2011 Ninth International Conference on Software Engineering Research, Management and Applications

Towards Automatic Generation of Ontology-based Antipattern Bayesian Network Models

Dimitrios Settas*, Antonio Cerone † and Stefan Fenz ‡
*United Nations University,
International Institute for Software Technology
Macau, SAR, China
Email: settdimi@iist.unu.edu
†United Nations University,
International Institute for Software Technology
Macau, SAR, China
Email: antonio@iist.unu.edu
‡Institute of Software Technology and Interactive Systems,
Vienna University of Technology and SBA Research
Vienna, Austria
Email:stefan.fenz@tuwien.ac.at

***Abstract*—Previous work has proposed the ontology-based semi-automatic generation of antipattern Bayesian Network (BN) models. The generated BN model can be used to illustrate the effects of uncertainty on antipatterns using Bayesian propagation. This can guide users in detecting particular antipattern attributes of importance based on uncertain ontological information. However, the proposed approach has been implemented in the Protege ontology editor environment and requires human intervention to specify how the BN model will be generated. The fully automated generation of ontology-based antipattern BN models still remains an open issue. SPARSE is an OWL ontology based intelligent system that assists software project managers in the antipattern detection process. In this paper, we propose the use of the resulting detected antipatterns of SPARSE, their attributes (i.e. causes, symptoms, consequences) and the ontological relationships between these attributes, in order to automatically generate BN models of the detected antipatterns. We illustrate how this approach can be implemented using an example of 8 antipattern attributes of 6 inter-related antipatterns detected using SPARSE.**

***Keywords*-Bayesian networks; antipatterns; ontology; intelligent systems;**

I. Introduction

Antipatterns [1, 2] are the latest generation of design pattern research and are mechanisms that describe how to arrive at a good (refactored) solution from a fallacious solution that has negative consequences [1]. These mechanisms can be used in software development, architecture and management. As a result, managers are able to avoid the specious solution(s) that have resulted in finding themselves in an unhealthy situation for the organization and the individual [2]. Using antipatterns, a software project can be managed more effectively by bringing insight into the causes, symptoms, consequences, and by providing successful repeatable solutions [1].

Several issues currently surround the technology of antipatterns and pose difficulties on their adoption and usage not just in project management but in all antipattern categories. Anyone can become an antipattern author. As a result, there exists a large number of unstructured and informal antipatterns in software development, architecture and management. Antipatterns do not appear isolated and are highly inter-related. However, antipatterns that are informally documented have no declared relationships between each other and often contradict and duplicate other antipatterns.

The World Wide Web currently contains a large number of documented project management antipatterns in blogs [1] and repositories [2]. The wide collection of project management antipatterns that is available offers a vast amount of project management knowledge on how to resolve antipatterns. The number of antipatterns is increasing to an extent that it cannot be effectively re-used and the different templates that can be used to document antipatterns impose further difficulties on software project managers. Furthermore, out of the many project management antipatterns that can be used during a software project, only a few are applicable while managing a specific project. This requires expertise on identifying antipatterns and requires managers to memorize a large number of antipatterns.

Semantic Web technologies and knowledge-based systems have recently provided antipatterns with new knowledge

[1]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/

[2]http://en.wikipedia.org/wiki/Antipattern, http://c2.com/cgi/wiki?AntiPa-tternsCatalog

978-0-7695-4490-8/11 $26.00 © 2011 IEEE
DOI 10.1109/SERA.2011.15

acquisition, representation and sharing options. The Protege and WebProtege ontology editors provided a framework to enrich the ontological knowledge base collaboratively and foster a community of software project management contributors. However, uncertainty concerns every aspect of software development and the antipattern ontology is not an exception. The knowledge that is used to enrich the antipattern ontology is based on past experience and often contains a certain amount of uncertainty. By including probabilistic information in the antipattern ontology, users can be given the option to specify the degree of uncertainty regarding the existence of an antipattern attribute or relationship. This knowledge is then shared with other developers who might be facing a similar problematic situation in a software project.

BNTab [3, 4] has been recently used in the antipattern ontology in order to generate Bayesian Networks in a semi-automatic manner. 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. However, BNTab is executed in the Protege ontology editing environment and requires users to define the classes and properties of the desired BN model.

SPARSE [5] is an intelligent system that can bring the software project managers' attention to focus on antipatterns that are specifically suitable to a specific software project. Hence, a software project manager who wishes to detect antipatterns will not require expertise to determine which antipattern is most likely to apper at a given moment. The approach proposed in this paper, aims to leverage the existing ontology-based BN model generation and integrate this technology in SPARSE. This will further enhance antipattern detection by allowing users to visualize the cause and effect relationships of the detected antipatterns and explore the Bayesian Network (BN) model using Bayesian updating. Furthermore, using the rule program of SPARSE, the BN model can be automatically generated without requiring human intervention.

This paper is divided in 6 sections, which are organized as follows: Section 2 describes the background, the related work and the literature review used in our research. Section 3 presents SPARSE and the antipattern detection mechanism and how it can be enhanced using BNs. Section 4 describes the extensions required to take probabilistic information into account in the antipattern ontology. Section 5 exemplifies the proposed approach using an example BN model of 8 antipattern attributes of 6 antipatterns which were detected using SPARSE. Finally, in Section 5, the findings are summarized, future work is proposed and conclusions are drawn.

II. Background and Related Work

A. Background

Bayesian networks are used in situations which require statistical inference and therefore they have been widely used in project management to address causality and uncertainty [6, 7]. Khodakrami et al. [8] made the first attempt to model project management using Bayesian Networks by showing how a BN model can be generated from a project's critical path method (CPM) network. BNs have been successfully used in order to support managerial decision making [7]. This BN model is used in a decision-support tool that allows a project manager to trade-off the resources used against the outputs (i.e. quality achieved) in a software project [7].

In previous work, Bayesian Networks (BNs) [9] and the formalism of ontology [5] 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. By applying the formalism of ontology, a common lexicon of term that can be communicated across people and software tools was defined [5]. This provided the basis for the application of further methodology in order to address similar antipattern ontologies and resulted in the implementation of SPARSE [5], an ontology-based intelligent system that can detect antipatterns based on the symptoms that appear during a software project.

In this paper, we intertwine the formalisms of Bayesian Networks (BNs) and Ontology with the results of SPARSE. The goal of this exploration is to automate ontology-based antipattern BN model generation. 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. Support for uncertainty is essential for the antipattern ontology 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 effect 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. This approach incorporates probabilistic information in the ontology [3] and allows the semi-automatic generation of BNs using BNtab[3]. SPARSE can further enhance and automate this process using its results as the input for BN model generation. This will result in a symptom-based antipattern BN model that

[3]http://stefan.fenz.at/ontology-based-generation-of-bayesian-networks/

is generated using ontological information and no human intervention at all.

Several approaches and frameworks have been proposed to support the study of uncertainty in ontologies. BayesOWL framework [10, 11] can translate an OWL ontology into a BBN structure, but is still under development. Other frameworks currently under development such as POWL [12] and PR-OWL [13] can extend OWL vocabulary for representing uncertainty with different expressiviness. These frameworks are being studied by the Uncertainty Reasoning for the World Wide Web Incubator Group (URW3-XG) [14] 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 to generate Bayesian Networks with Protege. The BNTab plug-in has been developed by Fenz [3] and enables users to efficiently generate Bayesian networks based on existing ontologies. In this paper, we propose the integration of BNTab in SPARSE in order to enhance the antipattern detection process using BNs.

B. Related Work

Expert systems have been widely used in many different settings including teaching [15], medicine [16] and safety critical systems [17]. The World Wide Web (WWW) contains numerous examples of expert systems. However, expert systems are not a panacea and can be wrong [18]. Adams [19] has presented some considerations for the expert system design that need to be addressed when the system is used via the WWW. The author concludes that the feasibility of providing expert system capabilities over the World Wide Web depends upon the particular situation for which the expert system is developed.

SPARSE has been developed to mimic or replace the reasoning and decision-making of a human expert in detecting antipatterns, but due to the nature and complexity of software projects in which such a system is deployed, they are doomed to make mistakes. In this paper, we address the issue of managing the uncertainty that is inherent in antipatterns using Bayesian Networks. Expert systems often use Bayesian analysis and a BN model can be considered an expert system itself. A Bayesian system has been considered an expert system for the prognosis at 24 h of head-injured patients of the intensive care unit [16]. The construction of a BBN incorporates the maintenance of a large database including all the critical variables corresponding to the specific clinical domain. The user views the changes at each step, thus being capable of deciding upon the necessary pieces of information in order to reach a certain belief threshold. The system produces results that are compatible with the opinions of medical experts regarding the prognosis of patients exhibiting certain patterns of clinical or laboratory data.

Guo [17] has discussed the application of Bayesian Networks (BNs) to a Safety Assessment Expert System. He concludes that most problems regarding safety standards originate from the uncertainty nature in safety assessment and proposed the use of BNs to represent knowledge and manage the uncertinty for safety assessment.

Despite the extensive body of expert or intelligent systems literature, the research summarized in this paper represents the first implementation of Bayesian Networks in an ontology-based intelligent system.

III. The Antipattern detection process

A complete description of SPARSE [5] is outside the scope of this paper. However, the underlying operation of the tool must be presented in order to understand the need and the benefits of enhancing it using Bayesian networks. There are three underlying technologies involved in SPARSE: (a) ontologies, through the use of the OWL ontology language, (b) DL reasoners, through the use of the Pellet DL reasoner and (c) production rule engines, through the use of the CLIPS production rule engine.

By defining a top level ontology for describing antipatterns together with probabilistic values on their attributes and relationships, we have advantage of achieving a collaborative definition of antipatterns, allowing software project managers to create antipatterns using new ontology instances of causes, symptoms and consequences or using any such attributes that have already been documented. The standard-based and open-world nature of OWL ontologies allow their extension, reuse and merge, creating an antipattern knowledge base that encapsulates different perspectives, according to the project they appear in. The ontology also allows semantic processing of antipatterns, based on DL algorithms (DL reasoners). DL reasoners ensure the semantic consistency of the ontology-based antipatterns, as well as the derivation of any implicit (hidden) knowledge that derives from the antipattern definitions. In SPARSE we have used the Pellet DL reasoner [20] as the underlying reasoning infrastructure.

The top level ontology also allows us to exploit the research that has been done on the combination of rules and ontologies. In that way, we are able to express richer semantic relationships among antipatterns, in a more declarative way. SPARSE is able to incorporate logical consequences expressed as SWRL rules [21] that are handled by the underlying Pellet DL reasoner. Finally, the ontology allows the exploitation of research in ontology based BN model generation [3]. In this paper we propose the integration of this work with the environment of SPARSE. This will allow the automatic creation of BN models of antipattern attributes and relationships based on the detected antipatterns. This will manage the uncertainty that exists in the ontology data collection process and will visualize the effect of a specific

attribute or relationship of an antipattern to other connected BN network nodes.

In SPARSE we make a distinction between the ontology inference rules and the domain rules that are used for deriving conclusions over the ontological knowledge (CLIPS production rules). The former are used at the ontology reasoning level and they are embedded into the DL reasoning procedure in order to infer the appropriate semantic relationships among antipatterns. The latter are used for defining the rule-based applications over the OO model of the extensional ontological knowledge, without altering the ontology itself. For example, the following production rule informs users about symptom objects that do not define any `symptomToConsequence` value.

```
(defrule validation-rule-check-symptoms
    (object
        (is-a Symptom)
        (title ?title)
        (symptomToConsequence $?cons &:
                (eq (length$ $?cons) 0)))
    =>
    (printout t "The symptom " ?title
"..." crlf))
```

A complete description of the antipattern ontology [5] is outside the scope of this paper. However it is important to understand the existing ontology before describing the required extensions in order to allow ontology-based BN generation. The ontology consists of 7 concepts, 21 roles (19 object and 2 datatype roles), 192 individuals and 7 SWRL rules.

The antipattern ontology consists of seven concepts. In addition to the four intuitive `Antipattern`, `Cause`, `Symptom`, `Consequence` and three antipattern-related concepts have been defined directly as subclasses of the `Antipattern` concept, that is,

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

The `Cause` concept is used in order to define the causes of the ontology. It is defined as the subclass of the intersection of three universal role restrictions.

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

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. 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 is 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:

$$\begin{aligned}\texttt{Antipattern} \sqsubseteq\ & \forall \texttt{hasCause.Cause} \sqcap \\ & \forall \texttt{hasSymptom.Symptom} \sqcap \\ & \forall \texttt{hasConsequence.Consequence} \sqcap \\ & \geq 1\ \texttt{hasCause.}\top \sqcap \\ & \geq 1\ \texttt{hasSymptom.}\top \sqcap \\ & \geq 1\ \texttt{hasConsequence.}\top\end{aligned}$$

The ontology roles allow the definition of basic knowledge related to antipattern causes, symptoms and consequences, as well as to their correlations. 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.

The antipattern ontology allows the definition of correlations among causes, symptoms and consequences. In this section, for simplicity, we describe only the roles that correlate a cause with a cause.

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 SWRL rules is able to infer all the relevant (implicit) causes for a specific antipattern following the `causeToCause` relations.

The main functionality of SPARSE is to detect and propose antipatterns based on a set of symptoms that users select from the antipattern ontology. The antipatterns that are returned can be classified into two categories:

- Symptom-based matched antipatterns. These are the antipatterns that contain one or more user-selected symptoms. The matching of antipatterns is performed by a set of production rules that traverse the antipattern objects of the COOL KB and select the ones that satisfy one or more user-selected symptoms.
- Relevant antipatterns. SPARSE proposes also a set of antipatterns that might be relevant to the symptom-

based returned antipatterns, examining their causes and consequences. More specifically, the algorithm finds antipatterns that have common causes and/or consequences with one or more symptom-based matched antipatterns.

Finally, an explanation mechanism presents to users in textual description the relationships that resulted in the inclusion of a specific antipattern in the result set. In the case of a symptom-based matched antipattern, SPARSE presents the user-selected symptoms that the antipattern satisfies, along with their category. In the case of a relevant antipattern to one or more symptom-based matched antipatterns, SPARSE presents all the relationships that the antipattern shares with the symptom-based matched antipatterns, along with their category.

BNs can supplement the results displayed to the user by allowing them to visualize the causes, symptoms and relationships of the detected antipatterns that have associated probabilistic values. The user can then observe which attributes are the key attributes of the model that affect the most nodes. Finally, BN editing software (e.g. Netica) can be used to assign values to BN nodes and explore the model using Bayesian propagation algorithms that are built into the software. This process is described in section 5.

IV. Probabilistic extensions to the Antipattern OWL Ontology

The required extensions to allow ontology-based BN model generation using BNTab included 6 new ontology concepts and 12 roles (object type). 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 `Consequence-AprioriProbability` 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 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.

The `AntipatternAprioriProbabilityhas-Antipattern` 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 `AntipatternhasAntipatternAprioriProbability` is a role of the antipattern concept and correlates an antipattern with specific probabilistic information of `AntipatternAprioriProbability` members. The `AntipatternAprioriProbabilityhasProbability` 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 resluting 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 or automated manner using the results of SPARSE. This process is described in the section 5.

V. Bayesian Networks in Antipattern Detection

Appropriate changes in the DL reasoner, production rule engines and ontologies need to be made in order to allow the incorporation of a BN model in the results of SPARSE. The OWL antipattern ontology documents antipatterns as ontology instances and defines the relationships with other antipattern attributes (i.e. causes, symptoms or consequences) through OWL properties. The semantic relationships that derive from the antipattern definitions are determined using the Pellet DL (Description Logic) reasoner [20]. The knowledge base of this reasoner is then transformed into the COOL object-oriented language of the CLIPS production rule engine [22]. This transformation is carried out in order 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. By using a combination of rules and ontologies, we are able to express richer semantic relationships that exist among antipatterns, in a more declarative manner and SPARSE is able to incorporate logical consequences expressed as SWRL rules that are handled by the underlying Pellet DL reasoner.

Using the same technology, the existence of a cause and effect relationship between BN model nodes can also be expressed as SWRL rules that are handled by the underlying Pellet DL reasoner. Finally, the Pellet DL reasoner ensures the semantic consistency of the ontology-based antipatterns and is the reasoning infrastructure that derives implicit

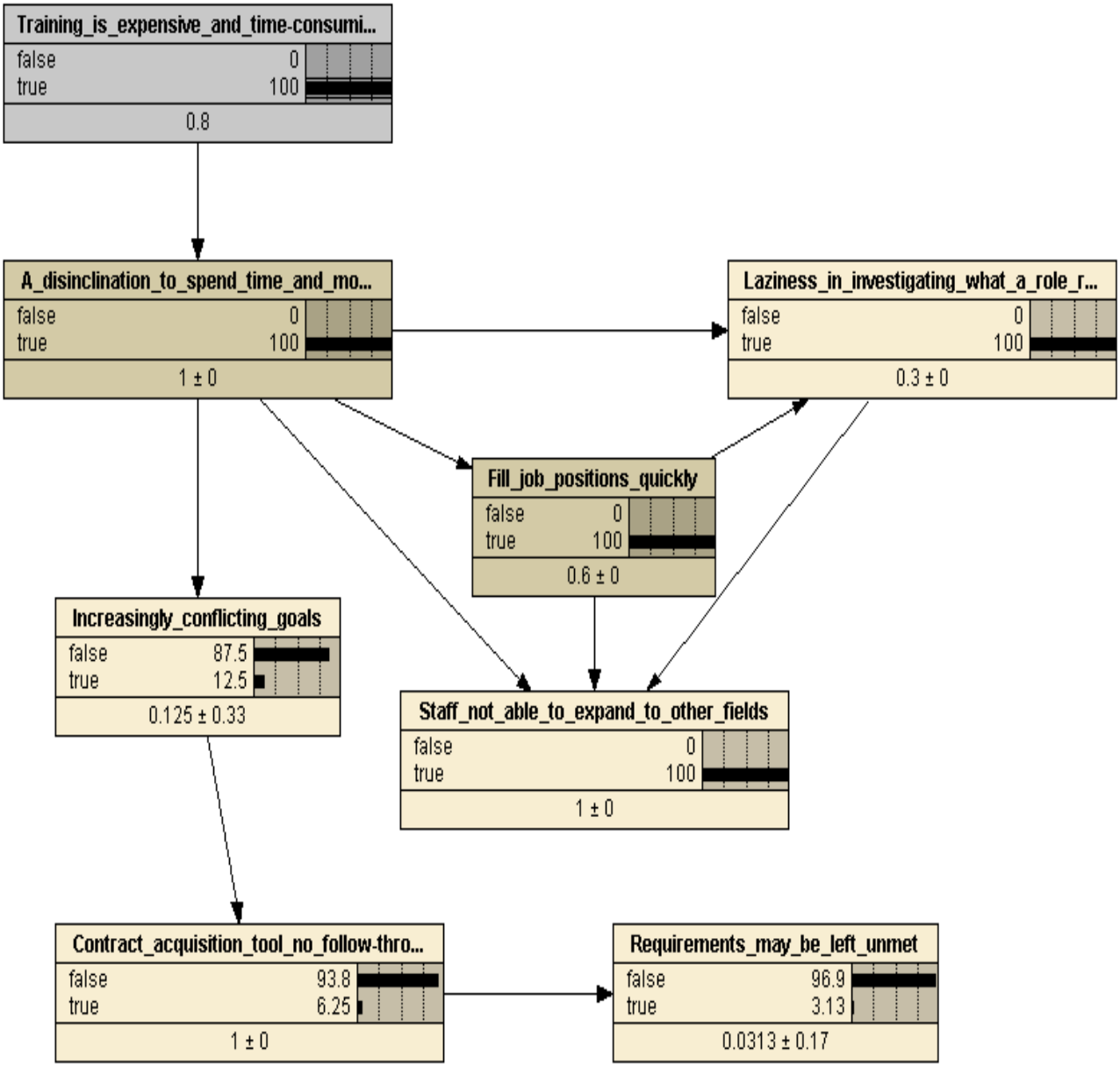

Figure 1. Example BN model of the attributes of the detected antipatterns

knowledge from the antipattern definitions. For example it ensures that for each created antipattern there is at least one associated primary cause, symptom and consequence. It cannot be expected for every cause, symptom, consequence and relationship to have probabilistic values defined in the ontology because this knowledge may not exist. However, for the antipattern attributes and relationship individuals that have an attached probabilistic value, Pellet DL reasoner will be used to derive this knowledge and display it together with the resulting detected antipatterns.

Regarding the ontology inference rules and the domain rules that are used for deriving conclusions over the ontological knowledge, new rules have to be constructed to specify which attributes will be the members of the BN model nodes and define their cause and effect relationships.

For example, rule (1) defines that if antipattern `anti` has symptom `s1` and symptom `s1` is related to symptom `s2`, then symptom `s2` also belongs to antipattern `anti`.

1) hasSymptom(?anti, ?s1) $\wedge$ symptomToSymptom(?s1, ?s2) $\rightarrow$ hasImplicitSymptom(?anti, ?s2)

In that way we give the opportunity to use an efficient and well-known production rule engine in order to develop the

rule-based application of SPARSE over a shared ontology in a hybrid manner.

The existing version of SPARSE can provide the user with a set of antipatterns that may exist in a software project according to the symptom(s) that the user selected from the predefined list of antipattern symptoms. The system retrieves, displays and provides further explanation for the detected antipatterns and their attributes according to the matching that was carried out by the set of production rules that traverse the antipattern objects of the COOL KB [5]. By enriching the ontology with further constructs that can define the probabilistic values of attributes and their relationships, this information can be used together with production rules that will retrieve the relevant attributes and relationships of the detected antipatterns and define their cause and effect relationships in the BN model. Using BNTab and Netica, this information can be used to automatically produce a BN model of antipattern attributes and/or relationships (for example the certainty of a link between a cause and a symptom of an antipattern). In this manner, the produced BN model (Figure 1) will only contain the relevant attributes and relationships of the detected antipatterns and will not contain BN nodes that are not connected with the detected antipattern nodes. In this section we provide an example of a BN model generated using BNTab.

The example model in Figure 1 contains 8 antipattern attributes that belong to 6 software project management antipatterns. 6 of these attributes are causes of antipatters and the remaining 2 attributes (”Staff not able to expand to other fields” and ”Requirements may be left unmet”) are symptoms of antipatterns. The probabilistic information that was used to enrich the values and CPTs of the BN nodes uses the probabilistic values contained in the antipattern ontology. These can be obtained based on expert judgement using the past occurence of these attributes or on probailistic occurence data from past software projects. The model in Figure 1 contains 10 cause and effect relationshps that show the influence that each node has to the BN model nodes. The edges that exist between nodes depend on the properties that are defined in ontology concepts. In this specific example the properties used are the `causeToCause`, `symptomToSymptom`, `symptoToCause` and `causeToSymptom`, as the model only contains causes and symptoms concepts. Each node has two categorical values which are ”True” and ”False”. Next to each value there is percentage indicating the possibility of each value. Users can explore the model by changing a value according to what they observed in past projects or what is occuring in the current software project. For example, by setting evidence to ”True=100%” to the node ”Training is expensive and time consuming”, they can explore how this affects the remaining connected nodes of the model. This value propagates to the node ”A disinclination to spend time and money training staff on the job” and using the BN d-seperation properties of a serial connection [9] affects the value of the node ”Increasingly conflicting goals”. In this serial connection only if evidence is set to the intermediary node, evidence can not be transmitted from the node ”Training is expensive and time consuming” to the node ”Increasingly conflicting goals”. The description of the other two d-separation properties in antipattern bayesian network models can be found in previous work [9].

Our model allows users to study how setting values to an antipattern cause affects other antipattern causes and symptoms. Users can then identifynodes that are of great importance and can emphasize on resolving specific antipattern causes, symptoms and/or consequences. Depending on the occurence or the existence of these antipattern attributes, a user can set values to BN nodes and then explore how this affects the nodes of interest. For example, a software project manager can deal with a cause of an antipattern individually and change its existence value to ”False”. The manager can then update the model again using Bayesian propagation, to explore how this affects the remaining antipattern attributes. Using a BN model based on the detected antipattern attributes, the possible probabilistic values of a node of interest (e.g., ”Requirements may be left unmet”) are explored by setting values to the other nodes of the model according to observations made for these nodes in the current or past software project(s).

VI. Conclusion

The automation of Bayesian Network model generation based on (i) the ontology, (ii) BNTab and (iii) the results of SPARSE is particularly useful and well suited to the domain of antipatterns. The resulting BN model can visualize antipattern attributes of the detected antipatterns and allows users to set specific probabilistic values to antipattern causes, symptoms, consequences and their relationships. It can also be used to illustrate how related antipattern attributes are affected using Bayesian propagation and help users in detecting particular nodes of importance. This can draw their attention to antipattern attributes that are of great importance to a specific project. The approach has been exemplified using an example of 8 antipattern attributes of 6 inter-related antipatterns detected using SPARSE.

The implementation and development of SPARSE to support automatic BN model creation will further motivate users to contribute more antipatterns to the antipattern ontology. Eventually, SPARSE will motivate the widespread use of antipatterns in software project development, architecture and management. Evaluation of the proposed software tool is necessary in order to determine its suitability as an intergrated tool that supports both antipattern detection and BN model generation.

REFERENCES

[1] W. Brown, H. McCormick, and S. Thomas, *AntiPatterns in Project Management*. Wiley Computer publishing, 2000.

[2] P. Laplante and C. Neil, *Antipatterns: Identification, Refactoring and Management*. Taylor and Francis, 2006.

[3] S. Fenz, A. M. Tjoa, and M. Hudec, "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*, I. C. Society, Ed., 2009, pp. 712 – 717.

[4] S. Fenz. [Online]. Available: http://stefan.fenz.at/ontology-based-generation-of-bayesian-networks/

[5] D. L. Settas, G. Meditskos, I. G. Stamelos, and N. Bassiliades, "Sparse: A symptom-based antipattern retrieval knowledge-based system using semantic web technologies," *Expert Systems with Applications*, vol. 38, no. 6, pp. 7633–7646, June 2011.

[6] C.-F. Fan and Y. Yu, "Bbn-based software project risk management," *The Journal of Systems and Software*, vol. 73, pp. 193–203, 2004.

[7] N. Fenton, W. Marsh, M. Neil, P. Cates, S. Forey, and M. Tailor, "Making resource decisions for software projects," in *Proceedings of the 26th International Conference on Software Engineering (ICSE'04)*. IEEE Computer Society, May 2004, pp. 397–406.

[8] V. Khodakrami, N. Fenton, and M. Neil, "Project planning: Improved approach incorporating uncertainty," in *European Academy of Management Annual Conference (EURAM 2005)*, 2005.

[9] D. Settas, S. Bibi, P. Sfetsos, I. Stamelos, and V. Gerogiannis, "Using bayesian belief networks to model software project management antipatterns," in *4th ACIS International Conference on Software Engineering Research, Management and Applications (SERA 2006)*, IEEE, Ed., 2006, pp. 117–124.

[10] Z. Ding, Y. Peng, and R. Pan, "A bayesian approach to uncertainty modelling in owl ontology," in *Proceedings of the 2004 Int.Conference on Advances inIntelligent Systems*, 2004.

[11] R. Pan, Z. Ding, Y. Yu, and Y. Peng, "A bayesian network approach to ontology mapping," in *Proceedings of the Fourth International Semantic Web Conference*, Springer, Ed., 2005.

[12] E. Hung, C.-C. Szeto, W. Fang, and Y. Deng, ""powl: A multi-level approach to represent uncertainty in semantic web ontology"," Department of Computing, Hong Kong Polytechnic University, Tech. Rep., 2009.

[13] R. N. Carvalho, K. B. Laskey, and P. C. G. Costa, "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)*, Springer, Ed., July 2010.

[14] K. J. Laskey and K. B. Laskey, "Uncertainty reasoning for the world wide web: Report on the urw3-xg incubator group," URW3-XG, Tech. Rep., 2008.

[15] C. Martincic and D. P. Metzler, "An expert system development environment for introductory ai course projects," *The Journal of Computing Sciences in Colleges, The proceedings of the Tenth Annual Consortium for Computing Sciences in Colleges Northeastern Conference*, 2005.

[16] G. Nikiforidis and G. Sakellaropoulos, "Expert system support using bayesian belief networks in the prognosis of head-injured patients of the icu," *Med Inform (Lond)*, vol. 23, no. 1, pp. 1–18, Jan-Mar 1998.

[17] B. Guo, "Knowledge representation and uncertainty management: applying bayesian belief networks to a safety assessment expert system," in *Proceedings of the International Conference on Natural Language Processing and Knowledge Engineering*, IEEE, Ed., 2003, pp. 114 – 119.

[18] J. Williams, "When expert systems are wrong," in *Proceedings of the ACM SIGBDP conference on Trends and directions in expert systems*, 1990.

[19] J. A. Adams, "The feasibility of distributed web based expert systems," in *Proceedings of the 2001 IEEE International Conference on Systems, Man, and Cybernetics*, 2001.

[20] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical owl-dl reasoner," *J. Web Sem.*, vol. 5, no. 2, pp. 51–53, 2007.

[21] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," W3C Member Submission, Tech. Rep., 2004. [Online]. Available: http://www.w3.org/Submission/SWRL/

[22] G. Meditskos and N. Bassiliades, "HOOPO: A hybrid object-oriented integration of production rules owl ontologies," in *18th European Conference on Artificial Intelligence (ECAI)*, A. IOS Press, Ed., 2008, pp. 729–730.