Knowledge Acquisition during Software Development: Modeling with anti-patterns.

Paraskevi Smiari, Stamatia Bibi, Ioannis Stamelos

Abstract Knowledge is a strategic resource; that should be timely acquired and exploited to manage and control software development. Software development is a knowledge intensive process characterized by increased uncertainty, presenting large variations among different development environments. Project uncertainty and volatility confounds the traditional knowledge-based processes since at any time traditional software project management techniques and patterns may be considered out of scope. In this chapter a dynamic and constantly adaptive knowledge encapsulation framework is presented. This framework analytically describes a) metric collection methods along with metrics that attribute to knowledge creation regarding successful software development b) representation mechanisms of the knowledge created in the form of anti-patterns c) Bayesian Network analysis technique for converting the data to knowledge allowing inference mechanisms for testing the applicability of the anti-pattern. The presented approach is demonstrated into a case study showing both its feasibility and applicability.

1 Introduction

Software development is a cross-disciplinary cognitive activity requiring knowledge from several different domains (Terry and Wayne 2005). Human knowledge-

Paraskevi Smiari
Department of Informatics & Telecommunications Engineering, University of Western Macedonia, Greece, e-mail: psmiari@uowm.gr

Stamatia Bibi
Department of Informatics & Telecommunications Engineering, University of Western Macedonia, Greece, e-mail: sbibi@uowm.gr

Ioannis Stamelos
Department of Computer Science, Aristotle University of Thessaloniki, Greece, e-mail: stamelos@csd.auth.gr

creating activities and past experience should contribute to the constantly evolving software development knowledge base according to Davenport et al. (Davenport and Prusak 2000). The key to systematic software development and process improvement is the decent management of knowledge and experience. Many techniques (Shepperd et al. 1996), (Lucia et al. 2005) have been applied to manage knowledge regarding software development in order to improve aspects of the whole process such as, cost estimation and project management, fault identification and quality of the outcome. These aspects are compound by vulnerabilities such as changing requirements, team dynamics and high staff turnover (Aurum et al. 2003).

Capturing and representing software project management knowledge while catering for its evaluation are some of the mechanisms of great importance. This book chapter provides a framework for effectively capturing the knowledge created during software development. This framework consists of three distinct phases:

- The first phase includes the data collection process that could attribute to knowledge creation. Adopting the appropriate qualitative and quantitative data collection approaches is of major importance for capturing the relevant information.
- The second phase is related to transforming the information and data collected from the previous step in the form of anti-patterns. Each anti-pattern represents a management problem and will be described as a set of symptoms, suggested solutions and identification factors.
- The final phase includes the representation of the anti-patterns created in the form of a Bayesian Network (BN) in order to be able to proceed to the validation process of the derived knowledge.

The final output of this framework will be an anti-pattern knowledge base in the form of Bayesian Networks available to managers and developers that can serve as a management toolkit.

Software project management anti-patterns propose frequently occurring solutions (Brown et al. 2000) to problems that have to do with flawed behavior of managers or extensive management practices that prevent a software project from successful (Laplante and Neil 2006) implementation. Nonetheless, the documentation of anti-patterns is conducted using informal or semi-formal (Eloranta et al. 2016) structures that do not readily encourage the reusability and sharing of knowledge. In addition, the number of defined anti-patterns and the number of printed documentation is expanding to the point that it becomes difficult for it to be effectively used. Thus better structured anti-pattern representations are required in order for them to become a widespread practice.

By applying the BN (Settas et al. 2006) representation formalism to anti-patterns we can gain useful insight about the knowledge created during software development and also perform post mortem analysis. The BN model offers a convenient mechanism to model and disseminate knowledge regarding software management anti-patterns which incorporates uncertainty. Re-factorings proposed by the anti-patterns can be tested to view their reflection to the process. The suggested model can be used by project managers to illustrate the effect of an anti-pattern solution in the process.

In order to reinforce the applicability of the proposed framework a case study is presented that exemplifies the process of creating the anti-pattern BN models. In particular, we use as a pilot the "BENEFIT" project, an ambitious, highly innovative cross disciplinary crowd-sourcing platform for tourism marketing, where the "tech-aware" anti-pattern was actually formulated and assessed. In brief the problem that BENEFIT project was faced from the early stages of implementation was that there were many technical conflicts and disagreements that inhibited project progress. Four new metrics were introduced that described better the development status namely: team synthesis, organizational structure, project integration and product innovation. The "tech-aware" anti-pattern was then formulated to test the impact of changing the organizational structure in order to better monitor technical conflicts.

This paper is organized as follows: section 2 describes analytically the three steps of the knowledge based framework providing guidelines for acquiring and modeling the knowledge in the form of BN anti-pattern models. Section 3 presents the BENEFIT case study were the three steps of knowledge acquisition and assessment are exemplified. Finally, in section 4, we conclude the chapter and summarize the knowledge acquisition framework.

2 The Knowledge Acquisition Framework

In this section, we present the framework that is suggested for guiding the process of acquiring knowledge during software development. The framework consists of three distinct phases a) collecting data during development b) modeling the data in the form of anti-patterns c) representing the anti-pattern with the appropriate BN model and assessing its applicability. Therefore, in Section 2.1 possible methods for collecting data along with the relevant metrics that can be valuable in assessing a software process are presented, in Section 2.2 the representation formalism of anti-patterns is described and in Section 2.3 the Bayesian Networks method is presented.

2.1 Collecting the relevant data

In this section we describe the methods that can be used to acquire the relevant data that can help us realistically capture the software development status quo of a team. Additionally possible sources of data and metrics are recorded in order to offer practitioners a library of metrics that can help in acquiring knowledge relevant to software development processes.

2.1.1 Data collection methods

In this section two complementary types of data collection methods will be presented namely qualitative and quantitative methods (Fenton and Bieman 2014). Both of these methods can be used to record primary data, data coming directly from the source (respondents, activities logged, crowd sourcing, etc.) or secondary data, historical data that come from experiments, aggregations and measurements.

Data collection approaches for qualitative research usually involve direct interaction and contact of the observer with the key individuals or groups whose opinion is considered valuable. Among the qualitative methods we can find a) interviews b) participant observation c) focus groups d) questionnaires/testing.

Interviews, which can be structured, semi-structured or unstructured are used to collect experience, opinion and emotions of the interviewee. A software project manager, formally or informally, "interviews" the members of the development team and holds the behavioral aspects of the responses. It should be mentioned that from the interviews large amount of data can be derived that can provide the "beat" of the development team. These data are rich, provide a wealth of information coming from the primary actors of the development process but due to their intangible nature remain often unrecorded and therefore unexploited. **Focus groups** method actually is a form of observation of a group of people that can interact and exchange opinions letting us gain useful insight regarding the participants' behavior and attitude. On the other hand modern paradigms of this method include data coming from online social networking platforms such as company blogs, employees' facebook, collaborative development tools, etc. **Participant observation** involves the intensive involvement of the observer with the software development team. Finally **questionnaires**, is among the most common methods to collect data that records the opinion of the respondents in the form of answers that can be unstructured or calibrated in a certain scale. In the case of software development process we should mention that interviews in the form of discussions, focus groups as blogs and collaborative tools and short questionnaire completion during and after the development of a project could provide a valuable source of data.

Quantitative methods on the other hand are less personalized and more unbiased. Among quantitative methods we can find a) surveys/polls b) automatic collection of data. **Surveys** are like questionnaires but they are completed anonymously, usually without personal contact and usually involve a large sample. In the case of project development teams that would mean that a survey could be an anonymous evaluation of certain aspects of the process during or after the completion of the project. **Automatic methods** may include direct data coming from systems that monitor or aid the development process and record information regarding the development progress (log files, nof active members, number of teleconference held etc.). Quantitative methods are more accurate and present greater objectivity compared to qualitative methods. Still both types of data collection methods are required since both human perceptions and numerical descriptions are necessary for fully recording the status of a software development process.

2.1.2 Software development metrics

Since software development process can present significant deviations compared to the traditional models found in literature (Dalcher et al. 2006) being able to identify the relevant metrics that will help us monitor, assess, update and finally improve a process is very crucial. The classical 4 Ps in software project management, **people, product, process** and **project** (Fenton and Bieman 2014) can help us identify the relevant metrics that can better describe the software process.

Metrics regarding the stakeholders of a software development project are necessary to address the coordination and cooperation efficiency of the team. Among the groups of **people** that affect the team efficiency are the senior manager, the project manager, the development team with all discrete roles (analyst, programmer, tester), the customer and the end-users. Metrics that can describe the efficacy of each role are presented in table 2.1. Typical examples of metrics describing the people involved in software development are those expressing the experience of the team (analysts' capabilities, familiarity with the application domain). Cultural characteristics also affect the performance of a team as well organized teams whose leadership encourage communication and knowledge exchange offering rewards are more productive compared to impersonal teams. User and customer involvement is also appointed by recent studies as a critical success factor of software projects. Therefore, customer participation and end user involvement are among the parameters that need to be quantified.

Table 2.1 People Drivers

People Drivers	Driver	Metric
Experience	Analysts capabilities	1-5 Scale
	Programmers experience	1-5 Scale
	Familiarity with the problem domain	1-5 Scale
Cultural issues	Reward mechanisms	1-5 Scale
	Collaboration	1-5 Scale
	Capable leadership	1-5 Scale
Stakeholders participation	Customer participation	1-5 Scale
	End-user involvement	1-5 Scale

Measurement of **process** attributes is important for establishing a baseline for assessing improvements and identifying possible process flaws. All methods, techniques, tools and supplements that may be used to support the development process should be quantified and recorded in order to be able to measure the efficiency of the development process. Among these attributes the use of CASE (Computer Aided Software Engineering) tools, the utilization of models, techniques and standards are the main aspects that define the level of support and observation of the development procedure. Following a well-defined and guided process customized to each company needs is crucial for delivering quality software within time and budget constraints. To achieve this target the process should be constantly measured

and improved to reach the quality goals of the company. Table 2.2 presents process drivers.

Table 2.2 Process Drivers

Process Drivers	Driver	Metric
Use of Case Tools	Versioning tools	% of usage
	Analysis & Design Tools	% of usage
	Testing Tools	% of usage
Management Process	Use of lifecycle models	Yes or No
	Managed development Schedule	1-5 Scale
Methodologies	Existence of best practices	1-5 Scale
	Software Reuse	% of the total LOC

Project attributes related to a software project include relevant variables that describe the type and the size of the project. The aggregation of project variables offers an indicator of the complexity of the project preparing the management for the risks and difficulties that may appear. Project attributes can be descriptive variables referring to the development type of the project, the application type and the user type of the application. Size attributes can be an initial assessment of functional requirements measured in function points or at later stages in the development process measured as Lines of Code (Boehm 1981). Table 2.3 summarizes project drivers.

Table 2.3 Project Drivers

Project Drivers	Driver	Metric
Type of project	Application Type	ERP, Telecom, Logistics, etc.
	Business Type	Medical, Public Sector, Transports, Media, etc.
	Development Type	New Development, Re-development, Enhance
User type	Level of usage	Amateur, Professional, Casual
	Number of Users	1-50, 50-200, 200-1000, $>$1000
Size	Source Code Lines	Lines of Code (LOC)
	Function Points	Number of Function Points

Product attributes are constraints imposed on the software by the hardware platform and the utilization environment. Such constraints include run-time performance, memory utilization, performance standards and transaction rates. Table 2.4 summarizes product metrics.

Table 2.4 Product Drivers

Product Drivers	**Drivers**	**Metric**
Technical attributes	Distributed Databases	1-5 Scale
	On-line Processing	1-5 Scale
	Data communications	1-5 Scale
	Back-ups	1-5 Scale
	Memory constraints	1-5 Scale
	Use of new, immature technologies	1-5 Scale
Non-functional requirements	Reliability	1-5 Scale
	Performance	1-5 Scale
	Installation Ease	1-5 Scale
	Usability	1-5 Scale
	Security	1-5 Scale

2.2 Forming the anti-pattern

A project manager needs to ensure that the management of the 4 Ps as described earlier is carried out effectively without any arising issues. In the occurrence of these issues, anti-patterns play a significant role due to their ability to describe commonly occurring solutions to the problems that lead to undesired results (Brown et al.2000), (Stamelos 2010). Anti-patterns propose re-factored solutions that can combat problems with reference to flawed behavior of managers or pervasive management practices that constrain a software project from being successful (Laplante and Neil 2006). Any reader who is not acquainted with anti-patterns can start with (Brown et al.2000), (Laplante and Neil 2006) as introductions to the matter.

Table 2.5 Pattern "Is Five the Optimal Team Size" central concept

Name	"Is Five the Optimal Team Size?"
Central Concept	It is agreed by most Agilists that smaller teams can be more functional and productive in comparison to larger teams. The definition of the optimal team size is, however, still a challenge. In order to produce more code, large teams are still being used. Nonetheless, a team size of 5 [Hazrati 2009] seems to satisfy all the conditions related to Scrum recommendations, Parkinson's Law, natural limit of short term memory and favorable communication channels. Software managers neglect to comprehend the influence of organizational and environmental matters on choosing the ideal team size.
Dysfunction	This anti-pattern is available to the board of managers or the software project manager, who picked the amount of an agile team without taking into account the characteristics of the organization and/or the software project.
Explanation	The purpose of this anti-pattern is the decision that the optimal team size is a team of 5 or a large team in order to produce more code.

One way, to identify potential problems and provide a refactored solution in a practical and reusable manner, is by capturing and representing tacit knowledge in

the form of an anti-pattern template. An anti-pattern template (Laplante and Neil 2006) is an informal presentation of the anti-pattern that depicts the management dysfunction and the remedies for all those engaged. The anti-pattern template carries out insight into the causes, symptoms, consequences and identification of the problem suggesting band-aids and refactoring. For example in Table 2.5 the encoding of the problems related with choosing the optimal agile team size is done by the "Is Five the Optimal Team Size?" anti-pattern (Hazrati 2009) (Table 2.6). The anti-pattern is described by a central concept, the dysfunction that it presents, a short explanation of the causes of the problem, a band-aid that suggests a short-term solution strategy, self-repair mechanisms for selecting the appropriate solutions, re factoring solutions and finally identification questions that will help a manager verify whether his team is suffering from the particular anti-pattern.

Table 2.6 "Is Five the Optimal Team Size" anti-pattern refactoring

BandAid	No band aid exists for this anti-pattern. No short term management strategy can be used to handle this problem.
SelfRepair	Project managers need to consider the environmental and organizational concerns that influence the ideal team size. They should start by identifying key issues that influence the success of a software project given the chosen team size.
Refactoring	It is important that before determining the size of an agile team. Software project managers should first determine the concerns that influence the optimal team size and how these concerns affect the progress of the project. Software managers have to accept that there is no generic team size that can be used for all agile projects and that each project has exclusive characteristics that need to be taken into account. Management needs to look out for the team location (collocated or not), the organization size, iteration length, and the existence of offshoring teams. Past project data can be used to detect how different team sizes behave in the same company. This will grant managers help to calculate the impact of the chosen team size on the success of the project.
Identification	The following questions should be answered with a "Yes" or "No". - Has the organization used the same agile team size repeatedly? - Does the agile project manager always use a team size of 5? - Does the agile project manager always use a large team size? If you responded "Yes" to one or more of these statements, your organization is probably suffering from "Is Five the Optimal Team Size?"

2.3 Modeling the anti-patterns with Bayesian Networks

In this section we present the Bayesian Network Models background theory (Jensen 2001) and provide a short example coming from the project management domain. Bayesian Network models known also as Bayesian Belief Networks are casual net-

works that form a graphical structure. These graphical structures consist of nodes and links between those nodes without, however, forming a cycle with each other. This is the reason why they belong in the Directed Acyclic Graphs (DAGs) family and these graphical structures are popular in the fields of statistics, machine learning, artificial intelligence and are used to handle uncertainty in software development application domains as process modeling (Bibi and Stamelos 2004), (Bibi et al. 2014), (Fenton et al. 2004), defect prediction (Okutan and Yildiz 2014) and cost estimation (Khodakarami and Abdi 2014),(Stamelos et al. 2003). Specifically, each node represents a random variable that has a finite set of mutually exclusive states. Furthermore, each link represents probabilistic cause-effect relations between the linked variables.

This type of network is used to track how a change of certainty in one variable can cause an effect on the certainty of other variables. The relation that links the two nodes can be seen on Bayes' rule:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

For each node A with parents B1, B2,... ,Bn there is attached an NxM Node Probability Table (NPT), where N is the number of node states and M is the product of its cause-nodes states. In this table, each column represents a conditional probability distribution and its values sum up to 1.

Example

In Figure 2.1 we present a simple BN model were two nodes "Mixed Personality" and "Design Correctness" affect the node "Development Time". Mixed Personality is an indicator of heterogeneous developer personality and its states are Yes or No. The Design Correctness measures the points in 1-10 scale, obtained by each pair for all the tasks and is divided in two categories, False for points below 5.4 and True for points above 5.4. The Development Time indicates the total time that took the pair to complete the tasks and it is divided in two categories, Low for total time below 90 minutes and High for total time above 90 minutes.

The BN model contains 3 variables and 2 connecting links. We have one converging variable connection. In a converging connection if nothing is known about Development Time except what may be inferred from knowledge of its influencing variables (parents) Mixed Personality and Design Correctness, then the parents are independent. This means that evidence on one of them has no influence on the certainty of the others. If Development Time changes certainty, it allows communication between its parents.

Specifically, if Mixed Personality is Yes and Design Correctness is True, then there is a 20% probability that the Development Time will be Low and 80% probability that it will be High.

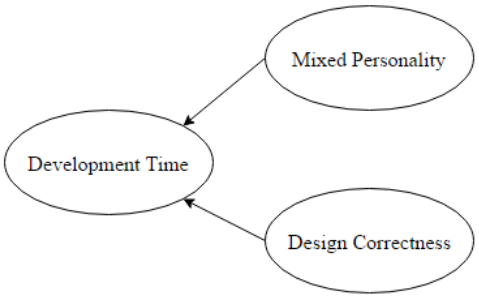

Fig. 2.1 A BN example for modeling "Development Time"

Table 2.7 Node Probability table for the BN model of figure 2.1.

	Mixed Personality	**Yes**		**No**	
	Design Correctness	**True**	**False**	**True**	**False**
Development time	**Low**	0.2	0.7	0.8	0.4
	High	0.8	0.3	0.2	0.6

3 The Case Study

This section exemplifies the framework of section 2 by presenting an actual case where the knowledge acquired during software development were modeled in the form of anti-pattern to be diffused and assimilated into new projects. In this section we will describe analytically the software development project under study. In section 3.2 we will present the data that were collected during development, in section 3.3 the new anti-pattern model and in section 3.4 the knowledge-based models representing the proposed anti-pattern.

3.1 CASE STUDY: The BENEFIT Platform

The BENEFIT Platform was designed to provide a solution catering to arise brand marketing, awareness and advertising capability of the tourism sector by providing an on-line company platform that will offer (a) specialized marketing toolkits available to the wider public, (b) advanced crowdsourcing tools to process tourism experience and review data in order to extract and present collective knowledge, (c) advanced forecasting models exploiting the tourism market sentiment to identify market trends and threats, (d) novel personalized recommendation systems to support marketing decisions according to the company's profile.

The project was funded by a Greek local tourism association in order to promote the tourism marketing of the area and arise awareness of the local tourism business sector regarding the arising trends of crowd- sourced tourism innovations. BEN-

EFIT was an ambitious project and involved the participation of 5 partners with different backgrounds, the first partner (Partner 1) was a university department specializing in the area of Marketing and Business Administration, the second partner (Partner 2) was a university department specializing in Informatics, the third partner (Partner 3) was a Web Company specialized in crowd sourcing applications, the fourth partner (Partner 4) was an on-line marketing company and Partner 5 was the Greek local tourism association. Partner's 1 role was to suggest marketing and socioeconomic models to predict market trends based on the data provided by the crowdsourcing tools, Partners 2 and 3 were responsible for designing and implementing the BENEFIT platform. Partner 4 and Partner 5 would pilot the platform and perform usability tests. All partners were responsible for providing initially the requirements of the BENEFIT platform and defining user stories to be implemented.

For the development of the project three distinct Committees were defined that would manage the development of the BENEFIT platform a) the Project Coordination Committee b) the Work packages committees and c) The Quality Assurance Management Committee. The Project Coordination Committee, consisting of representatives of all partners, was the principal management authority of the BENEFIT project having the final steering and controlling commitment to the project, making decisions on contractual, administrative and technical matters attempting to ensure timeliness and cost effectiveness. The Work Packages Committees was one for each work package and consisted of representatives from partners participating in each work package and were responsible for the monitoring the progress of each WP. The Quality Assurance Management Committee, including representatives from all partners, were responsible for providing internal quality assurance guidance to the BENEFIT project, outlining the policies, the purpose, the organization, the procedures and the responsibilities related to ensuring high quality performance of all activities and also collecting metrics for measuring and improving the process development.

3.2 Data collection

The Quality Assurance Committee (QAC) due to the fact that the same partners had a long series of projects in which they cooperated was determined to collect data and metrics that would help in the direction of improving the development and implementation process. Therefore from this project and on they decided to create a repository of anti-patterns coming from the "lessons-learned" during each project development (Silva et al. 2015) summarizing the knowledge created by past projects into the form of anti-patterns and adding also new knowledge created during the development of the BENEFIT implementation.

The QAC team decided that extra metrics would be important to better describe the complexity and specialty of BENEFIT project. Several additional metrics were recorded as the team synthesis, the structural organization, the tools integration com-

Table 3.1 The new metrics collected for the BENEFIT project

Metric	Type	Explanation	Possible Values
Team synthesis	People	This metric depicts the appropriateness of the team synthesis. Complementary teams consist of members with complementary skills that can work towards a certain direction, heterogeneous teams consist of members that have diverse orientation and homogeneous teams consist of members possessing the skills on the same domain.	Complementary teams Heterogeneous teams Homogeneous teams
Structure Organization	Process	Represents the organizational structure of the team emphasizing on whether the leader-ship is performed by a management committee, a technical committee or a combination of the two	Management Committee (M) Technical Committee (T) Combination (M+T)
Project Integration	Project	The cost of integrating different tools in a single application.	Low Average High values (depending on the integration complexity and time required.)
Product innovation	Product	Product innovation expresses the difficulty to perform to meet the requirements of an innovative application in our case is the importance of producing accurate estimation results forecasting behaviors on the tourism domain	Low Average High values (depending on the level of innovation required.)

plexity and the product innovation. These parameters are analytically described in table 3.1 and provided the key to form the anti-pattern presented in section 3.3.

3.3 The "tech- aware manager" anti-pattern

Due to the technical problems the QAC decided to form a Technical Management Committee who would be responsible for the overall technical management of the project, monitoring the advances performed, ensuring effective coordination of work packages and timely knowledge exchange. The QAC team formed the "tech-aware" antipattern as presented in tables 3.2 and 3.3 to depict the necessity of the Technical Management Committee.

The TMC assesses at first-level progress reports, resolve any internal technical conflicts between work packages and plans resources re-allocation if required. The

Table 3.2 Pattern "tech- aware" central concept

Name	"tech-aware"
CentralConcept	The manager of a software development project should possess leadership competencies and abilities to motivate, inspire and encourage the development team, regardless whether he possess technical competencies or not. That is true still that depends on the technical complexity of the project under development. On the other hand is a technical manager enough to lead a complex software application development? May be only in projects with small development teams?
Dysfunction	This anti-pattern is attributable to the project management board that cannot solve technical problems. Symptoms are: 1. Disagreements regarding technical issues 2. Conflicts in the allocation of resources 3. Difficulty in system integration 4. Limited knowledge dissemination between work packages
Explanation	The cause of this anti pattern is the fact that no technical experts have been formally appointed to monitor development progress and solve technical conflicts.

TMC members maintain a constant communication via audio/video conferencing, emails and schedule regular meetings every 3 to 4 months. The TMC consisted of the Project Coordinator and the Work Package Technical representatives and is chaired by the Technical Manager. The Technical Manager was a senior technical member of the team and was enrolled to lead the technical activities of the project, organize the TMC meetings, prepare the agenda, keep and share the minutes.

After this change in the project organizational structure the development of BENEFIT proceeded without conflicts and internal disagreements as the TMC monitored the development progress and mitigated any technical risks that occurred.

Table 3.3 The "tech- aware" anti-pattern refactoring

Band-Aid	Form a technical management committee that will contain representatives that possess technical knowledge from all conflicting stakeholders. Appoint a technical manager responsible for taking decisions.
Self-Repair	Project managers should examine technical complexity of a project and its size during the early stages of development. If they feel that their technical background is limited within the scope of the project severity they need to form a technical committee.
Refactoring	It is essential that before deciding on the organizational structure and management of a project to assess the unique characteristics of the application under development. Software project managers should first identify among others the technical issues that may jeopardize the success of the project, the team competencies, the technological risks, the integration costs. Based on all the special attributes of the project the organizational structure may include a technical committee, a quality assurance team or in more complex projects, an external advisory board. On the other hand in small agile projects the project manager can also be a technician. Software managers need to understand that there is no generic team organizational structure that can be used for all development projects.
Identification	The following questions should be answered with a "Yes" or "No". - Does the project depend on immature technology? - Does the project require increased integration efforts? - Is the development team heterogeneous with different back-grounds? - Is the development team disagreeing usually when taking technical decisions? - Is the project relatively complex compared to the other ones that the team has developed? If the answer is yes to at least two of these questions then the development team is suffering from this anti-pattern

3.4 Knowledge-based models of the "tech-aware" anti-pattern

In this section we model the anti-pattern presented in the previous section with the help of Bayesian Networks representation formalism in order to investigate the relationship between the identification factors that help us diagnose the problematic situations and test the impact of the refactored solution on the project progress. The proposed BN model presented in figure 3.1 consists of a set of nodes that represent People, Process, Project and Product drivers (P variables from now and on). Each one of these nodes is affected by the standard metrics described in tables 1 to 4 whose values are represented cumulatively in the first affecting node (for People

we have the node Team Competencies, for Process the node Process Organization, for the Project the node Project Complexity and for the Product the node Product Complexity).

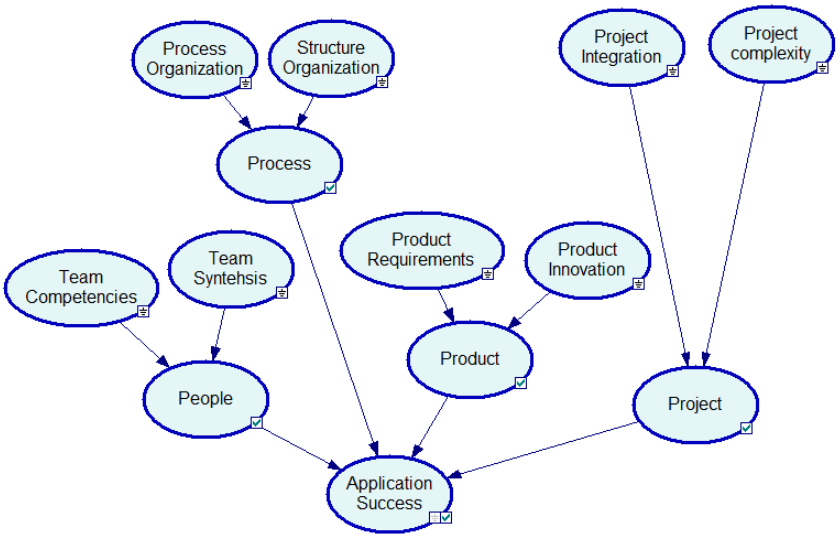

Fig. 3.1 The BN model of the "tech-aware" anti-pattern

In order to test the influence of the anti-pattern we need to insert in the BN model the new metrics identified in table 3.1 as nodes that describe the "tech-aware" anti-pattern and affect the relevant P variables. As a consequence four new nodes are inserted in the model: Team synthesis now affects People node, Organizational structure affects Process node, Project Integration affects Project node and Product Innovation affects Product node. At this point the standard drivers that affect application development success are now co-influencing along with the tech-aware anti-pattern nodes the application development success.

Figure 3.1 represents the Bayesian Network as it is formed after the representations mentioned previously. In this study the values of the Node probability tables were defined based on the historical data collected at phase 1 and on the experience and the knowledge gained during the development of past projects from experts (QAC team) that participated in the BENEFIT team and had in the past a long series of co-operations. The tools used to construct the network and create the node probability tables can be found in (Cheng 1998). Figure 3.2 presents the initial belief status of the network based on the node probability tables defined by the experts. For example this network provides the following information regarding the development of projects from the BENEFIT team: In 50% of the projects the team organization was considered to be high while 70% the projects where leaded solely by a man-

ager and in totally the people driver was considered in 46% of the cases to be high meaning that the development team and was well organized and structured. The rest of the information provided by the network is interpreted accordingly.

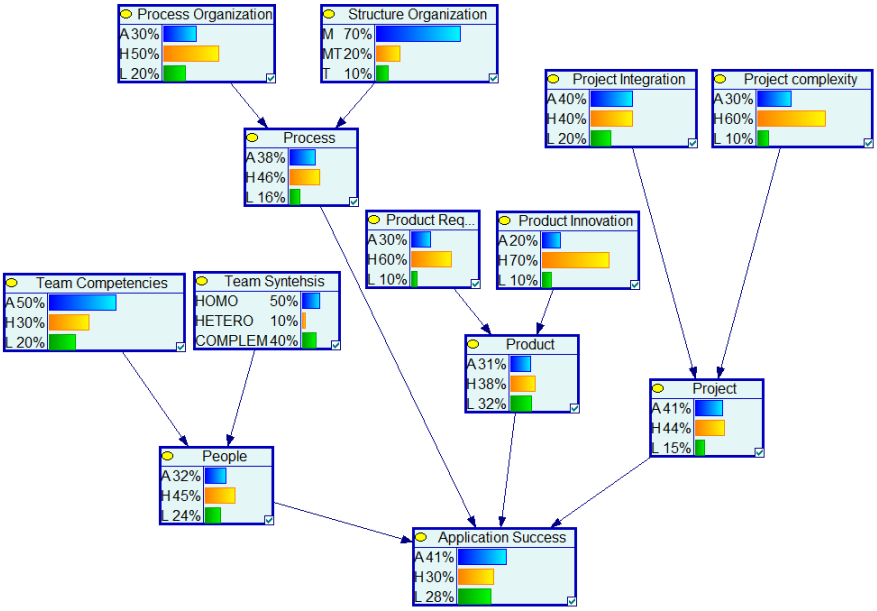

Fig. 3.2 The BN belief model for the BENEFIT team based on historical data

This Bayes Network can then be useful for applying inference. Certain inferences can be made to show how the change in the values of a metric can affect the values of another metric and, finally, reach some conclusions regarding good and bad practices in software project planning that lead to development success.

In order to apply inference, the answers to the identification questions of table 3.2 and table 3.3 are now expressed as values that initialize the anti-pattern nodes to a certain state that can more accurately and objectively describe current status of a project. Figure 3.2 shows an instance of the Bayes Network when it is instantiated with data coming from the BENEFIT project. We tested the BN model with the actual values of the metrics that represented the BENEFIT project and the Bayesian Network was updated to the one of figure 3.3. The BENEFIT platform was a highly innovative product, incorporating the need of integration of various tools and was developed by a team possessing complementary skills led both by a project management committee and a technical management committee. Changing the values of the relevant nodes we observe that the probabilities of the affected nodes also changed. The refactored solution to the “tech-aware” anti-pattern has a positive impact on the Application Success. On the other hand it would be interesting to test such a change in projects with low innovation, low project complexity and homogeneous develop-

ment teams. In such cases the probabilities depict that the appointment of technical management committee would be of no value.

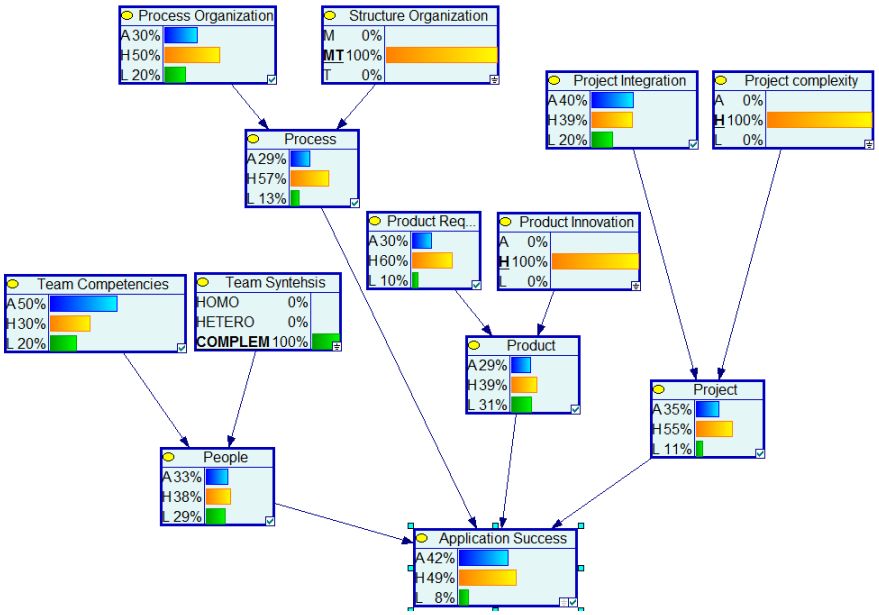

Fig. 3.3 The BN belief model of the BENEFIT project testing the "tech-aware" anti-pattern

4 Conclusions

In this paper we proposed a Knowledgebased framework for acquiring knowledge during software project development and model it in the form of anti-patterns represented by Bayesian Networks. The framework consists of the following phases a) acquire data during development b) identify problems and model them in the form of anti-patterns c) represent and assess the anti-pattern in the form of a Bayesian Network Model. BN modeling is particularly useful and well suited to the domain of anti-patterns because it provides a solid graphical representation of the probabilistic relationships among the set of variables. This approach offers the underlying reasoning engine to support project management decisions providing a formal model that can be used by project managers to illustrate the effects of uncertainty on a software project management anti-pattern.

The suggested approach takes into consideration the characteristics and the needs of the individual software organization under assessment and does not demand a large amount of resources and investment costs. The method provides a generic

Bayesian Network that models application development, which can be tailored to the needs of the development environment, applied. Bayesian analysis can make measurable each concept represented in the 4Ps of project management, People, Process, Project, Product. Bayesian Networks is an easily applied and comprehensible statistical tool that can provide very useful information to software managers. On the other hand the representation form of anti-patterns is a very descriptive and helpful tool in the hands of managers that can help them identify problems and provide easy and quick to launch re-factored solutions.

Acquiring a richer set of data from empirical investigations would be more helpful in creating knowledge from software development. A web-based community of software project management anti-pattern contributors would help in the direction of establishing a freely available, online knowledge base that could provide the tools to evaluate the impact of management decisions and decision support to software project managers worldwide.

References

1. Aurum A., Jeffery R., Wohlin C. and Handzic M. (Eds.)(2003) Managing Software Engineering Knowledge. Springer
2. Bibi, S. and Stamelos, I. (2004) Software process modeling with bayesian belief networks. In Online Proceedings of 10th International Software Metrics Symposium (Metrics 2004)
3. Bibi, S., Gerogiannis, V., Kakarontzas, G., Stamelos, I. (2014) Ontology based Bayesian Software Process Improvenent. ICSOFT EA 2014: 568-575
4. Barry W. Boehm (1981) Software Engineering Economics (1st ed.). Prentice Hall PTR, Upper Saddle River, NJ, USA.
5. Brown, W., McCormick, H., and Thomas, S. (2000) AntiPatterns in Project Management. Wiley Computer publishing.
6. Cheng, J. (1998) Power constructor system, http://www.cs.ualberta.ca/ jcheng/bnpc.htm.
7. Dalcher, D., Thorbergsson, H., and Benediktsson, O. (2006) Comparison of software development life cycles: a multi project experiment. IEE Proceedings - Software, Institution of Engineering and Technology 154 (3): 87-101.
8. Davenport, T. and Prusak, L. (2000) Working Knowledge How organizations manage what they know. Harvard Business School Press.
9. Veli-Pekka Eloranta, Kai Koskimies, Tommi Mikkonen (June 2016) Exploring ScrumBut-An empirical study of Scrum anti-patterns, Information and Software Technology, Volume 74: 194-203
10. Norman Fenton, James Bieman (2014) Software Metrics: A Rigorous and Practical Approach, Third Edition, CRC press
11. Fenton, N., Marsh, W., Neil, M., Cates, P., Forey, S., and Tailor, M. (2004) Making resource decisions for software projects. In Proceedings of the 26th International Conference on Software Engineering (ICSE'04): 397-406.
12. Hazrati, V. (2009) Is five the optimal team size?, http://www.infoq.com/news/2009/04/agile-optimal-team-size.
13. Jensen, F. (2001) Bayesian Networks and Decision Graphs. Springer.
14. Vahid Khodakarami, Abdollah Abdi (October 2014) Project cost risk analysis: A Bayesian networks approach for modeling dependencies between cost items, International Journal of Project Management, Volume 32, Issue 7: 1233-1245.

15. Laplante, P. and Neil, C. (2006) Antipatterns: Identification, Refactoring and Management. Taylor and Francis.
16. Lucia, D. A., Pompella, E., Stefanucci, S. (2005) Assessing effort estimation models for corrective software maintenance through empirical studies. Information and Software Technology, Elsevier 47 (1): 5-6
17. Okutan, A., Yildiz, O., (2014) Software Defect Prediction using Bayesian networks, Empirical Software Engineering, 19(1): 154-181.
18. Settas, D., Bibi, S., Sfetsos, P., Stamelos, I., and 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): 117-124.
19. Shepperd, M., Schofield, C., Kitchenham, B. (1996) Effort estimation using analogy. 18th International Conference on Software Engineering (ICSE' 96). ACM.
20. P. Silva, A. M. Moreno and L. Peters (May-June 2015) Software Project Management: Learning from Our Mistakes [Voice of Evidence], in IEEE Software, vol. 32, no. 3: 40-43.
21. Ioannis Stamelos (January 2010) Software project management anti-patterns, Journal of Systems and Software, Volume 83, Issue 1: 52-59.
22. Stamelos, I., Angelis, L., Dimou, P., and Sakellaris, P. (2003) On the use of bayesian belief networks for the prediction of software productivity. Information and Software Tech 45(1): 51-60.
23. Terry, F. and Wayne, S. (2005) 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 36(2): 28-42.