Expert Systems with Applications 39 (2012) 9041–9053

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

# Enhancing ontology-based antipattern detection using Bayesian networks

Dimitrios Settas [a,*], Antonio Cerone [a], Stefan Fenz [b]

[a] United Nations University, International Institute for Software Technology, Macau, SAR, China
[b] Institute of Software Technology and Interactive Systems, Vienna University of Technology and SBA Research, Vienna, Austria

ARTICLE INFO

*Keywords:*
Bayesian networks
Ontology
Antipatterns
Antipattern detection

ABSTRACT

Antipatterns provide information on commonly occurring solutions to problems that generate negative consequences. The antipattern ontology has been recently proposed as a knowledge base for SPARSE, an intelligent system that can detect the antipatterns that exist in a software project. However, apart from the plethora of antipatterns that are inherently informal and imprecise, the information used in the antipattern ontology itself is many times imprecise or vaguely defined. For example, the certainty in which a cause, symptom or consequence of an antipattern exists in a software project. Taking into account probabilistic information would yield more realistic, intelligent and effective ontology-based applications to support the technology of antipatterns. However, ontologies are not capable of representing uncertainty and the effective detection of antipatterns taking into account the uncertainty that exists in software project antipatterns still remains an open issue. Bayesian Networks (BNs) have been previously used in order to measure, illustrate and handle antipattern uncertainty in mathematical terms. In this paper, we explore the ways in which the antipattern ontology can be enhanced using Bayesian networks in order to reinforce the existing ontology-based detection process. This approach allows software developers to quantify the existence of an antipattern using Bayesian networks, based on probabilistic knowledge contained in the antipattern ontology regarding relationships of antipatterns through their causes, symptoms and consequences. The framework is exemplified using a Bayesian network model of 13 antipattern attributes, which is constructed using BNTab, a plug-in developed for the Protege ontology editor that generates BNs based on ontological information.



## 1. Introduction

Applying a previously successful solution to a similar problem is one of the established ways in which humans solve problems and is the main concept behind design patterns. After the emergence of design patterns, the pattern community slowly developed problem–solution pairs in which the solution is not beneficial and leads to further difficulties and problems. Antipatterns are the next generation of design pattern research and are related with patterns in the sense that design patterns can evolve into antipatterns. An antipattern is a new form of pattern that has two solutions. The first is a problematic solution with negative consequences and the other is a refactored solution, which describes how to change the antipattern into a healthy solution (Brown, McCormick, & Thomas, 2000). The second solution is what makes antipatterns beneficial. The difference is in the context: An antipattern is a pattern with inappropriate context and is particularly useful in the case of knowledge representation, because it captures experience and provides information on commonly occurring solutions to problems that generate negative consequences (Brown, Malveau, McCormick, & Mowbray, 1998). The process that is followed by a pattern to change its solution into a better one is called refactoring. This solution has an improved structure that provides more benefits than the original solution and refactors the system toward minimized consequences. There exist different categories of antipatterns. For the purposes of this paper, software project management antipatterns are used in order to exemplify the proposed approach.

In previous work, Bayesian Networks (BNs) (Settas, Bibi, Sfetsos, Stamelos, & Gerogiannis, 2006) and the formalism of ontology (Settas & Stamelos, 2007; Settas, Meditskos, Stamelos, & Bassiliades, 2011) have been used separately in order to produce statistical and extensible ontological models of antipatterns. Bayesian networks provided a framework for project managers, who would like to model the cause-effect relationships that underlie an antipattern, taking into account the inherent uncertainty of a software project. An antipattern BN model provides the precise mathematical model of a project management antipattern and can be used to measure and handle uncertainty in mathematical terms. By applying the formalism of ontology (Settas & Stamelos, 2007; Settas et al., 2011), a common lexicon of terms that can be communicated across people and software tools was defined. This provided the

* Corresponding author.
*E-mail addresses:* dsettas@csd.auth.gr (D. Settas), antonio@iist.unu.edu (A. Cerone), stefan.fenz@tuwien.ac.at (S. Fenz).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2012.02.049

basis for the application of further methodology in order to address similar antipattern ontologies and implement SPARSE (Settas et al., 2011), an ontology-based intelligent system that can detect antipatterns based on the symptoms that appear during a software project. This system has recently been proposed for teaching and learning using antipatterns (Settas & Cerone, 2011).

In this paper, we intertwine the formalisms of Bayesian Networks (BNs) and Ontologies. Ontology is a methodology that has been widely used for the development of expert systems (Liao, 2005). Bayesian networks are also widely used in expert systems for prognosis and prediction purposes (Ferreiroa, Arnaiza, Sierrab, & Irigoienb, 2012). The research goal in this paper is to further develop the existing antipattern ontology knowledge base by using Bayesian network analysis. Uncertainty concerns every aspect of ontologies (Taniar & Rahayu, 2006; Pan, Ding, Yu, & Peng, 2005) and it is one of the most important criteria that need to be taken into account in the selection of an appropriate knowledge representation (Richardson & Domingos, 2003). The task of capturing incomplete or uncertain antipattern knowledge is essential but is often not explored based on assumptions on the certainty and accuracy of software project data. For example, the certainty regarding the existence of the relationship between a specific cause and symptom of an antipattern. Support for uncertainty is essential for intelligent systems that support antipatterns, such as SPARSE because the creation of new antipatterns using the antipattern OWL ontology often relies on information from past project which comes from experience, memory and intuition. Antipattern ontology contributors might often rely on their own expert judgement to define how antipatterns might be linked, which has a clear impact on the effectiveness of the antipattern detection process. By incorporating BNs in the antipattern OWL ontology, we can represent probabilistic information regarding antipatterns and their attributes. As a result, the detection process can be improved by providing users not only a set of antipatterns that exist in a software project based on the symptoms they selected, but also to quantify, measure and visualize the uncertainty that lies behind antipattern attributes and relationships.

At the moment there exist a variety of ontology editors offering different functionality. Protege is one of the most widely used open source ontology editors and was chosen for the implementation of SPARSE and the antipattern OWL ontology (Settas et al., 2011). Further development of the ontology in previous work tackled the challenge of providing a web-based collaborative environment by using WebProtege[1] (Tudorache, Vendetti, & Noy, 2008), which is a web-client for Collaborative Protege. There is a variety of frameworks and tools under development in order to support the study of uncertainty in ontologies. BayesOWL framework (Ding, Peng, & Pan, 2004; Pan et al., 2005) can translate an OWL ontology into a Bayesian Belief Network structure, but it has still not been fully developed. Other frameworks currently under development such as POWL (Hung, Szeto, & Deng, 2009) and PR-OWL (Carvalho, Laskey, & Costa, 2010) can extend OWL vocabulary for representing uncertainty in different expressivities. These frameworks are being studied by the Uncertainty Reasoning for the World Wide Web Incubator Group (URW3-XG) (Laskey & Laskey, 2008) but at the moment they lack compatibility with OWL. In this paper, the issue of quantifying uncertainty in the antipattern ontology is addressed by using the BNTab Protege plugin. BNTab is the only plug-in that is currently available for Protege that can generate Bayesian networks based on ontologies. The BNTab[2] plug-in enables users to efficiently generate Bayesian networks based on existing ontologies.

The main contribution of this paper is the application of Bayesian networks on the antipattern OWL ontology. This approach incorporates probabilistic information in the ontology and allows the semi-automatic generation of BNs using BNTab. In that way, BNTab can illustrate the uncertainty that exists in antipatterns and antipattern attributes and enhance the antipattern detection process. Ultimately, the BNTab plugin will be used within the web-based collaborative version of the antipattern ontology (Settas, Stamelos, & Bassiliades, 2011) and antipattern contributors will be able to use the approach proposed in this paper simultaneously through the Web. This will ensure that all antipattern ontology users are editing the most up-to-date version of the ontology. SPARSE can benefit from this approach by including associated BNs with the resulting detected antipatterns in order to visualize the uncertainty of the data used for the detected antipatterns.

This paper is divided in six sections, which are organized as follows: Section 2 describes the background, the related work and the literature review used in our research. Section 3 introduces the reader to Bayesian networks and the BN analysis used in the remaining sections. Section 4 describes the antipattern ontology and presents a short description of the rules and reasoning used in SPARSE in order to detect antipatterns. Section 5 presents the process of extending the OWL ontology in order to describe probabilistic information. Furthermore, the application of BNTab in the antipattern ontology is described and the approach is exemplified using an example of a software project management antipattern. Finally, in Section 6, the findings are summarized, future work is proposed and conclusions are drawn.

## 2. Background and related work

### 2.1. Background

Bayesian networks and ontologies are different knowledge representation formalisms. In this paper these formalisms are intertwined, therefore it is important to understand how they can supplement each other at a syntactical but also inferential level. At a knowledge representation level, the antipattern ontology can determine the categories of things that exist in an antipattern and set the ontological commitments of the project manager, system architect or application designer (Davis & Szolovits, 1993). Both Ontologies and BNs offer a medium for efficient computation because they do not only represent knowledge, but also encode knowledge in a form that can be processed efficiently by software. Furthermore both BNs and ontologies offer a medium for human expression that can be used by project managers in order to have a common lexicon of terms with a solid mathematical foundation. The antipattern ontology encodes tacit software project management knowledge into a computer understandable form and allows the sharing and reuse of this knowledge by software tools, such as SPARSE which is a symptom-based antipattern retrieval knowledge-based system using semantic Web technologies.

Both BNs and ontologies have the ability to carry out reasoning. In the case of BNs, reasoning is carried out under uncertainty and by illustrating the uncertainty that surrounds antipatterns. The Bayesian capacity to draw strong inferences from sparse data cannot be ignored. However, it is the formalism of ontology that provided a sound basis for the further development of methods and software tools using ontology. By incorporating a BN model of an antipattern inside the ontology, managers can still take advantage of the mathematical analysis of BNs while using the formalism of ontology. The antipattern ontology served as the knowledge base for the development of the intelligent system (Settas et al., 2011) that can detect the antipatterns that exist in a software project.

A knowledge representation language has a syntactic and an inferential aspect. The syntactic aspect is notational and refers to

[1] http://protegewiki.stanford.edu/wiki/WebProtege
[2] http://stefan.fenz.at/ontology-based-generation-of-bayesian-networks/

the way in which one stores information in an explicit format. In Bayesian networks this refers to the notational aspect of cause and effect relationships and the probability values of the Conditional Probability Table (CPT). In the case of ontology this refers to the concepts and properties of the Ontology Web Language (OWL).

The inferential aspect refers to the way in which the explicitly stored information derives information that is implicit. However, knowledge representation formalisms are useless without the ability to reason with them (Sowa, 1999). Reasoning occurs using inferencing mechanisms. More specifically, reasoning is the "goal", whereas inferencing is the "implementation". In BNs, inferencing is carried out by means of Bayesian updating or by observing the values of a specific CPT. Reasoning under uncertainty using BNs studies and illustrates the uncertainty that surrounds antipatterns. Reasoning with ontologies can be carried out in various ways. Ontology provides automated reasoning techniques, which allow a computer system to draw conclusions from knowledge represented in a machine-interpretable form. This formalism has recently evolved in computer science as computational artifacts that can provide computer systems with a conceptual and computational model of a particular domain of interest. As a result, using ontology computer systems can base decisions on reasoning about domain knowledge, similar to humans.

In this paper, we propose extending the power of probabilistic reasoning provided by BNs using the formalism of ontology. BNs are used in the antipattern ontology because of the structural similarity between the Directed Acyclic Graph (DAG) of a BN and the structure of an OWL ontology: both of them are directed graphs, and direct correspondence exists between many nodes and arcs in the two graphs (Ding et al., 2004). According to Pan et al. (2005) a set of rules and procedures can be used for direct translation of an OWL ontology into a BN structure (a directed acyclic graph or DAG). The general principle underlying the structural translation rules is that all nodes in BN are translated to OWL classes and the arrows represent the influence relation (Pan et al., 2005).

However, the probabilities that are required in both translation and mapping can be obtained by using text classification programs, supported by associating with individual concepts relevant text exemplars retrieved from the web (Pan et al., 2005). In this paper we propose the process of incorporating BNs in an OWL ontology using the BNTab Protege plug-in and exemplify our approach using probabilistic information on software project management antipatterns. Software project managers can use data from past projects on the occurrence of antipatterns, causes, symptoms and consequences but can also supplement this data with expert judgement, which is one of the advantages of using BNs.

### 2.2. Related work

The research direction proposed in this paper builds upon the areas of antipatterns, ontology and Bayesian networks.

Anyone can become an antipattern author. As a result, there exists a large number of unstructured and informal antipatterns. Antipattern catalogues and blogs,[3] have covered several management personality and environmental antipatterns (Laplante & Neil, 2006) and different kinds of antipatterns on various Computer Science topics (Kis, 2002; Long, 2001; Kotzé, Renaud, & van Biljona, 2008; Krai & Zemlicka, 2007; Laplante, Hoffman, & Klein, 2007). The wide collection of antipatterns that is available offers a vast amount of knowledge on how to resolve antipatterns. Readers who are not familiar with antipatterns can refer to the work of Brown, Brown et al. (1998), Brown et al. (2000) and Laplante and Neil (2006) for an introduction to the topic.

The antipattern ontology (Settas & Stamelos, 2007; Settas, Sowe, & Stamelos, 2009) provides a description of antipattern concepts, attributes of concepts, relationships among concepts, constraints on these relationships, defining therefore knowledge reference structure of the domain of software project management antipatterns. At the moment, the technology of software project management antipatterns is supported by tools that can detect the existence of such antipatterns in software projects (Settas et al., 2011; Settas et al., 2011). While there is a considerable amount of literature about ontologies (Happel & Seedorf, 2006), the issue of incorporating Bayesian networks to study uncertainty in ontologies is still at an early stage of research and development.

The work of the Uncertainty Reasoning for the World Wide Web Incubator Group (URW3-XG) created by the World Wide Web Consortium (W3C), provided an important study (Laskey & Laskey, 2008) that characterized the range of uncertainty that affects reasoning on the scale of the World Wide Web, and the issues to be considered in designing a standard representation of that uncertainty. However, the report concludes that the work to date likely falls short of what would be needed to charter an effort to develop that representation. The group developed an Uncertainty ontology and proposed future work on how to extend it. Furthermore, the report identified the need for a representation for uncertainty models but it was beyond the scope of the current effort of URW3-XG to decide whether extensions to existing Semantic Web languages (e.g. OWL, RDFS, RIF) will be sufficient or whether new representation standards will be needed.

A candidate representation for uncertainty reasoning in the semantic web is Probabilistic OWL (PR-OWL) (Carvalho et al., 2010), an OWL upper ontology for representing probabilistic ontologies based on Multi-Entity Bayesian Networks (MEBN). A major design goal for PR-OWL was to achieve compatibility with OWL. However, this goal has not been achieved as yet, due to the lack of mapping in PR-OWL to properties of OWL. Moreover, although PR-OWL has the concept of meta-entities, which allows the definition of complex types, it lacks compatibility with existing types already present in OWL.

OWL has been extended with fuzzy set theory (Stoilos, Simou, & Stamou, 2006), which is a mathematical framework for covering vagueness, thus getting fuzzy OWL (f-OWL). The authors investigated several issues that arise from such an extension. More precisely, DL SHOIN was extended, in order to provide reasoning for f-OWL, present a mapping from f-OWL entailment to f-SHOIN satisfiability and at last provide a preliminary investigation on querying capabilities for f-SHOIN ABoxes. However this work is still ongoing as there is no support for ontology editors to effectively take advantage of this mathematical framework.

BayesOWL (Ding et al., 2004; Pan et al., 2005) is another extension to OWL that provides a set of rules and procedures for direct translation of an OWL ontology into a BN directed acyclic graph (DAG) and a method based on iterative proportional fitting procedure (IPFP) that incorporates available probability constraints when constructing the conditional probability tables (CPTs) of the BN. The translated BN, which preserves the semantics of the original ontology and is consistent with all the given probability constraints, can support ontology reasoning, both within and across ontologies as Bayesian inferences. A representation in OWL of probability information concerning the entities and relations in ontologies is also proposed. If ontologies are translated to BNs, then concept mapping between ontologies can be accomplished by evidential reasoning across the translated BNs. This approach to ontology mapping is seen to be advantageous to many

[3] http://en.wikipedia.org/wiki/Anti-pattern, http://c2.com/cgi/wiki?AntiPatternsCatalog, http://blogs.msdn.com/nickmalik/archive/2006/01/03/508964.aspx, http://blogs.msdn.com/nickmalik/archive/2006/01/19/PMAntipattern-Pardon-My-Dust.aspx, http://www.stevenlist.com/blog/.

existing methods in handling uncertainty in the mapping. However, this preliminary work has also not been implemented at a technical level in an ontology editing environment.

Zheng and Kim (2008) described Clinical Practice Guidelines with an ontology that contains uncertainty features, and proposed an algorithm which uses these features to construct the structure and the conditional probability tables of Bayesian networks. However, the proposed approach requires the creation of an ontology that provides the necessary properties to generate the BN and does not consider the usage of already existing domain ontologies.

Finally, Sadeghi, Barzi, and Smith (2005) use the descriptive information captured in an ontology to construct Bayesian decision models. The assumption behind this approach is that the underlying ontology is specifically created to model the considered decision problem and that one hypothesis can explain the states of evidence observed in the domain. The authors describe their approach in terms of building the ontology, implementing the BN model and acquiring the knowledge to build the knowledge base. However, no detailed description regarding the ontology-based derivation of the Bayesian network is provided.

Fenz, Tjoa, and Hudec (2009) developed the Protege BNTab plugin that enables users to generate Bayesian networks based on existing ontologies. Classes and/or individuals are converted to Bayesian network nodes, and the properties of the ontology are used to link the nodes. Further features include the specification of node weights and the automatic incorporation of existing findings into the Bayesian network. As BNTab is the only available implementation that generates Bayesian networks based on ontologies, we use it for the generation of antipattern Bayesian networks (Settas, Cerone, & Fenz, 2011).

## 3. Bayesian belief networks

The framework proposed in this paper generates semi-automatically BN models of antipatterns and antipattern attributes. Therefore it is important to introduce Bayesian networks and their basic properties, as the models that are produced have to be compliant with BN theory. Bayesian Belief Networks are a suitable formalism that can build models of domains with inherent uncertainty (Jensen, 2001). BNs are causal networks that consist of a set of variables and a set of directed links between variables and provide the means to structure a situation for reasoning under certainty (Jensen, 2001). Each variable has a finite set of mutually exclusive states. Furthermore, to each variable $A$ with influencing variables (parents) $B_1, \ldots, B_n$, there is attached the potential table $\Pr(A|B_1, \ldots, B_n)$. BNs represent causal relations between different events. The directions of the graph indicate the direction of the impact. Causal networks can be used in order to follow how a change of certainty in one variable may change the certainty for other variables. Formally, the relation between the two nodes is based on Bayes' rule

$$\Pr(B|A) = \frac{\Pr(A|B)\Pr(B)}{\Pr(A)} \quad (1)$$

This paper proposes the application of ontology-based generation of Bayesian networks (Fenz et al., 2009) in the domain of antipatterns. The methodology uses ontology concepts to create the nodes (variables) of the Bayesian network. In order to create cause and effect relationships, the method uses ontology relations in order to link the Bayesian network nodes. Finally the ontological knowledge base is exploited in order to calculate the conditional probability table calculation for each node of the network. This process is described in detail in Section 5.

It is important to understand the d-separation properties of a BN because the proposed ontology-based antipattern BN models have to correspond with an accurate perception of the world. The way that these properties are reflected in the BN model generated by the antipattern OWL ontology using BNTab, is described in Section 5. A, B, and C are examples of variables that have a number of states. In a causal network a variable represents a set of possible states of affairs. If variables $A$ and $B$ are $d$-separated, then changes in the certainty of $A$ have no impact on the certainty of $B$. If variables $A$ and $B$ are not $d$-separated, they are referred to as $d$-connected.

In a serial connection (Fig. 1) variable $A$ has an influence on $B$, which in turn has an influence on $C$. Evidence on $A$ will influence the certainty of $B$, which then influences the certainty of $C$. Similarly, evidence on $C$ will influence the certainty of $A$ through $B$. On the other hand, if the state of B is known, then the channel is blocked resulting to $A$ and $C$ becoming independent. $A$ and $C$ are $d$-separated given B, and when the state of a variable is known, then it is instantiated. It can be concluded that evidence may be transmitted through a serial connection unless the state of the variable in the connection is known (Jensen, 2001).

In a diverging connection (Fig. 2), influence can pass between all the influenced variables (children) of $A$ unless the state of $A$ is known (Jensen, 2001). Therefore $B$, $C$ and $E$ are $d$-separated given $A$.

The converging connection (Fig. 3) requires a little more care. If nothing is known about A except what may be inferred from knowledge of its influencing variables (parents) $B$, $C$ and $E$, then the parents are independent. This means that evidence on one of them has no influence on the certainty of the others. If A changes certainty, it allows communication between its parents. The conclusion is that evidence may only be transmitted through a converging connection if either the variable in the connection or one of its descendants has received evidence (Jensen, 2001). These three cases cover all ways in which evidence may be transmitted through a variable.

Ziv and Richardson have identified that BNs can be used for modeling uncertainty in software engineering because the graphical structure of BNs is consistent with that of software systems (Ziv & Richardson, 1997). Furthermore, BNs are able to reflect dynamic changes in a software system by means of Bayesian belief updating, which can be carried out automatically by software tools (Druzdzel, 2010). To use Bayesian networks, one must first specify prior belief values for network variables. Ideally, prior belief values are determined by collecting empirical, statistical data (Ziv & Richardson, 1997). However, Bayesian belief values can also be elicited from a domain expert who subjectively assesses them. Subsequent changes to belief values in the network are caused by new evidence, through Bayesian updating. There are numerous commercial tools that enable users to build BN models and run the program calculations. In this paper, BNTab is used in order to generate antipattern BN models which can then be further explored or modified using commercial or freely available BN editors.

## 4. Uncertainty in the antipattern ontology collection process

As already mentioned the antipattern OWL ontology is the knowledge base of SPARSE, the intelligent system that can detect antipatterns. A complete description of SPARSE (Settas et al., 2011) is outside the scope of this paper. However, a short description of the antipattern OWL ontology and its data collection process is necessary in order to understand why SPARSE needs to take into account probabilistic information and how it can benefit from such framework. The antipattern ontology documents antipatterns and how they are related with other antipatterns through


**Fig. 1.** Serial variable connection.

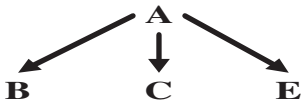


**Fig. 2.** Diverging variable connection.

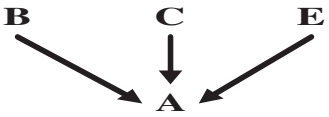


**Fig. 3.** Converging variable connection.

their causes, symptoms and consequences. The semantic relationships that derive from the antipattern definitions are determined using the Pellet DL reasoner and they are transformed into the COOL language of the CLIPS production rule engine. The purpose of this transformation is to create a compact representation of the antipattern knowledge, enabling a set of object-oriented CLIPS production rules to run and retrieve antipatterns relevant to some initial symptoms.

### 4.1. The antipattern OWL ontology

SPARSE provides an initial top-level antipattern OWL ontology as a starting point for defining antipatterns. The ontology originally consisted of seven concepts, 21 roles (19 object and 2 datatype roles), 192 individuals and seven SWRL rules. The current DL expressivity is $\mathcal{ALCHN}(\mathcal{D})$, that is, $\mathcal{ALC}$ with role hierarchies, cardinality restrictions and datatype properties. The ontology can be extended, both with OWL constructs and SWRL rules, according to user requirements. In this section we describe the default semantic capabilities that were originally provided through the ontology. Section 5 describes the probabilistic ontological extensions that were carried out in order to semi-automatically generate BN models by using BNTab.

### 4.2. Concept definitions

The antipattern ontology consists of seven concepts. In addition to the intuitive `Antipattern` concept, we have included three more concepts as subclasses of the `Antipattern` concept in order to demonstrate the way the ontology hierarchy can be extended with custom concepts according to managers' needs. In the rest of this section we analyze the concept definitions and semantics using the DL syntax.

The `Cause` concept is used in order to define the causes of the ontology. It has been defined as the subclass of the intersection of three universal role restrictions (see Section 4.3 for more details about the roles of the ontology):

$$\texttt{Cause} \sqsubseteq = \forall \texttt{causeToCause.Cause} \sqcap \forall \texttt{causeToSymptom.Symptom} \sqcap \forall \texttt{causeToConsequence.Consequence}$$

In that way, for a `Cause` instance, all of its values in the `causeToCause` role belong to the `Cause` concept, all of its values in the `causeToSymptom` role belong to the `Symptom` concept and all of its values in the `causeToConsequence` role belong to the `Consequence` concept.

The `Symptom` concept is used in order to define the symptoms of the ontology. It has been defined as the subclass of the intersection of four universal role restrictions:

$$\texttt{Symptom} \sqsubseteq = \forall \texttt{symptomToCause.Cause} \sqcap \forall \texttt{symptomToSymptom.Symptom} \sqcap \forall \texttt{symptomToConsequence.Consequence} \sqcap \geqslant 1\ \texttt{symptomToConsequence}.\ \top$$

The definition is similar to the `Cause` concept, apart from an additional restriction on the `symptomToConsequence` role that defines the existence of at least one value in the role.

The `Consequence` concept is used in order to define the consequences of the ontology and it has been defined as the subclass of a single universal restriction:

$$\texttt{Consequence} \sqsubseteq = \forall \texttt{consequenceToConsequence.Consequence}$$

The `Antipattern` concept is the root concept of the antipattern hierarchy and is defined in terms of at least one cause, symptom and consequence instance values in the corresponding roles:

$$\texttt{Antipattern} \sqsubseteq = \forall \texttt{hasCause.Cause} \sqcap \forall \texttt{hasSymptom.Symptom} \sqcap \forall \texttt{hasConsequence.Consequence} \sqcap \geqslant 1\ \texttt{hasCause}.\ \top \sqcap \geqslant 1\ \texttt{hasSymptom}.\ \top \sqcap \geqslant 1\ \texttt{hasConsequence}.\ \top$$

The other three antipattern-related concepts have been defined directly as subclasses of the `Antipattern` concept, that is,

$$\texttt{SoftwareDevelopment} \sqsubseteq \texttt{Antipattern}$$
$$\texttt{SoftwareArchitecture} \sqsubseteq \texttt{Antipattern}$$
$$\texttt{SoftwareProjectManagement} \sqsubseteq \texttt{Antipattern}$$

It is important at this point to mention that the concept and instance classification, as well as the consistency checking, are performed by the Pellet DL reasoner. Therefore, the open-world semantics are followed, that is, unstated information does not necessarily mean negated information. For that reason, the role restrictions we have described for the concept definitions do not act as constraints, in terms of database constraints. For example, it is valid to define an instance of the concept `Symptom` without specifying a `symptomToConsequence` value or an antipattern instance without a `hasSymptom` value, since the reasoner assumes that such a value exists but has not been stated yet (open world). However, all the values in the `symptomToConsequence` role of a symptom instance will be classified in the `Consequence` concept (necessary condition).

### 4.3. Role definitions

The ontology roles allow the definition of basic knowledge related to antipattern causes, symptoms and consequences, as well as to their correlations. Table 1 depicts a summary of the defined roles. In the rest of this section we describe in detail the definition of each role using the OWL RDF/XML syntax.

#### 4.3.1. Documentation roles

The ontology defines two datatype roles for providing human-readable textual descriptions for ontology instances. The `title` role can be used in order to define a short title for an instance and the `description` role can be used in order to provide a

detailed documentation. Intuitively, the title role can be used as the human-readable description of an ontology instance rdf:ID. Both roles are necessary for the successful interaction of the project manager with the interface of SPARSE. For that reason, SPARSE informs the user about the absence of values for any of the two roles for an instance..

```
⟨owl:DatatypeProperty rdf:ID="title"⟩
  ⟨rdfs:range rdf:resource="& xsd;string"/⟩
⟨/owl:DatatypeProperty⟩
⟨owl:DatatypeProperty rdf:ID="description"⟩
  ⟨rdfs:range rdf:resource="& xsd;string"/⟩
⟨/owl:DatatypeProperty⟩
```

*4.3.2. Cause, symptom and consequence object roles*

The antipattern ontology allows the definition of correlations among causes, symptoms and consequences. In that way, the ontology reasoning procedures, that is, the Pellet DL reasoner and the SWRL rules, are able to infer correlations that are not explicitly stated in the ontology, exploiting the antipattern knowledge that has been created by different managers. In this section, for simplicity, we describe only the roles that correlate a cause with a cause, a symptom and a consequence.

The causeToCause object role allows the correlation of a cause with another cause. In that way, there is no need to state explicitly all the causes of a specific antipattern. The ontology reasoning procedure through the SWRL rules that we describe in Section 4.5, is able to infer all the relevant (implicit) causes for a specific antipattern following the causeToCause relations.

```
⟨owl:ObjectProperty rdf:about="#causeToCause"⟩
  ⟨rdfs:domain rdf:resource="#Cause"/⟩
  ⟨rdfs:range rdf:resource="#Cause"/⟩
⟨/owl:ObjectProperty⟩
```

The causeToSymptom object role allows the correlation of a cause with a symptom. The ontology reasoning procedure on this role allows the inference of additional symptoms for a specific antipattern, following the cause-ToSymptom relationships of asserted or inferred causes.

**Table 1**
The roles of the antipattern ontology.

| Role | Description |
|---|---|
| title | The title of a resource |
| description | The description of a resource |
| causeToCause | Correlates two causes |
| causeToConsequence | Correlates a cause with a consequence |
| causeToSymptom | Correlates a cause with a symptom |
| symptomToSymptom | Correlates two symptoms |
| symptomToCause | Correlates a symptom with a cause |
| symptomToConsequence | Correlates a symptom with a consequence |
| consequenceToConsequence | Correlates two consequences |
| hasCause | Defines a cause for an antipattern |
| hasPrimaryCause | Defines a primary cause |
| hasSecondaryCause | Defines a secondary cause |
| hasImplicitCause | Defines an implicit cause |
| hasSymptom | Defines a symptom for an antipattern |
| hasPrimarySymptom | Defines a primary symptom |
| hasSecondarySymptom | Defines a secondary symptom |
| hasImplicitSymptom | Defines an implicit symptom |
| hasConsequence | Defines a consequence for an antipattern |
| hasPrimaryConsequence | Defines a primary consequence |
| hasSecondaryConsequence | Defines a secondary consequence |
| hasImplicitConsequence | Defines an implicit consequence |

```
⟨owl:ObjectProperty rdf:about="#causeToSymptom"⟩
  ⟨rdfs:domain rdf:resource="#Cause"/⟩
  ⟨rdfs:range rdf:resource="#Symptom"/⟩
⟨/owl:ObjectProperty⟩
```

The causeToConsequence object role allows the correlation of a cause with a consequence. Similar to the previous role, the ontology reasoning procedure on this role allows the inference of additional consequences for a specific antipattern, based on asserted or inferred causes.

```
⟨owl:ObjectProperty
  rdf:about="#causeToConsequence"⟩
  ⟨rdfs:domain rdf:resource="#Cause"/⟩
  ⟨rdfs:range rdf:resource="#Consequence"/⟩
⟨/owl:ObjectProperty⟩
```

Similar object roles to the above are also used for describing symptoms. In this paper we assume that for consequences, the antipattern ontology defines only the consequenceToConsequence role, since SPARSE uses a symptom based approach and also the relationships between consequences and symptoms are better described through the symptoms of an antipattern.

*4.3.3. Antipattern roles*

The hasCause, hasSymptom and hasConsequence roles are used to define the causes, symptoms and consequences of an antipattern, respectively. Note that due to the ontology reasoning procedure, an antipattern might result with more causes, symptoms or consequences than it has been initially defined with, as we have explained in the previous section.

```
⟨owl:ObjectProperty rdf:about="#hasCause"⟩
  ⟨rdfs:domain rdf:resource="#Antipattern"/⟩
  ⟨rdfs:range rdf:resource="#Cause"/⟩
⟨/owl:ObjectProperty⟩
⟨owl:ObjectProperty rdf:about="#hasSymptom"⟩
  ⟨rdfs:domain rdf:resource="#Antipattern"/⟩
  ⟨rdfs:range rdf:resource="#Symptom"/⟩
⟨/owl:ObjectProperty⟩
⟨owl:ObjectProperty rdf:about="#hasConsequence"⟩
  ⟨rdfs:domain rdf:resource="#Antipattern"/⟩
  ⟨rdfs:range rdf:resource="#Consequence"/⟩
⟨/owl:ObjectProperty⟩
```

For each one of the above roles, the ontology defines three subproperties of the form hasPrimaryX, hasSecondaryX and has-ImplicitX, with X being Cause, Symptom or Consequence. The values of the properties of the first two types are explicitly stated in the ontology, whereas the values of the properties of the third type derive after ontology reasoning. For example, the subproperties of the hasSymptom role are defined as:

```
⟨owl:ObjectProperty rdf:ID="hasPrimarySymptom"⟩
  ⟨rdfs:subPropertyOf rdf:resource="#hasSymptom"/
  ⟩
⟨/owl:ObjectProperty⟩
⟨owl:ObjectProperty rdf:ID="hasSecondarySymptom"⟩
  ⟨rdfs:subPropertyOf rdf:resource="#hasSymptom"/
  ⟩
```

```
⟨/owl:ObjectProperty⟩
⟨owl:ObjectProperty rdf:ID="hasImplicitSymptom"⟩
  ⟨rdfs:subPropertyOf rdf:resource="#hasSymptom"/
  ⟩
⟨/owl:ObjectProperty⟩
```

The purpose of the subproperty relationships is to allow an additional level of granularity during the definition of the antipattern's causes, symptoms and consequences that is used by the underlying antipattern retrieval mechanism to sort the results. For example, the antipatterns that match a specific symptom in their `hasPrimarySymptom` role are considered more relevant than the antipatterns that match the same symptom in their `hasSecondarySymptom` role. The implicit subproperties are used by the SWRL rules in order to insert the values they infer.

### 4.4. OWL ontology data collection

Using SPARSE, an antipattern can be categorized in any or all three different types of antipatterns: development, architecture or management. In the dataset used in the antipattern ontology there exist a total of 65 software project management antipatterns, nine development antipatterns and four architectural antipatterns. This implies that some antipatterns belong to two or more categories. Using SPARSE, an antipattern can be categorized in any or all three different types of antipatterns: development, architecture or management. Table 2 contains the sources of antipatterns used in the OWL ontology. The most popular antipattern repositories that exist on the Web at the moment are the Wikipedia Antipatterns Web page (Wiki-Community, & antipatterns community catalogue, xxxx), the Pattern Community Antipattern Cataloge (Wiki-Community & community antipattern catalogue, xxxx) and software project management blogs (Malik, 2006a, 2006b). The remaining antipatterns are provided by the main books on software project management antipatterns (Brown et al., 1998; Brown et al., 2000; Laplante & Neil, 2006). However, due to copyright restritions these antipatterns are not included in the ontology that is freely available on the Web and have been analyzed for research purposes only. Each antipattern is associated with at least one symptom and the total number of symptoms that exist in the ontology is 204. Also there exist 65 causes and 51 consequences. A BN model can be generated for any attribute of these antipatterns. For the purposes of this paper, a BN model has been generated based on 13 attributes of antipatterns that exist on the Web.

**Table 2**
On-line sources of antipatterns used in the antipattern OWL ontology.

| Source | Antipattern | Quantity |
|---|---|---|
| Wikipedia Antipatterns | Death March, Groupthink, Smoke and mirrors, Software bloat | 4 |
| Pattern Community | Analysis Paralysis, Architectre by | 25 |
| Wiki Antipattern Cataloge | Implication, An Athena, Appointed Team, Architects Dont Code, Blowhard Jamboree, Corn Cob, Carbon Copy His Manager, Death By Planning, Decision By Arithmetic, Dry Waterhole, Egalitarian Compensation, Email Is Dangerous, Emperors New Clothes, Fear Of Success, Glass Wall, Leading Request, Shoot The Messenger, Smoke And Mirrors, The BLOB, The Customers Are Idiots, Thrown Over The Wall, Train The Trainer, Untested But Finished, Yet Another Meeting Will Solve It | |
| Software Project Management Blogs | Pardon my dust, Project Managers who write specs | 2 |

### 4.5. SWRL Rules

SPARSE comes with a set of seven SWRL rules that are used by the Pellet DL reasoner in order to infer new antipattern causes, symptoms and consequences based on their correlations. Fig. 4 presents the set of the SWRL rules. Note that all inferred values are inserted into the corresponding implicit role. We follow such an approach since the explanation mechanism of SPARSE needs to know whether a value has been derived after ontology reasoning or it has been explicitly stated in the ontology, in order to generate the appropriate explanation message.

To exemplify, consider an ontology that defines an antipattern `antip` to have the symptom `syml` ( `hasSymptom (antip, syml)`) and that symptom `syml` is correlated with symptom `sym2` (`SymptomToSymptom (syml, sym2)`). By incorporating the SWRL rules in the ontology reasoning procedure, we are able to infer also that antipattern `antip` has the (implicit) symptom `sym2` (rule 7 in Fig. 4). Such type of reasoning is very useful since it enables the collaborative definition of antipattern relationships, drawing conclusions using relationships that have been modeled by different project managers, without requiring from them to be aware of the complete domain knowledge.

It is worth mentioning the OWL two language (Grau et al., 2008), an extension to OWL that allows the SWRL rules of SPARSE to be expressed directly as *property chains*. For example, the rule 7 of Fig. 4 is actually a property chain of the form

`hasSymptom o symptomToSymptom → hasImplicitSymptom`

that can be expressed in OWL 2 as

```
⟨rdf:Description⟩
  ⟨rdfs:subPropertyOf
  rdf:resource="#hasImplicitSymptom"/⟩
  ⟨owl:propertyChain rdf:parseType="Collection"⟩
    ⟨rdf:Description rdf:about="#hasSymptom"/⟩
    ⟨rdf:Description
  rdf:about="#symptomToSymptom"/⟩
  ⟨/owl:propertyChain⟩
⟨/rdf:Description⟩
```

## 5. Antipattern ontology-based BN generation

In this section we describe the process of extending the antipattern OWL ontology, in order to allow the semi-automatic generation of antipattern BN models. As already mentioned, the ontology acts as the knowledge base that enriches the BN with probabilistic knowledge. As a result, appropriate extensions need to be made in ontology concepts and roles using the OWL language and the Protege ontology editor. Finally the process of creating a BN model using the antipattern ontology is exemplified using an example software project management antipattern.

The process of using an ontology in order to generate a BN model (Fenz et al., 2009) is based on the following steps:

- A direct mapping of ontology classes and/or individuals of interest to BN nodes. Using BNTab, the ontology classes and/or individuals, which are relevant to the considered problem and should be represented in the Bayesian calculation schema are selected in order to establish the nodes of the Bayesian network.

```
1) hasSymptom(?anti, ?s) ∧ symptomToConsequence(?s, ?con)
    → hasImplicitConsequence(?anti, ?con)
2) hasSymptom(?anti, ?s) ∧ symptomToCause(?s, ?c)
    → hasImplicitCause(?anti, ?c)
3) hasCause(?anti, ?c1) ∧ causeToCause(?c1, ?c2)
    → hasImplicitCause(?anti, ?c2)
4) hasConsequence(?anti, ?c1) ∧
                      consequenceToConsequence(?c1, ?c2)
    → hasImplicitConsequence(?anti, ?c2)
5) hasCause(?anti, ?c) ∧ causeToSymptom(?c, ?s)
    → hasImplicitSymptom(?anti, ?s)
6) hasCause(?anti, ?c) ∧ causeToConsequence(?c, ?con)
    → hasImplicitConsequence(?anti, ?con)
7) hasSymptom(?anti, ?s1) ∧ symptomToSymptom(?s1, ?s2)
    → hasImplicitSymptom(?anti, ?s2)
```

**Fig. 4.** The SWRL rules of SPARSE.

- Ontology properties starting and ending between the selected classes and/or individuals are used to establish the cause and effect relationships between the Bayesian network nodes.
- Scale and weight relevant axioms are added to the ontology in order to determine potential states and weights of the Bayesian network nodes.
- Individuals of ontology classes and/or individuals which are represented by the Bayesian networks leaf nodes are used to derive and enter concrete findings in the Bayesian network.

Based on the existing antipattern ontology, the user has to select those classes and/or individuals which are relevant to the considered problem and should be represented by Bayesian network nodes. The following stage requires the user to configure the relationships between the BN nodes by selecting the ontological properties of interest. Each relation is checked for correctness to ensure that the considered property starts and ends at a class and/or individual, which has been selected in the previous phase. If this is true for the considered relation it can be used to connect the corresponding Bayesian network nodes. The potential relations can be derived automatically from the ontology, however the link direction (i.e. the determination of parent and child nodes) requires the human interpretation of the ontological property. After the nodes and their cause and effect links have been selected, the next phase assigns each node an appropriate scale of potential categorical values and its weight (if available) in the context of its siblings to support the conditional probability table generation.

In order to determine potential node scales requires the user has to define a new ontology class which defines the node scale. In the context of the antipattern probability determination the concepts Threepointlikertscale and Boolean provide a three-point likert scale with Low, Medium and High values and a True or False scale in order to allow the assignment of probabilistic antipattern information to antipatterns and antipattern attributes which are the causes, symptoms and consequences of antipatterns.

In the next step, the formal axioms of each high-level concept that was defined in the first phase can be queried. If such a concept has been found, the considered node can be equipped with the given scale (e.g. high, medium, low or true and false). After setting up the nodes, node links, and node scales the conditional probability tables (CPTs) for each node have to be calculated to finalize the Bayesian network model structure. A mathematical function is required which describes how the state of each node is influenced by the state of its parent nodes. At the moment, this knowledge is not provided by the antipattern ontology. The ontology provides the weight of each node in the context of its siblings. In the case of antipatterns and antipattern attributes the antipattern ontology provides an occurence rating for each antipattern or antipattern attribute (e.g. high, medium, and low).

As we are using boolean nodes we calculate their CPT by the following generic function:

$$P(N|X_1,\ldots,X_n) = (X_1/h_{X_1} + \ldots + X_n/h_{X_n})/n \quad (2)$$

Because, node-specific mathematical functions are not provided by the antipattern ontology, Formula 2 is used for the CPT calculation.

After creating a BN model that includes relevant nodes, node links, node scales, and node weights supporting the CPT calculation, the ontological knowledge needs to be exploited in order to provide concrete findings to the BN model. After defining the input nodes of interest, their individuals (instances) are selected to provide concrete findings to the Bayesian network.

### 5.1. Probabilistic extensions to the antipattern OWL ontology

A total of 18 new OWL ontology constructs were created in order to allow BN model generation for antipatterns, antipattern causes, symptoms and consequences. In this section we describe in detail the definition of each role using the OWL RDF/XML syntax.

More specifically, six new ontology concepts and 12 roles (object type) were required to make the appropriate extensions to the antipattern ontology in order to allow BN model generation using BNTab.

The `AntipatternAprioriProbability` concept is used in order to define the different instances of antipatterns that have an associated probability value. The object property assertions of this concepts is the specific antipattern itself and its occurence, which can be of type ThreepointLikertScale (low,medium or high).

Similarly the concepts `CauseAprioriProbability`, `SymptomAprioriProbability` and `ConsequenceAprioriProbability` were added in order to define the instances of antipattern attributes which have an associated probability value. Simarly to the `AntipatternAprioriProbability`, the concepts of the antipattern attributes also have two new object property assertions which define a specific attribute and its occurence measured with a probabilistic value scale.

The other two concepts required for BN model generation were the ontology concepts `Scale` and `ThreepointLikertscale` added to the ontology in order to determine potential states and

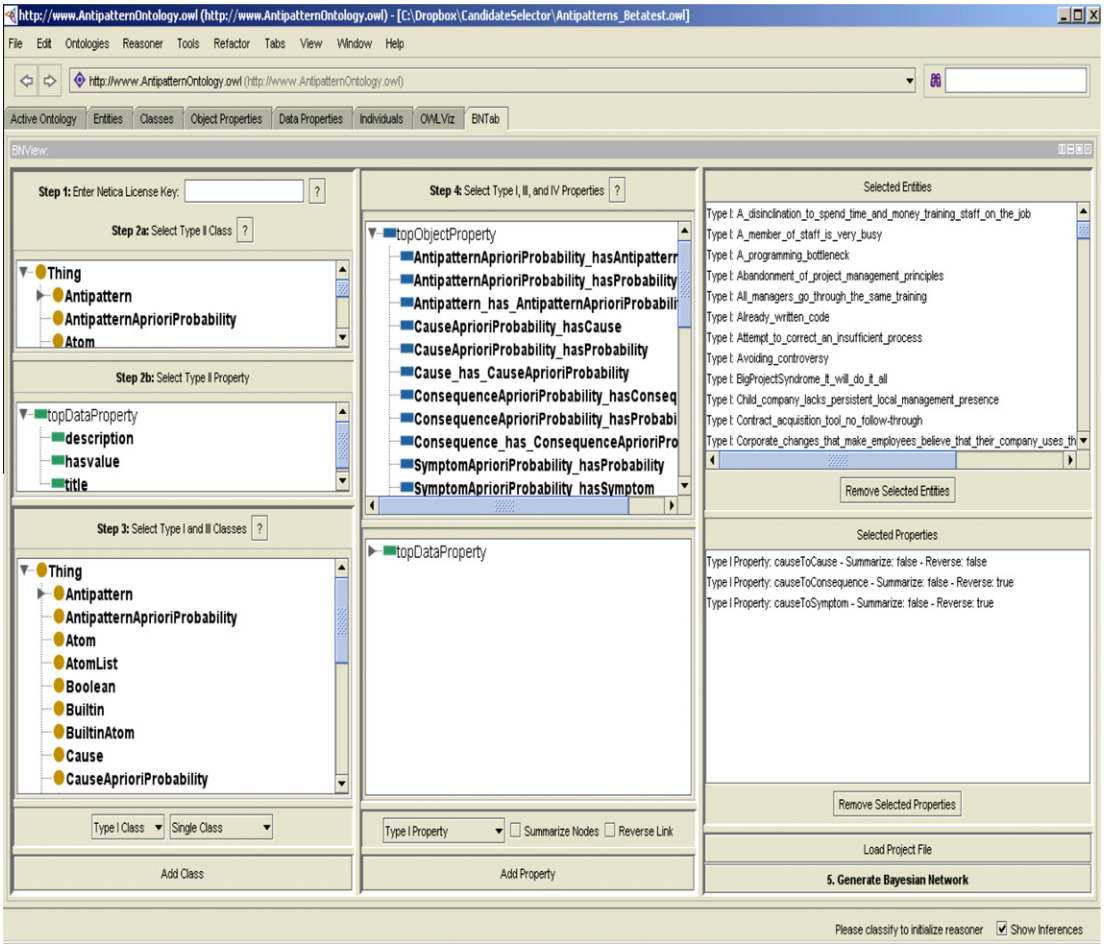


**Fig. 5.** Creating antipattern BN models using BNTab.

weights of the Bayesian network nodes. The boolean scale can be used to declare the occurence of an antipattern or antipattern attribute as True or False values, while the three point likert scale provides a Low, Medium and High scaling of occurence values. The ontology can be extended to handle different scales according to different user needs.

### 5.1.1. Antipattern attribute object role probabilistic extensions

As mentioned in Section 4, the ontology roles allow the definition of basic knowledge related to antipattern causes, symptoms and consequences, as well as to their correlations. The antipattern ontology allows the definition of correlations among causes, symptoms and consequences. Similarly, in this paper we extend these correlations with probabilistic information. In that way, the ontology reasoning procedures, that is, the Pellet DL reasoner and the SWRL rules, will be able in the future to infer correlations that are not explicitly stated in the ontology, exploiting the probabilistic antipattern knowledge that has been created by different managers.

The `AntipatternAprioriProbability_hasAntipattern` object role allows the correlation of an antipattern with an AntipatternAprioriProbability instance which aims to declare that a specific antipattern has associated probabilistic information on its occurrence. Similarly, the `Antipattern_hasAntipatternAprioriProbability` is a role of the antipattern concept and correlates an antipattern with specific probabilistic information of AntipatternAprioriProbability members.

The `AntipatternAprioriProbability_hasProbability` object role correlated an AntipatternAprioriProbability instance with probabilistic information. This information combined with the previous roles can be used to correlate an antipattern with specific probabilities of occurence according to the chosen scale concept.

The same roles have been defined for the cause, symptoms and consequences antipattern attributes. The resulting roles can associate any antipattern attribute of the ontology with probabilistic information that can be used by BNTab in order to generated a BN model in a semi-automated manner.

## 5.2. Example ontology-based BN model generation

This section exemplifies the semi-automatic creation of an antipattern BN model based on knowledge described in the antipattern ontology. After installing the BNTab plugin in the Protege ontology editor and installing the required Netica BN editing software, the first step required in the BN model creation process is to choose the Boolean scale value which will allow the BN model to display the possibility that an antipattern attribute exists with an associated "True" and "False" percentage. Then the property "hasvalue" is chosen as this property correlates any chosen ontology construct with a specific value, in this example a value from "0" to "1" indicates if possibility of occurence or existence of an antipattern attribute with "1" being "True" and "0" being "False". In the next step, ontology individuals with probabilistic information will be added

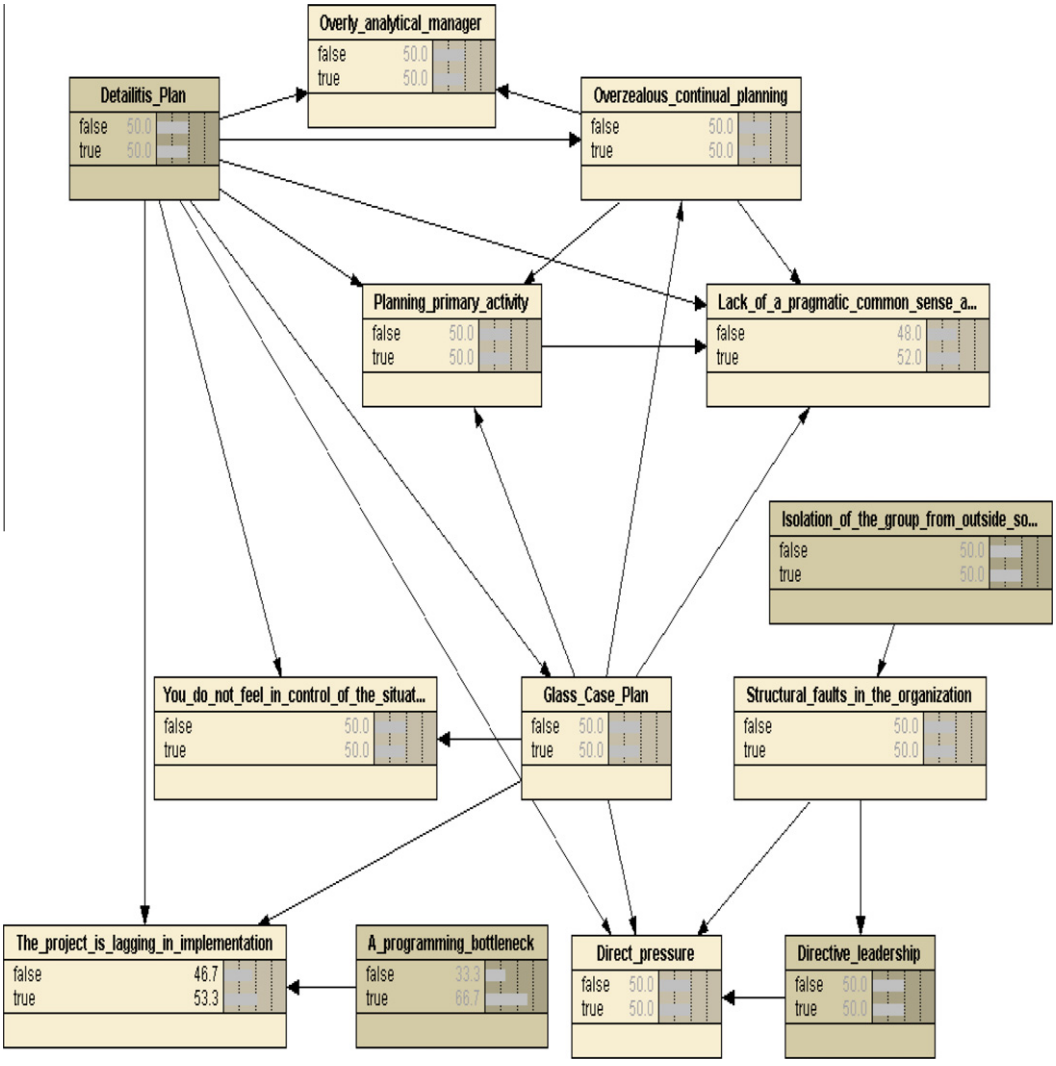


**Fig. 6.** Unprocessed BN model containing raw data values.

to the BN model. These are referred to as "classes" in BNTab. In the example of Fig. 5, the classes "Cause", "Symptom" and "Consequence" are chosen in order to create the BN model. The next step is to select the specific individuals of those classes that the user is interested in modeling as a BN. Fig. 6 illustrates the 13 attributes that were chosen for this specific example. The user has to select the corresponding properties to correlate the chosen attributes to the probabilistic knowledge contained in the ontology regarding those antipatterns. This is carried out by selecting from ten types of properties. These properties are include the object roles additions that were described in the previous sub-section. The property `Antipattern_hasAntipatternAprioriProbability` can be selected as it correlates an antipattern with specific probabilistic information of `AntipatternAprioriProbability` concept members. As already mentioned, the `AntipatternAprioriProbability` concept defines the different instances of antipatterns that have an associated probability value. The object property assertions of this concepts is the specific antipattern itself and its occurence, which can be of type ThreepointLikertScale (low,medium or high). The next property that can be selected is a probabilistic property, which is referred to as type-IV in BNTab. This property is the `AntipatternAprioriProbability_hasProbability` object role that correlates an AntipatternAprioriProbability instance with probabilistic information. The other seven properties correlate antipattern attributes and transform their relatioships to cause and effect relationships. For the example of Fig. 6 the properties `CauseToCause`, `CauseToConsequence` and `CauseToSymptom` were used in order to correlate the nodes of the BN model with relationships that define how the causes of a specific set of antipatterns are connected through their cause, symptom and consequence attributes.

By selecting "Generate Bayesian Network", the plug-in launches Netica BN editor and displays the resulting antipattern attribute BN model. The user can then save the file as a .DNE file and process it later on with any BN editing software. The resulting unprocessed BN model (Fig. 6) contains 13 nodes and 20 cause and effect relationships.

After compiling the BN model using Netica or any other BN editing software that is compatible with the .dne file type, the values of nodes are propagated through relationships and nodes according to their type of connection (Fig.7). It is important to examine the d-separation properties of the generated BN model because this affects the way that values are propagates and because the proposed model has to correspond with an accurate perception of the world. In this example, there exist four serial relationships. These are between the nodes "Isolation of the group from outside sources of information and analysis", "Structural faults in the organization" and "Directive leadership. Another serial relationship exists

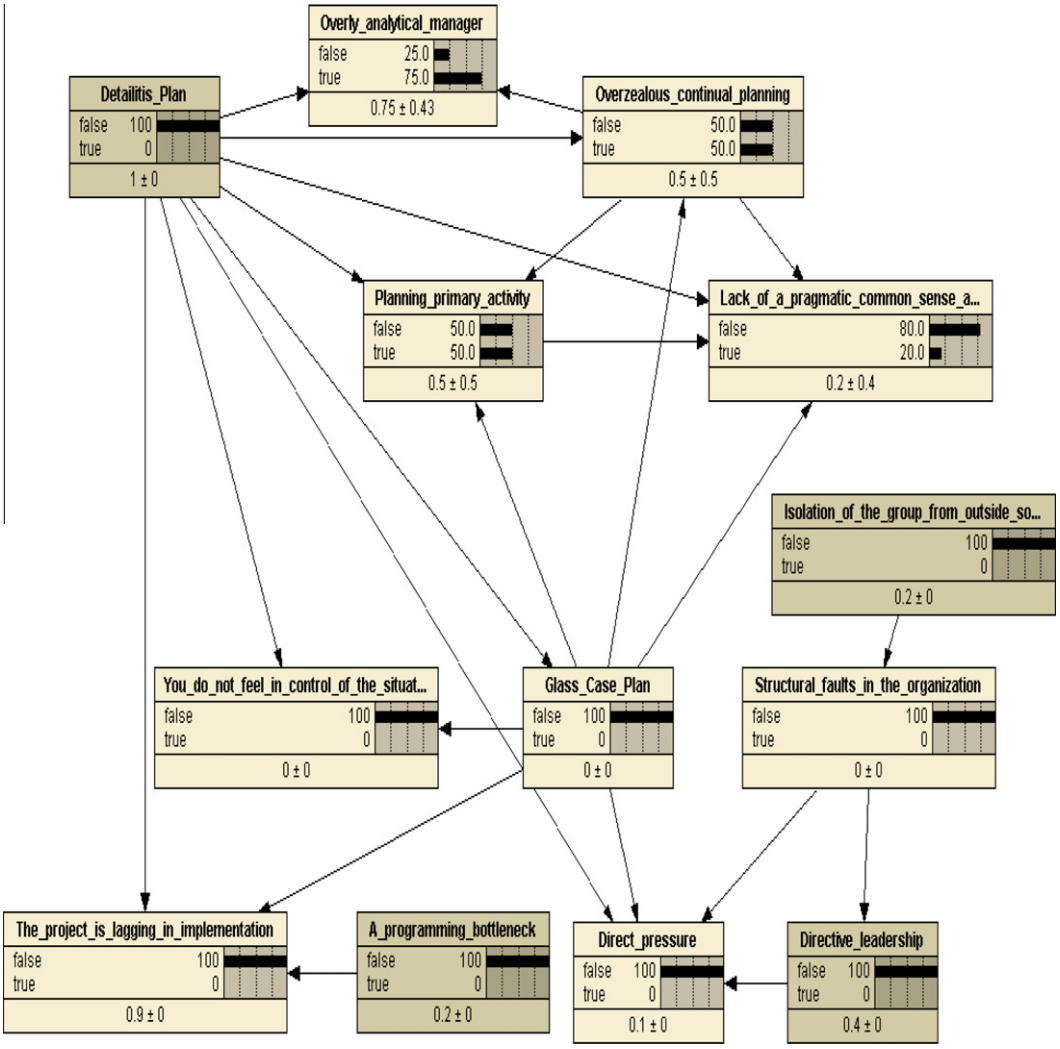


**Fig. 7.** BN model after evidence has been set to nodes.

between "Detailitis plan", "Planning primary activity" and "Lack of a pragmatic common sense approach to planning". The third serial connection exists between the nodes "Overzealous Continual Planning", "Planning primary activity" and Lack of a pragmatic common sense approach to planning". The last serial connection appears between nodes "Detailitis Plan", "Glass case plan" and "Direct Pressure". In these connections the first variable has an influence on the second, which in turn has an influence on the third. As a result, evidence on the first node influence the other nodes. However, if evidence is set on the middle node then channel is blocked resulting to the first and third node becoming independent.

Furthermore, the model contains three diverging connections. The first one exists between the nodes "Detailitis plan", "The project is lagging in implementation", "You do not feel in control of the situation", "Planning primary activity", "Overzealous continual planning" and "Over-analytical Manager". The second diverging connection exists between the nodes is between "Overzealous continual planning" and the nodes "Over-analytical Manager", "Planning primary activity" and "Lack of a pragmatic common sense approach to planning". The last diverging connection exists between the nodes "Glass Case Plan", "You do not feel in control of the situation", "The project is lagging in implementation" and "Direct Pressure". In a diverging connection, influence can pass between all the influenced variables of the parent node unless the state of the parent node is known. Therefore, setting evidence on the parent node, makes the children node independent.

The BN model also contains four converging connection, which require more attention because if no evidence is set to the child node then the parent nodes are independent. Hence evidence on one of them has no influence on the certainty of the others. If evidence is set to the child node, this allows communication between its parents. For example, a converging connection in which the child node is "Planning primary activity" and the parent nodes are "Detailitis Plan", "Overzealous continual planning" and "Glass Case Plan" illustrated that the parent nodes cannot influence each other if not evidence is set to "Planning primary activity". Other converging connections exist between the nodes "The project is lagging in implementation", "A Programming Bottleneck", "Detailitis Plan", "Glass Case Plan" and also the nodes "Direct Pressure", "Structural faults in the organization", "Glass Case Plan" and "Detailitis Plan". The last converging connection includes the nodes "Lack of a pragmatic common sense approach to planning", "Overzealous continual planning", "Detailitis Plan", "Planning primary activity" and "Glass Case Plan".

These three cases cover all ways in which evidence may be transmitted through a variable using Bayesian propagation which is automatically carried out using BN editing software. The model of Fig. 6 contains the raw data values of the BN model that are based on ontological probabilistic values. For example the node

"Isolation of the group from outside sources of information and analysis" has default values of 50% possibility of having a both a "True" and "False" existence or occurence in past projects. By setting evidence to nodes, the user can explore the results of setting probabilistic values to antipattern attributes to specific BN nodes. Using Bayesian propagation depending on the type of connection, the model will propagate influence through the nodes and will illustrate which BN nodes were affected by the set evidence. For example, using Netica (Fig. 7) a user can set the value "False = 100%" to the node "Isolation of the group from outside sources of information and analysis". This will influence the nodes "Structural Faults in the organization", "Directive Leadership" and "Direct Pressure". Similarly the remaining BN nodes will be influenced depending on their connections.

Alternatively, a BN model could be created to model the probability of occurence of an individual antipattern. The BN nodes in that case would be:

- `AntipatternAprioriProbability_hasProbability_1`. This has been generated using information of the `AntipatternAprioriProbability_hasProbability` object role and can have a Low, Medium or High value in the antipattern ontology using the "hasvalue" property.
- `AntipatternAprioriProbability_1`. This node contains a Conditional Probability Table in same categorical values as the target node as it directly affects its values. These are either "True" or "False" together with a specific percentage. These are 50% in both cases in an unprocessed model in which Bayesian updating has not been carried out.
- `Antipattern Node`. The results of this node when Bayesian updating has been propagated through the nodes, will illustrate whether the antipattern exists or not in a software project based on its occurence on past projects. This is particularly interesting as a user who whishes to know more about an antipattern or an antipattern attribute can quickly build a BN model to observe what other users have discovered regarding the occurence of this antipattern. Furthermore, this information can be used by SPARSE in order to rank antipatterns according to their occurence in past projects.

## 6. Conclusions

This paper studied the incorporation of BNs in the antipattern OWL ontology. The proposed approach incorporated probabilistic information in the ontology and allowed the semi-automatic generation of BNs using BNtab, a Protege plug-in. BNtab successfully illustrated the uncertainty that exists in antipatterns. The proposed methodology reduces the complexity of modeling Bayesian networks by using high-level concepts and relations to integrate relevant sub-concepts into the Bayesian network, and provides by the usage of ontologies the possibility of easily maintaining the underlying knowledge body of Bayesian networks.

The technology of antipatterns can benefit from this technology by using it with the web-based collaborative version (Settas et al., 2011) of the antipattern ontology. As a result, antipattern contributors will be able to produce BN models within the environment of Protege and add probabilistic information while editing or creating antipatterns. Ultimately, the SPARSE intelligent system will benefit from this approach by including associated BNs with the resulting detected antipatterns that can visualize the uncertainty of the data used for the detected antipatterns.

For the purposes of this work both a machine learning formalism and an ontological formalism were used. Bayesian modeling proved to be particularly useful and well suited to the antipattern ontology by capturing and by illustrating the cause and effect relationships between antipattern variables and by providing a solid graphical representation of the probabilistic relationships among the set of variables. This formalism allowed further advantages for example its use in cases of missing data, using expert judgement. The proposed construction model used a semi-automated BN construction process using BNtab to exemplify the approach.

Future work will redevelop SPARSE in order to take advantage of the proposed uncertainty based reasoning. Further limitations of the applied method (Fenz et al., 2009) are: (i) functions for calculating conditional probability tables are not provided by the ontology and have to be modeled externally and (ii) human intervention is still necessary if the ontology provides a knowledge model which does not exactly fit the domain of interest. Further research has to address these limitations to provide a more efficient method for creating and updating ontology-based antipattern Bayesian networks.

## References


Brown, W., Malveau, R., McCormick, H., & Mowbray, T. (1998). *AntiPatterns: Refactoring software, architectures, and projects in crisis*. Wiley Computer Publishing.

Brown, W., McCormick, H., & Thomas, S. (2000). *AntiPatterns in project management*. Wiley Computer Publishing.

Carvalho, R. N., Laskey, K. B., Costa, P. C. G. (2010). Compatibility formalization between PR-OWL and OWL. In *Proceedings of the first international workshop on uncertainty in description logics (UniDL 2010), held at the international joint conference on automated reasoning (IJCAR 2010)*.

Davis, H. R., & Szolovits, P. (1993). What is knowledge representation? *AI Magazine, 14*(1), 17–33.

Ding, Z., Peng, Y., & Pan, R. (2004). A Bayesian approach to uncertainty modelling in owl ontology. In *Proceedings of the 2004 international conference on advances in intelligent systems*, 2004.

Druzdzel, M. (2010). Genie2.0 system. Cited 26.1.10.

Fenz, S., Tjoa, A. M., & Hudec, M. (2009). Ontology-based generation of Bayesian networks. In *Proceedings of the third international conference on complex, intelligent and software intensive systems – international workshop on ontology alignment and visualization* (pp. 712–717).

Ferreiroa, S., Arnaiza, A., Sierrab, B., & Irigoienb, I. (2012). Application of Bayesian networks in prognostics for a new integrated vehicle health management concept. *Expert Systems with Applications, 39*(7), 6402–6418.

Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., & Sattler, U. (2008). OWL 2: The next step for OWL, web semantics: Science. *Services and Agents on the World Wide Web, 6*(4), 309–322.

Happel, H. J., & Seedorf, S. (2006). Applications of ontologies in software engineering. In *Proceedings of the 2nd international workshop on semantic web enabled software engineering (SWESE 2006), held at the 5th international semantic web conference (ISWC 2006)* (pp. 1–14).

Hung, E., Szeto, C. -C. Fang, W., & Deng, Y. (2009). *POWL: A multi-level approach to represent uncertainty in semantic web ontology*. Technical report, Department of Computing, Hong Kong Polytechnic University.

Jensen, F. (2001). *Bayesian networks and decision graphs*. Springer.

Kis, M. (2002). Information security antipatterns in software requirements engineering. In *Proceedings of the 9th conference on pattern language of programs (Plop)*.

Kotzé, P., Renaud, K., & van Biljona, J. (2008). Don't do this – Pitfalls in using anti-patterns in teaching human–computer interaction principles. *Computers and Education, 50*(3), 979–1008.

Krai, J., & Zemlicka, M. (2007). The most important service-oriented antipatterns. In *Proceedings of the international conference on software engineering advances (ICSEA)*.

Laplante, P., Hoffman, R., & Klein, G. (2007). Antipatterns in the creation of intelligent systems. *IEEE Intelligent Systems, 22*, 91–95.

Laplante, P., & Neil, C. (2006). *Antipatterns: Identification, refactoring and management*. Taylor and Francis.

Laskey, K. J., & Laskey, K. B. (2008). *Uncertainty reasoning for the world wide web: Report on the URW3-XG incubator group*. Technical report URW3-XG.

Liao, S.-H. (2005). Expert system methodologies and applications—a decade review from 1995 to 2004. *Expert Systems with Applications, 28*(1), 93–103.

Long, J. (2001). Software reuse antipatterns. In *ACM SIGSOFT software engineering notes* (Vol. 26, pp. 4).

Malik, N. (2006a). *Software project management antipattern blog, project managers who write specs*. <http://blogs.msdn.com/nickmalik/archive/2006/01/03/508964.aspx>.

Malik, N. (2006b). *Software project management antipattern blog, pardon my dust*. <http://blogs.msdn.com/nickmalik/archive/2006/01/19/pmantipattern-pardon-my-dust.aspx>.

Pan, R., Ding, Z., Yu, Y., & Peng, Y. (2005). A Bayesian network approach to ontology mapping. In *Proceedings of the fourth international semantic web conference*.

Richardson, M., & Domingos, P. (2003). Building large knowledge bases by mass collaboration. In *Proceedings of the 2nd international conference on Knowledge capture* (pp. 129–137).

Sadeghi, S., Barzi, A., & Smith, J. (2005). Ontology driven construction of a knowledge base for Bayesian decision models based on UMLS. *Study Health Technological Information* (116), 223–228.

Settas, D., & Cerone, A. (2011). An ontology based e-learning system using antipatterns. In Y. Cao, R. Cau, W. Nejdl (Eds.), *Proceedings of the 10th international conference on web-based learning (ICWL)* (pp. 243–252). LNCS 7048, Springer-Verlag.

Settas, D., Bibi, S., Sfetsos, P., Stamelos, I., Gerogiannis, V. (2006). Using Bayesian belief networks to model software project management antipatterns. In *4th ACIS international conference on software engineering research, management and applications (SERA 2006)* (pp. 117–124).

Settas, D., & Stamelos, I. (2007). Using ontologies to represent software project management antipatterns. In *Proceedings of the sofware engineering knowledge engineering conference (SEKE 2007)* (pp. 604–609).

Settas, D. L., G., Meditskos, Stamelos, I. G., & Bassiliades, N. (2011). Detecting antipatterns using a web-based collaborative antipattern ontology knowledge base. In C. salinesi, O. Pastor (Eds.), *Proceedings of the 5th ONTOSE 2011 workshop (international workshop on ontology, models, conceptualization and epistemology in social, artificial and natural systems)* (Vol. 83) (pp. 478–488). Lecture Notes in Business Information Processing, Springer.

Settas, D., Cerone, A., & Fenz, S. (2011). Towards automatic generation of ontology-based antipattern Bayesian network models. In *Proceedings of the 9th ACIS IEEE international conference on software engineering research and applications (Sera)*. (pp. 46–53). IEEE Computer Society.

Settas, D. L., Meditskos, G., Stamelos, I. G., & Bassiliades, N. (2011). Sparse: A symptom-based antipattern retrieval knowledge-based system using semantic web technologies. *Expert Systems with Applications, 38*(6), 7633–7646.

Settas, D., Sowe, S., & Stamelos, I. (2009). Addressing software project management antipattern ontology similarity using semantic social networks. *Knowledge Engineering Review, 24*(3), 287–308.

Sowa, J. F. (1999). *Knowledge representation: Logical, philosophical, and computational foundations*. Brooks/Cole.

Stoilos, G., Simou, N., & Stamou, G. (2006). Uncertainty and the semantic web. *Intelligent Systems, IEEE, 21*(5), 84–87.

Taniar, D., & Rahayu, J. W. (2006). *Web semantics and ontology*. Idea Group Publishing.

Tudorache, T., Vendetti, J., & Noy, F. N. (2008). Web-protege: A lightweight owl ontology editor for the web. In *Proceedings of the fourth workshop in the OWL: Experiences and direction (OWLED)*.

Wiki-Community, Wikipedia antipatterns community catalogue. <http://en.wikipedia.org/wiki/anti-pattern, http://en.wikipedia.org/wiki/Anti-pattern>.

Wiki-Community, Pattern community antipattern catalogue. <http://c2.com/cgi/wiki?antipatternscatalog, http://c2.com/cgi/wiki?AntiPatternsCatalog>.

Zheng, B. -Y. K. H. -T., Kim, H. -G. (2008). URSW, An ontology based Bayesian network approach for representing uncertainty in clinical practice guidelines. In F. Bobillo, P.C.G. da Costa, & P. Vojts (Eds.), *CEUR workshop proceedings. CEUR-WS.org* (Vol. 327).

Ziv, H., & Richardson, D. J. (1997). The uncertainty principle in software engineering. In *Proceedings of the 19th conference on software engineering (ICSE)*.