European Union
European Social Fund

OPERATIONAL PROGRAMME
EDUCATION AND LIFELONG LEARNING
investing in knowledge society
MINISTRY OF EDUCATION & RELIGIOUS AFFAIRS
MANAGING AUTHORITY
Co-financed by Greece and the European Union

NSRF 2007-2013
programme for development
EUROPEAN SOCIAL FUND

SPRINT SMEs PROJECT

SPRINT SMEs Project: Research in Software PRocessImprovemeNT Methodologies for Greek Small & Medium sized Software Development EnterpriseS

Work Package 3 (WP3): Design of SPINT SMEs Knowledge Base

Deliverable 3.2 (D3.2): Design of SPRINT SMEs Knowledge Base

Scientific Coordinator / Project Leader:

Vassilis C. Gerogiannis, PhD, Associate Professor, Dept. of Business Administration, Technological Education Institute of Thessaly, Greece

Authors:

- George Kakarontzas (WP3 Leader), PhD, Applications Professor, Computer Science and Telecommunications Department, Technological Education Institute of Thessaly, Greece
- Ioannis Stamelos, PhD, Associate Professor, Dept. of Informatics, Aristotle University of Thessaloniki, Greece
- Stamatia Bibi, PhD, Research Associate, Dept. of Informatics, Aristotle University of Thessaloniki, Greece
- Leonidas Anthopoulos, PhD, Assistant Professor, Dept. of Business Administration, Technological Education Institute of Thessaly, Greece

May, 2014

Research in SPRINT SMEs project is implemented through the Operational Program "Education and Lifelong Learning" and is co-financed by the European Union (European Social Fund) and Greek national funds in the context of the R&D program ARCHIMEDES III

Table of Contents

List of Figures

List of Tables

1. Introduction

Software Process Improvement (SPI) in the context of small medium-sized software development enterprises (SMEs) is gaining momentum in software engineering research (Pettersson et al., 2008). SPI is a challenging endeavour for most software SMEs aiming at preventing software project failures, reducing development costs and delivering high-quality software products/services consistent with end-customers' needs (Zahran, 1998). The importance of evaluation and improvement of particular software process areas is recognized by most SMEs but the lack of knowledge and resources prohibit SPI adoption and implementation. Software SMEs are often characterized by insufficient human resources, limited development and supporting environment and lack of budget. Therefore, for most SMEs SPI is a major challenge (Mishra & Mishra, 2009).

In this deliverable, a practical approach for supporting the improvement of selected software process areas which take place in a software SME is suggested. The approach is called SPRINT (Software PRocess ImprovemeNT) SMEs and adopts an ontology-based knowledge representation to capture the relevant data that describe a software process. The representation of a process tacit knowledge, through the use of a software process ontology, allows this knowledge to become accessible and transferable. The software process ontology is then represented and analysed in the form of a Bayes Network (BN) (Bibi & Stamelos, 2004). By adopting the BN formalism we can gain useful insight about the elements of the software process and perform post mortem analysis. The use of BNs enables the estimation of process measures (for example, process cost, quality or other measurable artefacts) and adequately handles uncertainty. Thus, the BN process representation can be used as a tool for experimenting with different process changes and testing their effects. In particular, the SPRINT SMEs approach consists of the following steps:

(i) Identification of software process areas of a SME and selection of specific areas which require mprovement.

(ii) Definition of a knowledge base that describes a process area under improvement.

(iii) Conceptualization and analysis of an ontology that represents the process domain.

(iv) BN analysis and suggestions for process improvement.

The deliverable structure is organised as follows. Section 2 provides a brief literature review on the use of ontologies and BNs for software process representation and analysis. Section 3 describes the steps of the SPRINT SMEs approach. Section 4 presents, as a proof of concept, a hypothetical example of applying the approach by using a publicly available data set (ISBSG). In the same section, the approach is validated by considering the software development process that takes place in a SME active in telecommunications area. Finally, in section 5, we conclude the deliverable and present ideas for future work.

2. Ontologies and Bayesian Networks for SPI

The concept of using BNs as predictive models in certain phases of software process is found in several research studies. For example, BNs have been used for handling uncertainty in defect prediction and software quality modelling (Fenton et al., 2002; Fenton et al. 2007). Okutan & Yildiz (2014) applies BNs to determine the probabilistic influential relationships among defect metrics and fault proneness in open source software projects. BNs have been applied for software cost estimation as well. For example, Stamelos et al. (2003) defined an empirically derived BN model for estimating the software development cost and evaluated it on the COCOMO data set. BN models are also useful for estimating the development cost of web applications (Mendes et al. 2007). A survey in research studies using BN models software cost estimation can be found in (Radlinski, 2010). As far as software process representation is concerned, BNs were adopted by (Bibi et al., 2010) to model a customized software development process in a case software company. The process representation through the use of a BN allowed the estimation of certain process aspects, such as defects and effort. BNs were also applied for modelling general software processes, such as the eXtreme Programming (XP) process (Abouelela & Benedicenti, 2010). In addition, the effect of project management anti-patterns on solving cooperation problems in a software development process has been modelled and analysed with the use of BNs in (Settas et al., 2006).

On the contrary, there are rather fewer studies that suggest the use of ontologies to represent a shared conceptualisation of a software process. In (Liao et al., 2005) an OWL-based ontology is suggested for capturing knowledge in software development processes. Falbo & Bertollo (2009) proposed an ontology that was specified with the use of a UML profile to define a vocabulary of concepts met in process quality models/standards, such as ISO/IEC 12207 and CMMI. Barcellos and Falbo (2009) reengineered a Software Enterprise Ontology based on the Unified Foundational Ontology (UFO) suggested by Guizzardi et al. (2008). These works were further extended by Brinquette et al. (2011) to address the conceptualisation of activities which take place in software project planning. Finally, Henderson et al. (2014) recently proposed an ontological infrastructure for representing, in a unified way, the software engineering standards developed under ISO/IEC SC7.

The SPRINT SMEs approach that is suggested in this deliverable utilizes mainly the generic Software Process Ontology proposed in (Brinquette et. al., 2011) with the aim to consider specific project, process and experience concepts. Also in SPRINT SMEs ontology we propose attributes that can be recorded to describe each of the above concepts along with operations (actions) that can be performed for each concept. Ontologies due to their deterministic nature are unable to adequately capture uncertainty. Thus, we consider uncertainty dimensions in the proposed software process ontology by synergizing the ontology with BNs. The benefits of this combination are twofold:

- Process area knowledge is combined with probabilistic information. The software process ontology offers a convenient framework to model and disseminate knowledge regarding the development process which incorporates uncertainty. BNs enable to analytically measure and handle this uncertainty.
- Changes proposed by the ontology actions can be tested to view their reflection to the process. Thus, the BN process model can be used by project/process managers to illustrate the effect of process changes.

3. A Knowledge based approach for supporting SPI activities in SMEs

The SPRINT SMEs approach follows a lightweight paradigm for efficiently improving certain process areas in the context of a software SME. The approach is tailored to the needs of individual SMEs as it is efficient, easily adoptable, non bureaucratic and independent of company's specific assets. The approach follows four steps described in the current section. It should be also noted that the SPRINT SMEs approach presents commonalities with established SPI approaches (Paulk et.al, 1994; ISO, 2013) and, in addition, offers a toolset (comprised by ontologies and BNs) to assist their application.

The first step of the approach involves the identification of a defective process area to be improved. The approach concentrates on supporting the improvement of particular process areas and not the complete software development process. We consider this decision more effective/efficient when addressed to software SMEs since the effort required to improve all aspects of a software process is often prohibitive in terms of time and cost and most SMEs do not possess neither the know-how nor the resources to achieve holistic improvement goals (Pettersson et al., 2008). Defining the software process area that will be set under assessment and improvement is a managerial decision that depends on the needs of a specific SME and the type of projects that it handles. For example, the area under improvement can be decided from traditional software lifecycle models: requirements engineering, design specification, programming and development, software testing, software project management etc.

The target of the second step is to specify and design a knowledge base that consists of information relevant to the knowledge required for improving the area(s) selected in the previous step. A knowledge base is a database that stores data and rules for knowledge management (Simari & Rahwan, 2009). Knowledge management (KM) refers to the set of practices adopted in an organisation to identify, create, represent, distribute, and enable adoption of insights and experiences (Nonaka &

Krogh, 2009). Such insights and experiences comprise knowledge, either embodied in individuals or embedded in organisations, such as processes or practices (Thomas, 1993). Using a KM approach, the tacit knowledge developed during the application of a software process is captured, stored, disseminated and reused, so that to achieve better quality and productivity. KM supports process management decisions, such as software process definition, human resource allocation and effort estimation of development activities as well as quality planning and control (Falbo et al., 2004). In a SPI project, the process manager should answer two main questions in order to create a knowledge base for the software process (Bibi et al., 2010): (i) which metrics can provide useful information for each particular process area? (ii) which projects will be considered to create a process area knowledge base?

The relevant literature points out numerous metrics to describe software processes (Kan, 2003). A well-known categorization of metrics involves project, process, product and personnel oriented metrics (Boehm, 1981). Regarding the projects that participate in the knowledge base, the manager should, for example, select the most relevant ones to the recent activity of the SME or the most recent ones. These project types are suggested since the process followed in these projects is likely to be repeated in the future. The manager should ensure that data of the selected projects are objectively and consistently recorded. It should be noted that the way to perform these types of activities (e.g. data collection) is not precisely specified by the SPRINT SMEs approach, since useful relevant guidelines are suggested by the generic SPI approach (e.g., ISO/IEC 12207) in the context of which SPRINT SMEs can be applied.

In the third step of the SPRINT SMEs approach we adopt an ontology-based paradigm (Katifori et al., 2007). Ontologies formally represent knowledge as sets of concepts within a domain by using a shared vocabulary to denote the types, properties and interrelationships of those concepts. Different complementary ontologies have to be developed to address knowledge in software process improvement projects (i.e., tacit and explicit knowledge, knowledge about projects, knowledge in projects and knowledge from projects). A generic structure of the software process ontology has been proposed by Brinquette et al. (2011) and it is depicted in Figure 1.

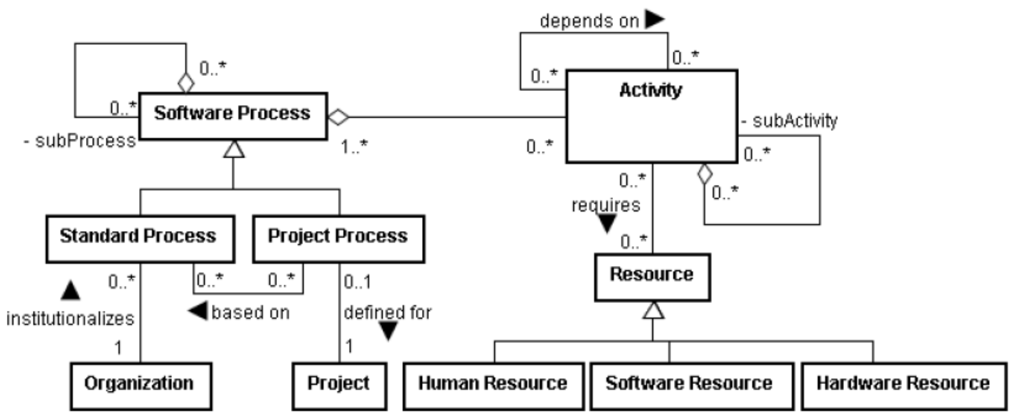

Figure 1: Software Process Ontology (Brinquette et al., 2011)

The SPRINT SMEs approach suggests three sub-ontologies to develop for covering three process improvement knowledge domains, respectively:

- Experience ontology: The experience ontology describes skills and qualifications required for performing specific improvement practices.
- Process ontology: The process ontology enables the definition of a hierarchical process structure and alternative process decompositions and dependencies.
- Project content ontology: The project content ontology supports the representation of information about the improvement of the project content which includes project artefacts (e.g. requirements artefacts, UML diagrams, source code components, etc.).

In the fourth step, the SPRINT SMEs approach utilises BNs to experiment with the ontologies defined in the previous step. A BN is a directed acyclic graph that represents a causal network consisting of a set of nodes and a set of directed links between them, in a way that they do not form a cycle (Jensen & Nielsen, 2007). Each node in a BN represents a random variable that can take discrete or continuous, mutually exclusive values according to a probability distribution, which can be different for each node. Each link in a BN represents a probabilistic cause-effect relation between the linked variables and it is depicted by an arc starting from the influencing variable (parent node) and terminating on the influenced variable (child node). The strength of the dependencies is measured by means of conditional probabilities depicted in the form of Node Probability Tables (NPTs).

BNs are helpful in software process evaluation and improvement since they offer (Bibi et al., 2010): i) a way to represent project/process attributes and identify their interrelationships, ii) capabilities for performing multiple attribute estimations, iii) results indicating confidence of the estimations, iv) solutions that can be easily interpreted and confirmed by intuition, and v) analytical methods that can be used alone or combined with expert judgment.

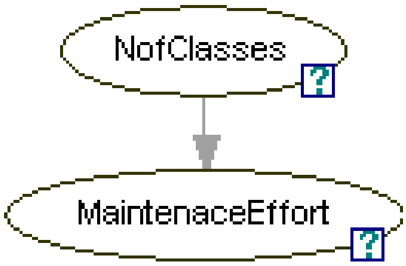

Figure 2: A BBN for software effort estimation

Table 1: The NPT of the node Maintenance Effort of Figure1

NofClasses		Low	High
Maintenance Effort	Low	0.7	0.45
	High	0.3	0.55

A simple BN example is presented in Figure 2. The model consists of two nodes. The first node (NofClasses) represents the number of classes in a software package and the second node (Maintenance Effort) represents the effort required for package maintenance. We consider that the values of these two nodes fall into two discrete categories (Low and High). For the node NofClasses, Low values range between 1 class and 10 classes, while High values represent packages with more than 10 classes (30 classes the most). For the node MaintenanceEffort, Low values range from 1 man month to 3 man months, while High values range from more than 3 man months up to 10 man months. A simple example to comprehend the NPT presented in Table 1 is the following: If the number of classes falls in the low category then there is 70% probability that the maintenance effort will also fall in the low category.

4. Validation of the Approach

4.1 Validation based on ISBSG data set

In this section we will present, as a proof of concept, a hypothetical example of applying the suggested approach. In the first step (identification of the area under improvement), we isolate project planning phase as the target of improvement attempts. During project planning, the project objectives are defined along with the project schedule and its activities. People to perform the project activities have to be allocated. Also project monitoring and control should be performed. This involves tracking the accomplishment of project activities and managing the necessary time to perform them. In particular, software project planning involves activities such as:

- Project process selection: This might involve the selection of a standard process such as RUP, SCRUM, ICONIX, XP or even hybrid methods that fit the particular needs of a specific company (Kruchten et al., 2003).
- Resource allocation: This task involves the selection of the development team, the allocation of people to tasks. Also in this task the selection of the necessary software tools and hardware equipments is performed.
- Project monitoring and controlling: They involve the necessary estimations relevant to the effort or the productivity required to complete a software project.

The next step is to define a knowledge base relevant to the process area under improvement. In order to create such a knowledge base, SMEs are advised to use their own empirical data coming from historical projects. If such data are not available, we can use publicly available data such as those coming from the ISBSG (International Software Benchmarking Standards Group) repository (www.isbsg.org), at least as a starting point until company-specific data are available. In the following, we will use metrics and data coming from ISBSG database. It is highly possible that a company that desires to estimate several aspects of software development will not possess a sufficient quantity of its own data. Therefore, using cross company data can be a starting point to manage and estimate a software development process.

The ontology of Figure 1 describes a general procedure to define a software process for a company’s project. The project manager should identify the activities that have to be performed to achieve the project goals. This is done by tailoring organizational standard processes, taking the project particularities and team features into account. The project process is the basis for the further project management activities. After defining the process, the project manager creates the network of project activities, define how long each activity will last, and allocate people to perform them. For a good understanding of these tasks, we need a shared conceptualization regarding software processes.

The generic ontology of Figure 1 is further extended to include process attributes and operations. Figure 3 depicts the class diagram of this extended ontology. In Figure 3, the class Software Process consists of certain attributes like Size, Effort, Complexity and Quality. The operations encapsulated in this class are Planning, Scoping, Assessing, Deciding, Measuring, Monitoring and Improving. The class Standard Process is associated with the metrics that show conformance to RUP, ICONIX or XP process models, while the class Project Process represents the use of a customized variation of these standard processes for a specific project. The class Organization is represented by metrics describing each individual SME. Such metrics may include the Size of the Organization, the Years of Experience and the Organization Type. The class Project defines project specific metrics, such as Development Type and Business Area Type. The Activity class represents standard activities performed in software development like Planning, Specification, Design, Build, Implementation and Testing. Depending on what area of project planning has to be improved, the Activity class may represent the relevant quality metrics for each activity or effort metrics (Deliverables, Milestones, etc.) for each activity. The class Human Resource is associated with metrics, such as Personnel skills and Roles for the Project Staff subclass or Expertise for the Manager subclass, while the class Software Resource is associated with metrics such as Use of Case Tools, Programming Language and Data Base. Finally, the class Hardware is associated with metrics, such as the Development Platform and the Architecture type.

Based on the instantiation of the ontology concepts presented in Figure 2, we apply BNs to experiment with the ontology data and find relationships among them. The aim of this step is to gain insights about how the project planning process can be improved. For this reason, we replace each class defined in the ontology class diagram of Figure 3 by relevant metrics derived from the ISBSG data set or we can use complementary metrics, if needed. Figure 4 shows the resulted BN model. BN tools (e.g., webdocs.cs.ualberta.ca/~jcheng/bnsoft.htm) can be helpful to redefine and analyse the structure of the BN model based on data derived from real projects. Data analysis results in probability tables that show how each node affects the neighbour ones. Certain inferences can show how changes in the values of a metric affect values of another metric and, finally, reach conclusions regarding good and bad practices in software project planning.

Figure 5 shows an instance of the BN that is instantiated with data derived from the ISBSG data base. This network was trained using actual data from 124 projects for which the activity phase effort data were recorded. Each node is accompanied by a Node Probability Table (NPT) that estimates its values according to the values of the parent node. The total effort value for all activities is dependent on the build and test effort. The implementation effort is mainly affected by the design effort.

Let assume, for example, that the manager's aim is to test situations under which the development effort has low values. In the BN of Figure 5 the evidence of low effort values can be inserted in the node SummaryWork Effort. The value of work effort will be minimized and, therefore, the values of the rest of the nodes will be altered to suit that inference. For example, low total effort values require relatively average effort values during planning and specification phases and low values during building and implementation phases. In this way, the manager is able to analyse the effect of such a conclusion. He/she can test in future projects whether high effort values in planning and specification phases can reduce the effort required during building and implementation phases.

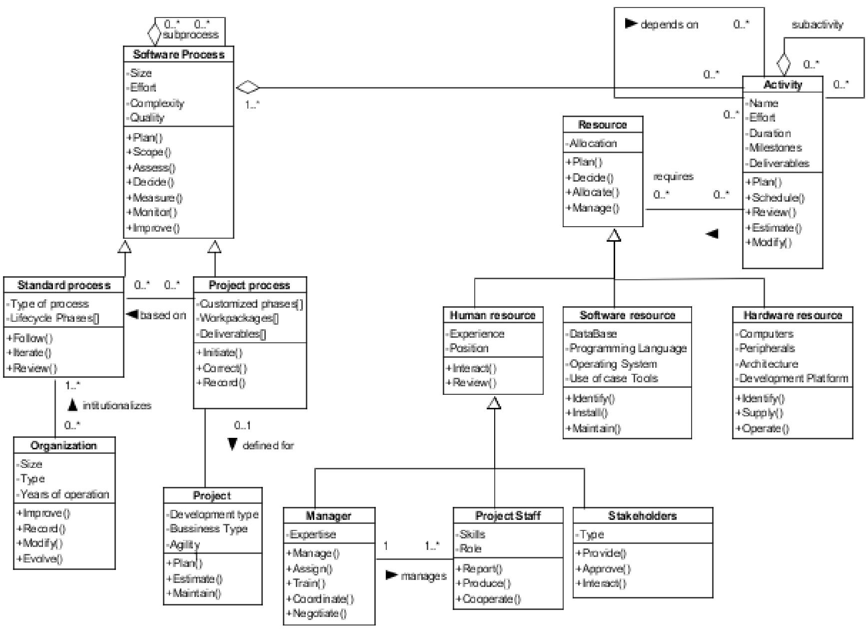

Figure 3: The extended software process ontology

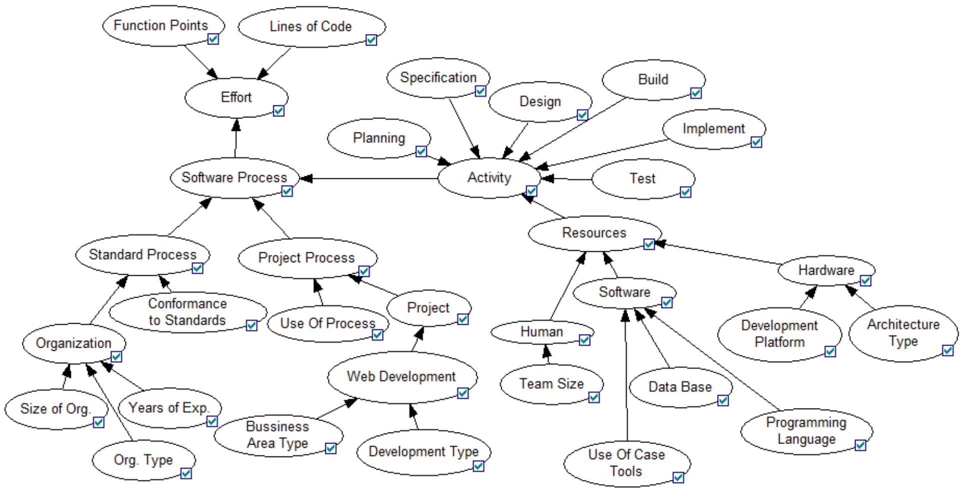

Figure 4: A Bayesian Network for the software process ontology presented in Figure 3

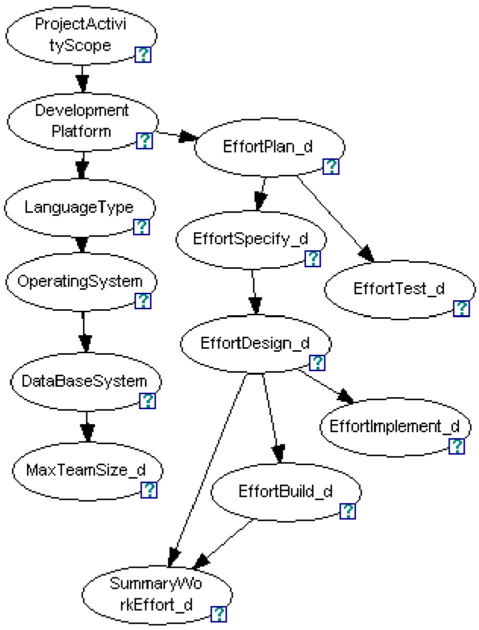

Figure 5: A Bayesian Network for ISBSG data set

4.2 Validation based on company specific data

In the following, we present an example of applying the SPRINT SMEs approach in a case study that took place in a Greek SME running projects in software telecommunications field. The study lasted one week. The company occupies almost 35 employees mainly scientific, technical and management personnel. In the case study we have followed the SPRINT SMEs approach to evaluate the company's project management and process improvement decisions. The first step was to identify the process areas that needed further support. For this reason, we interviewed three company's employees (project managers) with at least 5 years experience covering all aspects of company's activities. The employees pointed two areas of interest, namely effort/duration estimation and software reuse.

The second step was to develop a knowledge base that included all relevant information regarding the aforementioned process areas of interest. After the interviews, we selected to record metrics that are company specific and relevant to the telecommunication software that the company develops and also more general metrics, such as effort and size metrics. Then, we selected the historical projects that would participate in the analysis to define the required process models. We selected five recent projects that the managers considered more indicative of the current activity of the company. These projects offered information that could be retrieved even if we had to perform post-mortem analysis. The data that were collected involved software process, product and implementation metrics and they are presented in Table 2.

Table 2: Metrics of the knowledge base for the telecommunications company with low and high metrics' ranges.

Variable	Min	Categories
LOC	Lines of Code	L(≤12105),H(>12105)
Duration	# of months	L(≤9.5), H(>9.5)
Effort	# of months	L(≤5.50), H(>5.5)
P1Duration	Analysis & design phase, man months	L(≤4.5),H(>4.5)
P1Effort	man months	L(≤5),H(>5)
P2Duration	Coding & testing phase, man months	L(≤5),H(>5)
P2Effort	man months	L(≤3.5),H(>3.5)
TeamSize	# of people in the project	L(≤2),H(>2)
Reuse	% of reusage of previous project products	L(≤25%), H(>25%)

Reusability	% of the project products reused	L(≤35%), H(>35%)
TN_B	# of Blocks	L(≤3),H(>3)
TN_P	# of Processes	L(≤14),H(>14)
TN_ST	# of States	L(≤54),H(>54)
TN_PT	# of Process Types	L(≤1), H(>1)
TN_SYS (	# of Systems	L(≤0), H(>1)
TN_TMR	# of Timers	L(≤15), H(>15)
TN_BT	# of Block Types	L(≤0),H(>0)
TN_T	# of Data Types	L(≤0),H(>0)
TN_G	# of Gates	L(≤23),H(>23)
TN_CH	# of Channels	L(≤0),H(>0)
TN_BIP	# of Built in Procedures	L(≤8), H(>8)
TN_Ent_VS	# SDL Entities with Valid Suffix	L(≤49), H(>49)
TN_Ent_IS	# SDL Entities with Invalid Suffix	L(≤38), H(>38)

The third step resulted in a process ontology that represented the targeted improvement areas (effort/duration estimation and software reuse). To implement this step we have used parts of the ontology described in Figure 3. In general, for the ontology creation there can be several alternative solutions for each specific company. Therefore, we have used the generic ontology presented in Figure 3, as it is difficult for an SME to create its own process ontology from scratch. This generic ontology can be modified according to the needs of a specific company.

The fourth step was to design appropriate BNs based on the ontological representation of the knowledge base. To ensure better readability and clarity of the results, two BN models were created, one involving the effort estimation process and another one involving the software reuse process. The first BN is presented in Figure 6.

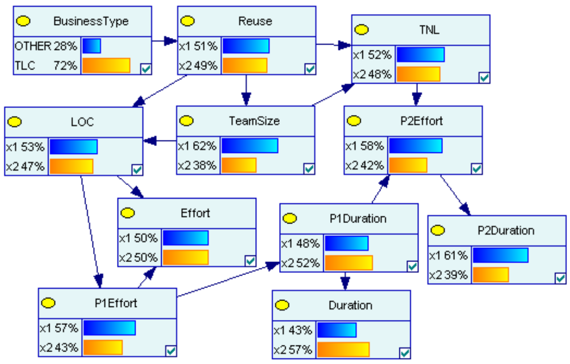

Figure 6: Software Process BN for effort estimation

In the BN of Figure 6 network nodes are shown as bar charts providing additional information for the data allocation at each node. This BN model demonstrated the following assertions: The total effort value mainly depends on the effort of the first development phase of a process that is often followed in the company's projects (P1Effort) and on the Lines of Code (LOC) written, apart from code written in Specification and Description Language (SDL). The company develops software using a mix of (i) graphical development with the use of SDL telecommunication modelling language and tools that execute directly the SDL models and (ii) programming in C language. The Lines of Code are affected by the percentage of reuse from previous projects which affects intuitively also the size of the development team. Larger teams produce more Lines of Code. A large percentage of reuse can reduce the actual number of new lines of code and the total effort value. The effort of the second development phase (P2Effort) that is followed in the company's projects mainly depends on TNL (Total Number of Lines) that correspond to lines written in SDL. The value of TNL is also affected by the percentage of reuse.

The NPT (Node Probability Table) of the node effort in the BN of Figure 6 is presented in Table 3. This table can be used for the estimation of the total effort required for the completion of a new project in the company. The total development effort of a new project is estimated to be high (second category) with probability 64% when the effort required for the first development phase is high and the number of Lines of Code is also high.

Table 3: NPT for effort estimation

P1Effort	X1		X2	
LOC	X1	X2	X1	X2
X1	0,75	0,42	0,36	0,31
X2	0,25	0,58	0,64	0,69

A second BN model (Figure 7) was developed during the case study to analyse the company’s software reuse process. A more conventional format is selected in Figure 7 to show this BN (nodes are depicted with icons). This model indicated that the variable TN_PT (Total Number of process types) actually affects the values of other code structure variables, such as the number of block types and the number of gates (these are all SDL specific metrics). According to the BN of Figure 7, the percentage of code from a particular project that can be reused is affected by the number of entities with invalid suffix, i.e., inappropriate naming choices (TN_Ent_IS). This result indicated that reuse heavily depends on the formality that the programmers adapt when naming the entities on the code. This intuitively affects the understandability of the code that enables further reuse.

Post-mortem analysis was applied on the BN model of Figure 7 and resulted in the following useful insights: The lower the number of code structure variables the greater the reuse. It seems that smaller parts of code can be more easily reused. According to the company's management, future projects are possible to breakdown to smaller autonomous packages that could perform different aspects of functionality. This decomposition would enable greater percentage of reuse. The company's management so far preferred the use of smaller teams, while there is also the possibility of using larger ones. The idea was that small teams can be more flexible, communicate better and produce more quickly results. It seems though from the analysis results that larger teams can produce results in shorter time and they are able to reuse larger percentage of code from previous projects. The management re-considered the initial opinion on utilisation of smaller teams and currently is validating the experimental results on larger teams.

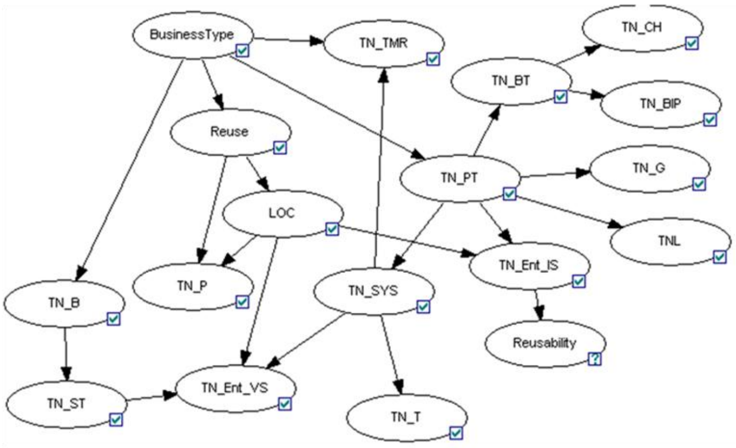

Figure 7: Software Process BN for reusability

5. Conclusions

This deliverable presented an approach to support software process improvement activities for software development SMEs. The 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 approach utilizes a generic ontology that is tailored to the needs of an SME and applies Bayesian network analysis to make measurable each concept that is represented in the process ontology. As a proof of concept, we have presented the approach application in a hypothetical project planning process by using publicly available project data derived from the ISBSG repository. The deliverable also presented the approach validation in a case study aimed to improve software effort estimation and reuse in a company that delivers hardware/software solutions in the telecommunications area. As future work the proposed approach will be further validated at a multiple case study involving Greek software SMEs, which show interest in improving their development practices and changing their role from bespoke to market-driven software product developers.

6. References–Bibliography

Abouelela, M. and Benedicenti, L., 2010. Bayesian Network based XP Process Modelling, *IJSEA Journal*, 1(3), 1-15.

Barcellos, M. P., Falbo, R. A., 2009. Using a Foundational Ontology for Reengineering a Software Enterprise Ontology. In *Advances in Conceptual Modeling - Challenging Perspectives*, *Lecture Notes in Computer Science* 5833, 179-188.

Bibi, S. Stamelos, I., Gerolimos, G., Kollias, V., 2010. BBN based Approach for Improving the Software Development Process of an SME - a Case Study, *Journal of Software Maintenance*, 22(2).

Bibi, S., Stamelos, I., 2004. Software Process Modeling with Bayesian Belief Networks. In *Proceedings of 10th International Software Metrics Symposium* (*Metrics 2004*), Chicago.

Boehm, B., 1981. *Software Engineering Economics*, Englewood Cliffs, Prentice-Hall.

Bringuente, A., Falbo, A., Guizzardi, G., 2011. Using a Foundational Ontology for Reengineering a Software Process Ontology, *Journal of Information and Data Management*, 2(3), 511-526.

Falbo, R., Bertollo, G., 2009. A software process ontology as a common vocabulary about software processes, *International Journal of Business Process Integration and Management* (*IJBPIM*), 4(4), 239-250.

Falbo, R., Borges, L. S. M., Valente, F. F. R., 2004. Using Knowledge Management to Improve Software Process Performance in a CMM Level 3 Organization. In *Proceedings of the International Conference on Quality Software* (*QSIC 2004*), 162-169.

Fenton, N., Krause, P., Neil, M., 2002, Probability modeling for software quality control, *Journal of Applied Non-Classical Logics*, 12(2), 173-188.

Fenton , N., Neil, M., Marsh, W. , Hearty,P., Marquez,D., Krause,P., Mishra, R., 2007. Predicting Software Defects in varying Development Lifecycles using Bayesian Nets, *Information & Software Technology* 49(1), 32-43.

Guizzardi, G., Falbo, R. A., Guizzardi, R. S. S., 2008. Grounding Software Domain Ontologies in the Unified Foundational Ontology (UFO): the Case of the ODE Software Process Ontology. In *Proceedings of the XI Iberoamerican Workshop on Requirements Engineering and Software Environments*, 244-251.

Henderson-Sellers,B., Gonzalez-Perez, C., Mc Bride,T. Low, G., 2014. An ontology for ISO software engineering standards: 1) Creating the infrastructure, *Computer Standards & Interfaces*, 36(3), 563-576.

International Organization for Standardization / International Electrotechnical Commission, 2013. *ISO/IEC FDIS 26550: Software and Systems Engineering - Reference Model for Product Line Engineering and Management*.

Jensen F., Nielsen, T., 2007. *Bayesian Networks and Decision Graphs*, Springer Verlag.

Kan, S., 2003. *Metrics and Models in Software Quality Engineering*, Pearson Education Limited.

Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., Giannopoulou, E., 2007. Ontology Visualization Methods - a Survey, *ACM Computing Surveys*, 39(4).
Kruchten, P., 2003. *The Rational Unified Process: an Introduction*, Addison-Wesley.
Liao, L., Qu, Y., Leung, H. K. N., 2005. A Software Process Ontology and its Application. In *Proceedings of the First International Workshop on Semantic Web Enabled Software Engineering*.
Mendes, E., 2007. The Use of a Bayesian Network for Web Effort Estimation. In *Proceedings of the International Conference on Web Engineering* (*ICWE 2007*), 90-104.
Mishra, D., Mishra, A., 2009. Software Process Improvement in SMEs: a Comparative View, *Computer Science and Information Systems*, 6(1), 111-140.
Nonaka, I., Krogh, G., 2009. Tacit Knowledge and Knowledge Conversion: Controversy and Advancement in Organizational Knowledge Creation Theory, *Organization Science*, 20 (3), 635–652.
Okutan, A., Yildiz, O., 2014. Software Defect Prediction using Bayesian networks, *Empirical Software Engineering*, 19(1), 154-181.
Paulk, M., Curtis, B., Chrissis, B., Weber,M., 1994. *Capability Maturity Model for Software: Guidelines for Improving the Software Process*, Addison-Wesley.
Pettersson, F., Ivarsson, M., Gorsheck, T., Ohman, P., 2008. A Practitioner's Guide to Lightweight Software Process Assessment and Improvement Planning, *Journal of Systems and Software*, 21(6), 972-995.
Radlinski, L., 2010. A Survey of Bayesian Net Models for Software Development Effort Prediction, *International Journal of Software Engineering and Computing*, 2(2), 95-109.
Settas, D., Bibi, S., Sfetsos, P., Stamelos, I., Gerogiannis, V. C., 2006. Using Bayesian Belief Networks to Model Software Project Management Antipatterns. In *Proceedings of the Fourth International Conference on Software Engineering, Research, Management and Applications* (*SERA 2006*), Seattle, 117-124.
Simari, G., Rahwan, I., 2009. *Argumentation in Artificial Intelligence*, Springer.
Stamelos, I., Angelis, L., Dimou, P., Sakellaris,E., 2003. On the Use of Bayesian Belief Networks for the Prediction of Software Productivity, *Information and Software Technology*, 45(1), 51-60.
Thomas R., 1993. A Translation Approach to Portable Ontology Specifications, *Knowledge Acquisition*, 5(2), 199–220.
Zahran, S., 1998. *Software Process Improvement: Practical Guidelines for Business Success*. Addison-Wesley.