Metamodeling of Bayesian networks for decision-support systems development

Isabel M. del Águila and **José del Sagrado** [1]

Abstract. The knowledge modeling and software modeling phases in Knowledge-Based System development are not integrable, in terms of representation, due to the different languages needed at the steps of the development. This paper focuses on bring closer these languages. By one hand, we define a meta model which contains the key concepts used in the definition of a knowledge model as a Bayesian network. On the other hand, we define an extension of UML using profiles that can bridge the gap in representation and facilitate the seamless incorporation of a knowledge model, as Bayesian network, in the context of a knowledge-based software development.

1 Introduction

Knowledge-based systems (KBSs) are characterized by their high risk, loose definition, poor structure and subjective requirements. These software systems were introduced in the early 1970s as expert systems from the field of artificial intelligence (AI) research. Originally their goal was to transfer expertise from domain experts to a some kind of knowledge base that may be integrable in a software system. However, nowadays knowledge engineering is changing as it turns towards the modeling approach. A KBS can be defined as "software that has some knowledge or expertise about specific, narrow domain, and is implemented such that Knowledge base and the control architecture are separated. Knowledge-Based Systems have capabilities that often include inferential processing (as opposed to algorithmic processing), explaining rationale to users an generating non-unique results" [19]. From this definition is easy to see the many roles played by the knowledge model (knowledge bases). Models provide an abstraction about reality and through this knowledge models the human experts problem solving approaches in order to be used in the development of software solutions. Knowledge models usually are described in a specific purpose language. There is not a standard, because it depends heavily on knowledge representation mechanisms (i.e. rules, semantic networks, frames). However, from a commercial point of view, the development of a software system focuses on customers, that is, in order to develop any software solution we need to gather requirements from customers and translate it into software functionalities. All the desired functionalities do not have to apply artificial intelligence techniques.Thus, software systems usually integrate a KBS with other needed software enterprise components.

These not knowledge based components are described using modeling languages from the software engineering domain (UML is the most used standard). This lead us to combine several modeling languages in the same project. Figure 1 shows the vision of a software development project from the points of view of a customer, a software engineer and a knowledge engineer. The knowledge engineer (Figure 1A) uses knowledge engineering methods to define the project's task, relegating to the background the tasks defined by software engineering. The software product that results from this process is a KBS. From another perspective, the software engineer (Figure 1B) systematically applies its skills, tools and software engineering methods to develop a software product (system), where knowledge is only another element. Finally, the customer's view of the project (Figure 1C) focuses on quality and the need of cooperation between the two engineerings and their own modeling languages [2, 3], so that the final software product properly covers all her/his needs. In other words, software components based and not based on knowledge must be integrated in shaping the software system for the end user.

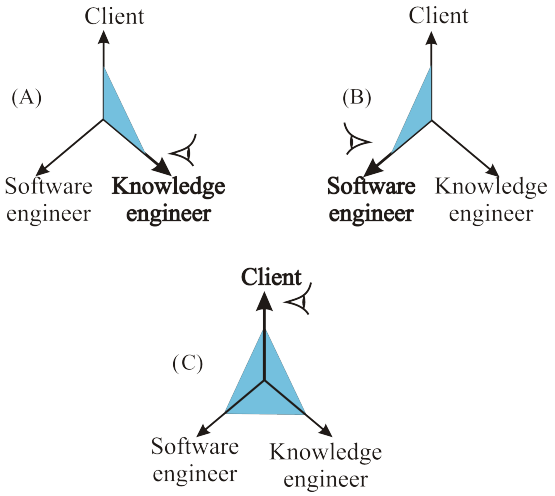

Figure 1. Different views of a software development project

The use of different modeling languages limits the applicability of one of the software development schemes more widespread in our day, Model-Driven Architecture (MDA) [13]. This approach, to information software systems development, separates specification of the system functions from implementation of these functions in a given platform, focusing on models as a higher level of abstraction during systems construction [5]. This fact leads developers to a significant decoupling between platform-independent models (PIMs) and platform-specific models (PSMs). Separation between specification and implementation is also a basic feature in KBS (see

[1] Department of Languages and Computation, University of Almería, Spain, email: imaguila@ual.es

[19]). Our problem is putting together different modeling language notations into a single platform independent model, so that the knowledge model and the UML model are expressed in a compatible format. Therefore, we need a extended PIM that include algorithmic and inference functionalities. A single language allows a great level of abstraction and makes easy the implementation process. The same PIM model can be translated to different platforms: the core goal of MDA.

This work propose a extension of UML when Bayesian Networks (BN) [21, 15, 16] are the representation selected by knowledge engineers to model the knowledge. We focus on BN because among all the knowledge modeling languages, Bayesian networks can be successfully used to represent expert knowledge on an uncertain domain. Today, BNs are expressed by applying its own algebraic notation, but if we want a BNs to be part of a software solution, we must be able to express themselves in the same language that is commonly used to model general purpose software (i.e. UML). In this paper we propose an approach which aims to improve the development of decision support system, reinforcing the aspects of BNs modeling. In order to achieve this goal, we introduce a BN metamodel and a UML profile as means to build the PIM for a KBS that represents knowledge based on a BN.

The rest of this paper is structured as follows. Section 2 introduces the basic fundamentals of Bayesian networks. The general MDA translation schema for BNs is explained in Section 3. Section 4 describe a basic metamodel for BNs (BayNet) that provides a specific and intuitive notation for modeling BN-based KBS. The extension of UML taking as basis the BN metamodel BayNet to create an UML profile for BNs (UBN) is studied in Section 5. Then, Section 6 shows how to apply UBN profile to the development of a simple pest control BN-based KBS. Finally, the conclusions and future works are exposed in Section 7.

2 Bayesian Networks Basics

A Bayesian network is a probabilistic graphical model that represents a set of random variables and their conditional dependencies via a directed acyclic graph. Because a Bayesian network is a complete model for the variables and their relationships, it can be used to answer probabilistic queries about them.

Formally, a BN [21, 15, 16] is a pair $(G, \mathbf{P})$ where

- $G = (\mathbf{V}, \mathbf{E})$ is a directed acyclic graph whose set of nodes $\mathbf{V} = \{\mathbf{X_1}, \mathbf{X_2}, \cdots, \mathbf{X_n}\}$ represents the system variables and whose set of arcs $\mathbf{E}$ represents direct dependence relationships among the variables, and
- $\mathbf{P}$ is a set of conditional probability distributions containing a conditional probability distribution $P(X_i|pa(X_i))$ for each variable X_i given its set of parents $pa(X_i)$ in the graph.

The joint probability distribution over $\mathbf{V}$ can be recovered from this set $\mathbf{P}$ of conditional probability distributions applying the chain rule as:

$$P(X_1, \cdots, X_n) = \prod_{i=1}^{n} P(X_i \mid pa(X_i)). \quad (1)$$

The process of obtaining the graph and the probabilities of a BN can be done either *manually*, from experts' knowledge on the domain, or *automatically*, from databases. In the first case, the elicitation of probabilities constitutes a bottleneck in the development of BNs [9].

A BN-based KBS needs to translate this algebraic notation to environments, such as Elvira [10], that consists of software packages for the edition, use and evaluation of BN. These environments provide class libraries that can be integrated as any other component into a software application (i.e. BN-based KBS). This application will be released to the end user and will contain an implementation of the network itself and network capabilities, such that inference from observed values of some variables.

In particular, the Elvira system [10] is tool to construct probabilistic decision support systems. Elvira works with Bayesian networks and influence diagrams and can operate with discrete, continuous and temporal variables. It has an easy to use Graphical User Interface (GUI) (see Figure 2). In addition you can edit, easily, the ASCII format of the Elvira systems to introduce your models. It has methods for inference, learning, abduction, fusion of knowledge and making decisions, and some tools to check the efficiency of the algorithms.

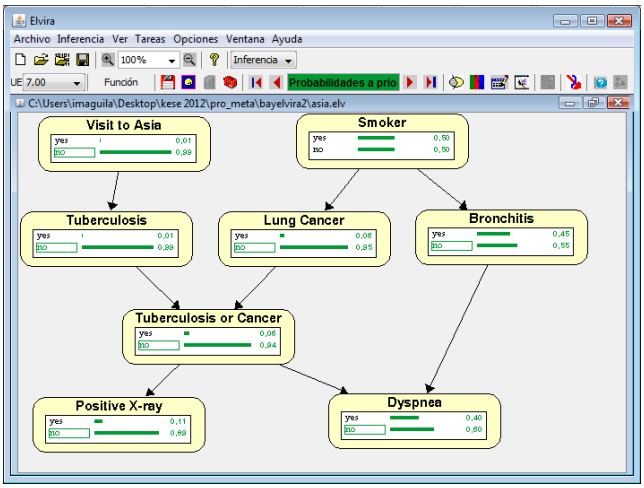

Figure 2. Elvira's main interface

The program Elvira has its own format for storing models, a parser, exact and approximate (stochastic and deterministic) algorithms for inference on both discrete and continuous variables, a graphical interface for building and evaluating Bayesian networks and influence diagrams, with specific options for canonical models (OR, AND, MAX, etc.), explanation of reasoning, decision making algorithms, learning (model building) from databases, fusion of networks, etc. Elvira is written and compiled in Java, which allows the program to run on different platforms and operating systems: Linux, MS-DOS/Windows, Solaris, etc.

2.1 Reasoning with Bayesian Networks

Probabilistic reasoning in BNs consists in computing the posterior probability distributions of some variables of interest $v_I \in \mathbf{V_I}$ given some observed variables $\mathbf{V_E}$ (this sets of findings is called *evidence*), $P(v_I|\mathbf{V_E})$. This process is performed via a flow of information through the network in any direction. If we give a *causal interpretation* to the links in the network (i.e. for an arc $X_i \rightarrow X_j$ we say that X_i is a *cause* of X_j and X_j is the *effect* of X_i), we can perform several types of reasoning [17]:

- *Diagnostic* reasoning, the evidence flows in the opposite direction to the arcs, from effects to cause (i.e. some effects receive evidence and some of their causes are the variables of interest).

- *Predictive* reasoning, the evidence flows in the direction of the arcs, from cause to effect (i.e. some causes receive evidence and their effects are the variable of interests).
- *Intercausal* reasoning, the evidence flows in all directions. This type of reasoning involves reasoning about mutual causes of a common effect (i.e. the effect and some of the causes receive evidence and some of the causes are the variables of interests).

Sometimes reasoning does not match into these three types, because any variable can be an interest variable and may be an evidence variable and information flows in either direction (i.e. the effects of a causes and the causes of the second receive evidence and the central cause is the variable of interest). This situation occurs when diagnostic and predictive reasoning are used simultaneously.

3 Bayesian Network as PIM

In terms of MDA vocabulary, a BN is a PIM, Elvira is a platform model into which a BN has to be translated in order to define the final PSM. Elvira software also offers Java support for the BN, that is to say, it provides the last level of the MDA approach (i.e. the code).

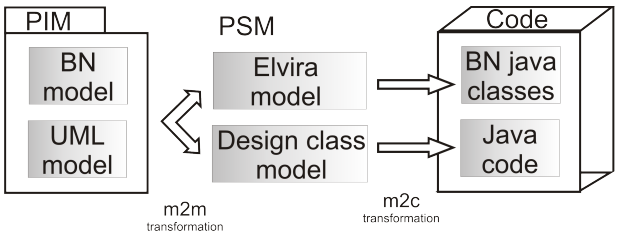

Figure 3. MDA for BN-based KBS

Figure 3 shows the translation model proposed. Both, the transactional (or interaction) PSM and the knowledge (or BN) PSM are expresed in terms of object-oriented design languages coming from Software Engineering (i.e. UML) and their translation into code is solved by means of a m2c translation. Many CASE tools, like Visual Paradigm, Microsoft Visual Studio or Enterprise Architect, already incorporate, at least to some degree, this kind of translation. But to use the power of MDA, is also necessary to express the PIM of these two parts of the software solution. There are modeling languages in the Software Engineering area that allow to express the PIM, but what about the BN? What is its language model? Is it UML compatible?

The MDA approach has been applied in other forms of knowledge representation as rules [7, 1]. From the metamodel of the rule-based languages have been defined UML profiles and automated tools, that translate the rules of PIM models into rules-based web-systems by combining Java Server Faces with Jess rule engine [6]. But, BNs lack of a modeling language compatible with UML that allows the application of MDA. Model transformation consists on the process of converting one model to another model of the same system in a different abstraction level: from PIM to PSM and from PSM to code. MDA tools allow these transformations to be automated and executed automatically.

Our goal is to create an UML-compatible modeling language for BNs. The Object Management Group (OMG) has defined two extension mechanisms for UML: metamodel model and profile extensions [14]. First extension mechanism involves the process of defining a new metamodel on which to build an entirely new language defined through the Meta-Object Facility (MOF) specification. But if we do not want to change UML semantics and only particularize some concepts, we can extend UML using a series of mechanism offered by the language itself: the profiles. [12].

We know that all the knowledge representation mechanisms are in themselves languages. So we can choose to build a totally new MOF language, but it may not respect the standard UML metamodel. This fact will prevent existing UML tools to manage the new language concepts in a natural way. To offer a proposal that also gives support to UML, leads to a greater number of users and reduce the learning curve in the new language. For all these reasons and agreeing with the proposals of several authors [18, 22, 4], we also propose the use of UML profiles. In any case, according to several studies [11, 12], the starting point for developing a UML profile is the metamodel for the platform or of the domain of the application that is going to be modeled. In our case, the starting point is a BN metamodel (called BayNet) that will allow us to specify how BN concepts are related to and represented in standard UML. Visual Paradigm is the CASE tool used to define this metamodel and its associated profile.

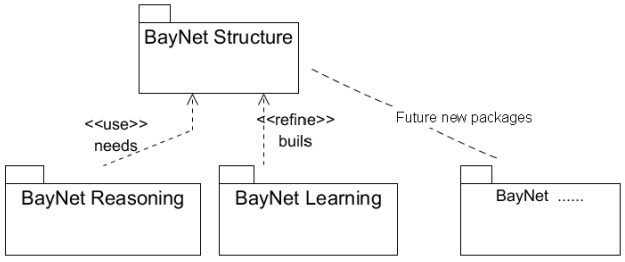

Figure 4. BayNet metamodel's basic structure

4 Metamodel for Bayesian Network

A metamodel includes domain entities, retrictions between them and limitations on the use of entities and relationships. BNs are complex in nature, beside its structure we have to face with complex concepts, as inference and learning, that have to be approached by successive approximations. This is the reason why we have split the metamodel in several packages as it is shown in Figure 4. The BayNet metamodel is the basis for providing a specific and intuitive notation for modeling BN-based KBS.

In a first approximation to BayNet, we only focus on BayNet structure (as we need to define the BN structure to define a model) and BayNet reasoning (as we need to carry out inference in order to reasoning with a BN) packages (see Figure 5). The BayNet structure package represents the basic components of a BN (BNet class): its qualitative (i.e. directed acyclic graph) and quantitative (i.e. set of probability distributions) parts. The qualitative part is represented by the class *Variable* and its self-association. An *Assignment* consists in assigning a *State* to a *Variable* modifying, accordingly, its marginal probability. The quantitative part is represented by means of the classes *Configuration* and *Relation*. For each given child-father association (*Configuration*) in the directed acyclic graph is assigned a conditional probability value (*Relation*). That is, we assign a

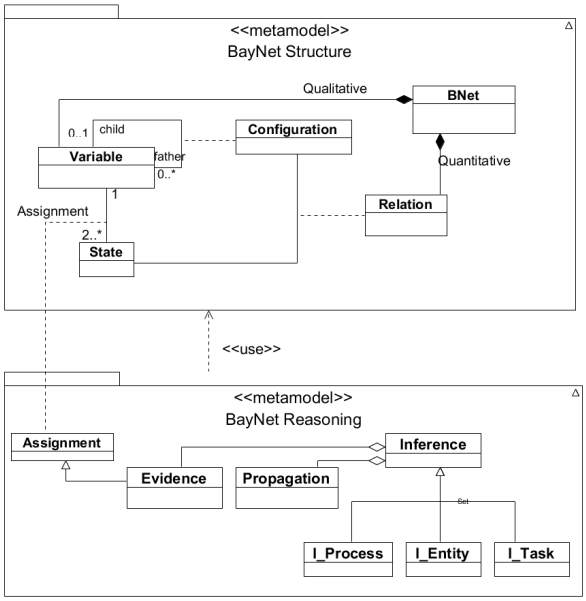

Figure 5. BayNet metamodel

probability value to each combination of values of a variable X_i and its parents $pa(X_i)$ in the graph, to define the conditional probability distribution $P(X_i|pa(X_i))$.

In BayNet reasoning, an inference can be view as a process (*I_Process*), a class able to carry out inferences (*I_entity*) or an operation inside a class (*I_task*). These three views allows to model different levels of abstraction in the decision tasks associated to a BN-based KBS. An *Inference* is an aggregation of the observed variables (*Evidence*) together with the execution of the operations needed to make evidence flow on the network and to compute the posterior probability distributions of the variables of interest (*Propagation*).

5 UBN profile

UML offers the possibility to extend and adapt its metamodel to a specific area of application through the creation of profiles. The BayNet metamodel is the basis that will provide a specific and intuitive notation for modeling BN-based KBS and including it in an UML project. UML profiles are UML packages with the stereotype $<< profile >>$. A profile can extend a metamodel or another profile while preserving the syntax and semantic of existing UML elements. It adds elements which extend existing classes. UML profiles consist of *Stereotypes*, *Constraints* and *Tagged Values*.

- A *stereotype* is a model element defined by its name and by the base class(es) to which it is assigned. Base classes are usually metaclasses from the UML metamodel, for instance the metaclass $<< Class >>$, but can also be stereotypes from another profile.
- *Constraints* are applied to stereotypes in order to indicate restrictions. They specify pre- or post conditions, invariants, etc., and must comply with the restrictions of the base class. Constraints can be expressed in any language, such as programming languages, natural language or Object Constraint Language (OCL).
- *Tagged values* are additional meta-attributes assigned to a stereotype, specified as name-value pairs. They have a name and a type and can be used to attach arbitrary information to model elements.

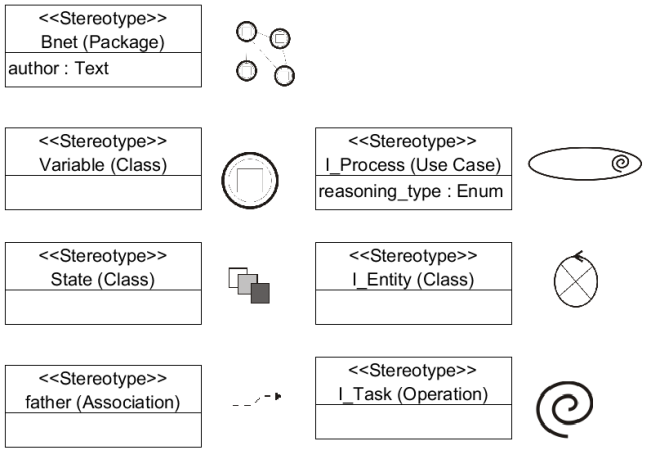

Figure 6. UBN stereotypes and icons

We use UML profile to define a UML Bayesian network profile (UBN). The aim of UBN is to define a language for designing, visualizing, specifying, analyzing, constructing and documenting the artifacts of knowledge-based systems, that represents it knowledge as a BN. The next step is to map the BayNet metamodel, described in the above section, to UML metaclases and make the necessary extensions. The mapping is a non-trivial task, because we need to know in deep how to apply the UML language. Most of concepts will map to stereotypes on a selected UML metaclass. Also we can define icons for most of the stereotypes, that allows the modeler to use intuitive symbols instead of UML shapes. Figure 6 shows the actual mapping with UML metaclasses.

Once the UBN is defined, it can be used in the software development of a particular application by defining a stereotyped dependency ($<< applyProfile >>$) between the UBN package an the package that is being under development for the application, as Figure 7 shows. A partial view of the class model of Elvira is included as it is needed in order to define the m2m translation between PIM and PSM (see Figure 3).

6 Case Study: A pest control BN-based KBS

This section shows how to apply UBN in an specific KBS development project. The project follows a development methodology described in [3], the process model proposed allows the seamless inclusion of Bayesian networks into the final software solution for an organizational environment. Let us begin with a brief description of the project to assist decision making in an agricultural domain. Our problem is related to pest control in a given crop under the regulation of Integrated Production. The

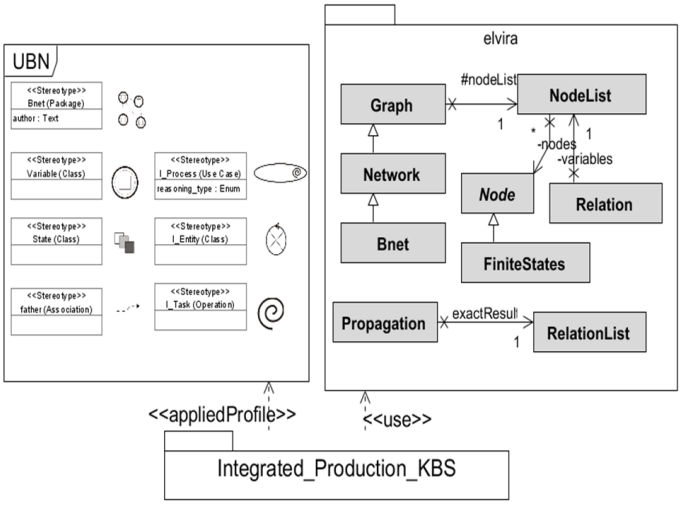

Figure 7. Bayesian network KBS modeling packages: pest control application

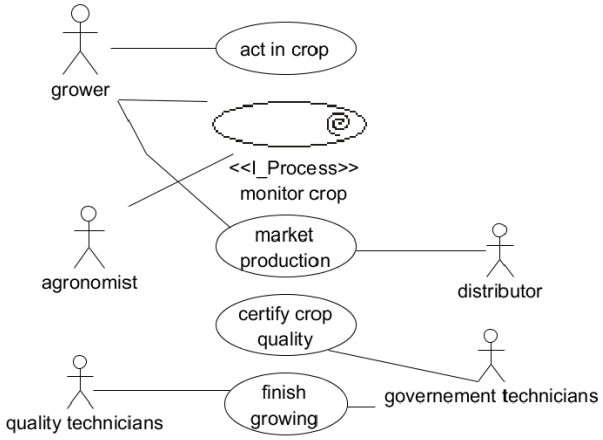

Figure 8. Use cases

Integrated Production Quality standard is adopted by a group of growers in order to achieve a quality production certification. The adoption of this standard forces growers to be disciplined in growing which involves intervention by technicians, marketing controls, and periodical inspection by the standard certification agencies.

The three main steps in the development of software systems that embed functionalities based and not based on knowledge, concerning the decision support process and the information management processes, are: *Requirement modeling* (RM), *Expert modeling* (EM) and *Specification of the software solution* (SSS)[3]. The first two are in charge of the definition on the PIM model according MDA (see Figure 3), and here is where UBN gets its value, because we can use only UML in order to execute RM and EM.

Software project development starts with business and RM modeling. The first activity consists of collecting, structuring and organizing all the relevant data for a concise and accurate definition of the problem to be solved by the software system. Integrated production involves handling and storing a huge amount of information about crops, and making decisions about all the tasks that have to be performed to fulfill the quality regulations.

We model the processes that are represented as use cases. The typical processes in an integrated production problem are shown in Figure 8. All tasks related to information required for quality management standards, without needing any knowledge based approach, are: Market Production, Act in Crop, Certify Crop Quality, and Finish Growing. All tasks related to pest control are performed by growers and agronomists in the Monitor crop process and represents the inference tasks that we need to model by means of a Bayesian network. The decision process when monitoring a crop is made at two levels. First, a decision is made on whether crop control action is necessary by sampling pests and estimating risk of attack. Then if it is decided that crop control action is required, the product (chemical or biological) to be applied has to be selected. The treatment advised has to respect natural enemies and other biological products previously used.

The next steep is to finish the PIM, using UML and UBN in order to define the system without considering platform level details. That is, from use cases we need to define the conceptual models. In this section, as specific case, we focus on the Monitor crop use case that can be described as the following informal scenario: Each week, the agricultural expert samples the crops condition and makes an estimate of the risk of pest attack. Crop sampling consists of direct observation and count of harmful agents in randomly selected plants. Where imbalance is detected, the expert advises treatment meeting the integrated production standard.

A crop is a complex system consisting of a plot of land, plants, a set of diseases and pests, and natural enemies that may be able to control them. The problem is to decide what treatment to apply, in order to maintain a balanced system. Figure 9, shows the UML class diagram obtained. Some of the classes in the model are variables of the Bayesian network (EM).

Within the scope of integrated production systems, when an agricultural expert visits a greenhouse, he writes down the date of the visit and samples the crop, including information about fauna, weather (wind, rain, etc.) and environment (weeds). The general schema for a crop-harmful agent pair consists of observing the crops condition and fauna. Crop condition is measured in terms of its phenology. The presence of fauna is important to estimate the intensity of the attack. The crop condition, along with the intensity of pest attack, determines the need for applying a plant health treatment or not. Periodical inspections of this kind are performed weekly. Figure 10 shows the BN structure elicited from the knowledge of the domain expert. Once the BN structure has been established, the probabilities are estimated completing the construction of the BN model. This expert modelling process has been successfully applied to determine the need of applying a treatment for olives' fly [8].

In order to select the set of relevant variables, we start from the initial conceptual model of the project that has to be refined. A first version of the PIM is shown in figure 9. Some of the modeling elements are stereotyped, using UBN, as nodes in a BN, Crop condition is measured in terms of its phenology. The presence of fauna is important to estimate the intensity of the attack. The crop condition, along with the intensity of pest attack, determines the need for applying a plant health treatment or not. Periodical inspections of this kind are performed weekly. Relationship that complete the qualitative part of the BN are shown as stereotyped associations of

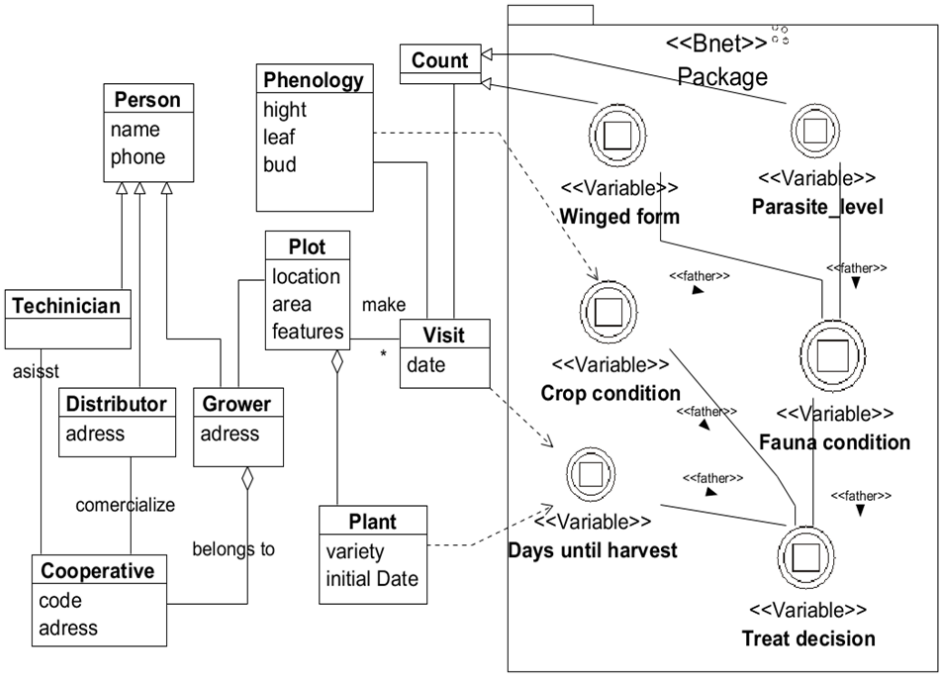

Figure 9. Partial view of the PIM

type $<< father >>$. Once the BN structure has been established, the probabilities are estimated (quantitative elicitation activity) based on a local government database of cases, completing the construction of the BN model. This expert modeling process has been successfully applied to determine the need of applying a treatment for the olives' fly (dacus olae) [20].

Finally, the specification of the software solution (SSS) represent a m2m translation that produces the PSM. Based on the PSM obtained, a m2c translation can be used to obtain a BN-based KBS in order to assists grower and technicians in pest control decision support tasks.

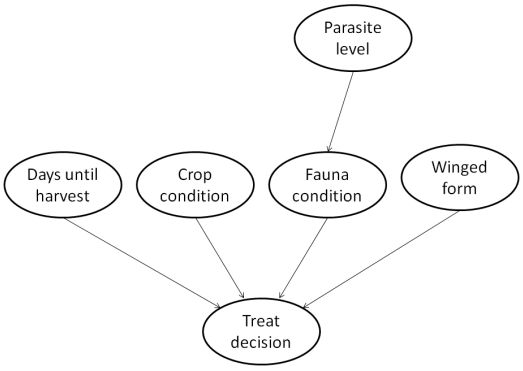

Figure 10. A BN structure for a crop-harmful agent pair

7 Conclusions

In this work we have presented a metamodel (BayNet) and an UML profile (UBN) for BN-based KBS modeling. This metamodel covers several important aspects for achieving the seamless inclusion of BN models into a final software solution for an organizational environment. The applicability of our solution has been tested in a simplified version of a real world problem: integrated production in agriculture.

Our proposal allows to manage a domain-specific language for BN without changing UML semantics. This can be view as a general framework to apply Model Driven Development, extending it to the BN-based KBS case. Developing a profile is a difficult task that implies to perform many steps. The next steps of this research will consist in defining an specification of constraints and operations using OCL, validate the profile using a CASE tool as Visual Paradigm and test it in a real-life development project that includes knowledge-base features.

ACKNOWLEDGEMENTS

This research has been funded by the Control crop Project (PIO-TEP-6174) from the Counseling of Economy, Innovation and Science, Government of Andalusia (Spain) and the Spanish Ministry of Education, Culture and Sport under project TIN2010-20900-C04-02.

REFERENCES

[1] Mohd Abdullah, Ian Benest, Richard Paige, and Chris Kimble, 'Using unified modeling language for conceptual modelling of knowledge-based systems', in *Conceptual Modeling - ER 2007*, eds., Christine Parent, Klaus-Dieter Schewe, Veda Storey, and Bernhard Thalheim, volume 4801 of *Lecture Notes in Computer Science*, 438–453, Springer Berlin-Heidelberg, (2007).

[2] Isabel María del Águila, Joaquín Cañadas, José Palma, and Samuel Túnez, 'Towards a methodology for hybrid systems software development', in *SEKE*, eds., Kang Zhang, George Spanoudakis, and Giuseppe Visaggio, pp. 188–193, (2006).
[3] Isabel María del Águila, José del Sagrado, Samuel Túnez, and Francisco Javier Orellana, 'Seamless software development for systems based on bayesian networks - an agricultural pest control system example', in *ICSOFT (2)*, eds., José A. Moinhos Cordeiro, Maria Virvou, and Boris Shishkov, pp. 456–461. SciTePress, (2010).
[4] Saartje Brockmans, Robert Colomb, Peter Haase, Elisa Kendall, Evan Wallace, Chris Welty, and Guo Xie, 'A model driven approach for building owl dl and owl full ontologies', in *The Semantic Web - ISWC 2006*, eds., Isabel Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Mike Uschold, and Lora Aroyo, volume 4273 of *Lecture Notes in Computer Science*, 187–200, Springer Berlin / Heidelberg, (2006).
[5] Alan W. Brown, 'Model driven architecture: Principles and practice', *Software and System Modeling*, **3**(4), 314–327, (2004).
[6] Joaquín Cañadas, José Palma, and Samuel Túnez, 'A tool for mdd of rule-based web applications based on owl and swrl', in *KESE*, eds., Grzegorz J. Nalepa and Joachim Baumeister, volume 636 of *CEUR Workshop Proceedings*. CEUR-WS.org, (2010).
[7] Joaquín Cañadas, José Palma, and Samuel Túnez, 'Defining the semantics of rule-based web applications through model-driven development', *Applied Mathematics and Computer Science*, **21**(1), 41–55, (2011).
[8] José del Sagrado and Isabel del Águila, 'Olive fly infestation prediction using machine learning techniques', in *Current Topics in Artificial Intelligence*, eds., Daniel Borrajo, Luis Castillo, and Juan Corchado, volume 4788 of *Lecture Notes in Computer Science*, 229–238, Springer Berlin / Heidelberg, (2007).
[9] Marek J. Druzdzel and Roger R. Flynn, 'Decision support systems', in *Encyclopedia of Library and Information Science*, ed., Allen Kent, volume 67, 120–133, Marcel Dekker, Inc., New York, NY, (2000).
[10] T. Elvira Consortium, 'Elvira: An environment for creating and using probabilistic graphical models', in *Proceedings of the First European Workshop on Probabilistic Graphical Models (PGM-02*, eds., J. Gómez and A. Salmerón, pp. 1–11, (2002).
[11] Lidia Fuentes and Antonio Vallecillo, 'An Introduction to UML Profiles', *The European Journal for the Informatics Professional*, **5**(2), (April 2004).
[12] Giovanni Giachetti, Francisco Valverde, and Oscar Pastor, 'Improving automatic uml2 profile generation for mda industrial development', in *ER Workshops*, eds., Il-Yeol Song, Mario Piattini, Yi-Ping Phoebe Chen, Sven Hartmann, Fabio Grandi, Juan Trujillo, Andreas L. Opdahl, Fernando Ferri, Patrizia Grifoni, Maria Chiara Caschera, Colette Rolland, Carson Woo, Camille Salinesi, Esteban Zimányi, Christophe Claramunt, Flavius Frasincar, Geert-Jan Houben, and Philippe Thiran, volume 5232 of *Lecture Notes in Computer Science*, pp. 113–122. Springer, (2008).
[13] O. M. G. Group, *MDA Guide Version 1.0.1*, document omg/03-06-01 edn., june 2003. http://www.omg.org/cgi-bin/doc?omg/03-06-01.
[14] O. M. G. Group, *UML Specification, Version 2.0*, 2005. http://www.omg.org/spec/UML/.
[15] Finn V. Jensen and Thomas D. Nielsen, *Bayesian Networks and Decision Graphs*, Springer Publishing Company, Incorporated, 2nd edn., 2007.
[16] Uffe B Kjrulff and Anders L Madsen, *Bayesian Networks and Influence Diagrams*, Springer New York, 2008.
[17] K. Korb and A. Nicholson, *Bayesian Artificial Intelligence*, Chapman and Hall, 2nd edn., 2010.
[18] François Lagarde, Huáscar Espinoza, François Terrier, and Sébastien Gérard, 'Improving uml profile design practices by leveraging conceptual domain models', in *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*, ASE '07, pp. 445–448, New York, NY, USA, (2007). ACM.
[19] Mary Lou Maher and R. H. Allen, *Expert System Components*, American Society of Civil Engineering, 1987.
[20] Francisco Javier Orellana, José del Sagrado, and Isabel María del Águila, 'Saifa: A web-based system for integrated production of olive cultivation', *Comput. Electron. Agric.*, **78**(2), 231–237, (September 2011).
[21] Judea Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.
[22] Bran Selic, 'A Systematic Approach to Domain-Specific Language Design Using UML', , *IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, 2–9, (May 2007).