dpunkt.verlag im Inter

Ina Schieferdecker ·
Stephan Goericke (eds.)

Setting Quality Standards

Since 1997 the international "Confer
Quality Engineering in Software Tec
(CONQUEST) has gathered together t
ware and quality engineering comm
discuss aspects of software quality
share experiences in industrial and
projects.

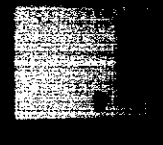
The book constitutes a selection of
sented speeches and lectures. They
first-hand information on the practi
application and ongoing developme
methods and techniques and provid
into real-life case studies along with
quality analysis and evaluation.

The proceedings of the CONQUEST
include

- Processes
- Experience Reports
- Next Generation Testbed for Medi
- Engineering
- Testing
- Secure Software Development
- QA for Finance IT
- Metrics
- Quality and Safety

CONQUEST 2008 is organized by th
national Software Quality Institute
in cooperation with Fraunhofer FO
the German Society for Informatic
kindly supported by the Ministry
mics of Brandenburg, the networ
rity and Safety made in Berlin-Br
(SeSamBB), and the ZukunftsAgen
denburg (ZAB).

ISBN 978-3-89864-567-6



Schieferdecker · Goericke
Setting Quality Standards

Ina Schieferdecker · Stephan Goericke (eds.)

Setting Quality Standards

Proceedings of the CONQUEST 2008

11th International Conference on
Quality Engineering in Software Technology

Potsdam 2008

dpunkt.verlag

Kindly supported by:

LAND BRANDENBURG
Ministerium für Wirtschaft

SeSamBB
Security and Safety made in Berlin-Brandenburg

ZAB ZukunftsAgentur Brandenburg

Ina Schieferdecker · Stephan Goericke (eds.)

Setting Quality Standards

Proceedings of the CONQUEST 2008

11th International Conference on
Quality Engineering in Software Technology
Potsdam 2008

The conference is organised by iSQI – International Software Quality Institute.

dpunkt.verlag

ASQF Arbeitskreis Software-Qualität und -Fortbildung e.V.

iSQI International Software Quality Institute

iSQI – International Software Quality Institute
Am Weichselgraben 19 · 91058 Erlangen · www.isqi.org

Ina Schieferdecker
ina.schieferdecker@fokus.fraunhofer.de

Stephan Goericke
stephan.goericke@isqi.org

Editor: Vanessa Wittmer
Copy-Editor: Julia Neumann, Prince George BC, Canada
Producer: Birgit Bäuerlein
Cover Design: Helmut Kraus, www.exclam.de
Printer: Koninkliijke Wöhrmann B.V., Zutphen, Netherland

Bibliografische Information Der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.ddb.de> abrufbar.

ISBN 978-3-89864-567-6

1st Edition

Ringstraße 19 B
69115 Heidelberg
Germany

543210

Greetings

Stel Pinhasov Beck
Commercial Attaché of the Embassy of Israel and
Director of the Israel Trade Center:

Shalom,

Dear Participants,

Welcome to this year's CONQUEST in Potsdam with its special focus on Israel.

As the Economic Representative of the State of Israel in Germany, I am especially pleased that Israel has been chosen to be iSQI's official partner country at CONQUEST 2008.

This is a great opportunity to present Israel's outstanding technologies to the international software development community.

At the same time, this year's CONQUEST offers an excellent platform to further intensify the bilateral relations in the software and security sector between Germany and Israel.

Israel's software and security industries' reputation as one of the most innovative in the world and the enormous amount of requests to my office from respective companies looking for business partners in Germany already give an idea about the strong potential of future co-operation.

That is why the Israel Trade Center and the Israel Export and International Co-operation Institute have put great efforts into bringing a delegation of Israeli companies to this year's CONQUEST. In particular, the Brandenburg meets Israel BusinessLink with its pre-scheduled B2B meetings will enable experts to get to know each other and to exchange potential business ideas.

I wish all of us an inspiring and fruitful conference and I'm looking forward to meeting you in Potsdam!

Sincerely,
Stel Pinhasov Beck

Greetings

Prof. Ina Schieferdecker, Conference Chair:

I am pleased to welcome you to CONQUEST 2008 in September 24-26, 2008 in Potsdam, Germany. Since 1997, CONQUEST – the International Conference on Quality Engineering in Software Technology – has been the platform for software professionals to get to know the latest methods and tools for quality engineering from industry and research, to listen to practical experiences and show cases, to see how quality engineering methods are successfully deployed in an industrial environment, to learn about recent tool and service offers, to share experiences and to discuss future directions.

CONQUEST 2008 will feature tutorials and presentations of internationally renowned speakers and high-quality peer-reviewed presentations from members of the quality engineering community. It provides a full picture of software quality in theory and practice.

It is a particular pleasure to celebrate at CONQUEST the 10th anniversary of the Software Testing Working Group at ASQF, which was created back in 1998 and currently runs over 20 meetings a year with more than 500 participants. It also contributed to the Certified Tester Initiative in Germany which, together with more than 30 other national testing boards, can currently draw on more than 90.000 certificates worldwide. All working group members, certified testers and interested parties are cordially invited to the Software Tester Welcome, which is supported by the German Testing Board (GTB). In addition, the GTB-

Forum will take place for the first time in Potsdam as an independent, but affiliated event of CONQUEST.

A variety of international speakers and participants will give you practical and research insights into current and topical aspects of quality engineering. This broad perspective will be presented by attendees from all over the world including Europe, Asia and America. Our exhibition on recent methods, services and tools will include 15 national and international companies from the software quality field and will give you detailed insights into their tools and services.

I wish us all a successful and enjoyable CONQUEST 2008.

Yours truly,
Ina Schieferdecker

Greetings

Stephan Goericke,
Director iSQI:

As the Director of the International Software Quality Institute, I'm very pleased to present this documentation issue for the CONQUEST 2008. Meanwhile, it's now the 11th time that software engineers from all over the world have accepted iSQI's invitation to come to be informed and to exchange information on the topic of Software Quality Improvement. This year's new topic concerns itself with "security and safety" – on the one hand, the area of software development and on the other, secure and safe software-based systems.

The 11th CONQUEST offers another novelty, about which I'm particularly pleased. This year, Israel has become the partner country of the conference, to commemorate the occasion of the 60th anniversary of the founding of the State of Israel and the good business relations which iSQI has made in recent years in the high-tech Mediterranean state. Many representatives of Israeli businesses and institutions are involved in the CONQUEST 2008. It is an honour for us to be able to further enhance business relationships and the transfer of knowledge between the two countries. Only then can we reach our goal to better software quality and institute better standards worldwide.

iSQI now supplies certification in over 40 countries, standards now exist in the areas Software Test, TTCN-3, Software Architecture, Project Management, Innovation Management, Configuration Management, Requirements Engineering, the SPICE-Assessors Certification following iNTACS, IT Security Manage-

ment, Secure Software Engineering, Information Security and soon also the V-Model XT. In addition, the iSQI has developed a standard on the basis of these "certified" programs, which incorporates the recognition of many years of practical experience and which allows for a flexible adaption in an individual job situation: QAMP (Quality Assurance Management Professional). If you're interested in this standard, you'll learn more about it during the CONQUEST.

I'd like to thank all those partners who supported us during this CONQUEST in Potsdam and also all those who helped to make the CONQUEST what it is: one of the most important specialized events in the area of software quality.

I wish you all the best and great success in your work and I would be very pleased to meet you at our future conferences and educational programs.

Sincerely,
Stephan Goericke

Program Committee

Conference Chair:

Ina Schieferdecker, Fraunhofer FOKUS, Germany

Program Committee:

Girts Baltaisbrencis, Exigen Services, Latvia
Manfred Broy, Technical University Munich, Germany
Wim Demey, FRSGlobal, Belgium
Winfried Dulz, University of Nuremberg-Erlangen, Germany
Karol Frühauf, INFOGEM, Switzerland
Anne Mette Jonassen Hass, DELTA, Denmark
Georg Heidenreich, Siemens, Germany
Bernard Homès, TESSCO Group, France
Dehua Ju, ASTI Shanghai, China
Dieter Landes, University of Applied Sciences Coburg, Germany
Peter Liggesmeyer, Fraunhofer IESE, Germany
Alon Linetzki, Best-Testing, Israel
Patricia McQuaid, California Polytechnic State University, USA
Pedro Merino, University of Malaga, Spain
Ludger Meyer, Siemens, Germany
Joanna Nowakowska, Oracle, Poland
Mohammad Nuruzzaman, Daffodil International University, Bangladesh
Barbara Paech, University of Heidelberg, Germany
Sachar Paulus, SAP, Germany
Helmut Pichler, ANECON, Austria
Klaus Pohl, University of Duisburg-Essen, Germany
Ita Richardson, University of Limerick, Ireland
Bj Rollison, Microsoft, USA
Alexander Rudolph, Ecolab, Austria

Sergii Riapolov, Ukrainian Scientific Center for Development of Information Technologies (ITDEV) under Ministry of Education and Science of Ukraine, Ukraine
Achim Schmidt, Beta Systems, Germany
Kurt Schneider, University of Hannover, Germany
Andreas Spillner, University of Applied Sciences Bremen, Germany
Maria Stefanova-Pavlova, CITT Global, Bulgaria
Angela Vozella, CIRA, Italy
Mario Winter, University of Applied Sciences Cologne, Germany
Walter Wintersteiger, Management & Informatik, Austria
Peter Zimmerer, Siemens, Germany

Additional Reviewers:

Stefan Wagner, Technical University Munich, Germany
Eva Geisberger, Technical University Munich, Germany
Marco Kuhrmann, Technical University Munich, Germany
David Cruz, Technical University Munich, Germany
Heiko Stallbaum, University of Duisburg-Essen, Germany
Times Illes-Seifert, University of Heidelberg, Germany
Lars Borner, University of Heidelberg, Germany
Jürgen Rückert, University of Heidelberg, Germany
Shamsuddin Ahammad, Daffodil Institute of IT (DIIT), Bangladesh

Table of Contents

Software Process Variant Characterization Using ISO/IEC 9126

T. Martínez-Ruiz · F. García · M. Piattini Velthuis

Departamento de Tecnologías y Sistemas de Informacíon (UCLM), Ciudad Real, Spain

Abstract

Software process lines allow us to generate processes which are tailored to the specific needs of each context. However, the problem lies in attempting to determine if the process variant that will occupy a variation point has the correct performance to customize the new process in the correct way. As a result, the goal of this work is to determine the characteristics of the elements that may vary in a process line, to thus facilitate the selection of variants. The characteristics that are applicable to the software process line variants can be determined by using the ISO 9126 standard, and which values will be evaluated in order to select the most appropriate variant depend on the context. From this study, we can infer that characterization is necessary for selecting the most appropriate variants to adapt processes correctly to each organization and each project.

1 Introduction

In recent years, software development organizations have begun to take an interest in both the quality of the products they develop and the capacity of their processes [Pia 06], due to the fact that software process quality depends on the software process [Fug 00]. From this interest in the latter, models such as CMMI and standards such as ISO 12207 or ISO 15504 have been developed. These describe generic processes for software organizations. Due to the diversity of existing software organizations and because "Just as there are no two identical software projects, there are no two identical software processes in the world" [Hum 89], the need to customize generic software processes to the project and the organization in which they are performed is clearly shown. In this respect, it is difficult to

apply defined generic process models in organizations without having previously adapted them to the specific situations in which they are going to be developed [Yoo 01].

With the aim of supporting this variety of software processes, software process lines have been defined. These are founded upon the creation of a set of processes which all have a common structure, but which are simultaneously differentiated by certain characteristics [Rom 05]. By means of determining a set of variants, each process can be customized to the context in which it will be performed. However, to be sure each variation point is occupied for the most suitable variant, it is necessary to characterize the capability of process variants. The values of variants characteristics identify their performance and how well they can customize the process. In this way it is possible select the most suitable variant to occupy each variation point, according to the behaviour the process requires in that point. The characteristics will be used to define dependences between process variants, by binding their characteristics.

In this paper, we present an analysis of the adaptation of the characteristics defined in the first part of the standard ISO/IEC 9126 to determine the software process lines variants characteristics. These characteristics, furthermore, offer the possibility of classification, and carry out searches in variants repositories.

The remainder of the paper is structured as follows: in Section Two we give an introduction to software process lines. In Section Three we present an analysis of the variants' characterization, in Section Four we present how to use them by means of OCL. Section Five shows an example, and, finally, Section Six contains our conclusions and future work. The quality characteristics defined to variants are listed in Appendix A.

2 Software Process Lines

Software process lines appear after the implementation of the systematic reuse present in software product lines. The advantages of software product lines, which can be extended to software process lines, are seen in [Cle 02].

Software process lines are formed from a set of very similar processes which are, nevertheless, different in certain aspects which distinguish them and allow the process to be adapted to either context. This approach supposes an evolution towards a more engineering-based approach through which to manage artifacts and processes. There are two types of process lines, depending upon the perspective used to build them: top-down and bottom-up [Rom 05].

In the top-down approach, software process lines are designed from a proved and documented process model which includes the best practices carried out in industry. This approach permits process standardization. By following the bottom-up approach, the software process line is created after an analysis of the commonalities and variations existing in a set of projects. The core process,

which is the part which is common to all the processes constructed by means of the process line, is the base from which the entire process is derived.

The mechanism through which variability is introduced into a software process line is based on variation points and variants. Variation points specify in which parts of the process variation may occur. Each variation point can be occupied by zero or more variants [Sin 04]. Variation points cannot imply a radical variation in the process. Variants are the parts of the process (or its components) that can occupy the variation points, and whose inclusion implies some changes in the original process. Different parts of a process can be seen as variants in a software process line, these are:

Activity: A component of a work developed in the execution of a process during the course of a project.
Resource: Something tangible, such as a template or a software program used to develop an activity to produce a product or resource. Human resources are also considered.
Sub-process: A set of related practices carried out through roles and automated elements, that produce output products by using resources and incoming products, and which contribute to satisfying the client's demands.
Role: A definition of the behaviour and responsibilities of an individual or a set of individuals who work together as a team, within the context of an organization.
Task: The smallest work unit subject to the responsibility of the person responsible for the project, which may include one or more actions. A task is a well-defined work assignment for one or more roles.
Work product/artifact: Any element that is generated or consumed by a process.

Dependences may appear between variants and variation points. These are constraints in the selection of variants for one or more variation points. Dependences can represent attributes such as quality [Sin 04] [Sin 07], and may be of inclusive and exclusive types.

An inclusive dependence from one variant to another specifies that to be able to occupy a variation point with the first variant, it is necessary for the second variant to occupy another variation point. For example, all the variants that a task contains will have an inclusive dependence towards the variants contained in the roles that carry them out.

An exclusive dependence specifies that for a variant to be able to occupy a variation point, the other variant must not occupy any variation point.

2.1 Process Generation from Software Process Lines

By following the top-down approach, a software process line is created by taking a capable, proved and documented process as the core process of the line. In this case, it must guarantee that the variants that occupy the variation points, and which create each new process of the line, achieve the needs of the process, in order to maintain consistency and to facilitate certification. Furthermore, the process designer knows what the needs of the process are, and he knows the software process line, its variation points and their variants. So he must get to know what variant can satisfy the process needs better than others. And when a variant is inserted in a variation point to create a process, the changes it entails for that process and the other variants are important, too.

To satisfy both points it is necessary to determine the characteristics of the variants in order to estimate their performance when they are part of a process. And to help to decide whether or not they fit the needs of the process in which they are participating. But nowadays there is no defined set of characteristics that can be applied to the variants of a software process line. Furthermore, since variants of a process line may be of several types – as many as different parts in a process can vary –, it is also necessary to define a set of characteristics designed to each of these types.

3 Variant characterization

Variants are the parts of the final process that allow to adapt to the context in which the process will be performed. They must be selected with the objective that they fit the process as well as possible. In Fig. 1 an illustrative example is shown using geometric figures.

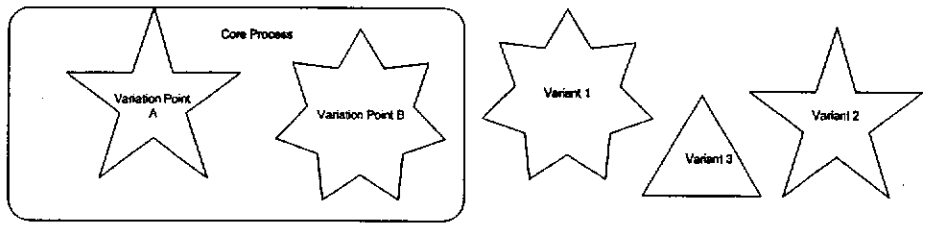


Fig. 1 *Representation of the fit between variants and variation points in a process line.*

As Figure 1 shows, variants have certain characteristics, which are represented by the shape of the figures. In this figure, we can see that the core process contains some variation points that can only be occupied by the variants with certain characteristics, and in Figure 1, this is represented by using the shape of each empty space.

In order to carry out the characterization of the variants in a software process line, we shall bear in mind the characteristics proposed in ISO/IEC 9126 [Iso 01]. Although these characteristics have been defined for software products, they can be applied to software process line variants, by taking into account the fact that they contain certain parts of the process model. Furthermore ISO/IEC 9126 characteristics are designed to check if software products satisfy customer and user needs. In this way an adaptation of these characteristics can be used to guarantee the variants that configure each process of a software process line are the most suitable to adapt it to the organization and project needs.

On the other hand, variants can be reused to create several processes of the same software process line. It implies that process lines will be greater and richer in variants. However, this also implies the necessity of storing these variants. In order to store variants in an ordered manner and to localize them efficiently, a repository of variants can be used, as is proposed in [Lu 07]. Thus, when it is necessary for a variant to occupy a variation point in the process line, it will be easily localized in the repository by means of the evaluation of the characteristics of the repository elements with regard to the needs of the variation point.

3.1 Application of the ISO 9126 characteristics to software process lines variants

We have used the first two steps of the analysis methodology proposed by Franch y Carvallo [Fra 03].This methodology uses six steps to build software quality models:

- **Step 0:** Defining the domain.
- **Step 1:** Determining quality sub-characteristic.
- **Step 2:** Defining hierarchy of sub-characteristics.
- **Step 3:** Decomposing sub-characteristics into attributes.
- **Step 4:** Decomposing derived attributes into basic ones.
- **Step 5:** Stating relationships between quality entities.
- **Step 6:** Determining metrics for attributes.

3.2 Methodology application

The Step 0 of the methodology has been undertaken and is summarized in the section on software process lines.

To carry out Step 1 of the methodology, to determine the applicable (sub)characteristics, we have taken those proposed by the ISO standard, and have evaluated whether or not they are applicable to each of the process model elements, which are the variants of the process line. We have applied the sub-characteristics to the six types of variants enumerated above (activity, resource, sub-process, role, task and work product). In this step some ISO/IEC 9126 char-

acteristics have been used by introducing little changes into their definition. In other cases, definitions have been modified to adapt it to variant needs. Other ISO/IEC characteristics have been deleted because they are not applicable to variants, and finally, some new characteristics have been added.

In Appendix A the quality characteristics determined to variants are shown. They are ordered according to their type in Tables 1 to 6 of Appendix A.

3.3 Related Variants

The previously defined characteristics refer to those of the variants according to their nature and utilization within the process. However, it is necessary to take into account the fact that the adaptation of a process does not involve separate variants, but that the final process will include a set of variants with dependences between them.

All the variants that have any inclusion dependence with others force these others to also occupy some variation point of the process line. Therefore the set of variants that are related with another set will be a new characteristic of the process line.

4 Application of characteristics

Starting from the characteristics determined for variants, OCL [Omg 04] can be used as a formal language with which to specify the constraints that bind these characteristics with the needs of the process that is generated by the software process line. Variants and variation points have been modelled as a UML Class Diagram (Fig. 2). In Figure 2 the metamodel for the definition of variants and variation points is shown according to the proposal in [Mar 08].

In the metamodel of Fig. 2 the variants are related with the variation point they can occupy. Then it is possible define OCL constraints inside this model, by considering element characteristics as attributes of the classes shown in the metamodel. These constraints between variation and variation points must model three aspects:

1. Variation point and variant must be specifications of the same process element. It can be done by means of the operation

   ```
   variant.oclIsTypeOf(type:oclType)
   ```

2. The dependences between variants must be satisfied in order to use variations correctly.
3. Behaviour of the variant may be adjusted to variation point behaviour needed by using constraints. In this way, constraints guarantee the variant that occupies a variation point is the most suitable to generate the ideal process for each project.

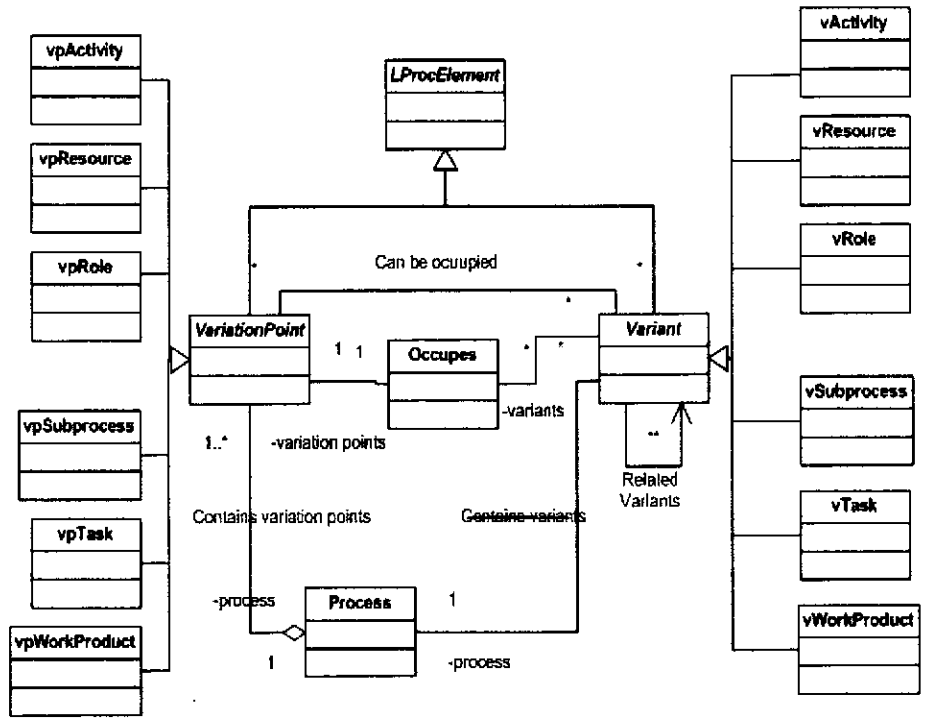


Fig. 2 Metamodel of software process line: variants and variation points

5 Example

To illustrate the use of the variants, an example is shown. The context of this example is the COMPETISOFT project. Because of space limitation, only the functionality sub characteristics are described. COMPETISOFT has the objective of defining a Software Process Improvement Framework orientated towards very small organizations, into which certain iberoamerican proposals are integrated. More than a hundred researchers are taking part in this project.

The process model included in the Methodological Framework is made up of ten processes, which are grouped into three categories (Fig. 3). The first is Top Management which provides the alignments which allow the organization to work. The second category, Management, includes the processes, projects and resources management. The Operations category deals with the management of development projects and software maintenance.

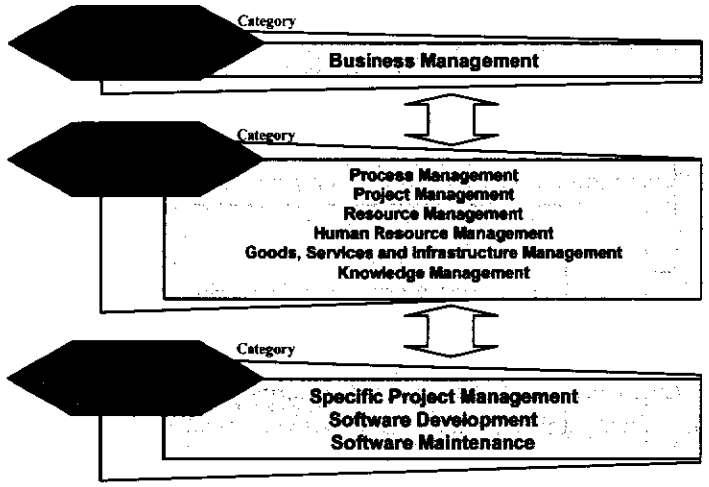


Fig. 3 *Categories and process in the COMPETISOFT process model*

An example of the application of functionality characteristics when some elements are added in a process is shown. By centring upon the COMPETISOFT process pattern, the definition of all the roles includes a section given over to abilities. When a variant of type role is evaluated in order to carry out a task or activity, certain things must be taken into account. In the first place, the *suitability* of its formation with regard to the activities that it is going to carry out must be evaluated, that is, whether it is able to carry out this task; the *accuracy* is measured to check that the abilities of the role do not exceed the needs of the task, so as not to waste resources. It is necessary to check whether it is able to work with tools and other roles (*interoperability*) and whether the capabilities of the role include the knowledge necessary to handle the tools that are used in the task (*standards agreement*).

When a new task is included in an activity as a variant, its *suitability* must be evaluated, that is, whether the work that is going to be developed in the task is consistent with the work being developed in the activity, and whether the results of the task are in accordance with those produced by the activity, that is, its *accuracy*. It is also necessary to evaluate whether the task has the *security* mechanisms to protect the information. Finally, it would be interesting to check whether the new task has a structure which can be adjusted to the activity's process model (*model compliance*) and whether it can be restructured to prevent failures (*flexibility*).

When a work product is included as a variant it is necessary to evaluate whether its content coincides with the activity in which it is developed (*suitability*), and whether it is sufficient (*accuracy*). One must also evaluate whether the work product can be elaborated by using the tools used by the activity (*interoperability*) and whether it is adjusted to the format standards defined in the organization (*format agreement*). Lastly, it is necessary to check whether it contains the mechanisms to guarantee the *security* of its contents.

In the COMPETISOFT model all the work products at the manager level are verified and validated. Therefore, all the variants included in a work product must have an *inclusion dependence*. This means that if the variations occupy a variation point within a process, then this point must also include two variation points which are occupied by a pair of variants that include the verification and validation activities of said work product. This means that the variants corresponding with the latter two activities form a part of the set of variants related to the variants of each product.

6 Conclusions and future work

In this work we have analyzed the adaptation of the ISO/IEC 9126 standard characteristics in order to reflect the characteristics of the elements that vary in a process line, although these are not always software products. In the adaptation process carried out, some of the ISO 9126 characteristics have been directly applicable, with a few modifications, others have been modified to a greater extent, and we have considered that certain process elements, owing to their nature, must not have determinate characteristics. Finally we have considered the dependence relationships between the variants.

With these characteristics, the aptitude of a variant with regard to the process in which it will take part can be evaluated in order to guarantee the process generated from a software process line is the most suitable for their context. With these characteristics, and by using OCL, constraints upon the process line can be specified which guarantee that all the processes derived from the process line will be fit. Furthermore, we have shown that these characteristics can be used to index the variants within a repository.

Future work will consist of integrating these characteristics into the mechanism that supports the management of the variability in a process line, and finding all the relationships between variant characteristics. We also wish to implement a tool that will support the creation of software process lines and the derivation of processes from them. These mechanisms will be included in this tool, and the related characteristics will be taken into account in order to manage the variants within the repository.

In this work, we have considered the standard elements of a process. However, in its last SPEM proposal, OMG has included more process elements, which are potential parts of a variant. The characteristics of these elements will also be analyzed. Furthermore, other factors such as productivity, cost and availability of a given variant will be analyzed to complement the quality factors for the evaluation of the process in which the variant takes part.

Acknowledgments

This work is partially supported by the investigation into software process lines sponsored by Sistemas Técnicos de Loterías del Estado S.A. within the framework of the agreement upon the Innovación del Entorno Metodológico de Desarrollo y Mantenimiento de Software, and by the projects ESFINGE TIN2006-15175-C05-05 financed by the Spanish Ministry of Science and Technology and INGENIO financed by the Junta de Comunidades de Castilla-La Mancha, Consejería de Educación y Ciencia, PAC08-0154-9262.

References

[Cle 02] P. Clements, L. Northrop: Software Product Lines. Practices and Patterns. Boston, Addison-Wesley. 2002

[Fra 03] X. Franch, J. P. Carvallo: Using Quality Models in Software Package Selection. IEEE Computer Society 20(1): 34-41. 2003

[Fug 00] A. Fuggetta. Software Process: A roadmap. International Conference on Software Engineering (ICSE), ACM Press. 2000

[Hum 89] W. Humphrey: Managing the Software Process, Addison-Wesley. 1989

[Iso 01] ISO: ISO/IEC 9126-1: Software Engineering-Software Product Quality-part 1: Quality Model. Geneva, Switzerland, International Organization for Standardization. 2001

[Lu 07] R. Lu, S. Sadiq. A Reference Architecture for Managing Business Process Variants. Proc. 9th International Conference on Enterprise Information Systems (ICEIS2007), Funchal-Madeira, Portugal. 2007

[Mar 08] T. Martínez-Ruiz, F. García, M. Piattini: Towards A SPEM 2.0 Extension to Define Appropriate process lines Variability Mechanisms. Sent to 6th International Conference on Software Engineering Research, Management and Applications. 2008

[Okt 07] H. Oktaba, F. Garcia, M. Piattini, F. Pino, C. Alquicira, F. Ruiz: Software Process Improvement in Small Latin-American Organizations: COMPETISOFT Project. IEEE Computer 40(10): pp. 21-28. 2007

[Omg 04] OMG: The Object Constraint Language Specification- v. 2.0, Object Management Group. 2004

[Pia 06] M. Piattini, F. Garcia, I. Caballero: Calidad de Sistemas Informáticos. Madrid, Ra-Ma. 2006

[Rom 05] D. Rombach. Integrated Software Process and Product Lines. SPW 2005, LNCS 3840, Teddington, UK, Springer-Verlag. 2005

[Sin 07] M. Sinnema, S. Deelstra: Industrial Validation of Covamof. Journal of Systems and Software **49**(1): 717-739. 2007

[Sin 04] M. Sinnema, S. Deelstra, J. Nijhuis, J. Bosch: Covamof: A Framework for Modelling Variability in Software Product Families. SPLC 2004 (Springer Verlag): 197-213. 2004

[Yoo 01] I.-C. Yoon, S.-Y. Min, D.-H. Bae: Tailoring and Verifying Software Process. Proceedings of the eighth Asia-Pacific Software Engineering Conference (APSEC.01): 202-209. 2001

Appendix A: Quality characteristics of variants[1]

Functionality: Capability of the activity to provide the functionality the process needs.
Suitability: The capability of the activity to carry out the necessary work in the process, within a determinate organization and project. Accuracy: The capability of the activity to provide the correct results. Security: The capability of the activity to offer the mechanisms that protect the data and information that it handles so unauthorized access can't take place. Flexibility: The capability of the activity to be restructured in order to support little changes in the context and correct failure effects. Model Agreement: The capability of the activity to adhere to an activity detailed in a process model, especially if this model is used by the core process of the process line.
Reliability: Capability of the activity to maintain its performance if the project plan fails.
Maturity: The capability of the activity to avoid failure when something faults in the project. Fault tolerance: The capability of the activity to be developed in cases something faults. Recoverability: The capability of the activity to reach the plan project after faults have occurred.
Usability: Capability of the activity to be used to create several processes.
Understandability: The capability of the activity to enable engineers to understand its application. Learnability: The capability of the activity to enable engineers to learn its application. Operability: The capability of the activity to be operated and controlled by the project manager.. Attractiveness: The capability of the activity to be attractive to the engineer.
Efficiency: Capability of the activity to provide the appropriate performance with the resources of the organization.
Time Behaviour: The capability of the activity to be realized in the time appropriate for the project and avoiding delays. Resource utilization: The capability of the activity to use the appropriate amounts and types of resources that are available for its execution.
Maintainability: Capability of the activity to be used as a base to create other variants.
Analysability: The capability of the activity to be analysed, in order to provide its reuse creating other activity variants. Changeability: The capability of the activity structure to be changed, in order to create a new variant.
Portability: Capability of the activity to be used in different environments.
Installability: The capability of the activity to be easily included into a process. Co-existence: The capability of the activity to co-exist with other variants, especially in multiple type variation points.

Tab. 1 *Quality characteristics of variants of type activity*

1 Note that all these characteristics include a *Compliance* sub-characteristic, which indicates the capability of the *element* to adhere related standards, conventions, style guides or regulations. These have not been included in the tables for space reasons.

Functionality: Capability of the resource to provide the capability needed into the project.
Suitability: The capability of the resource included in the variant, which indicates whether it provides the set of functionalities with which to perform the tasks and achieve the process goals. Accuracy: The capability of the resource included in the variant to provide the right results or effects with the correct precision. Interoperability: The capability of the resource included in the variant to interact with other resources. Security: The capability of the resource included in the variant to offer mechanisms that protect the data and information it handles so unauthorized access can't take place. Standards Agreement: The capability of the resource included in the variant to support standards, in the case of tools, or to know standards, in the case of members of the organization.
Reliability: Capability of the resource to maintain a specific level of performance under specific conditions.
Maturity: The capability of the resource to do its work correctly, with no faults. Fault tolerance: The capability of the resource to maintain its level of performance in case of project faults. Recoverability: The capability of the resource to re-establish a level of performance after a fault occurs.
Usability: Capability of the resource to be understood, learned and used [applicable to tools, not engineers].
Understandability: The capability of the resource to enable the engineers to be understood, and how it can be used. Learnability: The capability of the resource to enable engineers to learn its utilization. Operability: The capability of the resource to be operated and controlled by engineers and project manager. Attractiveness: The capability of the resource to be attractive to the engineers.
Efficiency: Capability of the resource to provide appropriate performance, relative to the amount of resources available.
Time Behaviour: The capability of the resource to provide an appropriate response consistent with the project plan. Resource utilization: The capability of the resource to use the set of resources available for the project.
Maintainability: Capability of the resource to be used as a base to create new variants.
Analysability: The capability of the resource to be analysed in order to provide its reuse to create other variants. Changeability: The capability of the resource to be modified in order to create a new variant.
Portability: Capability of the resource to be used in different environments.
Installability: The capability of the resource to be included into a process. Co-existence: The capability of the resource to co-exist with other variants, especially in multiple type variation points.

Tab. 2 *Quality characteristics of variants of type resource*

Functionality: Capability of the role to provide the functionality the process needs.
Suitability: The capabilities of the role included in the variant to carry out the appropriate set of tasks that permit the process goals to be achieved. Accuracy: The capability of the role included in the variant to provide the specific and correct results with the correct precision. Interoperability: The capability of the role included in the variant to interact with any other roles and tools. In the case of interacting with tools, this refers to the knowledge which is required to manage them. Standards Agreement: We believe that roles must adhere to standards, in this case, the organization members who must do it.
Reliability: Capability of the role to maintain a specific level of performance.
Maturity: The capability of the role to do its work correctly, with no faults. Fault tolerance: The capability of the role to maintain the level of performance of the project in case the project plan failures. Recoverability: The capability of the role to re-establish a level of performance after a fault occurs.
Usability: Capability of the role to be interpreted by the human resource.
Understandability: The capability of the role to enable engineers to understand it. Learnability: The capability of the role to enable the engineer to easily learn while it is been developed. Operability: The capability of the role to enable the engineer to use it. Attractiveness: The capability of the role to be attractive to the engineer.
Efficiency: Capability of the role to provide appropriate performance, relative with the amount of resources available.
Time Behaviour: The capability of the role to provide appropriate response in a consistent way with the project plan. Resource utilization: The capability of the role to use a set of resources available for the project.
Maintainability: Capability of the role to be used as a base to create new variants.
Analysability: The capability of a role to be analysed, in order to provide its reuse to create other variants. Changeability: The capability of a role to be modified in order to create a new variant.
Portability: Capability of the role to be used in different environments.
Installability: The capability of the role to be included into a process. Co-existence: The capability of the role to co-exist with other variants, especially in multiple type variation points.

Tab. 3 *Quality characteristics of variants of type role*

Functionality: Capability of the sub-process to provide the functionality the process needs.
Suitability: The capability of the sub-process to carry out the required work within a determinate organization and project. Accuracy: The capability of the sub-process to produce the expected results. Interoperability: The capability of the sub-process to interrelate with the process in which it is included and with the other processes that might be at the same hierarchical level. Security: The capability of the sub-process to offer mechanisms that protect the data and information it handles so unauthorized access can't take place. Flexibility: The capability of the sub-process to be restructured in order to support little changes in the context and correct failure effects. Model Agreement: The capability of the sub-process to adhere to a process model, especially if this model is used by the core process of the software process line.
Reliability: Capability of the sub-process to maintain its performance if project plan fails.
Maturity: The capability of the sub-process to avoid failures when something fails in the project. Fault tolerance: The capability of the sub-process to be executed in cases something fails. Recoverability: The capability of the sub-process to reach the plan project after faults have occurred.
Usability: Capability of the sub-process to be used as a base to create other variants.
Understandability: The capability of the sub-process to enable engineers to understand its application. Learnability: The capability of the sub-process to enable engineers to learn its application. Operability: The capability of the sub-process to be operated and controlled by the project manager. Attractiveness: The capability of the sub-process to be attractive to the engineer.
Efficiency: Capability of the sub-process to provide the appropriate performance with the resources of the organization.
Time Behaviour: The capability of the sub-process to be realized in the time appropriate for the project and avoiding delays. Resource utilization: The capability of the sub-process to use the appropriate amount and types of resources that are available, to be executed.
Maintainability: Capability of the sub-process to be used as a base to create other variants.
Analysability: The capability of the sub-process to be analysed, in order to provide the reuse of the sub-process creating other sub-process variants. Changeability: The capability of the sub-process structure to be changed, in order to create a new variant.
Portability: Capability of the sub-process to be used in different environments.
Installability: The capability of the sub-process to be included into a process. Co-existence: The capability of the sub-process to co-exist with other variants, especially in multiple type variation points.

Tab. 4 *Quality characteristics of variants of type sub-processes*

Functionality: Capability of the task to provide the functionality the process needs.
Suitability: The capability of the task to carry out the necessary work in the activity in which it is included, within a determinate organization and project. Accuracy: The capability of the task to provide the correct results. Security: The capability of the task to offer mechanisms that protect the data and information it handles so unauthorized access can't take place. Flexibility: The capability of the task to be restructured in order to support little changes in the context and correct failure effects. Model Compliance: The capability of the task to adhere to one of the tasks specified in a model process, especially if this model is that which is followed by the core process.
Reliability: Capability of the task to maintain its performance if project plan fails.
Maturity: The capability of the task to avoid failures when a failure occurs in the project. Fault tolerance: The capability of the task to be developed in cases something is failing. Recoverability: The capability of the task to reach the plan project after faults have occurred.
Usability. Capability of the task to be used to create several processes.
Understandability: The capability of the task to enable engineers to understand its application. Learnability: The capability of the task to enable engineers to learn its application. Operability: The capability of the task to be operated and controlled by the project manager. Attractiveness: The capability of the task to be attractive to the engineer.
Efficiency: Capability of the task to provide the appropriate performance with the resources of the organization.
Time Behaviour: The capability of the task to be realized in the time appropriate for the project and avoiding delays. Resource utilization: The capability of the task to use the appropriate amounts and types of resources that are available for its execution.
Maintainability: Capability of the task to be used as a base to create other variants.
Analysability: The capability of the task to be analysed, in order to provide its reuse creating other task variants. Changeability: The capability of the task structure to be changed, in order to create new variants.
Portability: Capability of the task to be used in different environments.
Installability: The capability of the task to be easily included in a process. Co-existence: The capability of the task to co-exist with other variants, especially in multiple variation points.

Tab. 5 *Quality characteristics of variants of type task*

Functionality: Capability of the work product to include the data and the format needed.
Suitability: The capability of the work product, which indicates whether it contains the elements/information necessary to develop the activities for which this product was configured as an incoming product, within a determinate organization and project. Accuracy: The capability of the work product which indicates whether it contains the expected results. Interoperability: The capability of the work product to be produced and/or processed by automatic tools. In the case of documentation, the characteristic refers to whether it is written in a format that can be processed by people and machines. Security: The capability of the work product to offer mechanisms that protect the data and information it handles so unauthorized access can't take place. Format Agreement: The capability of the work product to adhere to the format and content rules defined within the organization.
Reliability: Capability of the work product to maintain the level of performance.
Maturity: The capability of the work product to avoid failures. Recoverability: The capability of the work product to be re-established after a failure in the organization.
Usability: Capability of the work product to be understood and easily used.
Understandability: The capability of the work product to be understood, and how it can be used in the project. Learnability: The capability of the work product to enable engineers to understand its content. Operability: The capability of the work product to be produced or used. Attractiveness: The capability of the work product to be attractive.
Efficiency: Capability of the work product to be produced/used within the project.
Time Behaviour: The capability of the work product to be produced or used inside the activity or task planned. Resource utilization: The capability of the work product to use resources available for the activity/task in which it is produced or used.
Maintainability: Capability of the work product to be used a base to create new variants.
Analysability: The capability of the work product to be analysed, in order to provide its reuse to create new variants. Changeability: The capability of the work product to be modified in order to create new variants.
Portability: Capability of the work product to be used in different environments.
Installability: The capability of the work product to be included into a process. Co-existence: The capability of the work product to co-exist with other variants, especially in multiple type variation points.

Tab. 6 *Quality characteristics of variants of type work product*

Quality Plans for Measuring Testability of Models

H. Voigt · B. Güldali

Software Quality Lab, University of Paderborn, Germany

G. Engels

sd&m AG, software & management, Germany ·
Software Quality Lab, University of Paderborn, Germany

Abstract

For models used in model-based testing, the evaluation of their testability is an important issue. Existing approaches lack some relevant aspects for a systematic and comprehensive evaluation. Either they do (1) not consider the context of software models, (2) not offer a systematic process for selecting and developing right measurements, (3) not define a consistent and common quality understanding, or (4) not distinct between objective and subjective measurements.

We present a novel quality management approach for the evaluation of software models in general that considers all these aspects in an integrated way. Our approach is based on a combination of the Goal Question Metric (GQM) and quality models. We demonstrate our approach by systematically developing a short quality plan for measuring the testability of software models.

1 Introduction

Model-based testing (MBT) advocates the systematic use of software models in particu-lar for testing activities, e.g. generation of test cases, test oracles, and test execution environments [HL03]. If a software model is well suited for testing activities, we speak of a *testable* model. Thus, *testability* of software models is an important quality indicating the degree to which a software model facilitates test-