ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2005, Vol. 34, No. 2 A

DEVELOPMENT OF CASE TOOLS FOR SOFTWARE PROCESS IMPROVEMENT

Eduardas Bareiša, Eimutis Karčiauskas, Tomas Blažauskas

Software Engineering Department, Kaunas University of Technology
Studentų 50-404, LT-51368 Kaunas, Lithuania

Abstract. In this paper issues of creating case tools, implementing capability maturity models, for software process improvement are analyzed. A range of aspects of using the maturity models were analyzed to this end; these aspects were taken into consideration when designing the case tools structure. The group of existing case tools is reviewed, their benefits and limitations are displayed. The inner structures for organizing the model and data under selection are analyzed. Solutions and issues regarding creation of capability maturity models development environments are discussed. Recommendations for the development of analogous tools are presented.

1. Indroduction

Software Capability Maturity Models is a rapidly growing area of software engineering. Its origins seek the second part of the eighth decade in USA when a short description of process maturity framework and the maturity questionnaire were created. This questionnaire was used to estimate the places of the organization's software development process to be improved. After a while, having applied the collected data of the assessment of software products development processes the Capability Maturity Model for Software (SW-CMM) was created. This model was a success, as well the need for applying this model in other areas (e.g., system engineering) kept growing. Consequently this model was „split", i.e. a few versions of this model (SE, SSE, SA, IPD, People) were developed. Later there emerged a new „integrated" CMM version – CMMI ([6]). In parallel with the models of the CMM ([8]) family the "European reply" to these models have been created: ISO/IEC 15504 (SPICE) ([9],[10]), starting with the year 1993. The main difference from the SW-CMM was not the evaluation of organizations, but of key processes (a continuous representation). It is worth mentioning that the experience with ISO/IEC 15504 standard was used in the development of the CMMI model. The CMMI model has both cascade and continuous representation; therefore it is compatible with the ISO/IEC 15504 model. Several models and guidelines have been produced to support assessments and subsequent improvement process ([1],[3],[4],[5],[7]).

Observing the evolution of software quality standards, models and methodologies the following stages may be distinguished: creation, split or development of separate versions, integration and consolidation. Generally the split is influenced by its successful application in a certain area. Then it is attempted to adapt in other areas. Eventually there raises a need to have a generalized solid model, consequently an integrated model is being created, thereby considering the contemporary decisions of the competitors (the mentioned American CMM and the European ISO/IEC 15504 (SPICE) are constantly being developed incorporating each other's experience).

Beyond any doubt all these changes in a relatively short period (less than 20 years) brought in some confusion not only for enterprises (that wish to implement a standard for maintaining the software development process), but also for software developers that produce instrumental tools for facilitating the implementation of capability maturity models. It was more the whole evolutional process mentioned above than the created different models (ISO9000, CMM and ISO/IEC 15504) that brought in the confusion. Maybe this is the reason why the number of software suitable for a wide range of consumers is not so huge this day.

Developers of tools for Capability Maturity Models are interested in model development frameworks. Unfortunately, commercial software developers often do not reveal the methodologies they use for tools creation and material development; whereas related Open Source software is developed by separate persons and in most cases it use no mature development methods. Having initiated the project of creation tools supporting assessment, definition and implementation of mature software process based on SPICE model, in the first phase, we focused on creating the effective collaborative development environment expecting that the companies and communities of the universities participating in the project will be capable to maintain

the model's development after the project is terminated.

Our experience in creating model development environment will be presented throughout the paper. Section 2 covers an overview of existing related software. The merits and demerits of existing commercial and open source software are investigated. In section 3 a distributed architecture of model development and software process improvement tools is presented. Section 4 describes issues on organising model data structure. In section 5 we investigate the means, which can be used to organize the model material. In section 6 we discuss the issues related to collaborative model development environment. Section 7 completes the paper with concluding remarks.

2. Software overview

All software for implementing and maintaining capability maturity models can be divided according to the model under maintenance: CMMI or ISO/IEC 15504 (SPICE). The other aspect of division is open source software and commercial software. Analysing open source software the attention can be drawn to the fact that the better-known tools are developed for the models of the CMM family (according to the survey presented in [2]). Beyond any doubt there exists some non-commercial tools for ISO/IEC 15504 (SPICE). But in most cases this is exploratory software. The need for tools, enabling efficient and reliable capture of assessment data is addressed in the SPICE project through notion of an Assessment Instrument. A number of tools emerged to support the data collection and storage procedures (for example the Seal tool [12]) [11]. Other researchers investigated effective ways to visualize acquired assessment data. The software by Robin Hunter (Strathclyde University) can be mentioned as an example; this software is created using MS Access and its aim is to explore software process visualization (in the paper [13]). We can state that there is no strong open source society which would be capable of developing software maintaining capability maturity models constantly. Generally such enterprises and persons whose main activity is the assessment are producing open source tools; for this purpose they develop their own tools. It goes without saying that such tools cannot match up to professional commercial software in point of the versatility. Open source software is more oriented either at the representation of the model itself, either at the basic calculations (i.e. calculations that are prerequisite to the assessments); meanwhile, the visualization subsystem of the assessment data is developed less than in commercial software, the attention paid to the maintenance of the process implementation and improvement is low.

Analysing commercial software the main attention was paid to the commercial tools of one of the leading companies HM&S; these tools are designed for working both with ISO/IEC 15504 (SPICE 1-2-1, SPICE LITE, SynQuest) and with CMMI (CMM-QUEST). Such commercial software is distinguished by a relatively huge price (from 700$ to 12.000$ depending on the licence). Its main features are a well-organized subsystem of the assessment presentation, a sufficient amount of tools for maintaining the process improvement, integration with other tools (this is exceptionally highlighted). Nevertheless, despite of all the advantages that allow affirming the existence of efficient assessment and maintenance tools nowadays, not inconsiderable demerits are observed as well. Therefore we can expect a growth of tools and more variety in tools organization (architecture).

3. Distributed architecture of tools

Selecting the tool architecture was mainly influenced by the striving to fulfill the needs of various user groups – from those who study software engineering to those who participate in the evaluation process.

A major part of the known tools, discussed in Section 2, operate as a standalone program or a package that includes both the model information and the repository of data under selection and analysis and the additional material. Of course, a tool organized in such a way can be used effectively in developing and maintaining software development processes in enterprises; as well it can be used for assessments. However such tool organization is not applicable to the multi-user collaborative development of model. Organizing a continuous development of the model and having some institutions under participation it is more convenient to organize software of a distributed architecture.

Developing system architecture the objective was to detach the information about the model and its material from the information of accounts and assessments. Such a detachment enables:

- a more flexible possibility for updating;
- better possibilities for security: it is possible to use the model information that is updated globally and contained on the external server, whereas the data about firm and firm assessments can be stored in the inner servers of the firm;

We specially did not present the particular deployment diagram in Figure 1, because there can be various deployment scenarios when using separated services. As the model, its material and tools are being developed by several separate organizations, therefore an interface for data exchange has to be provided. This interface is realized on the basis of the web-service. Both software of portals and client executable programs that support the web-services interface can connect to it. This interface enables the organizations to create the development and analysis tools themselves adjusting the material placed in the external servers to the organization needs. "Model services registry" shown in Figure 1 is a package that contains web-services (functions) for performing operations with data of the model and its material. "Account and

assessment services registry" shown in Figure 1 is a package that contains web-services (functions) for performing operations with client information and data of assessments he performs. "Assessment tool (application)" is client software for performing evaluations. It is used in case the users prefer using the environment that is provided by the executable program instead of a web-page. "Assessment tool (portal)" is an Internet page for data gathering and evaluations. "Integrated Development Environment" is a tool for model and its material development. "Development and analysis tools" are utility client programs for material analysis.

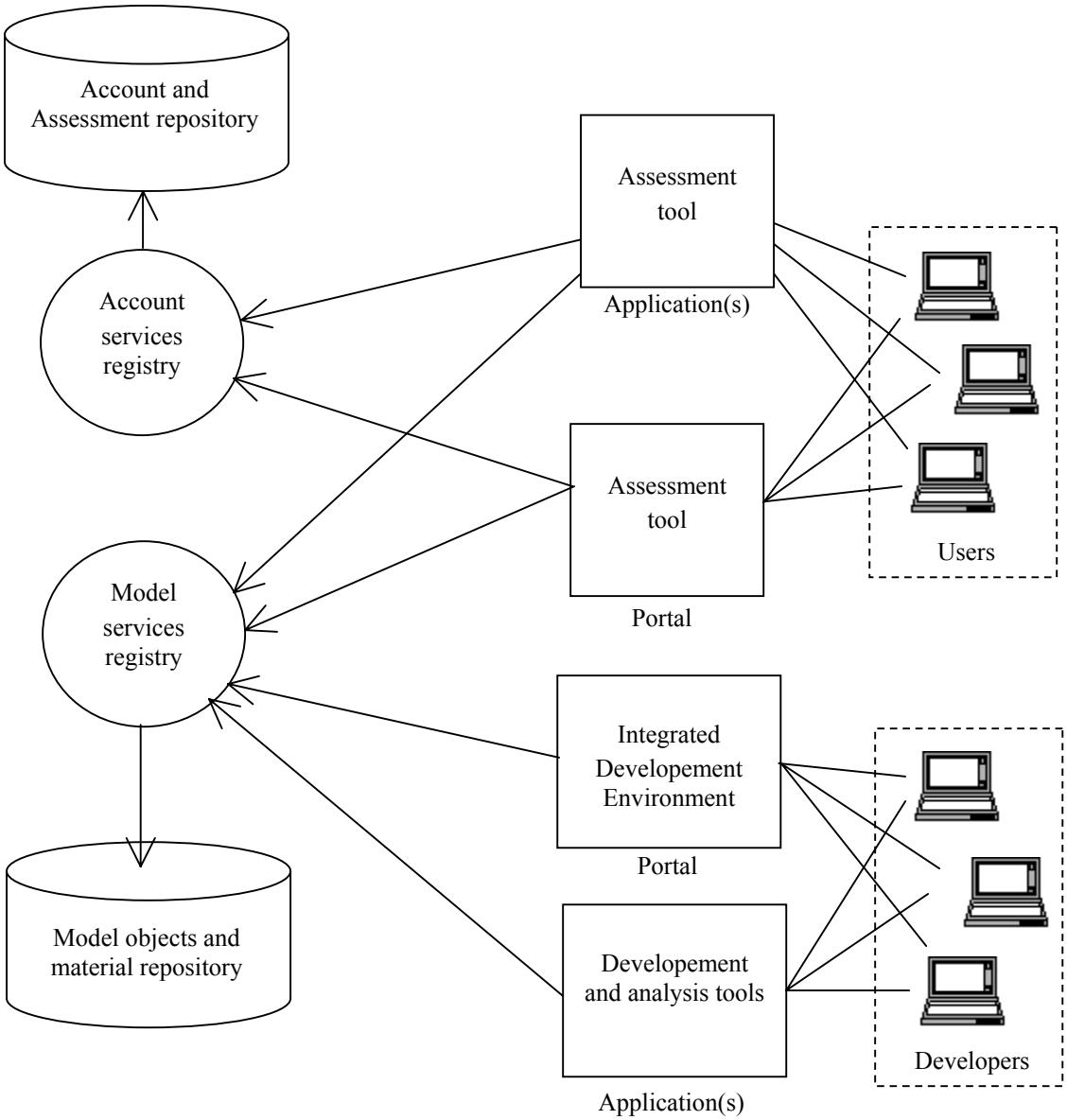

Figure 1. Distributed tools and web-service based architecture

4. Issues on organizing model data structure

In our implementation the MySQL database is used for storing both data of the model under development, material and assessment data. As well this choice was influenced by the requirement for the usage of free technologies that guarantee a high capacity and portability.

The major problem rises in choosing the model structure representation in the database. Two ways were considered:

1. the structure of the database has to reflect the model structure;
2. the model structure has to be detached from the structure of the database.

The organization of the detachment of the model structure from the database would enable the model developers to realize various models or to perform structural modifications without the help of programmers. In other words the markup would allow

controlling not only the features of the model elements themselves but as well their interrelationships. However such flexibility would cause quite a lot of new problems: the absence of a defined structure (in the database) would make the model accuracy control difficult, would require more resources for representing the model structure, additional tags for defining relations would emerge, there would be a need for creating additional tools for representing information during the development.

Despite the freedom the detachment of the model structure from the structure of the database would give the first way was chosen. This choice was influenced by the requirements imposed on our project earlier; these requirements say that the structure of the database should reflect the model structure; furthermore, the database tools should perform a partial data control.

5. Material organization issues

The material of capability maturity models as any other material that has complex relations among a large number of information elements has to be presented on the grounds of hypertext documents. One of the simpler ways is to present material in HTML documents – this enables to present it in web pages, as well there exist components in various programming environments that allow presenting material of HTML documents in executable programs, material can be presented in various systems. Though the storage of material in a form of HTML documents is not convenient, as:

- Generally the material is not separated from the visualization, consequently it is difficult to present in different environments;
- Tags in HTML documents interlace with the material a lot quite often, so the material is hard to read when editing. It is called a „tag soup“ problem;
- Visual editing tools often create false and substandard HTML documents, therefore problems of presenting may occur later;
- The HTML language is not extensible, consequently it is difficult to distinguish custom information elements in documents for an automated analysis of the material.

We think these are the reasons why it is better to create such markup that:

- Would allow separating material representation from markup in order to present the material in various environments;
- Would conform to the “fast markup” (or Wiki) technology in order to make the usage more productive, the depicted material easier to read and to reduce the data flow among the systems during the transfer of data;
- Would enable to identify model elements.

A major part of these objectives can be reached by using the XML markup. The material described in this language can be easily transformed to HTML documents; the language expandability would enable to define model elements; the XML format would allow exchanging data among the remote systems without difficulty. However, the XML markup does not conform to the “fast designating” technology, the non-processed material is hard to read. The overall markup system used in our prototype for the development of the model can be divided into the following main parts:

- The markup of text formatting;
- Object, resource markup;
- Scenario description markup;
- Formula interpretation and result representation markup;
- Scripting markup (php extension).

The subset of MediaWiki tags was selected for formatting the material under development (for text formatting). MediaWiki is an open source software, originally written for the Wikipedia ([14]) project, therefore a part of the MediaWiki system was integrated into the project of the prototype under development. The tags of Wiki formatting are designed for text formatting, therefore a separate subset of tags was created to describe the model and the information elements of the model.

Both the information elements of the model, files of various format and free (non-depending to the model elements) texts of various purposes can be objects and resources in the environment of the model development. The following objects that are used for material structuralization and supplementation are realized in the system:

- ***Category***. A tag that describes the category of processes. Its usage creates a reference to a formal description of the category in the formatted text.
- ***Process***. A tag that describes the nominal process. Its usage creates a reference to a formal description of the process in the formatted text.
- ***Practice***. A tag that describes the basic practice. Its usage creates a reference to a formal description of the basic practice in the formatted text.
- ***Work product***. A tag that describes the work product. Its usage creates a reference to a formal description of the work product in the formatted text.
- ***Term.*** It is used for short descriptions of the terms under usage. If the term used in the material is on the term table, its description is presented as an alternative text.
- ***Picture***. A picture as a term as well is a free object that is not related with other objects (the dependence to the descriptions of the objects mentioned above is not being formed). Pictures are stored in

non-structuralized storage (in one directory), therefore it is strongly recommended to provide the pictures or at least the keys of their relationship with meaningful names.

- ***Path.*** This object is used in such a case when the relational path to the object server in the directory structure has to be created. It is useful when, e. g., one desires to refer a picture using the HTML designating.
- ***Media***. It can be any multimedia object. It is unrelated object as well.
- ***Free object***. It is a textual resource that is not assigned to the formal material of the model. Generally this is the material used for an informal description of the model.

The scenario scripting markup subset was designed on purpose to use it for user training. There is a possibility to describe various scenarios of work with the program, to keep the track of user actions and to react to them in an appropriate way. The actions performed by the user are being memorized; therefore there is a possibility to react to the user training progress by selecting the learning material for the user. Moreover, the scenario marking subset can be used for an interactive representation of informal material of the model.

Often during the modeling certain calculations have to be performed, controlled and presented. Our system has several ways for doing that, e. g. the usage of scenarios or the programming possibility. However, a separate markup is formed for the ease of application. This markup is created so that the description of the mathematical formula would conform to the traditional marking used in programming languages.

Situations may occur when the possibilities of the system marking mentioned above are not enough for realizing the idea. Developers have a possibility to program just in the material under editing. The interpretation of the written program is transferred to the php interpreter and the system prepares the variables and translates them into a subset of the php language. We have not decided yet whether to leave this possibility in the final version or to remove it due to the possible problems of security. Though this possibility will probably be removed due to a too huge hazard – the problems could be explored on purpose to distinguish a safe subset of the programming language.

6. Model development environment

Creating the model development environment the main attention was paid to the analysis of Wikipedia development environment whose main features are:

- It is a multi-user collaborative development environment that has no developed hierarchy of rights (a major part of the material can be edited by anyone);
- It has a textual (not visual) environment of fast editing; this environment uses a special Wiki markup;
- The users create development templates in the same environment;
- It has tools for controlling the system resources, for keeping the track of changes, for material analysis, for user control.

The Wikipedia project uses this environment and it exists successfully, however the requirements for systems differ and consequently it is not enough to use the development environment just what it is. We think that the main disadvantage of the Wikipedia project is that the development environment is "width"-oriented (the amount of the material) and not quality-oriented. As a result of such orientation the material can be edited practically by anyone; there is no defined process of the material development and the formed groups of users use their own methodologies; the absence of the developers' hierarchy burdens material and pattern quality assessment and validation (qualified developers are usually involved in unnecessary disputes); the absence of project control tools does not allow developing the material purposefully (material development is spontaneous), to keep the track on the progress.

Considering the advantages and the disadvantages of the Wikipedia project and the other requests of project participants the following requirements for the model development environment were formed:

- The development environment has to provide with a model material editor, which enables to automate those formatting actions that are performed frequently;
- The development environment must have a developed system of authorization and material control; this system enables to define the rights for the material under development;
- The development environment has to provide a subsystem of changes tracking;
- The development environment has to provide a subsystem of the project control;
- The development environment has to provide a subsystem of patterns and other utility resources development;
- The development environment has to provide the import/export subsystem.

These subsystems had to be improved by revising the requirements for the development environment and the need for additional subsystems during the development of the prototype.

Developing the prototype a large attention was paid to the editor of the model's material, because the editor's convenience influences the usability of the development environment in essence. In the first stage of developing the prototype a textual editor is intended to be created by automating the text formatting functions and the model element identification insertion

functions. Considering the features of editor handling the decision will be made if a visual editing environment is worth to be composed. Composing a visual editing environment a range of problems would occur as during the editing the text under formatting in the client program has to be converted to the Wiki markup and vice versa when the resources distributed in client area and in server area will be used. Therefore a visual editing subsystem will be composed only under the necessity.

The authorization and material control system has to meet the requirements for document management systems in substance. Each information element of the model of the material, defined by a unique identification, has to be treated as a separate document. The subsystem has to ensure the document (material) load, withdrawal for editing, return, lock, right setting and other functions performed by document management systems.

The track changes tool is the main tool of the Wikipedia project; this tool has a huge influence on improving the material quality. One of work techniques recommended by the Wikipedia developers is to look over the list of the latest changes, to make comments on the changes performed, to complement the changed material with a new content. Thus, starting with a small initial content, the material is being developed to a thorough resource. The key requirement for the track changes tool is a good mechanism of setting and presenting the differences among the versions; such a tool enables to track changes in a short time.

The project control subsystem is required for ensuring a purposeful development of the model and its material. It has to ensure the setting of work inclusion and their priority, terms, the required resources, executables, and the execution state. Depending on the demand, the project control subsystem can be designed as a constituent of the development environment or a template for work tracking can be composed and filled in the system according to the users' convention. The latter version is chosen in the prototype – the need for the project control subsystem will be evaluated upwards.

The templates can be used both in the model development (e. g. the development of templates for the project control) and for the assessments (e. g. the formation of the assessment report for the export can be performed using templates). The templates should be composed using the same editor of the material, however the template information itself should be detached from the model and its information.

The demand for import/export tools emerged having noticed that the Office programs (such as Excel, Access) are useful in analyzing the information elements of the model under development and the development of additional tools is not purposeful in that case. Furthermore, project developers and assessors have mastered the Office programs; it is more convenient to perform a partial analysis in the Office programs namely. A fair amount of the open source software is developed by the assessors and is adjustable for the Office programs namely (e. g. CMM Browser is for MS Access, IME tools – for MS Excel). In fact all the presented material can be exported. The material is always being formed as a HTML document; such a document does not include the elements from the development environment. However such a method is more suitable for importing into the environment of the MS Word program. A separate subsystem for exporting various information elements from (and importing into) the MS Excel program was created; this subsystem allows to exchange data by using the CSV[1] format.

7. Conclusions

In this paper a short overview of existing software is given. Various aspects of creating CASE tools for software improvement process were investigated; a big attention is paid to the maturity model development environment. Creating the model development environment it is particularly important to take into consideration that:

- A part of capability maturity models are still being improved and transformed intensely;
- The enterprises that plan the assessment often treat the capability maturity models as a framework for software development; they desire not only the formal material of the model but also specific examples, templates that are not provided by the formal material of the model;
- The enterprises endeavor to store the assessment data in their servers;
- The enterprises store the assessment data – as many other reports – in the Office documents.

Generalized solutions and schemes are presented in this paper; those aimed to ensure that:

- The model, its material and the additional information material will have to be under a constant development;
- The formal, information and systemic material of the model and the assessment data have to be separated;
- The material can be distributed with a view to the flexibility of the material updating and matching up with the unwillingness of the enterprises to store the assessment data in the external servers;
- The access to the model information elements and material can be realized on the basis of the web-services in order to be able to create independent tools;
- Tools and development methodologies for the model development environment have to be oriented at the quality of the model material;

[1] CSV – Comma Separated Values, a textual file format usually used to exchange data.

- Both the model development environment and the assessment system must have the interface with the Office programs or a possibility to import/export data.

The success of collaborative model and its material development will largely depend on how model developers will accept model development environment. Further research must be performed in order to propose methodologies and effective set of tools for constant model and its material development.

References

[1] **M.C. Paulk, C.W. Weber, B. Curtis, M. B. Chrissis.** The Capability Maturity Model: Guide-lines for improving the software process. *Pittsburg: Addison Wesley*, 1993.

[2] **E. Karčiauskas, T. Blažauskas**. Brandaus programų kūrimo proceso programinės įrangos apžvalga. *Informacinės Technologijos* 2004 *konferencijos medžiaga*, *Kaunas*, 2004.

[3] **P. Kuvaja, A. Bicego.** BOOTSTRAP – an European assessment methodology. *Software Quality Journal*, *Vol*.3, 1994.

[4] **B. McFeeley.** IDEAL: A User's guide for Software Process Improvement. *SEI, Pitsburg, Handbook CMU/SEI-96-HB-001*, 1996.

[5] **K. Caputo.** CMM Implementation Guide – Choreographing Software Process Improvement. *Massachusets: Addison Wesley*, 1998.

[6] **SEI**. CMMI for Systems Engineering/Software Engineering, Version 1.02. *CMU/SEI-200-TR-019, Carnegie Mellon University/Software Engineering Institute, Pittsburg*, 606.

[7] **R.B. Grady.** Successful Software Process Improvement. *Prentice Hall*, 1997.

[8] **M.C. Paulk, C.W. Weber, B. Curtis, M. B. Chrissis.** Capability Maturity Model for Software. *SEI*, 1991.

[9] **ISO**. ISO/IEC 15504 Draft Standard for Software Process Assessment (Parts 1-9). *International Standards Organisation*, 1995.

[10] **K. El Emam, J.N. Droin, W. Melo.** SPICE: The Teory and Practice of Software Process Improvement and Capability Determination. *IEEE*, 1997.

[11] **R. Hunter, G. Robinson, I. Woodman**. Tool support for software process assessment and Improvement. *Software Process: Improvement and Practice*, *John Wiley and Sons Ltd*., 1997.

[12] **R. Him Lok, A.J. Walker.** Automated tool support for an emerging international software process assessment standard, *ISESS* 97, *IEEE Computer Society, Wlnut Creek,* 1997, 25-35.

[13] **R. Hunter, C. McCallum**. A visual approach to software process improvement. *Software Process Improvement* '98 (SPI98), *Monte Carlo*, 1998.

[14] Wikipedia. *http://www.wikipedia.org*.