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 PRocess ImprovemeNT Methodologies for Greek Small & Medium sized Software Development EnterpriseS

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

Deliverable 3.1 (D3.1): Specification 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

January, 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

1. Introduction

The objective of this deliverable is to specify and design a Knowldge Base to support the Software Process Improvement of small-medium sized software development enterprises (SMEs). This Knowldge Base will be used to form an ontology that best describes the software process. The proposed ontology will be evaluated on Open Source projects. This deliverable presents the specification of the Knowledge Base that consists of information relevant to the knowledge required for Software Process Improvement procedures.

The objective of SRPINT-SMEs approach is to develop a practical framework for the improvement of software processes which take place in Small & Medium sized Software Development Enterprises (SW SMEs) in order to increase their competitiveness. The framework concentrates on improving selected process domains such as the Requirements Engineering process (RE) and Project Planning. The following stages of a SPRINT-SME approach are supported by the framework's analysis methods:

(i) Assessment of software process domains and selection of defective ones for further analysis and improvement
(ii) Definition of a knowledge base that better describes the software domain under improvement
(iii) Creation and further analysis of the ontology that represents the domain
(iv) Experimentation and suggestions for improvement

The framework will be particularly useful for software SMEs interested in implementing lightweight and flexible software process improvement (SPI) projects based on critical improvement issues, tailored in cost/time/resource constraints and consistent with their individual needs.

In Section 2 of this deliverable, we describe the four-steps SPRINT- SME approach for software process improvement. In section 3, we present the SPI Knowledge base. In section 4, we define metrics that can be used in the SPI Knowledge Base. In Section 5, we present a simple example of the knowledge base application in improving the domain of Project Planning.

2. The SPRINT- SMEs Approach

In this section, we will introduce the SPRINT- SMEs approach for the improvement of the software process of a small medium sized software company. The SPRINT-SMEs approach is a lightweight methodology for efficiently improving certain process domains of SMEs. Our approach is tailored to the needs of SMEs as it is efficient, easily adoptable, non bureaucratic and independent of company specific assets.

We suggest a four step process improvement model that consists of the following steps:

1. Select domain for SPI. Definition of the particular software development domain that needs improvement.

2. Define a SPI knowledge base. Selection of attributes that describe the particular domain and definition of metrics that better represent these attributes.

3. Ontology analysis. Form an ontology that describes better the relationships among the attributes of the SPI knowledge base. Application of tools and methods to better analyze an ontology that is feeded with data coming from the SME.

4. Experimentation and improvements. Exploit the results of the previous step to find problematic and defective domain areas. Make changes, experiment and suggest improvements.

A. Define the Domain of Improvement

The first step of the SPRINT-SMEs approach is to define the defective domain that will be investigated. The domain identified will then be set as the target of the improvement efforts. The starting point of popular SPI models is the specification and improvement of the quality of the total development process.

An objectively measurable specification of software process quality is a prerequisite for such types of models. However, the definition of software process quality is not always the same and does not apply to all types of companies. Additionally, the effort required to improve all aspects of software process is often prohibitive in terms of time and cost for most SMEs since they do not possess neither the know-how nor the resources to achieve such improvement goals.

Defining the software process domain that will be set under observation is a managerial decision and depends on the needs of the SME and the type of projects that it handles. For example, the domain under improvement can be decided from the traditional software lifecycle models: requirements engineering, design specification, programming and development, software testing, software development management etc.

The selection of one or more of these domains for process improvement will then define the company specific aspects that need to be specified to continue with the SPINT-SMEs approach. Some of the domains of improvement that the SPINT-SMEs approach focuses are:

- Requirements Engineering
- Software Process Estimation
- Software Project Quality
- Personnel Management

B. Create a SPI Knowledge Base

The objective of this step is to specify and design a Knowledge Base that consists of information relevant to the knowledge required for Software Process Improvement procedures of the domain (s) pointed by the previous step. A knowledge base [17] is a database that stores data for knowledge management. Knowledge management (KM) [18, 19] comprises a range of strategies and practices used in an organisation to identify, create, represent, distribute, and enable adoption of insights and experiences. Such insights and experiences comprise knowledge, either embodied in individuals or embedded in organisations, such as processes or practices [20].
Knowledge bases are commonly used to complement a procedure for sharing information among members of a community. They might store critical enterprise data, personnel information, process metadata, troubleshooting information,

knowledge tags, or answers to frequently asked questions. Typically, a search engine is used to locate information in the system, or users may browse through a classification scheme.

Using a KM approach, knowledge created during software process can be captured, stored, disseminated and reused, so that better quality and productivity can be achieved. KM can be used to better support management activities, such as software process definition, people allocation and estimation, software development activities, such as requirement analysis and test case design, and quality assurance activities, such quality planning and control.

C. Ontology Analysis

In order to design the structure of the SPRINT-SMEs Knowledge Base we follow an ontology-based approach. In computer and information science, an ontology formally represents knowledge as a set of concepts within a domain, using a shared vocabulary to denote the types, properties and interrelationships of those concepts [21], [22].

Ontologies are structural frameworks for organizing information and they are used in artificial intelligence, Semantic Web, systems engineering, software engineering, biomedical informatics, library science, enterprise bookmarking, and information architecture as a form of knowledge representation about the “world” of interest or some part of it. The creation of domain ontologies is also fundamental to the definition and use of an enterprise architecture framework [22].

Different complementary ontologies have to be developed to address the full spectrum of knowledge in SW process improvement projects (i.e., tacit and explicit knowledge, knowledge about projects, knowledge in projects and knowledge from projects). The SPRINT-SMEs approach suggests three sub-ontologies covering three distinct process improvement knowledge domains:

- **Experience ontology:** The experience ontology describes skills and qualifications required for performing specific improvement practices.

- **Process ontology:** The process ontology enables defining a hierarchical process type structure and alternative process decompositions and dependencies (for example, it is possible to state that “Requirements Traceability” is dependent on “Requirements Specification”).

- **Project content ontology:** The project ontology supports the representation of information about the improvement of the project content which includes project artifacts (e.g. source code, UML diagrams etc.) and the project content in general. Examples are: “no of class diagrams = 40”, “number of use cases in the use case model = 30”, “persistence framework = Hibernate” etc.

D. Experiment and Improve.

In this step, we suggest the use of formal notations and tools to represent and experiment with the ontologies defined in the previous step of the approach. Some of the methods that can be used are: (i) UML for ontology representation which is a well-known standard in the software development community, (ii) clear and rigorous semantics provided by the definition of a metamodel itself, and (ii) use of tools for the generation and verification of the SPRINT-SMEs models. Especially the use of tools alleviates the difficulty of process description verbosity. For example, a human role or a phase in a process is described once and the same definition is reused, whenever the description of the same role/phase is required.

Tools and methods that can be used to further analyse and experiment with the ontology can come from the traditional group of CASE tools (such as process framework tools, simulation tools, etc.) or can be tools that provide an estimation and inference mechanism (such as probabilistic techniques, fuzzy logic based techniques, Social Network analysis tools, etc.). The SPRINT- SMEs approach, for example, focuses, among others, on two representation tools a) the open source Eclipse Process Framework (EPF) Composer and b) the Bayesian Networks Analysis tools.

In particular, a tool that provides the above mentioned capabilities is the open source Eclipse Process Framework (EPF) Composer. EPF Composer includes the following advantageous features to be exploited by the SPRINT-SMEs approach: (i) A Method Content Authoring Feature, to define process roles, tasks, work products and guidances, independently from the process definition, a feature which makes these method ingredients reusable throughout the process description. (ii) A Process Authoring feature, to describe processes as sequences of tasks (performed by roles) which produce work products. (iii) A Process Configuration Feature, for incorporating the process descriptions into packages (plugins) to be re-utilized in the definition of other processes. This functionality allows the customisation of processes to particular contexts and it very important since a SW development company may follow similar but slightly different methods and processes.

On the other hand, SPRINT-SMes exploits the advantages from powerful estimation techniques such as the Bayesian Belief Networks (BBNs) [15]. BBNs can be helpful since they can provide: i) a way to represent project/process attributes and identify their interrelationships, ii) capabilities for multiple attribute estimations, iii) results indicating confidence of the estimation, iv) solutions that can be easily interpreted and confirmed by intuition, and v) a formal method that can be used alone or combined with expert judgment.

In the following sections we will present analytically the metrics that can be collected in order to create the SPI Knowledge base.

3. Experience knowledge base

The experience knowledge base consists of data that refer to the personnels general experience, the personnels experience with process models and the personnels experience with product development.

3.1 Personnel experience metrics

– Analysts capabilities

This metric refers to the analyst capabilities and experience measured either in years of experience or in number of projects that the analyst participated.

– Programmers capabilities

This metric refers to the programmers capabilities and experience measured either in years of experience or in number of projects that the programmer participated.

– Management experience

This metric refers to the managers capabilities and experience measured in years of experience in managing software projects.

3.2 Process experience metrics

– Experience with RUP. Rational Unified Process (RUP) is an iterative software development process framework that is not a single concrete prescriptive process. RUP is an adaptable process framework, intended to be tailored by the development organizations and software project teams that will select the elements of the process that are appropriate for their needs. Therefore the personnel and manager's experience in implementing the RUP process is important and can affect the success of a software process.

- Experience with Object Oriented Analysis, Design and Programming. The application of Object Oriented Design principles requires certain experience and skills. This experience can be measured in the number of projects that the development team has completed using OOD or the years of experience.

3.3 Product experience metrics

- Experience with the type of application. This metric defines whether the team has experience with certain types of applications. This can be measured either by a Boolean variable (yes, no) or in number of relevant projects completed or in years of experience.

- Experience with programming language. This metric defines whether the team has experience with certain programming languages. This can be measured either by a Boolean variable (yes, no) or in number of projects that have been implemented with the particular language or in years of experience.

4. Process knowledge base

The process knowledge base consists of data that describe the various artifacts produced in each development phase. We will present metrics describing the requirements and analysis phases, the design and implementation phase and the testing phase. Also metrics deriving from the planning phase and general progress metrics will be presented.

4.1 Planning attributes

– Effort

Provides the total effort in hours recorded against the project.

– Productivity delivery rate

Project productivity delivery rate in hours per work product for the development team.

– Team Size

The average number of people that worked on the project, (calculated where available from the team sizes per phase).

– Noof Use Cases

The number of use cases that were recorded during analysis and finally implemented during development.

– No of function points

The number of function points recorded for the project during the planning phase and the total number of function points calculated after implementation.

– No of internal meetings

Frequency of internal meetings for progress discussion and project control.

– Use of Case tools

The number of Case tools used and the percentage of work done with the help of Case tools.

– Lines of Code produced

The number of the source lines of code (SLOC) produced by the project.

– Planning Documents

The documents or other work products produced during the planning activity.

– Conformance to software standards models

Software standards typically define a series of actions and documentation structures and contents required to deliver quality software and software processes. Software standards are maintained by international organisations. Software developers are typically required to be formally and externally assessed in order to achieve certification to these standards. This attribute indicates if the project was performed under CMM or ISO or SPICE processes. Where available, details such as the level or version, and year of certification can be included.

4.2 Analysis & Requirements

– Number of requirements.

Can be measured as the number of use cases defined or us the number of functions recorded.

– Non- functional requirements.

The number of non functional requirements.

– Number of requirements analysts and reviewers.

The number of analysts that participate in the requirements elicitation is important as based on this number certain metrics can be defined regarding the requirements understandability. For example requirements understandability can be defined as the number of the requirements understood by all reviewers to the total number of requirements.

4.3 Design

- Use of design patterns. Design patterns can speed up the development process by providing tested, proven development paradigms. The use of them cann be measured in the number of design patterns implemented for a certain project
- Type of documentation produced. In this phase we can record the types of documents or diagrams produced. For example we can record whether class, sequence or data flow diagrams have been produced.

4.4 Development

- Development Techniques

Techniques used during development. (e.g.: JAD, Data Modelling, OO Analysis etc.). These techniques have not been recorded as being phase specific and therefore may apply to any part of the development lifecycle.

4.5 Testing

- Type of testing

The type of testing refers to the testing techniques applied like black box testing, white box testing, unit testing, integration testing, system testing, load testing, stress testing

- No of Test Cases

The number of test cases applied.

- Effort for Testing

The effort required for completing the testing phase.

- Correction effort

The effort required for correcting the problems identified.

5. Project knowledge base

The project knowledge base consists of data that are project specific. Such data refer to metrics and data relevant to the projects environment, type, architecture, users and quality.

5.1 Project Environment Attributes

– Development Platform

Defines the primary development platform, (as determined by the operating system used). Each project can be classified as: PC, Mid Range, Main Frame or Multi platform.

– Language Type

Defines the language type used for the project: e.g. 3GL, 4GL, Application Generator etc.

– Primary Programming Language

The primary language used for the development: JAVA, C++, PL/1, Natural, Cobol etc.

– Hardware

This is the primary technology hardware platform used to build or enhance the software (i.e. that used for most of the build effort).

– Operating System

This is the primary technology operating system used to build or enhance the software (i.e. that used for most of the build effort).

– Integrated Development Environment

This is the primary Integrated Development Environment, a development environment integrating a range of tools to aid the processes of designing, constructing and testing the software; typically, incorporating graphical and component based development techniques.

- Debugging Tool

This is the primary technology debugging tool used to build or enhance the software (i.e. that used for most of the build effort), otherwise (if known) it is whether the project used a debugging tool.

- Data Base System

This is the primary technology database used to build or enhance the software (i.e. that used for most of the build effort), otherwise (if known) it is whether the project used a DBMS.

- Component Server

This is the primary technology object/component server used to build or enhance the software (i.e. that used for most of the build effort); otherwise (if known) it is whether the project used an object/component server.

- Web Server

This is the primary technology HTML/Web server used to build or enhance the software (i.e. that used for most of the build effort); otherwise (if known) it is whether the project used an HTML/Web server.

- Message Server

This is the primary technology E-Mail or message server used to build or enhance the software (i.e. that used for most of the build effort), otherwise (if known) it is whether the project used an E-Mail or message server.

5.2 Project Type attributes

- Development Type

This field describes whether the development was a new development, enhancement or redevelopment.

- Organisation Type

This identifies the type of organisation that submitted the project. (e.g.: Banking, Manufacturing, Retail).

- Business Area Type

This identifies the subset within the organisation being addressed by the project. It may be different to the organisation type or the same. (e.g.: Manufacturing, Personnel, Finance).

- Application Type

This identifies the type of application being addressed by the project. (e.g.: information system, transaction/production system, process control.)

- Package Customisation

This indicates whether the project was a package customisation. (Yes, No or Don't Know).

- Degree of Customisation

If the project was based on an existing package, this field provides comments on how much customisation was involved.

5.3 Project Architecture

- Architecture

A derived attribute for the project to indicate if the application is Stand alone, Multi-tier, Client server, or server, or Multi-tier with web public interface.

- Client Server?

Indicator of whether the application or product requires more than one computer to operate different components or parts of it. (Yes, No or Don’t Know).

– Client Roles

The roles performed by the computers that provide interface to the software’s external users.

– Server Roles

The services provided by the host/server computer(s) to the software application or product.

– Type of Server

A description of the server to the software application or product.

– Client/Server description

A description of the architecture of the client/server software application or product.

– Web Development

A derived indicator of whether the project data includes any comment that it is a web-development.

5.4 Project Quality

– Defects Delivered

Defects reported in the first month of use of the software. This metric splits into the number of Minor, Major and Extreme defects reported.

– Total Defects Delivered

Total number of defects reported in the first month of use of the software. This column shows the total number of defects reported (Minor, Major and Extreme).

– Defect Density

Defects per 1000 FP calculated as Total Defects Delivered * 1000 / Functional Size

Measures the quality of software in terms of defects delivered in unit size of software. It is defined as the number of Defects per 1000 Functional Size Units of delivered software, in the first month of use of the software. It is expressed as Defects per 1000 Functional Points.

5.5 Project User attributes

– User Base - Business Units

Number of business units (or project business stakeholders) serviced by the software application. Where the application covers multiple sets of users, a Business Unit is where a distinct set of business rules applies to a distinct set of application users.

– User Base - Locations

The number of physical locations being serviced/supported by the installed software application. A ‘distinct installation’ is an individual installation of the complete software system.

– User Base - Distinct Users

The total number of end users that can invoke the application.

– User Base - Concurrent Users

The total number of end users using the system concurrently. The term concurrent end users applies to a single distinct installation.

– Intended Market

This field describes the relationship between the project’s customer, end users and development team.

6. Example

In this section we will present a simple example of applying the SPRINT- SMEs approach. We follow four steps in order to validate and demonstrate the applicability of the proposed approach.

The first step of the approach involves the identification of the domain under improvement. We isolated project planning phase as the target of the improvement attempts. We selected to use this domain in our example as it is pointed out to be a very defective task that cannot be easily performed in SMEs. During project planning the project process need to be defined along with the 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 the activities and managing the necessary time to perform them. Software project planning involves activities such as:

- Project process selection: This might involve the selection of standard processes such as RUP, SCRUM, ICONIX or even hybrid methods that fit the particular needs of the company.
- 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: Usually involves the necessary estimations relevant to the effort or productivity required to complete the project.

The next step in the SPRINT-SMEs approach is to define a knowledge base relevant to the domain under improvement. In order to create such a knowledge base, an SME is advised to use each own empirical data coming from historical projects. In case such data are not available we suggest the use of publicly available data such as those coming from the ISBSG database [https://www.isbsg.org/] .

In this generic example we use software project metrics and data derived from ISBSG. It is highly possible that a company that wants to estimate several aspects of software development will not possess a sufficient quantity of its own data. Therefore, using cross company data is a starting point in order to manage and estimate a software development process. Additionally, cross company data can be useful when a company is adopting a new technology and lacks experience. Cross company data may contain projects utilizing the particular technology and can provide support on estimation and implementation issues. Table 1 summarizes some of the metrics that can be used to form the project planning knowledge base.

The third step of the SPRINT- SMEs approach involves the identification of an ontology relevant to the project planning phase. The Software Process Ontology (SPO), originally developed in [23], was built aiming at establishing a common conceptualization for software organizations to “talk” about software processes. It was divided into four sub-ontologies, namely: activity, resource, procedure and software process ontologies. Figure 1 shows a segment of the first version of this ontology that includes concepts from the activity, resource, and software process sub-ontologies. We selected this segment, because we are interested in the ontology’s concepts that are more relevant for project planning.

TABLE I. METRICS IN THE PROJECT PLANNING KNOWLEDGE BASE

Metric	**Possible Values**
Function Points	Continuous values
Max team size	Continuous values
Development Type	NewDevelopment ,Re-development, Enhancement
Language Type	3GL, 4GL, ApG
Primary Programming Language	ApG, 4GL, ACCESS, C, C++, CLIPPER, COBOL, CSP, EASYTRIEVE, JAVA, NATURAL, ORACLE OTHER, PERIPHONICS, PL/I, POWERBUILDER, SQL, TELON, VISUAL BASIC
Organisation Type	Aerospace/Automotive, Banking, Communication, Community.Services, Computers, Electricity,Gas,Water, Financial.Business, Government, Insurance, Manufacturing, OTHER, Professional Services, PublicAdministration, Transport&Storage, Wholesale&Retail Trade, Defence, Electronics.
Database Management System	ACCESS, ADABAS, DB2 ,IMS, OBJECTSTOR, ORACLE, OTHER
Development Platform	MF, MR, PC
How Methodolgy Acquired	Developed/purchased, Developed Inhouse, Purchased
Application Type	DSS, Elect.Data.Interch., Executive.I.S, MIS, Network.M, Office.I.S, OTHER, Process.Control, Real-time, Transaction/Production
Business Area Type	Accounting, Banking, Engineering, Financial, FineEnforcement, Insurance, Inventory, Legal, Logistics, Manufacturing, OTHER, Personnel, Research&Development, Sales&Marketing, Telecommunications
Implementation Date	1989- 2001

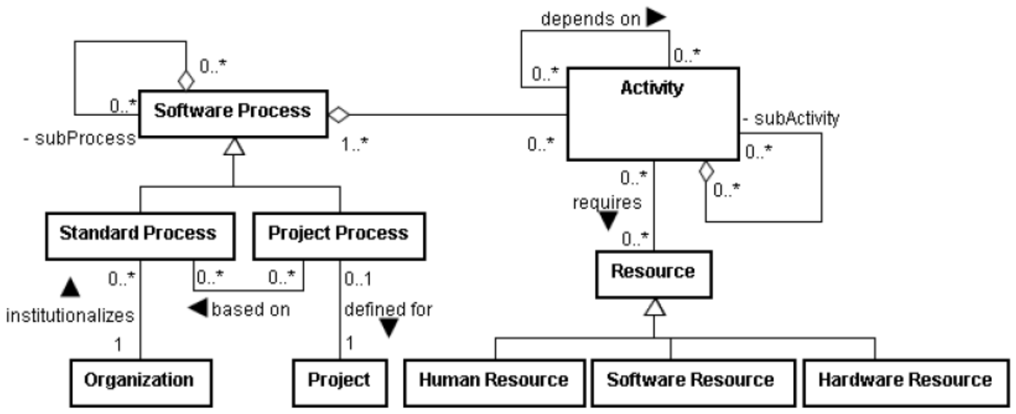

Figure 1: Software process ontology [23]

The ontology in Figure 1 describes a procedure to define a process for a software project. The project manager should identify the activities that have to be performed in order 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 has to create a network of activities, define how long each activity will last, and allocate people to perform it. For a good understanding of these tasks, we need a shared conceptualization regarding software processes.

Based on the ontology of Figure 1 we use Bayesian Networks in order to experiment with the data represented in the ontology and find useful relationships among them to gain insights about how the project planning process can be improved. For this reason, we replace each class defined in the UML diagram of Figure 1 by appropriate metrics from Table 1 or use complementary metrics if needed.

The node software process can be accumulated by software metrics relevant to the effort required to complete a software project. These can be the effort, the software size, the lines of code. The node Standard Process can be represented with the metrics that show conformance to RUP, ICONIX or XP process models while the node Project Process can represent the use of a customized variation of these standard processes for a specific project. The node Organization can be represented by metrics describing the SME. Such metrics may include the size of the organization, the years of experience, the organization type. The node Project can be identified with project specific metrics such as development type, business area type, Web development. The Activity node can be represented with the standard activities performed in software development like Planning, Specification, Design, Build, Implementation and Testing. Depending on what aspect of project planning has to be improved, these nodes can represent the relevant quality metrics for each activity or effort metrics for each activity. The node Human Resources can be represenetd with metrics like the team size, while the node Software Resources can be represented by metrics such as Use of Case Tools, Programming Language, Data Base. Finally, the node Hardware can be replaced by metrics such as the Development Platform and the Architecture type.

Figure 2 represents the Bayesian Network as it is formed after the representations mentioned previously. This Bayes Network can then be useful for applying inference. For example, specific tools can be used to redefine the structure of the network based on data from real projects. These data can be further analyzed to create probability tables that show how each node can affect the neighbor ones. 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.

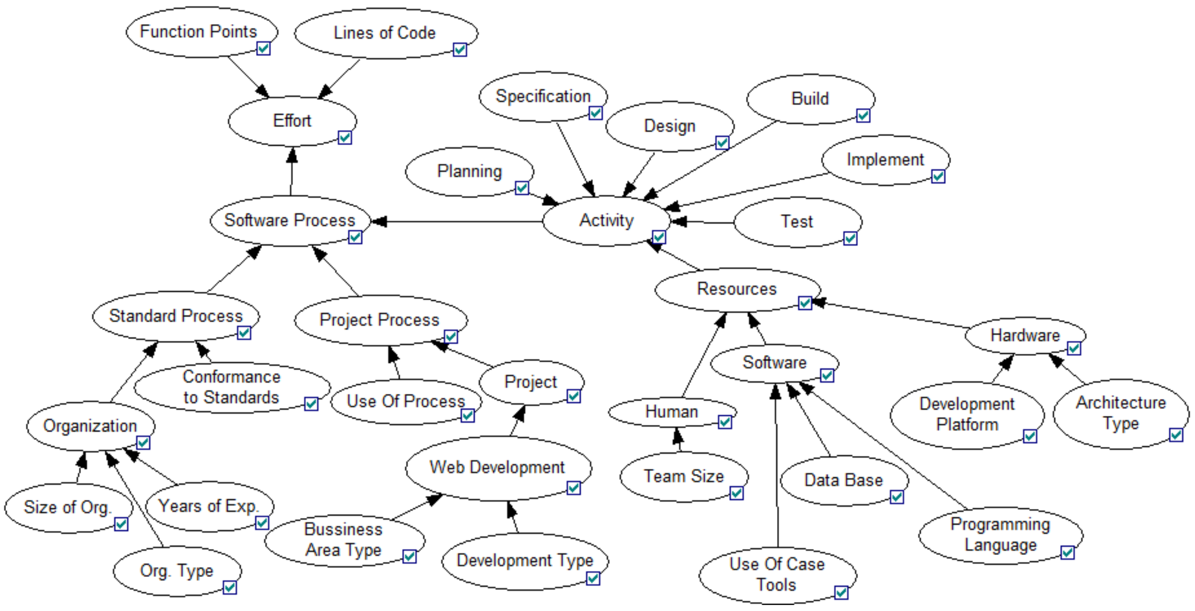

Figure 2: A Bayesian Network for the Software process Ontology presented in Figure 1.

7. Conclusions

This deliverable specified the details of a Knowldege Base that can be used as the basis of software process improvement (SPI) efforts in small-medium sized software enterprises (SMEs). Experience, process and project metrics and data relevant to software development lifecycle are recorded in this Knowledge base. Also a lightweight framework, namely the SPRINT-SMEs approach, has been presented in order to guide and support software process improvement for SMEs. The SPRINT-SMEs approach is based on a four-step assessment and improvement process of a particular software process domain. The approach takes into consideration the characteristics and the needs of an individual software organization under assessment and does not demand a large amount of resources and investment costs.

8. References - Bibliography

[1] 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.

[2] Basili, V.R. (1985), Quantitative Evaluation of Software Methodology, University of Maryland, College Park, Maryland.

[3] Cattaneo, F., Fuggeta, A., Sciuto, D. (2001), Putting Coherence in Software Process Assessment and Improvement. Software Process Improvement and Practice, 6 (1): 3-22.

[4] CMMI (2002), Capability Maturity Model Integration (CMMI). CMMI for Systems Engineering, Software Engineering, Integrated Product-Process Development and Supplier Sourcing (http://www.sei.cmu.edu/cmm/cmm.html).

[5] El Emam, K., Drouin, J. N., Melo, W. (1998), SPICE: The Theory and Practice of Software Process Improvement and Capability Determination. IEEE Press.

[6] Zahran, S. (1998), Software Process Improvement: Practical Guidelines for Business Success. Addison-Wesley.

[7] Mishra,A., Mishra, D. (2006), Software Quality Assurance Models in Small and Medium Organisations: a Comparison. International Journal of Information Technology and Management, 5(1): 4-20.

[8] Software Process Engineering Metamodel (2005), www.omg.org.

[9] Kruchten. P. (2003) The Rational Unified Process: An Introduction (3nd Edition), Addison-Wesley.

[10] Wiegers, K.E., Sturzenberger, D.C. (2000), A Modular Software Process Mini-Assessment Method. IEEE Software 17(1):62-70.

[11] Richardson, I. (2002), SPI Models: What Characteristics are Required for Small Software Development Companies? Software Quality Journal, 10(2): 1573-1367.

[12] Niazi, M., Wilson, D. , Zowghi, D. (2006), Critical Success Factors for Software Process Improvement Implementation: An Empirical Study. Software Process Improvement and Practice, 11(2):193–211.

[13] Gerogiannis, V. C. & Ipsilandis, P. G. (2007), Multi Objective Analysis for Timeboxing Models of Software Development. Proceedings of the 2nd ICSOFT Conference (International Conference on Software and Data Technologies), Barcelona, Spain, pp. 145-153.

[14] Gerogiannis, V. C., Kakarontzas, G., Stamelos, I. (2006), A Unified Approach for Software Process Representation and Analysis. Proceedings of the 1st ICSOFT Conference (International Conference on Software and Data Technologies), Setubal, Portugal, pp.127-132.

[15] Settas, D., ,Bibi, S., Sfetsos, P., Stamelos, I., Gerogiannis, V.,C. (2006), Using Bayesian Belief Networks to Model Software Project Management Antipatterns. Proceedings of the Fourth International Conference on Software Engineering, Research, Management and Applications (SERA 2006), Seattle, Washington, USA, IEEE Press, pp. 117-124.

[16] Wieringa, R., Ebert, C. (2004), Guest Editors' Introduction: RE'03: Practical

Requirements Engineering Solutions. IEEE Software 21(2):16-18.

[17] Guillermo Simari, Iyad Rahwan, Argumentation in Artificial Intelligence, Springer, 2009.

[18] http://en.wikipedia.org/wiki/Knowledge_base

[19] Nonaka, Ikujiro (1991). "The knowledge creating company". Harvard Business Review 69 (6): 96–104.

[20] Nonaka, Ikujiro; von Krogh, Georg (2009). "Tacit Knowledge and Knowledge Conversion: Controversy and Advancement in Organizational Knowledge Creation Theory". Organization Science 20 (3): 635–652. doi:10.1287/orsc.1080.0412

[21] Thomas R. (June 1993). "A translation approach to portable ontology specifications". Knowledge Acquisition 5 (2): 199–220.

[22] Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., Giannopoulou, E.: Ontology Visualization Methods - A Survey. ACM Computing Surveys, 39, 4, Article 10 (2007).

[23] Bringuente, A. C. O., Falbo, R. A., Guizzardi, G. (2011), “Using a Foundational Ontology for Reengineering a Software Process Ontology”. Journal of Information and Data Management, vol. 2, n. 3, pp. 511-526.