIS treatment.

5.3.3 Identify the uncertain events of nature

After defining each possible decision step and its actions, we have to determine the possible set of uncertain events that can occur when an action is taken. Based in the NCCN 2014 guideline we identify two possible uncertain events. After the surgery decision the patient can have positive or negative estrogen receptors, also after the Tamoxifen decision step and before the decision tree nodes, we consider the probabilities of a patient's relapse. A summarized result is presented in Table 3.

Events	Uncertain events
Estrogen receptors	Yes, No
Relapse	Yes, No

¹⁷ www.nccn.org

Table 3: Set of possible uncertain events for teh DCIS treatment.

5.3.4 Create the decision flow

Using the software presented in this work, we define the structure of the decision tree. The complete tree definition can be seen in Appendix 8.4. Figure 32 shows a partial definition of the DCIS tree, the decision of prescript the Tamoxifen adjuvant therapy or not. The probabilities of the uncertain events and the function utilities have not been set yet in this design phase. Dashed lines resent that the decision path created does not have a utility function assigned.

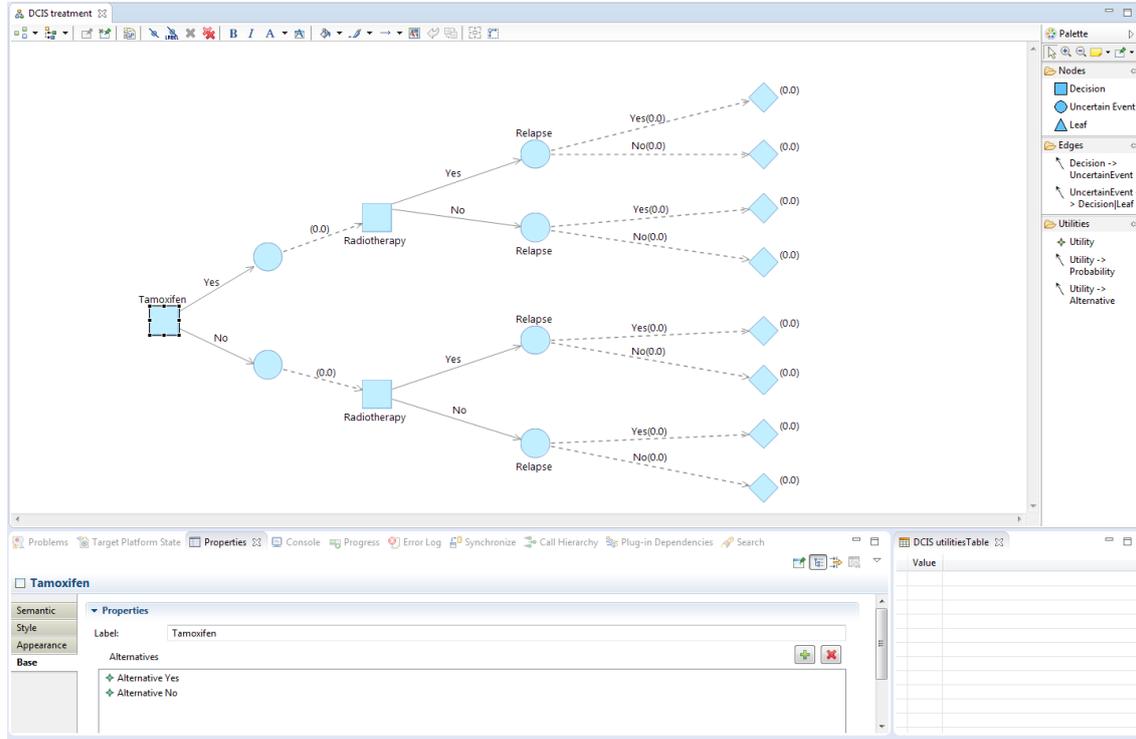


Figure 32: Creation of the decision flow used in the DCIS modeling. The values of the probabilities and the utilities have not been set yet.

5.3.5 Metric to measure benefit and the utility functions

It is of extreme importance using a correct estimator to measure the utility when solving the decision trees because the utility function quantifies the consequences of the decision made and depending on the estimator selected our decision path can change drastically. We have to take into account that we are working in a clinical environment, so only using a cost-based function would not be appropriate to measure the utility of the guideline defined. A common measure used in these environments is the disease-free survival rates or the quality adjusted life years (QALY).

Moreover, to measure the effectiveness of the treatments, the clinicians resort to the clinical evidence. There exist different levels to rank clinical evidence, the most reliable evidence is the evidence gathered at least from one properly designed randomized controlled trial (RCT), while the least reliable one is the evidence gathered from opinions of respected authorities, clinical experience or reports from expert committees.

The evidences used in the case study are based on Kaplan-Meier curves. The Kaplan-Meier estimator is a non-parametric estimator of the survival function. It measures the patient's survival for more than an estimated time t . We selected this estimator because it is very common to find RCTs that use them to measure the treatments proposed in the trial. Thus,

based on the studies of Allred et al. [4], and Donker et al [11] and combining the information extracted from their respective Kaplan-Meier curves and applying the utility theory we manage to extract the utilities as presented in Table 4.

Surgery	ER	Tamoxifen	Radiotherapy	Relapse	Utility
Lumpectomy	ER+	Yes	Yes	Yes	0.2348
Lumpectomy	ER+	Yes	Yes	No	5.765
Lumpectomy	ER+	Yes	No	Yes	0.4455
Lumpectomy	ER+	Yes	No	No	5.582
Lumpectomy	ER+	No	Yes	Yes	0.5066
Lumpectomy	ER+	No	Yes	No	5.4934
Lumpectomy	ER+	No	No	Yes	0.455
Lumpectomy	ER+	No	No	No	5.1319
Lumpectomy	ER-	Yes	Yes	Yes	0.8653
Lumpectomy	ER-	Yes	Yes	No	5.1319
Lumpectomy	ER-	Yes	No	Yes	1.642
Lumpectomy	ER-	Yes	No	No	4.971
Lumpectomy	ER-	No	Yes	Yes	0.6256
Lumpectomy	ER-	No	Yes	No	5.3744
Lumpectomy	ER-	No	No	Yes	1.187
Lumpectomy	ER-	No	No	No	5.2037

Table 4: Utility values for each leaf of the decision tree created. Utilities based on Kaplan-Meier curves presented in randomized control trials.

5.3.6 Estimate the probabilities

In section 4.6 we presented the probabilistic models that are going to be used in this case study and in section 5.2 we trained the models and saw how to use it in combination with the software. In this case, the naïve Bayes models will calculate the probabilities of the patient’s relapse and introduce it in the tree algorithm when calculating the optimal decision. For calculate such probabilities, the models need data. This data is going to be extracted by the mediator from a simulated electronic health record (EHR). This document is in structure the same as an EHR but the data contained in it belongs to simulated patients.

After defining the data needed by the probabilistic model, a mapping file has to be created to tell the mediator what data retrieve and in which node. The complete mapping file defining the source of the data paths for the probabilistic models can be seen in Appendix 8.4.2.

For the probabilities regarding the estrogen receptors it has been used the populations’ prevalence, assigning a 75% for positive ER and a 25% for negative ER.

5.3.7 Solve the decision problem for each step

Figure 33 presents the decision’s tree solution proposed by Criterion for this case study. A small screen presenting the alternative recommended and the utility value calculated for that alternative is showed to the user. Moreover, the path representing the optimum alternative is marked, so the user can decide to continue the tree execution following the recommendations or not.

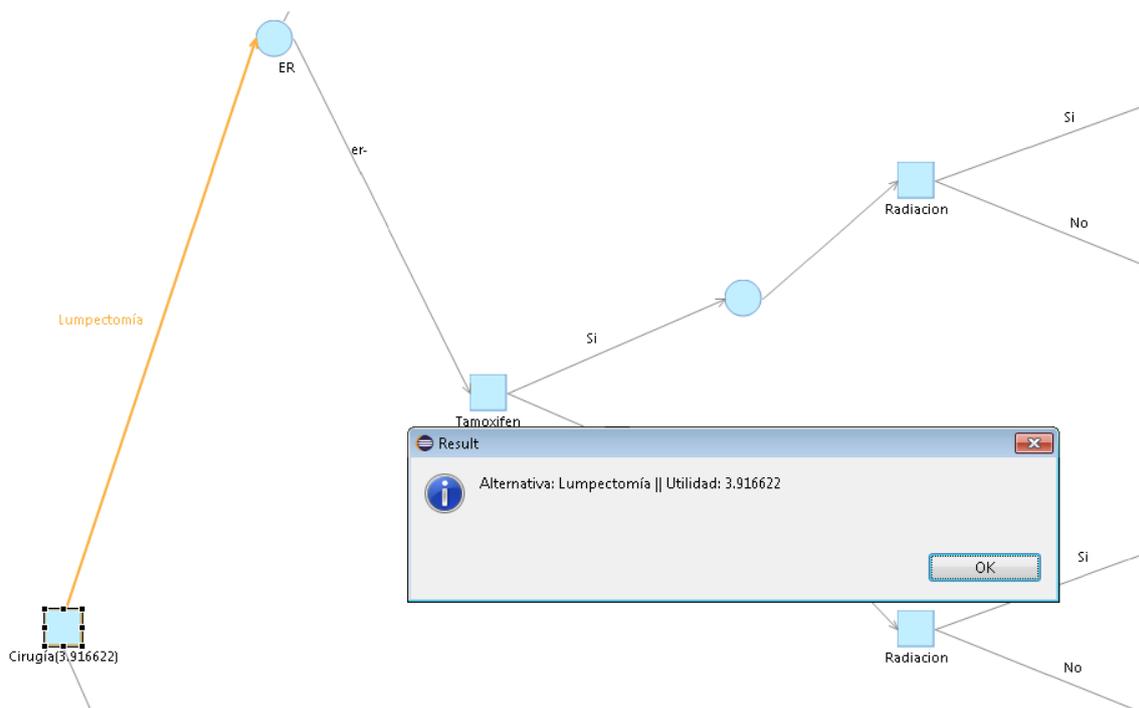


Figure 33: Solution presented to the user when solving the decision tree in Criterion.

The Bayesian theory used in the resolution of the tree algorithm does not force the user to follow the path of maximum utility. The system is always going to calculate the path of maximum utility for the current selected decision in the diagram, so if the user decides to take a different path from the recommended one, the system is going to use the data from previous decisions as facts that will affect the decision process and the utility estimation.

Following this step by step execution of the tree we can also see what the recommended alternatives would be when facing situations that are not presented in the randomized controlled trials (RCT), either because the trial was not conducted or because a non-ethical situation is presented (i.e. performing a mastectomy versus conservative surgery).

Particularly, in the case of a patient with negative estrogen receptors, the Tamoxifen adjuvant therapy is not recommended. But if the physician decides to give the patient the Tamoxifen, the next decision node do not recommends the radiotherapy, as shown in Figure 34. This situation is not treated in any RCT that we know of, and could establish the bases to analyze clinical cases that don't even need to exist.

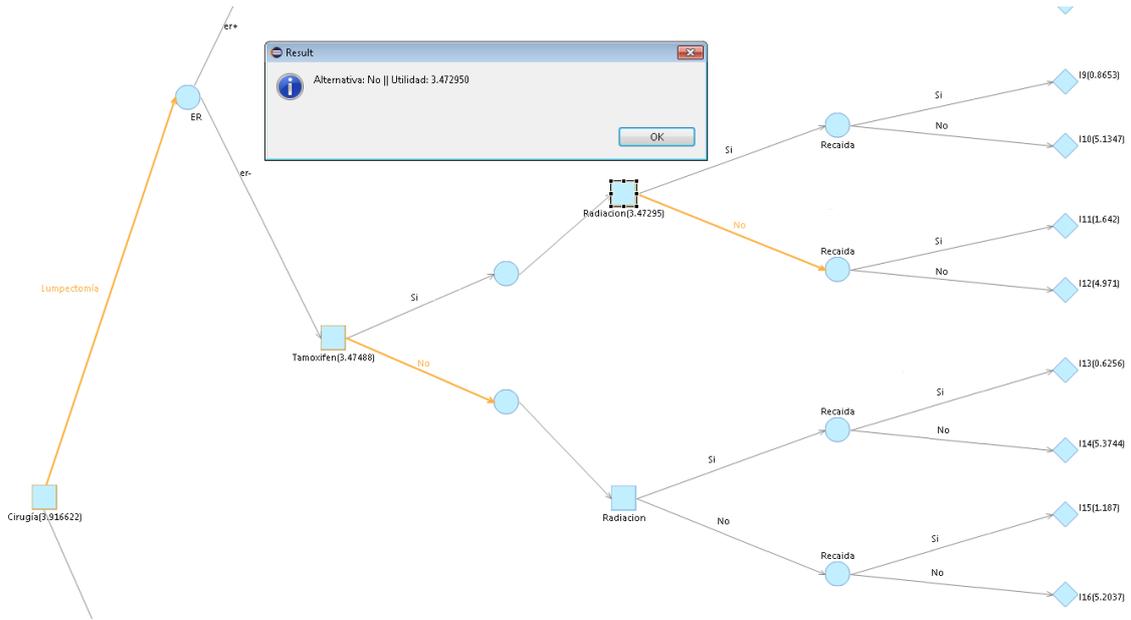


Figure 34: Decision tree suggesting to not give radiotherapy to a patient that has the estrogen receptors negative and has been treated with Tamoxifen. The previous suggested paths are also highlighted. Note that the decision tree advises to not give Tamoxifen to the patient, but the software allows to calculate the utility for the decision anyway.

6 DISCUSSION AND CONCLUSIONS

6.1 CDS-DATOR

In Section 3.2 we analyzed how several systems were connected to their respective data sources. We described how all of these systems accessed the data through one or more combinations of the more common technologies/architectures (SQL, XML and WebServices). In section 4.4.1, we also saw that CDS-DATOR provides a methodology in which data sources can be directly accessed based on technology-specific loaders in order to facilitate the canonical model construction. In this section we are going to discuss how some of these systems could benefit from the integration of CDS-DATOR with their respective data source managers.

We already saw that the Shaphire technology relies in the sources managed by the GLIF language. Although their methodology has been proved to work [37], the fact that it is tied to a specific guideline definition language (GLIF3) raises the same problem as the faced in the PROforma case study: what do we have to do if we want to define a new data source? Should we try to adapt the language to the new sources? In section 5.1.1 it was demonstrated that creating a source mediator independent of the language implementation can be done. This shows that not only the Shaphire technology, but all the systems that use GLIF, like the system presented in the work of Kazemzadeh et al. [25], could be benefit by the mediator being implemented for the GLIF language.

On the other hand, it is widely acknowledged that the utilization of standards is a prerequisite to be able to achieve semantic interoperability in any domain. Furthermore, CDSSs very often require abstract terms that are not supported directly by raw data sources. This involves some type of abstraction processing, such as transforming patient data from a low-level quantitative form to a high-level qualitative description (e.g., abstracting an episode of hyperkalemia from a set of time-stamped observations of the level of potassium in blood), concept abstractions by means of classification hierarchies (e.g., antibiotics instead of amoxicillin) or the definition of abstract terms for a set of basic concepts (e.g., metastatic solid tumor). Correctly managing the abstractions is a complex task that may benefit from being treated alone, so a separation between raw and semantic treatment of data could be of use. Currently, there are several tools that perform different types of abstractions (such as KDOM [39], IDAN [6]) or that provide standardization and basic abstraction (such as LinkEHR [34, 33]).

As previously exposed, data is not managed at a semantic level by the CDS-DATOR architecture. The main focus of CDS-DATOR is to facilitate functional interoperability, i.e., to ease the access to source data regardless of the data model (relational, XML, etc.) and the access method (web services, JDBC, etc.). In order to offer a complete and smooth integration between clinical data sources and CDSSs, we need to complement our tool with other modules that provide semantic interoperability services. These services are necessary when there is a need to use and aggregate data across heterogeneous data sources that may employ different schemas structures, terminologies, or level of detail to represent similar data [53].

CDS-DATOR has been successfully used in a prototype for the determination of patient eligibility based on clinical data stored in a real-life EHR system [34]. The work presented in that article describes a comprehensive approach for dealing with the interoperability of CDSSs and EHR based on EHR standards and archetypes. Specifically, it makes use of LinkEHR to provide a standard-based virtual health record over the EHR whose contents need to be integrated and used in the CDSS. The prototype included a CDSS for patient eligibility determination in the

PROforma language. CDS-DATOR provided a data access layer to allow the smooth access of the Proforma execution engine to the standard-compliant XML documents generated by LinkEHR.

The methodology used integrating LinkEHR with the CDS-DATOR and the PROforma engine can also be used in other systems that implement semantic/temporal mediators, like the IDAN mediator in the MEIDA system. Although IDAN is a complete and proved mediator [6], it may also take advantage of the CDS-DATOR. IDAN communicates with MEIDA through the data access module (DAM). Integrating the CDS-DATOR with the DAM would allow MEIDA simplify the queries for static and immediate data, and would provide IDAN with more sources to pass the request for complex data.

Other benefit for all the systems that would integrate the mediator is the mapping reusability. CDS-DATOR provides a methodology in which data sources can be directly accessed based on technology-specific loaders in order to facilitate the canonical model construction. In addition, CDS-DATOR simplifies the model access, providing all of them a standard way to access it. This means that the systems using the mediator could access other canonical models from other systems thus favoring data exchange. Moreover, the CDSSs that use the same canonical model, the data mediator (i.e., CDS-DATOR) can access the data sources in the same way. A data mapping could be shared and reused from multiple CDSSs and all CDSSs could access all the data sources without having to be restricted to any particular one. In systems of Kazemzadeh et al. and Curiam the patient's data is accessed and afterwards an XML document is generated to create queries over it. It is reasonable to think that if the two systems implement the same data mediator, the data mappings between this two different CDSSs could be shared, reducing the time to codify the mappings and allowing data mapping reusability.

6.2 NAÏVE BAYES MODELS

We can conclude that given the limitations (low number of cases and imbalanced classes), the Naïve Bayes models show a good performance when assuming uniform priors. However, these drawbacks should be considered when analyzing the results and more data should be gathered to reinforce the usefulness of the models.

Finally, although the predictive models can be embedded in a decision support system for improving breast cancer treatment, these NB models can also be used independently to help clinicians to improve the postoperative treatment and patient follow-up. A simple web interface, as part of the HEDECAMA project (reference IPT-2011-1126-900000), was created to allow the physicians test the results and validate the models (see Appendix 8.2).

6.3 CRITERION

The meta-model programming of the system significantly reduces the development time and simplifies the maintenance of the software. On the other hand, the development framework used (i.e. the eclipse modeling tools) presents a steep learning curve and you need deep understanding of the provided libraries to generate things outside the pre-established patterns.

The software is designed as a set of plug-ins of the eclipse environment, providing the developers and the users with all the functionalities that Eclipse has by default. Although forcing the software to be integrated in eclipse limits its accessibility. The next steps on the development of Criterion are going to be focused on providing a web access to the software, so no installation on the user's computer is needed.

Obtaining the utilities for medical evidence is not always an easy task, most of the times data is incomplete or insufficient. Other times, it would be immoral or against the ethics to do a study only to get enough data to confirm a result. These led us to create a decision theory to fill these gaps in the medical knowledge. With the decision tree complete, the probabilities adapted and the utilities set, we were able to compute the optimum decision path and the expected utility for the alternative suggested, proving that the software works and that is a viable alternative to develop and execute Bayesian decision trees.

6.4 OBJECTIVES ACHIEVED

The main contributions of this work can be summarized in the following points:

1. Mappings for common data sources and easy extension to new ones. In addition to designing the mediator to manage any type of data source, we also designed and architecture to facilitate the most usual mapping definitions. The mediator also allows coders to define new specific loaders for new data sources.
2. Data mediation or wrapping with any CDSS with minimal integration effort. Only a wrapping of the CDSS data layer is required to interact with the CDS-DATOR.
3. Separation between the CDSS encoding process and the mapping definition. The CDSS data access is defined in an external mapping that is not integrated in the CDSS. This allows the CDSS knowledge base files (e.g., a CIG file or a classification model) to be treated separately from the data access definitions.
4. Methodology that helps transforming a clinical practice guideline in a computer interpretable guideline, in particular, transforms a CPG into a Bayesian decision tree.
5. Capability of computing the expected utility for each sequence of decisions, obtaining the optimum path, which is the one that maximizes the expected utility or minimizes the expected cost.
6. Integration of predictive models in the decision process in a transparent way to the user, which permits achieving personalized decisions based on the data of the patient.
7. Capability to calculate the tree's solution at any moment of the tree temporal line. So the user can take a decision not suggested by the software and still calculate the next best decision based on the data available.

6.5 FURTHER WORK

Several aspects can be considered as future lines of work. Regarding the mediator, a schema extension to permit type conversion prior to sending the data to the CDSS. Right now, this task is left to the system implementing the mediator, so providing a unit conversion module would improve the functionality of the system and would reduce the processing time in the CDSS that implements it. Training more probabilistic models with more data would be also required to provide the user with more decision-making power. Having different models would allow us to combine them, giving the decision tree even more adaptability.

Also, a simplification of the Criterion interface could be necessary for the system to be used by physicians. The system has been tested by a physician, but always under the supervision of a computer expert. This situation is not always possible and to avoid it we should ease the usage of the program. To finish, the tendency of software development is to present the software as a service instead of a standalone application to be installed in the client, so a web version of Criterion could be also a great improvement to facilitate the usage of the application.

7 BIBLIOGRAPHY

- [1] Health Level 7. Arden syntax v2.9, September 2014.
- [2] A.A. ; Othman Z. Al-Aidaros, K.M. ; Bakar. Naïve bayes variants in classification learning. pages 276 – 281. IEEE, 2010.
- [3] O. Alagoz, H. Hsu, A. J. Schaefer, and M. S. Roberts. Markov decision processes: a tool for sequential decision making under uncertainty. *Med Decis Making*, 30(4):474–483, 2010.
- [4] D. C. Allred, S. J. Anderson, S. Paik, D. L. Wickerham, I. D. Nagtegaal, S. M. Swain, E. P. Mamounas, T. B. Julian, C. E. Geyer, J. P. Costantino, S. R. Land, and N. Wolmark. Adjuvant tamoxifen reduces subsequent breast cancer in women with estrogen receptor-positive ductal carcinoma in situ: a study based on NSABP protocol B-24. *J. Clin. Oncol.*, 30(12):1268–1273, Apr 2012.
- [5] J. M. Bernardo. *Bayesian theory*. Wiley, Chichester, Eng. New York, 1994.
- [6] D. Boaz and Y. Shahar. A framework for distributed mediation of temporal-abstraction queries to clinical databases. *Artif Intell Med*, 34(1):3–24, May 2005.
- [7] A. A. Boxwala, M. Peleg, S. Tu, O. Ogunyemi, Q. T. Zeng, D. Wang, V. L. Patel, R. A. Greenes, and E. H. Shortliffe. GLIF3: a representation format for sharable computer-interpretable clinical practice guidelines. *J Biomed Inform*, 37(3):147–161, Jun 2004.
- [8] Aziz A Boxwala, Mor Peleg, Samson Tu, Omolola Ogunyemi, Qing T Zeng, Dongwen Wang, Vimla L Patel, Robert A Greenes, and Edward H Shortliffe. Glif3: a representation format for sharable computer-interpretable clinical practice guidelines. *Journal of Biomedical Informatics*, 37(3):147 – 161, 2004.
- [9] ETH Zurich Zurich Switzerland ; Cheng Soon Ong ; Stephan K.E. ; Buhmann J.M. Brodersen, K.H. ; Dept. of Comput. Sci. The balanced accuracy and its posterior distribution. pages 3121 – 3124. Pattern Recognition (ICPR), IEEE, August 2010.
- [10] I. Colombet, A. R. Aguirre-Junco, S. Zunino, M. C. Jaulent, L. Leneveut, and G. Chatellier. Electronic implementation of guidelines in the EsPeR system: a knowledge specification method. *Int J Med Inform*, 74(7-8):597–604, Aug 2005.
- [11] M. Donker, S. Litiere, G. Werutsky, J. P. Julien, I. S. Fentiman, R. Agresti, P. Rouanet, C. T. de Lara, H. Bartelink, N. Duez, E. J. Rutgers, and N. Bijker. Breast-conserving treatment with or without radiotherapy in ductal carcinoma In Situ: 15-year recurrence rates and outcome after a recurrence, from the EORTC 10853 randomized phase III trial. *J. Clin. Oncol.*, 31(32):4054–4059, Nov 2013.
- [12] Marilyn J. Field and Kathleen N. Lohr. *Clinical Practice Guidelines: Directions for a New Program*. 1990.
- [13] Marilyn J. Field and Kathleen N. Lohr. *Guidelines for Clinical Practice: From Development to Use*. 1992.
- [14] American Society for Testing and Materials. Standard specification for guideline elements model version 3 (gem iii), September 2014.

- [15] J. Fox, V. Patkar, and R. Thomson. Decision support for health care: the PROforma evidence base. *Inform Prim Care*, 14(1):49–54, 2006.
- [16] J. Fox and R. Thomson. Decision support and disease management: a logic engineering approach. *Trans. Info. Tech. Biomed.*, 2(4):217–228, December 1998.
- [17] David Garlan, Felix Bachmann, James Ivers, Judith Stafford, Len Bass, Paul Clements, and Paulo Merson. *Documenting Software Architectures: Views and Beyond*. Addison-Wesley Professional, 2nd edition, 2010.
- [18] E. German, A. Leibowitz, and Y. Shahar. An architecture for linking medical decision-support applications to clinical databases and its evaluation. *J Biomed Inform*, 42:203–218, Apr 2009.
- [19] J. M. Grimshaw and I. T. Russell. Effect of clinical guidelines on medical practice: a systematic review of rigorous evaluations. *Lancet*, 342(8883):1317–1322, Nov 1993.
- [20] R. Hanka. Guidelines in general practice. Information overload on GPs’ desks must be overcome. *BMJ*, 318(7192):1212, May 1999.
- [21] M. Hauskrecht and H. Fraser. Planning treatment of ischemic heart disease with partially observable Markov decision processes. *Artif Intell Med*, 18(3):221–244, Mar 2000.
- [22] A. Hibble, D. Kanka, D. Pencheon, and F. Pooles. Guidelines in general practice: the new Tower of Babel? *BMJ*, 317(7162):862–863, Sep 1998.
- [23] Krapcho M Neyman N Aminou R Waldron W Altekruze SF Kosary CL Ruhl J Tatalovich Z Cho H Mariotto A Eisner MP Lewis DR Chen HS Feuer EJ Cronin KA Edwards BK (eds) Howlader N, Noone AM. *Seer cancer statistics review, 1975-2008*, 2011.
- [24] P. D. Johnson, S. Tu, N. Booth, B. Sugden, and I. N. Purves. Using scenarios in chronic disease management guidelines for primary care. *Proc AMIA Symp*, pages 389–393, 2000.
- [25] Reza Sherafat Kazemzadeh, Kamran Sartipi, and Priya Jayaratna. A framework for data and mined knowledge interoperability in clinical decision support systems. *IJHISI*, 5(1):37–60, 2010.
- [26] A.A. Bakar K.M. Al-Aidaros and Z. Othman. Medical data classification with naive bayes approach. *Information Technology Journal*, 11:1166–1174, 2012.
- [27] J. Komulainen, I. Kunnamo, P. Nyberg, M. Kaila, T. Mantyranta, and M. Korhonen. Developing an evidence based medicine decision support system integrated with eprs utilizing standard data elements. Riva del Garda, Italy, 2006.
- [28] I. Kononenko. Machine learning for medical diagnosis: history, state of the art and perspective. *Artif Intell Med*, 23(1):89–109, Aug 2001.
- [29] G. Kosimbei, K. Hanson, and M. English. Do clinical guidelines reduce clinician dependent costs? *Health Res Policy Syst*, 9:24, 2011.
- [30] G.B. Laleci and A. Dogac. A semantically enriched clinical guideline model enabling deployment in heterogeneous healthcare environments. *Information Technology in Biomedicine, IEEE Transactions on*, 13(2):263–273, 2009.

- [31] J. Lopez, D. Palacios-Alonso, S. Tortajada, A. Moreno, E. Casitas, J. M. García-Gómez, R. González Otal, A. Pérez-González, A. Martínez, C. L. Parra Calderón, E. Rivin, S. Leal, and M. J. Ortiz Gordillo. Computerized decision support system and naïve bayes models for predicting the risk of relapse in breast cancer. ASTRO, September 2014.
- [32] Esther Lozano, Mar Marcos, Begoña Martínez-Salvador, Albert Alonso, and Josep Alonso. Experiences in the development of electronic care plans for the management of comorbidities. In David Riaño, Annette ten Teije, Silvia Miksch, and Mor Peleg, editors, *Knowledge Representation for Health-Care. Data, Processes and Guidelines*, volume 5943 of *Lecture Notes in Computer Science*, pages 113–123. Springer Berlin / Heidelberg, 2010.
- [33] José Alberto Maldonado, Catalina Martínez Costa, David Moner, Marcos Menárguez-Tortosa, Diego Boscá, José Antonio Miñarro Giménez, Jesualdo Tomás Fernández-Breis, and Montserrat Robles. Using the researchehr platform to facilitate the practical application of the ehr standards. *Journal of Biomedical Informatics*, 45(4):746 – 762, 2012.
- [34] Mar Marcos, José A. Maldonado, Begoña Martínez-Salvador, Diego Boscá, and Montserrat Robles. Interoperability of clinical decision-support systems and electronic health records using archetypes: A case study in clinical trial eligibility. *Journal of Biomedical Informatics*, 46(4):676-89, 2013.
- [35] Silvia Miksch, Yuval Shahar, and Peter Johnson. Asbru: a task-specific, intention-based, and time-oriented language for representing skeletal plans. In *UK, OPEN UNIVERSITY*, pages 9–1, 1997.
- [36] A. A. Montgomery, T. J. Peters, and T. Fahey. Reasons physicians do not follow clinical practice guidelines. *JAMA*, 283(13):1685; author reply 1686, Apr 2000.
- [37] O. Nee, A. Hein, T. Gorath, N. Hulsmann, G. B. Laleci, M. Yuksel, M. Olduz, I. Tasyurt, U. Orhan, A. Dogac, A. Fruntelata, S. Ghiorghe, and R. Ludwig. SAPHIRE: intelligent healthcare monitoring based on semantic interoperability platform: pilot applications. *Communications, IET*, 2(2):192–201, February 2008.
- [38] Sociedad Española de Neumología y Cirugía Torácica. *Normativa para la espirometría forzada*. Ediciones Doyma, SA, Barcelona.
- [39] M. Peleg, S. Keren, and Y. Denekamp. Mapping computerized clinical guidelines to electronic medical records: knowledge-data ontological mapper (KDOM). *J Biomed Inform*, 41:180–201, Feb 2008.
- [40] A. Pérez-González. Arctic conference on dual-model based clinical decision support and knowledge management. UNN Universitetssykehuset Nord-Norge, May 2014.
- [41] Mary Tork Roth and Peter Schwarz. A wrapper architecture for legacy data sources. Technical report, IBM Almaden Research, 1997.
- [42] D. Palacios A. Pérez-González J.M. García-Gómez R. González Otal C.L. Parra Calderon A. Martínez Garcia A. Moreno Conde M.J. Ortiz Gordillo S. Tortajada, J.L. Lopez Guerra. http://www.postersessiononline.com/173580348_eu/congresos/estro33/aula/-ep_712_estro33.pdf, April 2014.
- [43] Carlos Sáez, Adrián Bresó, Javier Vicente, Montserrat Robles, and Juan Miguel García-Gómez. An hl7-cda wrapper for facilitating semantic interoperability to rule-based clinical

decision support systems. *Computer Methods and Programs in Biomedicine*, 109(3):239–249, 2013.

[44] Carlos Sáez, Juan Miguel García-Gómez, Javier Vicente, Salvador Tortajada, Miguel Esparza, Alfredo T. Navarro, Elíes Fuster-García, Montserrat Robles, Luis Martí-Bonmatí, and Carles Arús. A generic Decision Support System featuring an assembled view of predictive models for Magnetic Resonance and clinical data. *Magnetic Resonance Materials in Physics, Biology and Medicine*, 21:483, Sep 2008.

[45] Carlos Sáez, Juan Miguel García-Gómez, Javier Vicente, Salvador Tortajada, Elíes Fuster-García, Miguel Esparza, Alfredo T. Navarro, and Montserrat Robles. Curiam BT 1.0, decision support system for brain tumour diagnosis. *Magnetic Resonance Materials in Physics, Biology and Medicine*, 22:538, Oct 2009.

[46] Carlos Sáez, Juan Miguel García-Gómez, Javier Vicente, Salvador Tortajada, Jan Luts, David Dupplaw, Sabine Van Huffel, and Montserrat Robles. A generic and extensible automatic classification framework applied to brain tumour diagnosis in healthagents. *The Knowledge Engineering Review*, 2011. Vol. 26:3, 283:301.

[47] Carlos Sáez, Luis Martí-Bonmatí, Ángel Alberich-Bayarri, Montserrat Robles, and Juan M. García-Gómez. Randomized pilot study and qualitative evaluation of a clinical decision support system for brain tumour diagnosis based on SV 1H MRS: evaluation as an additional information procedure for novice radiologists. *Computers in Biology and Medicine*, 45:26–33, February 2014.

[48] G. D. Sanders, R. F. Nease, and D. K. Owens. Publishing web-based guidelines using interactive decision models. *J Eval Clin Pract*, 7(2):175–189, May 2001.

[49] B. Seroussi and J. Bouaud. Using OncoDoc as a computer-based eligibility screening system to improve accrual onto breast cancer clinical trials. *Artif Intell Med*, 29(1-2):153–167, 2003.

[50] Y. Shahar, S. Miksch, and P. Johnson. The Asgaard project: a task-specific framework for the application and critiquing of time-oriented clinical guidelines. *Artif Intell Med*, 14(1-2):29–51, 1998.

[51] R. Shaker, P. Mork, M. Barclay, and P. Tarczy-Hornoch. A rule driven bi-directional translation system for remapping queries and result sets between a mediated schema and heterogeneous data sources. *Proc AMIA Symp*, pages 692–696, 2002.

[52] S. Skonetzki, H. J. Gausepohl, M. van der Haak, S. Knaebel, O. Linderkamp, and T. Wetter. HELEN, a modular framework for representing and implementing clinical practice guidelines. *Methods Inf Med*, 43(4):413–426, 2004.

[53] Walter Sujansky. Heterogeneous database integration in biomedicine. *Journal of Biomedical Informatics*, 34(4):285 – 298, 2001.

[54] W. E. Sumner, L. G. Koniaris, S. E. Snell, S. Spector, J. Powell, E. Avisar, F. Moffat, A. S. Livingstone, and D. Franceschi. Results of 23,810 cases of ductal carcinoma-in-situ. *Ann. Surg. Oncol.*, 14(5):1638–1643, May 2007.

- [55] P. Terenziani, S. Montani, A. Bottrighi, M. Torchio, G. Molino, and G. Correndo. The GLARE approach to clinical guidelines: main features. *Stud Health Technol Inform*, 101:162–166, 2004.
- [56] S. W. Tu, J. R. Campbell, J. Glasgow, M. A. Nyman, R. McClure, J. McClay, C. Parker, K. M. Hrabak, D. Berg, T. Weida, J. G. Mansfield, M. A. Musen, and R. M. Abarbanel. The SAGE Guideline Model: achievements and overview. *J Am Med Inform Assoc*, 14(5):589–598, 2007.
- [57] S. W. Tu and M. A. Musen. Modeling data and knowledge in the EON guideline architecture. *Stud Health Technol Inform*, 84(Pt 1):280–284, 2001.
- [58] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, March 1992.

8 APPENDIX

8.1 INPUT/OUTPUT SCHEMAS OF THE BAYESIAN DECISION LIBRARY

8.1.1 Input schema

Following is presented the input schema created for the Bayesian decision library. This schema establishes the structure for the XML instances that can be defined and accepted as correct inputs by the library.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!--
  == Created by Alfonso Pérez <alpegon3@upvnet.upv.es>
  == and Salvador Tortajada <vesaltor@upvnet.upv.es>
  == Grupo de Informática Biomédica,
  == Universidad Politécnica de Valencia, Spain
  == 02/2013
-->
<xsd:schema targetNamespace="urn:decisionTree.bmg.ibime.upv.es"
  xmlns="urn:decisionTree.bmg.ibime.upv.es"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  attributeFormDefault="unqualified" elementFormDefault="qualified">
  <!-- ===== -->
  <!-- ===== Element Declarations -->
  <!-- ===== -->
  <xsd:element name="DecisionTree">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="decisions">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element maxOccurs="unbounded" name="decision"
type="Decision"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="alternatives">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element maxOccurs="unbounded" name="alternative"
type="AlternativeElem"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="uncertainEvents">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element maxOccurs="unbounded" name="uncertainEvent"
type="UncertainEventElem"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="probabilities">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element maxOccurs="unbounded" name="probability"
type="ProbabilityElem"
            />
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="utilities">
          <xsd:complexType>
```

```

                <xsd:sequence>
                    <xsd:element maxOccurs="unbounded" name="utilityFunction"
type="UtilityFunctionElem"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:sequence>
            <xsd:attribute name="rootDecisionId" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>
    <!-- ===== -->
    <!-- ===== Complex Type Definitions -->
    <!-- ===== -->
    <!-- ~~~~~ -->
    <!-- Node -->
    <!-- ~~~~~ -->
    <xsd:complexType abstract="true" name="Node">
        <xsd:attribute name="id" type="xsd:string" use="required"/>
        <xsd:attribute name="name" type="xsd:string" use="required"/>
    </xsd:complexType>
    <!-- ~~~~~ -->
    <!-- Decision -->
    <!-- ~~~~~ -->
    <xsd:complexType abstract="true" name="Decision">
        <xsd:complexContent>
            <xsd:extension base="Node"/>
        </xsd:complexContent>
    </xsd:complexType>
    <!-- ~~~~~ -->
    <!-- DecisionLeaf -->
    <!-- ~~~~~ -->
    <xsd:complexType name="DecisionLeaf">
        <xsd:complexContent>
            <xsd:extension base="Decision"/>
        </xsd:complexContent>
    </xsd:complexType>
    <!-- ~~~~~ -->
    <!-- DecisionNode -->
    <!-- ~~~~~ -->
    <xsd:complexType name="DecisionElem">
        <xsd:complexContent>
            <xsd:extension base="Decision">
                <xsd:sequence>
                    <xsd:element name="alternativeList">
                        <xsd:complexType>
                            <xsd:sequence>
                                <xsd:element maxOccurs="unbounded"
name="alternativeId" type="xsd:string"/>
                            </xsd:sequence>
                        </xsd:complexType>
                    </xsd:element>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <!-- ~~~~~ -->
    <!-- Alternative -->
    <!-- ~~~~~ -->
    <xsd:complexType name="AlternativeElem">
        <xsd:complexContent>
            <xsd:extension base="Node">
                <xsd:sequence>
                    <xsd:element name="uncertainEventsList">
                        <xsd:complexType>
                            <xsd:sequence>

```

```

                                <xsd:element maxOccurs="unbounded"
name="uncertainEventId" type="xsd:string"/>
                                </xsd:sequence>
                                </xsd:complexType>
                                </xsd:element>
                                </xsd:sequence>
                                <xsd:attribute name="utilityFunctionID" type="xsd:string"
use="required"/>
                                </xsd:extension>
                                </xsd:complexContent>
                                </xsd:complexType>
                                <!-- ~~~~~ -->
                                <!-- UncertainEvent -->
                                <!-- ~~~~~ -->
                                <xsd:complexType name="UncertainEventElem">
                                <xsd:complexContent>
                                <xsd:extension base="Node">
                                <xsd:attribute name="utilityVarID" type="xsd:string" use="required"/>
                                <xsd:attribute name="probabilityID" type="xsd:string"
use="required"/>
                                <xsd:attribute name="nextDecisionID" type="xsd:string"
use="required"/>
                                </xsd:extension>
                                </xsd:complexContent>
                                </xsd:complexType>
                                <!-- ~~~~~ -->
                                <!-- UtilityFunction -->
                                <!-- ~~~~~ -->
                                <xsd:complexType name="UtilityFunctionElem">
                                <xsd:sequence>
                                <xsd:element name="utilityEntry" minOccurs="0" maxOccurs="unbounded">
                                <xsd:complexType>
                                <xsd:sequence>
                                <xsd:element name="key" minOccurs="0" maxOccurs="unbounded"
type="xsd:string"/>
                                <xsd:element name="value" minOccurs="0" type="xsd:double"/>
                                </xsd:sequence>
                                </xsd:complexType>
                                </xsd:element>
                                </xsd:sequence>
                                <xsd:attribute name="utilityVarID" type="xsd:string" use="required"/>
                                <xsd:attribute name="utilityFunctionID" type="xsd:string" use="required"/>
                                </xsd:complexType>
                                <!-- ~~~~~ -->
                                <!-- Probability -->
                                <!-- ~~~~~ -->
                                <xsd:complexType abstract="true" name="ProbabilityElem"/>
                                <!-- ~~~~~ -->
                                <!-- DiscreteProbability -->
                                <!-- ~~~~~ -->
                                <xsd:complexType name="DiscreteProbabilityElem">
                                <xsd:complexContent>
                                <xsd:extension base="ProbabilityElem">
                                <xsd:sequence>
                                <xsd:element name="probabilityEntry" minOccurs="0"
maxOccurs="unbounded">
                                <xsd:complexType>
                                <xsd:sequence>
                                <xsd:element name="key" minOccurs="0"
maxOccurs="unbounded"
type="xsd:string"/>
                                <xsd:element name="value" minOccurs="0">
                                <xsd:complexType mixed="true">
                                <xsd:sequence>
                                <xsd:element name="data" minOccurs="0"
maxOccurs="unbounded">

```

```

                                <xsd:complexType>
                                    <xsd:attribute name="value"
type="xsd:anySimpleType" use="required"/>
                                </xsd:complexType>
                            </xsd:element>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:sequence>
<xsd:attribute name="probabilityID" type="xsd:string"
use="required"/>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:schema>

```

8.1.2 Outputschema

The following schema structures the output of the decision library. By using this schema to define the output structure other systems know what to expect as result and the library can be easily adapted.

```

<?xml version="1.0" encoding="UTF-8" ?>
<!--
    == Created by Alfonso Pérez <alpegon3@upvnet.upv.es>
    == and Salvador Tortajada <vesaltor@upvnet.upv.es>
    == Grupo de Informática Biomédica,
    == Universidad Politécnica de Valencia, Spain
    == 02/2013
-->
<xs:schema
    targetNamespace="urn:hedecama.ibime.upv.es"
    xmlns="urn:hedecama.ibime.upv.es"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    attributeFormDefault="unqualified"
    elementFormDefault="qualified">
    <!-- ===== -->
    <!-- ===== Element Declarations -->
    <!-- ===== -->
    <xs:element name="BayesianDecisionTreeOutput">
        <xs:complexType>
            <xs:sequence>
                <xs:element minOccurs="1" maxOccurs="unbounded"
name="decision" type="Decision"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <!-- ===== -->
    <!-- ===== Complex Type Definitions -->
    <!-- ===== -->
    <!-- ~~~~~~ -->
    <!-- Decision -->
    <!-- ~~~~~~ -->
    <xs:complexType name="Decision" mixed="true">
        <xs:sequence>
            <xs:element minOccurs="0" maxOccurs="unbounded"
name="decision" type="Decision"/>

```

```

</xs:sequence>
  <xs:attribute name="utility" type="xs:float" use="required"/>
</xs:complexType>
</xs:schema>

```

8.2 NAÏVE BAYES WEB INTERFACE

The results of the Naïve Bayes probabilistic models can be checked in the web:

<http://158.42.167.95:8080/StaticNaiveBayesUI>

This web presents a graphical user interface where the user can select between several cases and for each of them select different treatments. The pie chart presents the output of the probabilistic model for the treatment proposed in the bottom table of the left. The bar charts show the probabilities of the patient if other treatment is applied to the patient. Furthermore, the tabs of the top allow the user to change the probabilistic model in use. Three models are available, loco-regional relapse, remote relapse and a mix of the two previous.

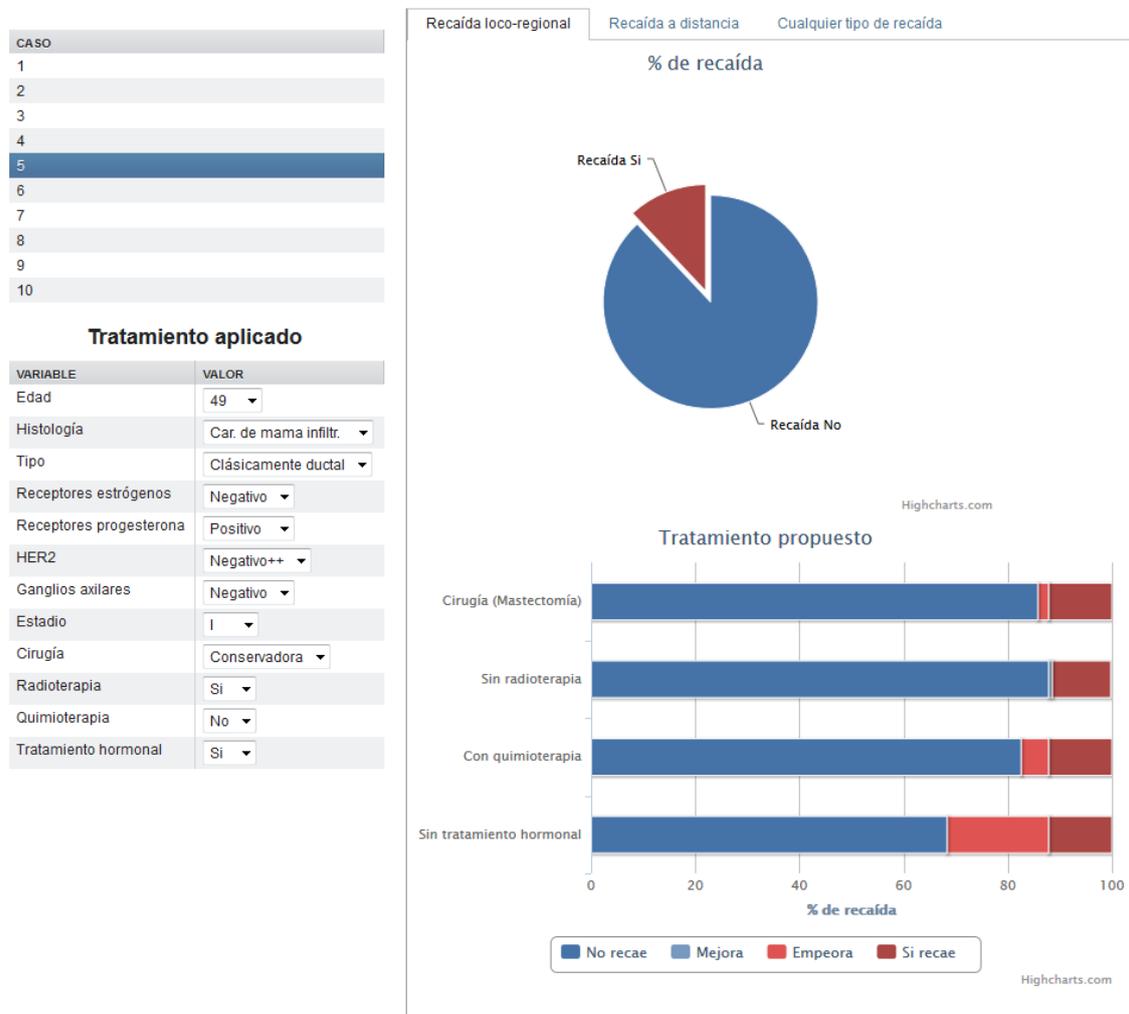


Figure 35: Web created to present the results from the Naïve Bayes classifier models

8.3 DATA MEDIATOR

8.3.1 PROforma case study

```
<?xml version="1.0" encoding="UTF-8"?>
<CDSDC xmlns="urn:cdsdc.bmg.ibime.upv.es"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:web="urn:webserviceloader.cdsdc.bmg.ibime.upv.es"
  xmlns:xpath="urn:xpathloader.cdsdc.bmg.ibime.upv.es"
  xsi:schemaLocation="urn:main.cdsdc.bmg.ibime.upv.es mainSchema.xsd" cdsName="CHF-
  COPD_management_1.pf">

  <preload id="patientEHR">
    <dataLoader xsi:type="web:WebServiceLoader">
      <web:soapService>
        <web:wSDLURL
xsi:type="URL">http://localhost:9901/PatientHistory?wsdl</>
        <web:operationName xsi:type="SimpleValue">getRecord</>
        <web:parameters xsi:type="ContextValue">patientID</>
      </web:soapService>
    </dataLoader>
  </preload>

  <stage id="compatible_clinic_enquiry">
    <data id="Age">
      <dataLoader xsi:type="xpath:XPathLoader">
        <xpath:source xsi:type="PreLoadValue">patientEHR</>
        <xpath:xpath
xsi:type="SimpleValue">../../x:observation/x:code[@code='1001']/../x:value/@value</>
        <xpath:namespaces xsi:type="SimpleValue">x=urn:h17-org:v3</>
      </dataLoader>
    </data>

    <data id="chronic_cough">
      <dataLoader xsi:type="xpath:XPathLoader">
        <xpath:source xsi:type="PreLoadValue">patientEHR</>
        <xpath:xpath
xsi:type="SimpleValue">../../x:observation/x:code[@code='1002']/../x:value/@value</>
        <xpath:namespaces xsi:type="SimpleValue">x=urn:h17-org:v3</>
      </dataLoader>
    </data>

    <data id="chronic_respiratory_failure">
      <dataLoader xsi:type="xpath:XPathLoader">
        <xpath:source xsi:type="PreLoadValue">patientEHR</>
        <xpath:xpath
xsi:type="SimpleValue">../../x:observation/x:code[@code='1003']/../x:value/@value</>
        <xpath:namespaces xsi:type="SimpleValue">x=urn:h17-org:v3</>
      </dataLoader>
    </data>

    <data id="chronic_sputum_production">
      <dataLoader xsi:type="xpath:XPathLoader">
        <xpath:source xsi:type="PreLoadValue">patientEHR</>
        <xpath:xpath
xsi:type="SimpleValue">../../x:observation/x:code[@code='1004']/../x:value/@value</>
        <xpath:namespaces xsi:type="SimpleValue">x=urn:h17-org:v3</>
      </dataLoader>
    </data>

    <data id="risk_factors">
      <dataLoader xsi:type="xpath:XPathLoader">
        <xpath:source xsi:type="PreLoadValue">patientEHR</>
        <xpath:xpath
xsi:type="SimpleValue">../../x:observation/x:code[@code='1005']/../x:value/@value</>
        <xpath:namespaces xsi:type="SimpleValue">x=urn:h17-org:v3</>
      </dataLoader>
    </data>
  </stage>
</CDSDC>
```

```

        </dataLoader>
    </data>

    <data id="dyspnea">
        <dataLoader xsi:type="xpath:XPathLoader">
            <xpath:source xsi:type="PreLoadValue">patientEHR</>
            <xpath:xpath
xsi:type="SimpleValue">//x:observation/x:code[@code='1006']/../x:value/@value</>
            <xpath:namespaces xsi:type="SimpleValue">x=urn:h17-org:v3</>
        </dataLoader>
    </data>
</stage>

<stage id="spirometry_DSS_enquiry">
    <data id="Weight">
        <dataLoader xsi:type="xpath:XPathLoader">
            <xpath:source xsi:type="PreLoadValue">patientEHR</>
            <xpath:xpath
xsi:type="SimpleValue">//x:observation/x:code[@code='1007']/../x:value/@value</>
            <xpath:namespaces xsi:type="SimpleValue">x=urn:h17-org:v3</>
        </dataLoader>
    </data>

    <data id="FVC">
        <dataLoader xsi:type="xpath:XPathLoader">
            <xpath:source xsi:type="PreLoadValue">patientEHR</>
            <xpath:xpath
xsi:type="SimpleValue">//x:observation/x:code[@code='1008']/../x:value/@value</>
            <xpath:namespaces xsi:type="SimpleValue">x=urn:h17-org:v3</>
        </dataLoader>
    </data>

    <data id="FEV1">
        <dataLoader xsi:type="xpath:XPathLoader">
            <xpath:source xsi:type="PreLoadValue">patientEHR</>
            <xpath:xpath
xsi:type="SimpleValue">//x:observation/x:code[@code='1009']/../x:value/@value</>
            <xpath:namespaces xsi:type="SimpleValue">x=urn:h17-org:v3</>
        </dataLoader>
    </data>

    <data id="Gender">
        <dataLoader xsi:type="xpath:XPathLoader">
            <xpath:source xsi:type="PreLoadValue">patientEHR</>
            <xpath:xpath
xsi:type="SimpleValue">//x:observation/x:code[@code='1010']/../x:value/@value</>
            <xpath:namespaces xsi:type="SimpleValue">x=urn:h17-org:v3</>
        </dataLoader>
    </data>
</stage>

<stage id="spirometry_results_enquiry">
    <data id="FEV1_FVC_postbronchodilator">
        <dataLoader xsi:type="web:WebServiceLoader">
            <web:restService>
                <web:restURL xsi:type="URL">http://localhost:7080/fev1Fvc</>
                <web:datatype xsi:type="SimpleValue">text/plain</>
                <web:parameters xsi:type="ContextValue">FEV1</>
                <web:parameters xsi:type="ContextValue">FVC</>
            </web:restService>
        </dataLoader>
    </data>

    <data id="FEV1_prediction">
        <dataLoader xsi:type="web:WebServiceLoader">
            <web:restService>

```

```

        <web:restURL xsi:type="URL">http://localhost:7080/fev1Pred</>
        <web:datatype xsi:type="SimpleValue">text/plain</>
        <web:parameters xsi:type="ContextValue">FEV1</>
        <web:parameters xsi:type="ContextValue">FVC</>
        <web:parameters xsi:type="ContextValue">Age</>
        <web:parameters xsi:type="ContextValue">Gender</>
    </web:restService>
</dataLoader>
</data>
</stage>
</CDSDC>

```

Code 7: XML instance used in the PROforma proof of concept (see section 5.1.1.3). The preload at the beginning loads the patient's data from a web service using the patientID from the CDSS. Afterwards, that the data is retrieved from cache using an Xpath route through the XpathLoader. The last two stages are used to load the data from a CDSS that is using a web service to communicate with the guideline.

8.3.2 Curiam case study

```

<?xml version="1.0" encoding="UTF-8"?>
<CDSDC xmlns="urn:cdsdc.bmg.ibime.upv.es"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:sql="urn:sqlloader.cdsdc.bmg.ibime.upv.es"
  xsi:schemaLocation="urn:cdsdc.bmg.ibime.upv.es SQLLoader.xsd"
  cdsName="curiam_classif">

  <stage id="curiam_engine">

    <data id="SET">
      <dataLoader xsi:type="sql:SQLLoader">
        <sql:connection
xsi:type="URL">jdbc:mysql://localhost/testDB</sql:connection>
        <sql:sql xsi:type="SimpleValue">SELECT v.value FROM Patients p,
Variables v WHERE p.ID="'<query xsi:type="ContextValue">patientID</query>'" AND
p.Variable="SET" AND v.ID IN (SELECT( p.ID || '_' || p.Variable) FROM
Patients);</sql:sql>
      </dataLoader>
    </data>

    <data id="DET">
      <dataLoader xsi:type="sql:SQLLoader">
        <sql:connection
xsi:type="URL">jdbc:mysql://localhost/testDB</sql:connection>
        <sql:sql xsi:type="SimpleValue">SELECT v.value FROM Patients p,
Variables v WHERE p.ID="'<query xsi:type="ContextValue">patientID</query>'" AND
p.Variable="DET" AND v.ID IN (SELECT( p.ID || '_' || p.Variable) FROM
Patients);</sql:sql>
      </dataLoader>
    </data>
  </stage>
</CDSDC>

```

Code 8: XML instance used in the Curiam proof of concept (see section 5.1.2.2). The system only have one stage where the data is requested. Two mappings are defined to retrieve the data. The SQL loader permits defining dynamic variables, allowing the same mapping to be reused for different cases.

8.4 DCIS CASE STUDY

8.4.1 Diagram

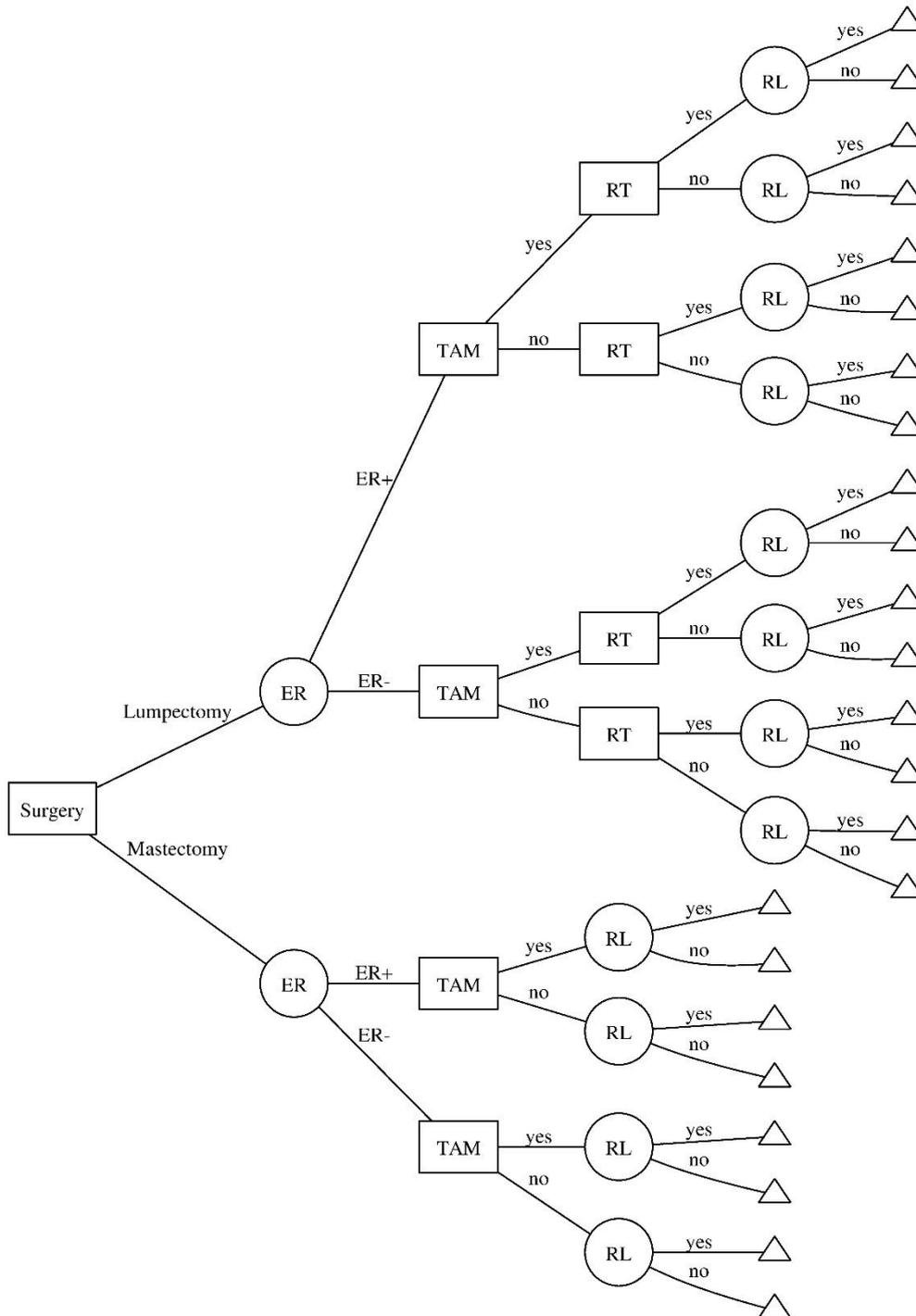


Figure 36: Complete decision tree diagram.

8.4.2 Naïve Bayes model mapping file

A data preload is done at the beginning of the guideline execution. When the data flow arrives at the node identified by the *stage id*, the data is extracted from the preloaded XML following the paths specified.

```
<?xml version="1.0" encoding="UTF-8"?>
<CSDC xmlns="urn:cdsc.bmg.ibime.upv.es"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:web="urn:webserviceloader.cdsc.bmg.ibime.upv.es"
xmlns:xpath="urn:xpathloader.cdsc.bmg.ibime.upv.es"
xsi:schemaLocation="urn:main.cdsc.bmg.ibime.upv.es mainSchema.xsd">

  <preload id="ehrPreload">
    <dataLoader xsi:type="web:WebServiceLoader">
      <web:restService>
        <web:restURL
xsi:type="URL">http://localhost:8080/ehrServer/patientEhr/</web:restURL>
        <web:datatype xsi:type="SimpleValue">text/xml</web:datatype>
        <web:parameters xsi:type="SimpleValue"/>
      </web:restService>
    </dataLoader>
  </preload>

  <stage id="nbProbModel">
    <data id="age">
      <dataLoader xsi:type="xpath:XPathLoader">
        <xpath:source xsi:type="PreLoadValue" preLoadId="ehrPreload"/>
        <xpath:xpath
xsi:type="SimpleValue">string(//x:observation/x:code[@code='1001']/../x:value/@value)
<xpath:xpath/>
        <xpath:namespaces xsi:type="SimpleValue">x=urn:h17-
org:v3<xpath:namespaces/>
      </dataLoader>
    </data>
    <data id="histology">
      <dataLoader xsi:type="xpath:XPathLoader">
        <xpath:source xsi:type="PreLoadValue" preLoadId="ehrPreload"/>
        <xpath:xpath
xsi:type="SimpleValue">string(//x:observation/x:code[@code='1002']/../x:value/@value)
<xpath:xpath/>
        <xpath:namespaces xsi:type="SimpleValue">x=urn:h17-
org:v3<xpath:namespaces/>
      </dataLoader>
    </data>
    <data id="type">
      <dataLoader xsi:type="xpath:XPathLoader">
        <xpath:source xsi:type="PreLoadValue" preLoadId="ehrPreload"/>
        <xpath:xpath
xsi:type="SimpleValue">string(//x:observation/x:code[@code='1003']/../x:value/@value)
<xpath:xpath/>
        <xpath:namespaces xsi:type="SimpleValue">x=urn:h17-
org:v3<xpath:namespaces/>
      </dataLoader>
    </data>
    <data id="estrogen_receptors">
      <dataLoader xsi:type="xpath:XPathLoader">
        <xpath:source xsi:type="PreLoadValue" preLoadId="ehrPreload"/>
        <xpath:xpath
xsi:type="SimpleValue">string(//x:observation/x:code[@code='1004']/../x:value/@value)
<xpath:xpath/>
        <xpath:namespaces xsi:type="SimpleValue">x=urn:h17-
org:v3<xpath:namespaces/>
      </dataLoader>
    </data>
    <data id="progesterone_receptors">
```

```

        <dataLoader xsi:type="xpath:XPathLoader">
            <xpath:source xsi:type="PreLoadValue" preLoadId="ehrPreload"/>
            <xpath:xpath
xsi:type="SimpleValue">string(//x:observation/x:code[@code='1005']/../x:value/@value)
<xpath:xpath/>
            <xpath:namespaces xsi:type="SimpleValue">x=urn:h17-
org:v3<xpath:namespaces/>
        </dataLoader>
    </data>
    <data id="her2">
        <dataLoader xsi:type="xpath:XPathLoader">
            <xpath:source xsi:type="PreLoadValue" preLoadId="ehrPreload"/>
            <xpath:xpath
xsi:type="SimpleValue">string(//x:observation/x:code[@code='1006']/../x:value/@value)
<xpath:xpath/>
            <xpath:namespaces xsi:type="SimpleValue">x=urn:h17-
org:v3<xpath:namespaces/>
        </dataLoader>
    </data>
    <data id="affected_axillary_lymph ">
        <dataLoader xsi:type="xpath:XPathLoader">
            <xpath:source xsi:type="PreLoadValue" preLoadId="ehrPreload"/>
            <xpath:xpath
xsi:type="SimpleValue">string(//x:observation/x:code[@code='1007']/../x:value/@value)
<xpath:xpath/>
            <xpath:namespaces xsi:type="SimpleValue">x=urn:h17-
org:v3<xpath:namespaces/>
        </dataLoader>
    </data>
    <data id="stage">
        <dataLoader xsi:type="xpath:XPathLoader">
            <xpath:source xsi:type="PreLoadValue" preLoadId="ehrPreload"/>
            <xpath:xpath
xsi:type="SimpleValue">string(//x:observation/x:code[@code='1008']/../x:value/@value)
<xpath:xpath/>
            <xpath:namespaces xsi:type="SimpleValue">x=urn:h17-
org:v3<xpath:namespaces/>
        </dataLoader>
    </data>
    <data id="surgery">
        <dataLoader xsi:type="xpath:XPathLoader">
            <xpath:source xsi:type="PreLoadValue" preLoadId="ehrPreload"/>
            <xpath:xpath
xsi:type="SimpleValue">string(//x:observation/x:code[@code='1009']/../x:value/@value)
<xpath:xpath/>
            <xpath:namespaces xsi:type="SimpleValue">x=urn:h17-
org:v3<xpath:namespaces/>
        </dataLoader>
    </data>
    <data id="radiotherapy">
        <dataLoader xsi:type="xpath:XPathLoader">
            <xpath:source xsi:type="PreLoadValue" preLoadId="ehrPreload"/>
            <xpath:xpath
xsi:type="SimpleValue">string(//x:observation/x:code[@code='1010']/../x:value/@value)
<xpath:xpath/>
            <xpath:namespaces xsi:type="SimpleValue">x=urn:h17-
org:v3<xpath:namespaces/>
        </dataLoader>
    </data>
    <data id="quimiotherapy">
        <dataLoader xsi:type="xpath:XPathLoader">
            <xpath:source xsi:type="PreLoadValue" preLoadId="ehrPreload"/>
            <xpath:xpath
xsi:type="SimpleValue">string(//x:observation/x:code[@code='1011']/../x:value/@value)
<xpath:xpath/>

```

```
        <xpath:namespaces xsi:type="SimpleValue">x=urn:h17-
org:v3<xpath:namespaces/>
    </dataLoader>
</data>
<data id="hormones">
    <dataLoader xsi:type="xpath:XPathLoader">
        <xpath:source xsi:type="PreLoadValue" preLoadId="ehrPreload"/>
        <xpath:xpath
xsi:type="SimpleValue">string(//x:observation/x:code[@code='1012']/../x:value/@value)
<xpath:xpath/>
        <xpath:namespaces xsi:type="SimpleValue">x=urn:h17-
org:v3<xpath:namespaces/>
    </dataLoader>
    </data>
</stage>
</CDSDC>
```