Using Bayesian Belief Networks to Model Software Project Management Antipatterns

Dimitrios Settas, Stamatia Bibi, Panagiotis Sfetsos, Ioannis Stamelos
Dept. of Informatics,
Aristotle University of Thessaloniki
Thessaloniki, Greece
{dsettas,sbibi,stamelos}@csd.auth.gr, sfetsos@it.teithe.gr

Vassilis Gerogiannis
Department of Project Management, Technological Education Institute
411 10, Larisa, Greece
gerogian@teilar.gr

Abstract

In spite of numerous traditional and agile software project management models proposed, process and project modeling still remains an open issue. This paper proposes a Bayesian Network (BN) approach for modeling software project management antipatterns. This approach provides 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. The approach is exemplified through a specific BN model of an antipattern. The antipattern is modeled using the empirical results of a controlled experiment on Extreme Programming (XP) that investigated the impact of developer personalities and temperaments on communication, collaboration-pair viability and effectiveness in pair programming. The resulting BN model provides the precise mathematical model of a project management antipattern and can be used to measure and handle uncertainty in mathematical terms.

1. Introduction

Ineffective project management has been recognised as a major factor contributing to software project failure [1]. Software project success or failure can be attributed to the incorrect handling of one or more project variables: people, technology and process [2], [3]. There exist many uncertainties in these variables and current software engineering techniques are unable to eliminate them [4]. Managing this uncertainty is very important because this would lower the risk of the project being unsuccessful [2].

Project management antipatterns suggest commonly occurring solutions [5] to problems regarding dysfunctional behaviour of managers or pervasive management practices that inhibit software project success [6]. These mechanisms can manage all aspects of a software project more effectively by bringing insight into the causes, symptoms, consequences, and by providing successful repeatable solutions [3]. Though it is widely accepted that uncertainty influences software project management, antipatterns have not been used to explicitly capture uncertainty in a project management model. Based on project management models, tools that can greatly assist managers in carrying out project management tasks can be developed [7]. Such tools can assist in determining how realistic a schedule is, can identify the need for additional resources, allow the analysis of various alternative approaches and identify various mixes of staff skills and experience levels [1]. Recent modern software project management tools focus on relationships between communication and project states [7]. Thus, even if project managers have no management documents communication logs can indicate automatically project states by the proposed tool [7]. Particularly, in agile software development, such tools are valuable for managers because developers should complete their project in a short period without creating various documents for management. However, these

Part of this work is supported by the "Model based Integrated Environment to Support Simulation in Software Project Management project" (MISSION-SPM), which is co-funded by the European Social Fund and national resources (EPEAEK II programme).

tools do not address the issue of managing uncertainty. Uncertainty has been taken into account by several authors, who have used Bayesian Networks in software engineering project management [4], [8], [9], [10], [11].

In this paper we suggest the use of Bayesian Networks for modelling software project management antipatterns. Bayesian Belief Networks provide a natural, logical and probabilistic framework to depict project management antipatterns. The benefit of modelling a project management antipattern into a BN is to bring the full weight and power of BN analysis to bear on the problem of managing uncertainty in software projects. Furthermore three considerations justify the decision of modelling antipatterns with the BNs:

- The formalism of Bayesian Networks enables the development of a precise mathematical model of an antipattern. We believe that project management antipatterns are effective in dealing with uncertainty and BNs enable us to measure and handle this uncertainty in mathematical terms.
- The suggested model can be used by project managers to illustrate the effects of uncertainty on a project management antipattern.
- The proposed antipattern BN model can be used by project managers who have scarce statistical data of present or past projects and who wish to supplement the model using expert judgement.

In particular we define the Extreme Programming (XP) "Shaken but not stirred" project management antipattern and model it using a Bayesian network in order to study how it can manage uncertainty in an XP project. Communication is one of the four most important values of XP. However, XP accepts the observation that communication problems exist in projects [12]. In view of that, this paper presents a BN model of an antipattern designed to evaluate the impact of developer personalities and temperaments on communication and collaboration. We chose this issue to demonstrate the power of antipatterns modelled through BNs in this study. This paper is divided in 4 sections, which are organized as follows: section 2 describes the background, the related work and the literature review used in our research. Section 3 describes the Bayesian network model of the antipattern, the antipattern itself and the interpretation of the data. Finally in section 4 we summarize our findings and draw our conclusions.

2. Background and Related Work

2.1 Background

It is crucial that a project manager is aware of the issues associated with managing people, technology and the process. Antipatterns are particularly useful in such cases, because they describe commonly occurring solutions to problems that generate negative consequences [5]. By listing project management antipattern catalogues [3], [6], [15] project managers can identify potential problems and provide a refactored solution in a practical and reusable manner. An antipattern is a new form of pattern that has two solutions. The first is a solution with negative consequences and the other is a refactored solution, which describes how to change the antipattern into a healthy solution [5]. According to Brown et al. [5], an antipattern can be the result of a manager not having sufficient knowledge or experience in solving a particular problem.

Antipattern catalogues, have covered several management [3], personality and environmental antipatterns [6]. By documenting XP antipatterns [15], project coaches and managers are able to solve a newly encountered problem by applying a previously successful solution to a particular problem. XP is arguably the best known and most popular agile development methodology and pair programming is the most controversial aspect of XP [14]. According to the pair programming practice, every line of code should be created by a pair of developers sitting side by side. The aim is to transfer skills and knowledge between the two developers. However, there exist underlying human factors that need to be identified and understood in order to control, predict and understand the problems that arise from the limited guidance that has been so far provided for specific occasions. Therefore, an active community has been developed around XP in order to address such issues using antipatterns [15].

2.2 Related Work

Three XP antipatterns have been recently presented as an experience report, in the hope that they can be shared in the XP community [15]. It was concluded that the encountered problems when introducing XP were caused by the people, the team and the organization and not the technology itself. Such occasions for example, can occur when the senior programmer individually modifies the code of a particular pair of programmers because it does not meet the coding standards of the team [15]. This results in an unsatisfied pair of developers with little motivation. Therefore the refactored solutions to the proposed "AntiPractices" suggest turning the coach's eyes to the subtleties of people. Antipatterns in XP have not been adequately investigated. There has been no investigation on the impact of uncertainty on an XP antipattern. Furthermore, to the best of our knowledge this is the first attempt to use a graphical representation of a statistical model of an antipattern. We consider the impact of heterogeneous developer personalities and temperaments as an antipattern in order to study how this can manage uncertainty in XP.

The issue of finding the best mix of developer personalities has been widely researched [13], [16], [17], [18], [19]. According to these findings, one of the main factors that contribute to poor performance is software project team composition [17]. Belbin [16] concluded that teams need a balance of different types of people and proposed a self-assessment questionnaire that can help to identify which role a person is best suited. According to Stepanek [18], many programmers enjoy programming alone because they tend towards introversion. Williams [19] suggested that the most beneficial form of pairing is with two programmers of roughly the same ability and highlights the problem that mixed abilities are unavoidable. Finally, empirical investigation [13] has shown that heterogeneous developer personalities and temperaments affect pair effectiveness and especially communication, collaboration and pair viability.

Bayesian networks are used in situations that require statistical inference and have been widely used in project management to address causality and uncertainty [4], [8], [9], [10], [11]. Khodakrami et al. [10] 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 [9]. 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 [9]. It has been suggested [8] that BNs are effective in representing software process models having as an output the estimation of software effort. BNs are highly visual tools that are able to facilitate effective planning, control and operational management of the process [8]. The use of BNs has also been proposed to support expert judgment in software cost estimation and proved to be suitable for modelling uncertainty, and produce reasonable and safe estimates [11].

2.3. Uncertainty in XP

Uncertainty occurs in software engineering for various reasons and stems from multiple sources. Ziv and Richardson [20] proposed that there can be uncertainties in human participation, which is a key element of agile development. This inherent uncertainty is closely related to the fundamental values of XP. The foundation of agile development is based on the assumption that the business environment is unstable. Therefore this kind of development is not based on theory, but is derived from the experience of successful project teams [18]. As a result agile development is not overwhelmed by high change. The manifesto for agile software development has established a set of four values [12]. The first value of the agile manifesto ”individuals and interactions over processes and tools” outlines the importance of human participation. The second value is ”working products over comprehensive documentation”. The third value is ”customer collaboration over contract negotiation”. Finally the last value is ”responding to change over following a plan”. The prime directive of agile development is to ”embrace change” [12]. The faster the pace of change, the greater the uncertainty will be and hence the lower the probability that a software project will be successful. The closest second directive for agile development is ”embrace communication and feedback” [12]. This directive is illustrated in such practices as pair-programming, on-site customer and daily meetings, which are human activities.

3 The Antipattern Bayesian Network Model

3.1. Bayesian Networks (BNs)

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 [21]. Each variable has a finite set of mutually exclusive states. Furthermore, to each variable A with influencing variables (parents) B_1,.., B_n, there is attached the potential table Pr(A | B_1,..,B_n). These directed network graphs 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 (1).

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

It is important to understand the d-separation properties of a BN because the proposed antipattern BN model has to correspond with an accurate perception of the world. These properties are reflected in the proposed BN model of the antipattern (See Fig. 4). 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, we say 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 [21].

$$\begin{tikzcd} A \arrow[r] & B \arrow[r] & C \end{tikzcd}$$

Figure 1. Serial variable connection

In a diverging connection (Fig. 2), influence can pass between all the influenced variables (children) of A unless the state of A is known [21]. Therefore B, C and E are d-separated given A.

$$\begin{tikzcd} & A \arrow[dl] \arrow[d] \arrow[dr] & \\ B & C & E \end{tikzcd}$$

Figure 2. Diverging variable connection

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 [21]. These three cases cover all ways in which evidence may be transmitted through a variable.

$$\begin{tikzcd} B \arrow[dr] & C \arrow[d] & E \arrow[dl] \\ & A & \end{tikzcd}$$

Figure 3. Converging variable connection

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 [20]. 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 [24]. 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 [20]. 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. For the purposes of the BN models, i.e. for their graphical representations and various probability calculations, we used the Belief Network Powersoft Data Preprocessor [23], Belief Network Powersoft BN Power-Constructor [22] and Genie 2.0 [24] software tools.

3.2. Antipattern Definition

According to Brown [5] an antipattern is a literary form that describes a commonly occurring solution to a problem that generates decidedly negative consequences. Brown et al. [5], [3] was the first to provide the structure of an antipattern. Following the informal presentation style of project management antipatterns ([3], [6], [15]), Laplante's [6] template (see Table 1) will be used to identify the dysfunction and remedies for all those involved.

- **Name:** ”Shaken but not stirred”.
- **Central Concept:** A rush to use pair programming without ensuring that the developer pairs have mixed developer personalities and temperaments. Generally, any project manager or coach who does not use personality and temperament inventories.
- **Dysfunction:** This antipattern is attributable to a single individual. The project manager or the coach, who does not arrange the pairs according to mixed personalities, but usually only care in having experienced programmers mixed with inexperienced ones.
- **Explanation:** We use the term ”Shaken but not stirred” because it reflects the fact that the developers are only ”shaken” in pairs according to their skills but are not ”stirred” according to their personality and temperament. The main causes are the lack of experience and knowledge of the project manager on mixing developer personalities. This clearly has a negative effect on knowledge and skill sharing hence affecting the development time, communication, collaboration, design correctness and the quality of the software.
- **Band-Aid:** There is no band aid for this antipattern. No short term coping strategy can be applied to deal with this problem. Rearranging the pairs again without using personality tests will only make the situation more dysfunctional.
- **Self-Repair:** Project managers should mix the developer pairs accordingly as soon as possible. Therefore they have to quickly identify symptoms such as poor communication and collaboration of the pairs and pay attention to the possible complaints of the developers.
- **Refactoring:** Personality and temperament sorter tests such as the Keirsey Temperament Sorter test must be

Table 1. Antipattern structure.

Name	A short name that conveys the antipattern's meaning.
Central Concept	The short synopsis of the antipattern in order to make the antipattern identifiable.
Dysfunction	The problems with the current practice.
Explanation	The expanded explanation including causes and consequences.
Band-Aid	A short term coping strategy for those who don't have the influence nor time to refactor it.
Self-Repair	The first step for someone perpetuating the antipattern.
Refactoring	The required changes in order to remedy the situation and their rationale.
Identification	An assessment instrument consisting of a list of questions for diagnosis of the antipattern.

used in order to identify and interpret developers' personalities and temperaments. Pairs should be rearranged immediately according to their personality and temperament in order to improve software quality by improving pair communication, collaboration and pair effectiveness.

- **Identification:** The following questions should be answered with a "Yes" or "No". Does the XP team consistently deliver low quality software? Is there evidence of lack of shared knowledge of the project design? Have the developers reported any communication / collaboration problems to the project coach? Do the developers insist on working individually? If you responded "Yes" to one or more of these statements, your organization is probably suffering from "Shaken but not stirred".

3.3. Building the Antipattern BN Model

As already mentioned, BNs can be constructed using empirical data and/or expert judgment. For the purposes of the antipattern BN model, the empirical data of a controlled experiment [13] has been used in order to specify prior belief values for network variables. Bayesian belief values could have been elicited from a domain expert, who would subjectively assess them. However, this approach was not used as there was sufficient empirical data for our study. The participants in those experiments were eighty four undergraduate students from the 4th semester of the SE course separated randomly into two groups of pairs according to their personality [25] and temperament inventories. The Keirsey Temperament Sorter test [6] was used to identify and interpret students' temperaments in the data set that was used in the antipattern BN model (see Fig 4). The objective in both experiments was to design, code and test in Java two tasks of a well known object-oriented application.

The data from the two experiments were grouped in one single data set containing the eight identified fields (variables) required for our model (see Table 2). The process that was followed then involves: a) pre-processing of the training data by specifying which fields need to be discretized b) construction of the Bayesian Network using the Belief Network Powersoft BN PowerConstructor software tool [22] c) Bayesian update in order to explore the effect of the antipattern on resolving uncertainty.

Data pre-processing involved discretization of continuous fields into a finite number of intervals (see Table 2) using the default equal width binning method of the Belief Network Powersoft Data Preprocessor [23] software tool. The discretized fields were selected to be included in the BN. The Belief Network Powersoft BN PowerConstructor [22] tool was then used in order to provide our domain knowledge and construct the Bayesian Network. The BN model (see Fig.4) resulted from a three-phase belief network construction algorithm [22]. The discretized fields were selected to be included in the BN. Domain knowledge was provided on root and leaf fields. Root fields are fields that do not have any parent and leaf fields are fields that do not have any child. The default threshold was used in order to detect only strong relations between variables. Domain knowledge was also provided on the ordering of fields (variables) in order to indicate that a field is an indirect cause of another field. This procedure speeded up the construction process and defined the directed cause-effect relations between the variables. The results of the model were improved by providing domain knowledge compared with a model that contained no domain knowledge at all. However, even without providing any domain knowledge, PowerConstructor [22] managed to detect correct non-directed cause-effect relations. Genie 2.0 [24] was then used for further processing (see Fig. 4) in order to set evidence to data (see Table 3) and produce meaningful results. By updating the belief values every time evidence was set, the model was tested to verify its correctness. This technique allowed us to set evidence (Set Evidence =Yes) in the Mixed Personality and Mixed Temperament variables in order to illustrate the impact of uncertainty on an XP antipattern (see Table 3).

3.4. Data Analysis and Interpretation

The resulting BN model contains 8 variables and eight connecting links including 3 diverging connections, 2 serial connections and a converging connection (Fig. 4). The

Variable name	State name
Mixed Personality: An indicator of heterogeneous developer personality.	**NO:** Non-mixed developer personalities **YES:** Mixed developer personalities
Mixed Temperament: An indicator of heterogeneous developer temperament according to Keirsey's descriptive groupings[6].	**NO:** Non-mixed developer temperaments **YES:** Mixed developer temperaments
Personality Type: 16 Personality types according to the Myers-Briggs Type Indicator questionnaire [25].	**ENFJ, ENFP, ENTJ, ENTP, ESFJ, ESFP, ESTJ, ESTP, INFJ, INFP, INTJ, INTP, ISFJ, ISFP, ISTJ, ISTP**
Development Time: The total development time that took the pair to complete the 2 tasks (measured in minutes).	$<$ **75**: Reduced development time **75 - 105**: Medium range development time $>$ **105**: Increased development time
Total Communication Transactions: The total number of transactions in requirements gathering, specification and design changes, coding, unit testing, code and design reviews.	$<$**39**: Reduced communication **39 - 66**: Medium range communication $>$**66**: Increased communication
Collaboration: The collaboration grade with the partner developer (measured in points).	$<$**2.33**: Reduced collaboration **2.33-3.66**: Medium range collaboration $>$**3.66**: Increased collaboration
Design Correctness: Measured in points obtained by each pair for both assignments. The points were measured on a ratio scale based on checklists to ensure objective assessment of the participants'results.	$<$**4.33**: Reduced design correctness **4.33-6.66**: Medium range design correctness $>$**6.66**: High design correcntess
Quality: The quality of code variable used the the acceptance tests scores of both tasks as an indicator.	$<$**83.33**: Low quality **83.33-136.66**: Medium range quality $>$**136.66**: High quality

Table 2. Variable and state description for the BN model

Bayesian network model was then validated in order to ensure that the 3 diverging connections of the mixed personality, mixed temperament and design correctness variables correspond with our perception of the proposed antipattern. As already mentioned, according to this d-separation property, setting evidence to one of those three parent variables blocks the communication of the corresponding children variables. For example, setting evidence to the design correctness variable, blocks the communication of the collaboration variable and the quality variable. As a result, setting evidence to the collaboration variable, does not affect the quality variable.

Furthermore, the proposed BN model includes two serial connections between the variables Personality type, Design Correctness, Collaboration and Mixed personality, Mixed temperament, Total Communication transactions. These variables are d-separated by setting evidence to the variable in the connection (middle variable). As a result, setting evidence to the Mixed temperament variable blocks the connection between the two other variables and setting evidence to one variable, does not change the belief values of the other variable. Finally, there is a very interesting converging connection that links the variables Personality type, Mixed personality, Mixed temperament with the variable Design Correctness as a child. This indicates that knowledge about the personality type of one developer does not provide any information on whether the pair has mixed temperament or personality. However, if we set evidence to the consequence, which is the Design Correctness variable, then the parent variables become d-connected and as such, can communicate.

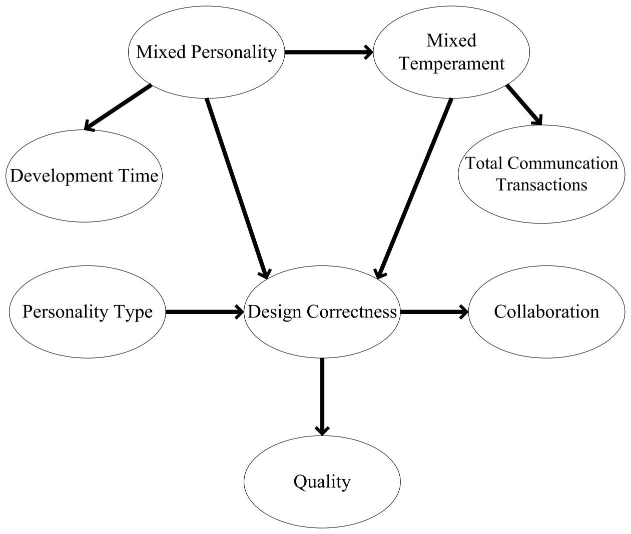

Figure 4. Antipattern BN model.

After ensuring that the model represents the antipattern "Shaken but not stirred" and that its properties reflect the

Raw Data values	Antipattern values
Mixed Personality: N 22%	N 0%
Y 78%	Y 100%
Mixed Temperament: N 47%	N 0%
Y 53%	Y 100%
Development time: x<75 7%	x<75 7%
x 75−105 57%	x 75−105 65%
x>105 35%	x>105 28%
Design Correctness:	
x<4.33 36%	x<4.33 24%
x 4.33−6.66 36%	x 4.33−6.66 41%
x>6.66 28%	x>6.66 35%
Total Communication Transactions:	
x<39 42%	x<39 14%
x 39−66 49%	x 39−66 72%
x>66 9%	x>66 14%
Collaboration: x<2.33 24%	x<2.33 20%
x 2.33−3.66 21%	x 2.33−3.66 20%
x>3.66 55%	x>3.66 60%
Quality: x<83.33 22%	x<83.33 17%
x 83.33−136.66 30%	x 83.33−136.66 27%
x>136.66 48%	x>136.66 56%

Table 3. Comparison of BN variables

results of the controlled experiment [13], it was attempted to illustrate the effects of the antipattern on resolving uncertainty. By comparing the raw data BN model that contains no set evidence with the utilized antipattern BN model (Table. 3), which contains the update beliefs for the set evidence (Mixed Personality = Yes, Mixed Temperament = Yes) very interesting observations were made on how the certainty of the BN variables was affected by the proposed antipattern. The certainty of shortening the longest development by using the proposed antipattern was improved significantly. The probability of developing the two tasks in the upper interval of more than 105 minutes has decreased by 7%, increasing the probability of achieving a lower development time in the medium range interval (75 and 105 minutes) significantly (8%). The proposed antipattern also had a strong effect on the certainty of improving design correctness, which can explain the effects of the antipattern on the collaboration and software quality. This was indicated by decreasing the probability of low range (<4.33) values by 12% and increasing the medium range (4.33-6.66) and high range (>6.66) values by 10%. Furthermore, a significant increase of the certainty in achieving an increased number of total communication transactions was illustrated by using the proposed antipattern. The probability of increasing the medium range (39 - 66) and high range (>66) values of the total communication transactions in the two tasks has increased by 28%, while the probability of having a reduced number of total communication transactions (<39) has decreased by 28%. The results also indicated the positive effect of the proposed antipattern on the certainty of improving collaboration. The probability of having of a low range (<2.33) value of collaboration has decreased by 4%. Additionally the probability of achieving a high range (>3.66) value has increased by 5%. Furthermore, the probability of a medium range (2.33-3.66) value has decreased by 1%, which is not a significant finding compared with the findings concerning the higher and lower range interval values. Finally, the probability of the Quality variable was the most interesting one to observe. The probability of having of a low range (<83.33) value on the two acceptance tests has decreased by 5%. The probability of a medium range (83.33-136.66) value has decreased by 3%, which is not a very significant finding. Most importantly, the probability of achieving a high range (>136.66) percentage on both acceptance tests has increased by 8%. These results clearly indicate the positive effect of the proposed antipattern on the certainty of achieving a higher percentage on both acceptance tests. This illustrates the statistical improvement of the software quality variable using the proposed antipattern.

4. Conclusion

In this paper we proposed a Bayesian network approach for modelling software project management antipatterns. Considering the heterogeneous developer personalities and temperaments as an antipattern, our model explicitly captured project management uncertainty. Bayesian modelling was particularly useful and well suited to the domain of antipatterns because it provided a solid graphical representation of the probabilistic relationships among the set of variables. This formalism served as the underlying reasoning engine to support project management antipatterns and develop a precise mathematical model that can be used by project managers to illustrate the effects of uncertainty on a project management antipattern. The proposed antipattern BN model can be used in situations where there is scarce statistical data. As a result project managers can use their expert judgment to model an antipattern.

The applied antipattern BN model (Table 3) has illustrated better communication and collaboration, reduced development time, increased design correctness and eventually a significant improvement in software quality. This finding can help project managers to improve the effectiveness of pair developers by identifying and matching the developers' personality and temperament types to their potential roles and tasks, effectively exploiting their differences in pair formations and rotations.

It is therefore suggested that a richer set of data from empirical investigations would be more helpful in representing project management antipatterns. Besides that, an advanced graphical tool support will be very necessary for the analysis of complex Bayesian Networks. Finally, the use of BN modeled antipatterns in the absence of statistical data must also be investigated.

References

[1] Fox Terry L., Spence J. Wayne, ”The Effect of Decision Style on the Use of a Project Management Tool: An Empirical Laboratory Study”, *The DATA BASE for Advances in Information Systems*, Vol. 36(2), pp. 28 - 42, Spring 2005.

[2] Robert Hughes, Michael Cotterell, *Software Project Management*, Second Edition, McGraw-Hill Publishing Co., 1999.

[3] William J. Brown, Hays W. ”Skip” McCormick III, Scott W. Thomas, *AntiPatterns in Project Management*, Wiley Computer publishing, 2000.

[4] Chin-Feng Fan, Yuan-Chang Yu, ”BBN-based software project risk management”, *The Journal of Systems and Software*, Vol. 73, pp. 193-203, 2004.

[5] William J. Brown, Raphael C. Malveau, Hays W. ”Skip” McCormick III, Thomas J. Mowbray, *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis*, Wiley Computer publishing, 1998.

[6] Philip A. Laplante, Colin J. Neil, *Antipatterns: Identification, Refactoring, and Management*, Taylor and Francis, 2006.

[7] Hanakawa Noriko, Okura Kimiharu,”A project management support tool using communication for agile software development”, in *Proceedings of the 11th Asia-Pacific Software Engineering Conference (APSEC’04)*, 2004, pp. 316-323.

[8] Bibi, S. and Stamelos, I. ”Software Process Modeling with Bayesian Belief Networks”,in *Proceedings of 10th International Software Metrics Symposium (Metrics 2004)*, 14-16 September 2004, Chicago, USA.

[9] Norman Fenton, William Marsh, Martin Neil, Patrick Cates, Simon Forey Manesh Tailor, ”Making Resource Decisions for Software Projects”, in *Proceedings of the 26th International Conference on Software Engineering (ICSE’04)*, May 2004, Edinburgh, United Kingdom, pp. 397-406.

[10] Khodakrami V., Fenton N., Neil M., ”Project Planning: Improved approach incorporating uncertainty”, Track 15, *European Academy of Management Annual Conference (EURAM 2005)*, TUM Business School, Munich, Germany, May 4-7th 2005.

[11] Stamelos, I., Angelis, L., Dimou, P., Sakellaris, P. ”On the use of Bayesian belief networks for the prediction of software productivity”, *Information and Software Tech*, Vol. 45(1), pp. 51-60, 2003.

[12] Craig Larman, *Agile and Iterative Development. A Manager’s Guide*, Addison-Wesley, 2004.

[13] Panagiotis Sfetsos, Ioannis Stamelos, Lefteris Angelis, Ignatios Deligiannis, ”Investigating the Impact of Personality Types on Communication and Collaboration-Viability in Pair Programming - An Empirical Study”, *XP 2006 conference*, to appear.

[14] Matt Stephens, Doug Rosenberg, *Extreme Programming Refactored: The Case Against XP*, Apress, 2003.

[15] Kuranuki Y., Hiranabe K., ”Antipractices: AntiPatterns for XP Practices”, *Agile Development Conference*, 2004.

[16] Meredith Belbin, *Management Teams: Why They Succeed or Fail*, Butterworth-Heinemann, 1996.

[17] Narasimhaiah Gorla, Yan Wah Lam,”Who should work with whom? Building effective software project teams”, *Communications of the ACM*, Vol. 47(6), pp. 79-82, June 2004.

[18] George Stepanek, *Software Project Secrets. Why Software Projects Fail*, Apress, 2005.

[19] Laurie Williams, Robert Kessler, *Pair Programming Illuminated*, Addison-Wesley Professional, 2002.

[20] Hadar Ziv, Debra J.Richardson, ”The uncertainty principle in Software Engineering”, in *Proceedings of the 19th Conference on Software Engineering (ICSE 1997)*, Boston, MA, 1997.

[21] Finn V. Jensen, *Bayesian Networks and Decision Graphs*, Springer, 2001.

[22] Cheng, J., 1998, ”PowerConstructor System”. `http://www.cs.ualberta.ca/~jcheng/bnpc.htm`

[23] Cheng, J., 2000, ”PreProcessor System”. `http://www.cs.ualberta.ca/~jcheng/prep.htm`

[24] Druzdzel, M., 2005, ”Genie 2.0 System”. `http://genie.sis.pitt.edu/about.html`

[25] Myers Isabel, *Manual: The Myers-Briggs Type Indicator*, Palo Alto, California: Consulting Psychologists Press, 1975.