Tool Support for the Typed Generic Process Model

D. Rodriguez, R. Harrison, M. Satpathy
The University of Reading
Reading, RG6 6AY, UK
E-mails: {d.rodriguez-garcia, r.harrison, m.satpathy}@rdg.ac.uk

J. J. Dolado
The University of the Basque Country
20009, San Sebastian, Spain
E-mail: dolado@si.ehu.es

Abstract

The Typed Generic Process Model (TGPM) [10] is a generic template which can be instantiated to produce the quality model for any software process. Consequently, the idiosyncrasies of each of the software processes can be addressed, and further, such instantiated quality models can form the basis of assessment and improvement of the concerned processes. The distinctive feature of the TGPM is that it makes the relationship between the product quality and process quality transparent; it provides a platform to integrate and analyse product and process metrics in an explicit manner. Since a process structure may be hierarchical, tool support is necessary to manage the volume of information associated with a process and the associated products. In this paper, we present such a tool and show how this tool can provide effective guidance to a broad spectrum of software personnel including developers, managers and quality assurance engineers.

1. Introduction

A measurement programme is essential in order to assess, predict and improve the quality of software products and processes. In this paper, by a process we will mean any software activity associated with the development and maintenance of software: from requirement analysis through to maintenance. Process models like the Capability Maturity Model (CMM) [4], ISO/IEC 12207 [7], ISO 9001/9000-3 [6] etc. are not specific enough to cater to the needs of the whole spectrum of software processes. The reasons are: (i) the nature of processes vary widely, ranging from requirement analysis to maintenance (ii) most of the models are more oriented towards enhancing the maturity of an organization and take a monolithic view of the overall development process, and finally (iii) such process models, while emphasizing on the process activities, often put too little importance on the products which are the results of the process activities.

Product quality aspects are usually addressed by models such as ISO 9126 [5], FURPS+ model [3] etc. However, how such product models are related to the above process models, has not been properly addressed. In order to alleviate these problems, Satpathy et al. [10] have defined a Typed Generic Process Model (TGPM). The generic model is a generic template which in turn could be instantiated to generate the customized model of any individual process. This instantiation mechanism clearly spells out the idiosyncrasies of the individual process that needs special attention and further, it makes the relationship between process model and product quality explicit through the dual interpretation of process attributes. By dual interpretation, they mean that a process attribute may have two aspects: a process aspect and a product aspect.

The TGPM defines a process as a relationship between a set of input products and a set of output products. As we will discuss later, a process may consist of subprocesses. Because of the hierarchical nature of processes, and that each process or subprocess has input products and output products, the volume of information that we can associate with the whole hierarchy can be very high. And depending on a requirement, we need to extract the relevant information from the database associated with the process hierarchy and perform necessary computation on them to derive certain results. Therefore, a tool support becomes imperative when we deal with the processes under TGPM.

SEGESOFT is a project management tool [13], which provides an environment for training project managers. The goal of this tool is to have a uniform structure so that

analytical techniques can be incorporated into the structure smoothly. The system collects and records both actual and simulated project data and implements different techniques such as machine learning, project tracking, dynamic modelling, etc. The basic assumption of this work is that management decisions should be supported by integration of different sources of information. In this paper, we have upgraded this tool to accommodate the distinctive features of the TGPM for quality assessment and improvement.

The organization of the paper is as follows. Section 2 describes the related work. Section 3 explains the TGPM; in section 4, we explain the rationale for tool support. The characteristics of the tool are explained in Section 5 and a case study is discussed in Section 6. Finally, in Section 7, we present our conclusions and future work.

2. Related Work

ISO/IEC 9126 [5] describes a generic model for specifying and evaluating the quality of software products. The model isolates six factors, called Functionality, Usability, Reliability, Efficiency, Maintainability and Portability; and the quality of a product is defined in terms of the quality of the above factors. Each factor may in turn be defined by a set of subfactors. The FURPS+ model [3] used by HP is quite similar to ISO/IEC 9126.

Focusing on product quality alone may not guarantee that an organization will deliver products of good quality. Products are created by processes. So, based on an orthogonal view that improving the quality of a process will deliver products of good quality, many models have been developed. Prominent among them are the CMM [4] and ISO 9001 [6]. Models like BOOTSTRAP [9] and SPICE/ISO 15504 [8] are variants of the CMM. ISO/IEC 12207 [7] does a classification of all processes associated with the software development and offers general guidelines which can be used by software practitioners to manage and engineer software. The scope of most of these standards cover an entire organization.

The GQM method [1, 12] proposes a measurement plan for assessing quality of entities like products, processes or people. It starts with a set of business goals and the goals are progressively refined through questions until we obtain some metrics for measurement. The measured values are then interpreted in order to answer the goals. Existing approaches choose a quality model from those that exist so as to generate the business (or the primary) goals of the GQM formulation for any individual product or process.

Focusing on either process quality or product quality alone is not sufficient. In a European-wide awareness survey 65% of the respondents agreed that certification against ISO 9000 is not enough to guarantee the quality of software products [2] 40% of the respondents agreed that a combined certification of products and processes is necessary, and almost all of the models fail to make the relationship between process quality models and product quality clear. TGPM makes such a relationship clear by taking the dual nature of process attributes into account. A brief description of TGPM will be presented in the next section.

3. The Typed Generic Process Model (TGPM)

The TGPM gives types to products and processes. A product is an entity, which a process (e.g. any software activity) produces as output. Products may also be fed to processes as inputs. Table 1 enumerates some of the products and their types. Product ***rd*** belongs to the set of requirement documents, ***spc*** belongs to the set of specifications and so on. TGPM models the set of requirements by the set PRE_ELICIT_REQ. TGPM defines a process as a relation from a set of products to another set of products; the set of relations from *m* input products to *n* output products is denoted by:

$$IP_1 \mathbf{x}\ IP_2 \mathbf{x} \ldots \mathbf{x}\, IP_m \leftrightarrow \quad OP_1 \mathbf{x}\, OP_2 \mathbf{x} \ldots \mathbf{x}\, OP_n$$

where IP_i and OP_j are the types of the *i-th* input product and the *j-th* output product respectively. Table 2 shows the types of a few processes. A process may have many type definitions. For example, if an organization develops formal specifications from requirements only, then the type of the formal specification process (FS process) is: ***RD*** $\leftrightarrow$ ***FS***. On the other hand, if an organization (in relation to a project) also decides to interview the users for developing a FS, then the type of FS process would be: ***PRE-ELICIT-REQ*** X ***RD*** $\leftrightarrow$ ***FS.***

req: *PRE-ELICIT-REQ*	// Set of pre elicitation requirements.
rd: *RD*	// Set of requirement documents
spc: *SPEC*	// Set of specifications
des: *DESIGN*	// Set of designs
doc: *DOC*	// Set of documentations
impl: *IMPL*	// Set of implementations
des:: *DESIGN*	// Set of designs
mod: *MODULES*	// Set of set of program modules
testdata: *TESTDATA*	// Set of test data

Table 1 Types of some products

Requirement analysis: *PRE-ELICIT-REQ* ↔ *RD*
Specification: *RD* ↔ SPEC
Formal design: *RD* x *FS* ↔ *FD*
Maintenance: *RD* x *SPEC* x *DES* x *IMPL* x *DOC* ↔ *RD* x *SPEC* x *DES* x *IMPL* x *DOC*

Table 2 Types of some processes

TGPM then formalizes the internal structure of a process. A process may be composed of subprocesses. A subprocess is also a process in the sense that it takes a product set as input and gives out a product set as output. Whether a process should be decomposed into subprocesses or not is decided by the process designer. When considered from a process point of view, the inputs and outputs of subprocesses are called intermediate products. A subprocess which is not decomposed further is called an atomic process. An atomic process is defined as a set of process steps, like the steps of an algorithm. The steps usually do not have rigorous definitions: a process executer usually has degrees of flexibility while executing them. Figure 1 shows the internal details of the testing process. An organization is expected to have its own definition of the internal structure of a process. For a more detailed discussion on the internal process model, refer to [10].

In Figure 1, testing process takes an implementation (a set of program modules) and a set of test cases as input and produces a tested implementation and Test results as output. As shown in the figure, testing process consists of two subprocesses: unit testing and integration testing. Unit testing consists of two atomic processes: black box testing and white box testing. Integration testing is an atomic process. Black box testing and white box testing can proceed in parallel, both producing their respective test results. For simplicity, we will consider the only scenario where the implementation passes the test cases. After the unit testing, the program modules are integrated by the integration process. Since the process of integration is not a part of the testing process, following the notations of TGPM, it has been shown outside the scope of the testing process. The integrated modules and the test cases are then fed to the subprocess integration testing which produces a tested implementation along with test results.

Once the internal structure of a process (or a subprocess) is in place, the TGPM is next defined in terms of eight factors: Functionality, Usability, Efficiency and Estimation, Visibility and Control, Reliability, Safety, Scalability, and Maintainability; each subfactor is in turn defined by a set of subfactors. The definitions of the subfactors of TGPM can be found in [10].

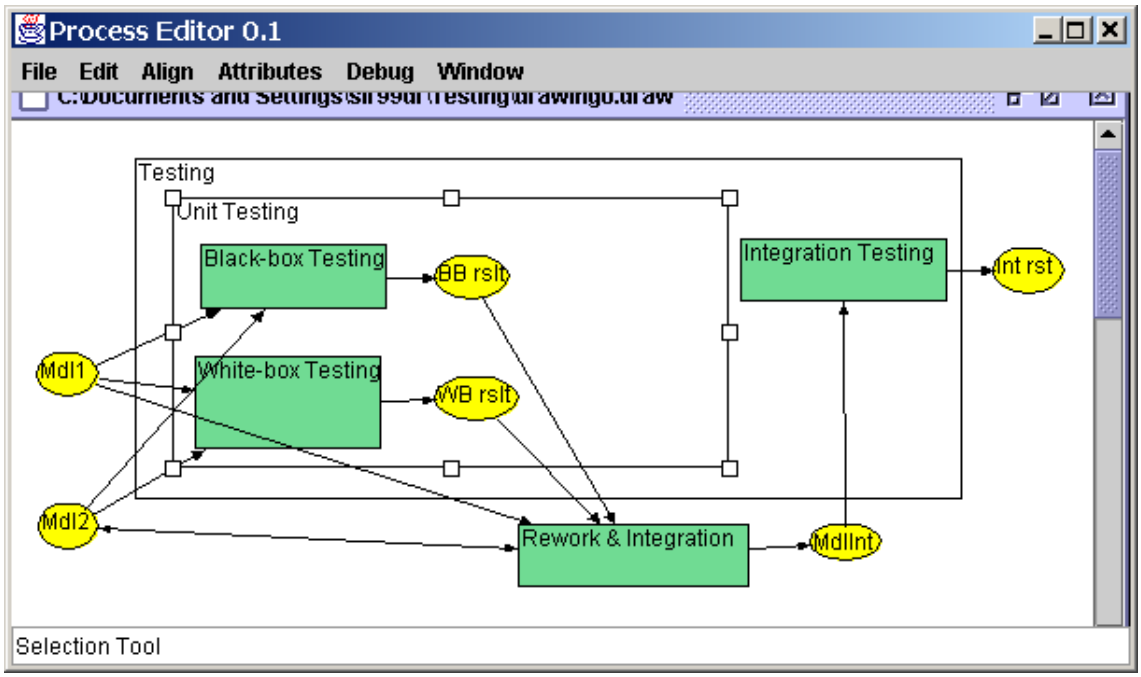

Figure 1. Testing processes and products associated

The most important aspect of TGPM is that it defines attributes from a dual perspective. Consider, for instance, the understandability attribute. The two perspectives are (i) the logical concepts of a process should itself be understandable to the process executer and further (ii) the process should make its output product sets understandable. When we consider the formal specification process, the method of creating the formal specification and the formal specification language itself must be understandable to the specifier, and further the specification process must make the formal specification understandable to its users. In conclusion, any process assessment must consider such a dual perspective. TGPM could be seen as a template with three parameters:

TGPM (input-product-set-type,
output-product-set-type,
application-domain)

The input product and the output product sets have been discussed earlier. By application domain we mean whether it is a safety-critical application, a real-time application, a business application etc.

The customisation of TGPM proceeds in two steps: (i) the substitution step and (ii) the refinement step. In the substitution step, we substitute the parameters with their actual bindings. Let us take the example of the FS process and let its type be: ***RD↔FS***. The substitution is illustrated by the expression:

TGPM [*RD* / input-prod-set-type]
[*FS* / output-prod-set-type]

What the above expression signifies is that, all occurrences of the input product set in the definitions of the factors and the subfactors in the definition of TGPM are substituted by RD (requirement document). Similarly, all occurrences of output product set are substituted by FS (formal specification). Thus, at the end of the substitution step, we have a crude definition of each of the subfactors of the process concerned.

For the *application-domain parameter*, suppose we have a safety critical application. Now, with the knowledge that we are dealing with a FS process in a safety critical application, the refinement step refines the crude definitions that we have obtained after the substitution step. The result then will be the customized quality model for the FS process. As an illustration, consider the first part of the subfactor 'completeness'. After the substitution step, we obtain the following definition: *the degree to which the process transforms all of the functionalities of the RD into the FS*. In the refinement step we know that it is the FS process and the application domain is the 'safety critical application'. Further FS

process achieves transformation through specification; and at the RD level, a functionality is understood by a 'feature'. So the refined definition is: *the degree to which the FS process specifies all of the features (including the safety critical features) of the RD in the FS.*

The distinctive feature of TGPM is the dual perspective of a process attribute. In plain terms, it says how good a process attribute is in relation to the process itself and then how it is related to the output product(s) of the process and also to the end-product. This duality point of view identifies certain process features such as (a) process faults, (b) process understandability, (c) process scalability, (d) process reliability, (e) process stability etc. These features were not properly addressed by any of the previous process models.

4. Rationale for Tool Support

Following the conventions of TGPM, we may need to keep the following categories of information in relation to a process.

1. A process is either an atomic process or it is defined in terms of subprocesses. So, documentation is necessary to specify the process structure.

2. A process is defined as a relation between a set of input products and a set of output products. Some documentation is necessary to say what such products are.

3. A process can have a hierarchical structure (see Figure 1). Each process or subprocess takes a set of input products and a set of output products. If we follow the ISO 9126 product quality model, then each such product can have quality factors, subfactors. Further each subfactor may be associated with metrics. So we need to store all such information which may be associated with the products. Since a process can be hierarchical, the number of such products and hence the volume of information associated with them could be significant.

4. Any process can have an instantiated model (instantiated TGPM). Then, in relation to the instantiated model, each process will have quality factors and subfactors. Further corresponding to each subfactor there may be metrics. So, each process should store all such information.

Since there can be a number of processes, each process can be hierarchical and each subprocess will have the above 4 types of information associated with them, it is clear that the volume of information can be very high, and therefore tool support is essential.

5. Tool Support for TGPM

SEGESOFT is a project management tool [13] which integrates simulation and dynamic modelling, knowledge discovery, quality models etc so that a manager can obtain a broad spectrum of information which can make project management easier. We have upgraded this tool so that it can incorporate TGPM features, and we will refer this upgraded tool as Upgraded SEGESOFT (or USEGESOFT). We will now describe the distinctive features of this USEGESOFT in relation to the TGPM.

USEGESOFT provides a database where all information associated with processes, products and their relationships can be stored. It also provides a graphical editor using which one can draw diagrams. The tool also helps in building and maintaining the internal structure of a process. The diagram in Figure 1 that represents testing process has been drawn using this editor. Since the internal structure of a process could be hierarchical, the tool also supports nested hierarchical diagrams. Figure 3 represents a snapshot of the graphical editor which supports 'drag and drop' mechanism.

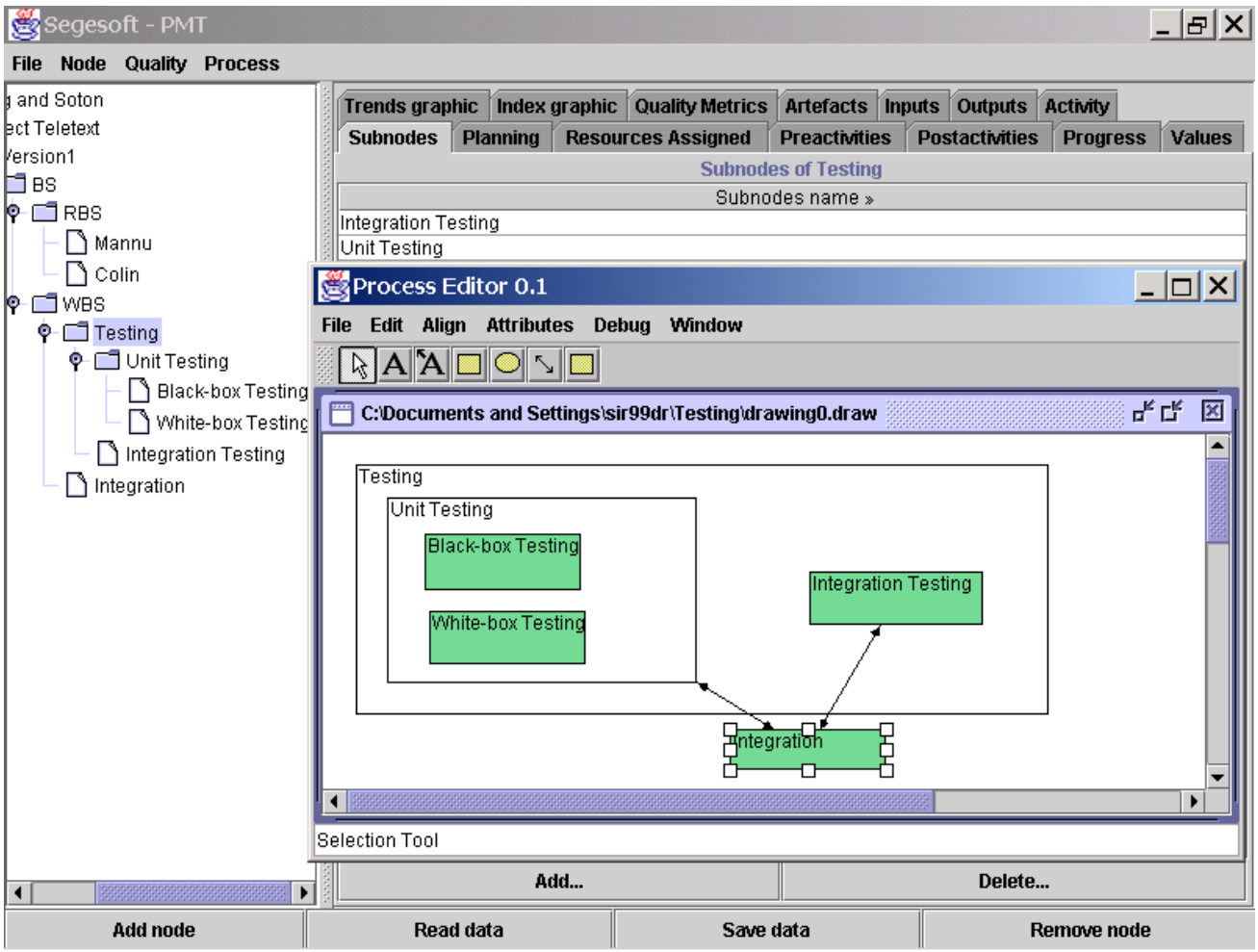

Figure 2. Drag and drop mechanism of the Process Editor

As mentioned in the last section, we need to store four types of information in relation to a process. USEGESOFT tool provides an adequate documentation mechanism to specify, store and display all the four types of information which can be associated with a process or subprocess. This information can remain hidden, or it can be displayed through a pop up menu. For example, when a process component is selected in the Process Editor, a context menu associated with it allows us to store or visualize all information associated with that component. The snapshot of Figure 3 shows the high-level information of the testing process through a pop-up menu. It is also possible to follow from the pop-up menu in a nested fashion to extract further information, if any. Alternatively, information can also be obtained by following the trail of entities from the main window. Figure 4 demonstrates this.

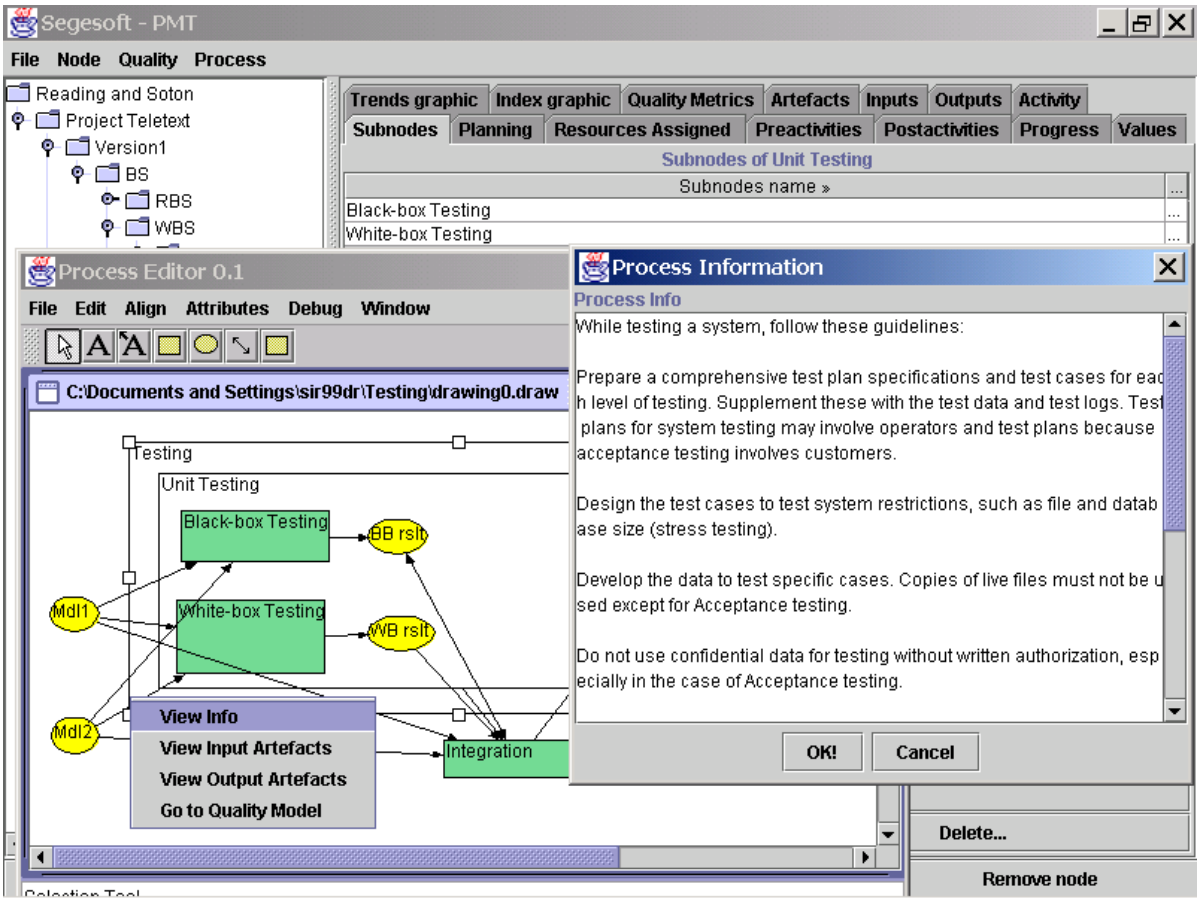

Figure 3 Accessing the process information from the popup menu in the Process Editor

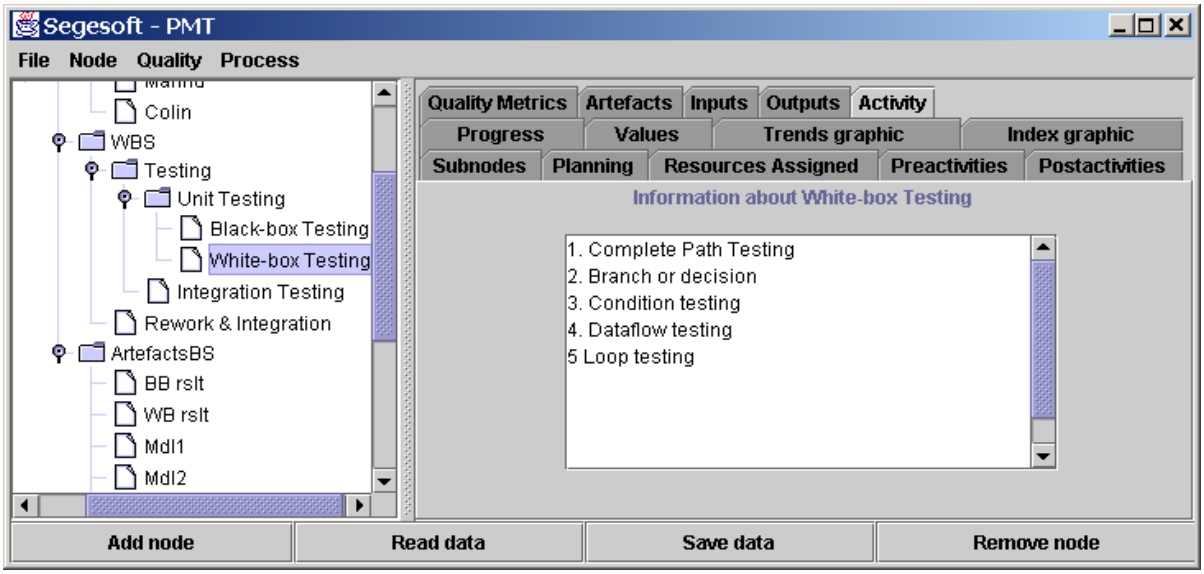

Figure 4. Atomic Information from USEGESOFT Main Window

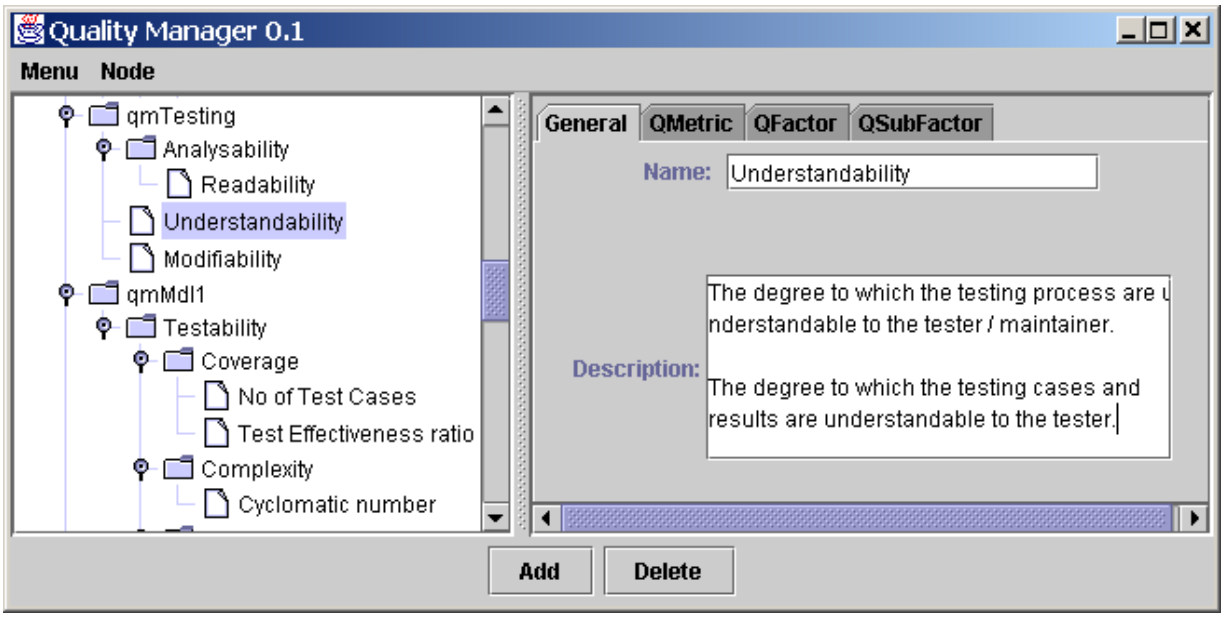

Figure 5. Process quality attributes according to TGPM

Figure 5 shows how definitions of various process quality attributes according to the TGPM, along with their dual definitions, can be obtained.

5.1. Information Retrieval and inference mechanism

USEGESOFT stores a lot of information in relation to a process and the associated products. Depending on a requirement, we need to obtain relevant information from the database, which stores all the documentation. And further, we need to use this information to do some graphical display or make inferences by using some statistical modelling approach. Currently, as quality information is stored using the open standard XML (Extensible Markup Language) [14] we are able to extract this information manually or transform it using XSL (Extensible Stylesheet Language) [15]. In this way it is possible to manipulate or display the data using a spreadsheet or other data processing applications. More automated extraction of this information using a script language and their visualization is a part of our ongoing work.

6. Case Study

Satpathy et al. [11] has discussed a case study in which they took the teletext module of a new generation TV from Philips Electronics and specified it formally using B and informally using UML. Then they have done a comparative study of both the specifications. They have performed their specification in the TGPM framework and they also have collected various metrics for future analysis. We have used our USEGESOFT tool in creating all diagrams, and storing all relevant information associated with those diagrams. For this case study.

7. Conclusions and Future Work

The TGPM is a generic process model, which is instantiated to obtain process models for individual processes. The amount of information that needs to be stored with processes could be very high. USEGESOFT is a tool that provides a framework for storing various artefacts associated with the instantiated process models. We have discussed how information can be stored and how, in response to a specific requirement, relevant information can be retrieved to make inferences. We have also used the USAGESOFT tool for a case study to show the effectiveness of our tool.

At this stage, the information retrieval and its display are more or less manual. We intend to use a query language and retrieve the necessary information

automatically by using a script language. This is a part of our ongoing work.

Acknowledgements

This work has been supported by The University of Reading and the EPRSC – EMPAF ER/L87347. Also this work is based on a project management tool developed by the SEGESOFT Group, supported by CICYT TIC99-0351 (Spain). Thanks to Wolfram Kaiser for answering questions about the JHotDraw package.

8. References

[1] V. R. Basili, G. Caldiera, and H. D. Rombach, "The Goal Question Metric Paradigm," in *Encyclopedia of Software Engineering*: John Wiley & Sons, Inc., 1994, pp. 528-532.

[2] G. Bazzana and M. Pioti, "Process and Product Measurement," in *Better Software Practice for Business Benefict*, R. Messnarz and C. Tully, Eds. Los Alamitos: IEEE Comp. Society, 1999, pp. 151-176.

[3] R. B. Grady and D. L. Caswell, *Software Metrics: Establishing a Company-wide Program.* Englewood Cliffs, N.J.: Prentice-Hall, 1987.

[4] W. S. Humphrey, "Introduction to Software Process Improvement," Software Engineering Institute CMU/ SEI- 92- TR- 7, June 1992 (Revised June 1993).

[5] ISO, "Information technology -- Software product evaluation," ISO (International Organization for Standardization) ISO/IEC 9126, 1991.

[6] ISO, "Quality management and quality assurance standards," ISO (International Organization for Standardization) ISO 9000-1:1994.

[7] ISO, "Information technology -- Software life cycle processes," ISO (International Organization for Standardization) ISO/IEC 12207, 1995.

[8] ISO, "Information technology -- Software process assessment," ISO (International Organization for Standardization) ISO/IEC TR 15504, 1998.

[9] P. Kuvaja, *Software process assessment and improvement : the BOOTSTRAP approach.* Oxford: Blackwell Business, 1994.

[10] M. Satpathy, R. Harrison, C. Snook, and M. Butler, "A Generic Model for Assessing Process Quality," presented at 10th International Workshop on Software Measurement (IWSM'00), Berlin, Germany, 2000.

[11] M. Satpathy, C. Snook, R. Harrison, and M. Butler, "A Comparative Study of Formal and Informal Specifications through an Industrial Case Study," presented at IEEE/IFIP Workshop on formal Specifications of Computer Based Systems (ECBS), Washintong, D.C., 2001.

[12] R. v. Solingen and E. Berghout, *The goal/question/metric method: a practical guide for quality improvement of software development.* Maidenhead: McGraw-Hill, 1999.

[13] J. Tuya, P. Fernández, M. A. Prieto, J. Aguilar, I. Ramos, J. Riquelme, F. Ferrer, M. Toro, M. Ruiz-Carreira, D. Rodriguez, M. Satpathy, R. Harrison, A. Ruiz de Infante, J. J. Dolado, R. Matilla, and M. A. Álvarez, "Integration of Information in a Training Environment for Software Project Management," presented at Software Quality Management 2001 SQM'01, Loughborough, UK, 2001.

[14] W3C, "Extensible Markup Language (XML) 1.0. W3C Recommendation", World Wide Web Consortium 1998.

[15] W3C, "XSL Transformations (XSLT). Version 1.0 W3C Recommendation", World Wide Web Consortium, 1999.