DEFINING AND USING A METAMODEL FOR DOCUMENT-CENTRIC DEVELOPMENT METHODOLOGIES

Manuel Bollain and Juan Garbajosa
System and Software Technology Group, Technical University of Madrid (UPM), E.U. Informatica
Km. 7 Cra. Velencia, E-28031 Madrid, Spain

Keywords: Document centric software engineering, document-driven software engineering, product and process modelling.

Abstract: The concept of software product is often associated to software code; process documents are, therefore, considered as by-products. Also very frequently, customers primarily demand ”results” and, in a second place, documentation. Development efforts are then focused on code production at the expense of documents quality and corresponding verification activities. As discussed within this paper, one of the root problems for this is that documentation in the context of methodologies is often described with not enough level of detail. This paper presents a metamodel that faces this problem. It is an extension of ISO/IEC 24744, the metamodel for methodologies development. Under this extension, documents can become the drivers of the methodology activities. Documents will be the artifact in which method engineers should focus for methodology development, by defining its structure and constraints. Developers will put their effort in filling sections of the documents as the way to progress in the process execution. This process execution will be guided by those documents defined by the method engineers. This approach can be, as well, the basis for a new approach to a Document-Centric Software Engineering Environment.

1 INTRODUCTION

In software production, work products are both programs and documents. Methodologies describe the activities and tasks for producing such documents and programs but in many cases, organizations have problems for following the defined process (if any). Time schedule delays force developers to skip over document production, and related activities, trying to release a ”operational” product in time. This is a common situation that pushes companies to adhere to process maturity models, and attain a maturity level. Maturity models maybe a baseline to guide the efforts to cope with the documentation issue, but not a complete solution, even in the case of defining a specific maturity model for the documentation process as in (Visconti and Cook, 1993), more in scenarios with strong time constraints or high pressure. Technicians like to solve technical problems, and associated documentation is often considered as the ”boring” part of the work. For all these reasons, documentation quality is often neglected. We can also consider applying agile methods and therefore to reduce the effort in documentation production, and maintaining some control over documentation quality and completeness, but the ongoing debate still remains the same: ”Working software over comprehensive documentation” (Boehm, 2002).

One of the root problems is that the existing predefined processes and methodologies consider documents as one of their *natural outputs*. Very often they simply provide guidelines on how to perform the process considering proper documents as resulting from performing good process practices. The problem with this assumption is that, being true, the granularity level is too coarse. It is necessary to establish, as formally as possible, the existing relationship between documents and the rest of the elements of the software process at a level of detail enough to consider the production of documents not simply as a result of activities but as a part of the process itself.

Within this paper, the concepts of process and

Bollain M. and Garbajosa J. (2007).
DEFINING AND USING A METAMODEL FOR DOCUMENT-CENTRIC DEVELOPMENT METHODOLOGIES.
In *Proceedings of the Second International Conference on Evaluation of Novel Approaches to Software Engineering* , pages 33-40
DOI: 10.5220/0002586400330040

methodology as described in (Gonzalez-Perez and Henderson-Sellers, 2005) will be followed, and applied in the context of ISO 24744:2007 emerging standard (ISO, 2007). Following this standard, *a document is a durable depiction of a fragment of reality*. However it is possible to be more concrete and consider that a document will be composed of sections and subsections. It is possible to describe models in terms of documents, assuming that documents are structured; UML class model is an example when described using OCL. It is just necessary to describe the model in a textual fashion. In our approach, all the activities must correspond with a product, that is, a document; we adopt a strategy in which all the process outputs can be represented as documents, in other words, a document-centric perspective of the software process as in (Luqi Zhang et al., 2004) and (Bollain et al., 2003). According to this, all the activities will produce a "touchable" result in form of a document; this includes source code. The approach to this research work is based on specifying the relationship between documents and the rest of the elements of the software process.

It is common that standards define activities as lists of tasks in which the verb *to document* participates at the beginning or the end of the list. For example, in ISO 12207 (ISO, 1995), we can go to section 5.3.4.1 that reads *The developer shall establish and **document** software requirements, including the quality characteristics specifications, described below [...].* As discussed above, in many cases, the description of an activity includes the act of documenting its execution results. This takes us to a situation in which, in practice, the tools used for assisting in an activity have some sort of utility intended for report or document generation. This report or document will contain the results of the activity, that is what is usually understood as the software product. This makes the activity execution and to document two different issues. If we accept that all software products will be documents, corresponding to previously defined templates, and that activities will be defined in accordance to that templates, the act of document creation will mean performing the corresponding activities or tasks. With this approach we found the following advantages:

- All the project information is within documents, following the template established by the method engineer and there is no need of further work for documenting the performed activities. This will require to define modelling formalisms in terms of documents as outlined in (Bollain et al., 2003). Stakeholders models adopted in software process models such as (Sharp et al., 1999) and (Robinson and Volkov, 1997) can be partially supported as well with this approach. Defining roles for different participants or stakeholders could be achieved using different points of view of each stakeholder for different documents or document sections.
- Documentation preparation is timely. This is ISO/IEC 12207, section 6.4.2.7 compliant. As progress in the development process is through documents fulfillment, it is not necessary verify if documentation preparation is timely.

This paper is structured as follows: this section, 1, is an introduction. Related work is analysed in section 2. The necessary ISO/IEC 24744 standard extensions for achieving a document-centric methodology definition, the outcome classes and types and the new relationships will be explained in section 3. How to make a usage of this extension for defining a document-centric methodology will be developed in section 4, and finally, some conclusions, advantages of the approach, pitfalls and future work will be presented in section 5.

2 RELATED WORK

Document-centric approach has been addressed such as in (Luqi Zhang et al., 2004) and more recently in (Rausch et al., 2005). Reference (Luqi Zhang et al., 2004) introduces a document centric approach and concludes that a document driven approach is feasible and useful; however the approach is different in the sense it is not based on a study on process metamodelling but rather on how documents can be translated from a user oriented form into machine equivalent forms more oriented to be processed and how to get advantage of this. Reference (Rausch et al., 2005) introduces some ideas in line with this work but they are not developed in detail. Reference (Nguyen and Munson, 2003) presents and environment in which documents play an essential role, but the work described is oriented to support web applications. XML documents contain the information of the project, that is, documents and data are the same as described in (Nguyen et al., 2003). This approach makes it possible to support traceability and configuration management on the document itself. In all the cases, the source code is still independent from the project documents.

The use of documents to drive the software process is presented in (Bollain et al., 2003) and (Alarcón et al., 2004); it can be considered as a precedent research work to this one, but the theoretical background is not provided within here; these references

also introduce which are the requirements for documents to support models.

There are several techniques for defining and supporting process. Process definition usually lies on Process Modeling Languages (PML) based on some linguistic paradigms such as (Cass et al., 2000), Petri nets or logical languages; several of them are discussed in (Fuggetta, 2000). Others are component-based software process support (Gary et al., 1998), role-based approaches (Kopka and Wellen, 2002), coordination rules (Ciancarini, 1995); some interesting reflections can be found in (Barnes and Gray, 2000). Even process support could based on tools integration mechanisms, like in (Pohl et al., 1999).

In (Gonzalez-Perez and Henderson-Sellers, 2005), several approaches on process metamodelling are discussed: a well accepted modelling language as UML (OMG, 2001) deal with modelling issues but neglect process, while widespread methodological frameworks such OPEN (Atkinson and Kune, 2000) or Extreme Programming] (Beck, 1999) emphasize the process side and are less detailed when it comes to work product. In the beginning of 2007, ISO/IEC 24744:2007 (ISO, 2007) Metamodel for development methodologies was published; it offers a balance between process and product in the methodology definition issue. This standard has been the basis for defining our metamodel in which process and product will be closely related.

Other related works include document extraction from central repositories like in (Gray et al., 1999), (Henrich, 1996)and (Singh and Han, 1997). Document contents is based on project information, but just as a result of querying the project repository. Also Hypertext linking management in HTML documents has been studied in (Devanbu et al., 1999) and (Oinas-Kukkonen, 1999), where part of the project data, like traceability information, is in the document, but the document info is still the result of querying a repository.

3 A METAMODEL FOR SUPPORTING DOCUMENTS AS *FULL* PRODUCTS

ISO/IEC 24744 standard (ISO, 2007), has a three domain context: the metamodel domain, the methodology domain and the endeavour domain, as shown in figure 1. According to ISO/IEC 24744 an endeavour is an Information-based domain development effort aimed at the delivery of some product or service through the application of a methodology.

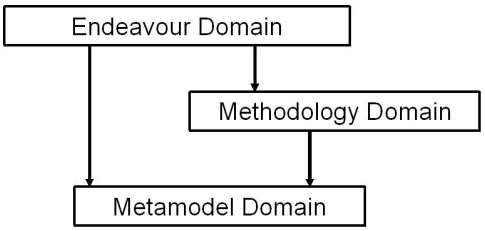

Figure 1: Three domains layers conforming to ISO/IEC. 24744:2007

Metamodels are useful for specifying the concepts, rules and relationships used to define methodologies. A methodology is the specification of the process to follow together with the work products to be used and generated, plus the consideration of the people and tools involved, during an Information-based domain development effort. A methodology specifies the process to be executed, usually as a set of related activities, tasks and/or techniques, together with what work products must be manipulated (created, used or changed) at each moment and by whom, possibly including models, documents and other inputs and outputs. In turn, specifying the models that must be dealt with implies defining the basic building blocks that should be used to construct these models.

Most metamodelling approaches define a metamodel as a model of a modelling language, process or methodology that developers may employ. Following this conventional approach, classes in the metamodel are used by the method engineer to create instances (i.e. objects) in the methodology domain and thus generate a methodology. However, these objects in the methodology domain are often used as classes by developers to create elements in the endeavour domain during methodology enactment. This apparent contradiction is solved by conceiving a metamodel as a model of both the methodology and the endeavour domains. Thus, we find a dual-layer modelling, one layer in the methodology domain and other in the endeavour domain. Modelling a methodology element in both layers should be performed by means of *Clabjets*, that is a dual entity that is a class (for the endeavour layer) and an object (for the methodology layer) at the same time

ISO/IEC 24744 is an open and flexible standard and has been designed to be extended. Therefore the approach to define in detail the existing relation between documents, tasks and other process elements was to extend ISO/IEC 24744 to provide the method engineer with a new tool. ISO/IEC 24744 classes are *immutable*. It is allowed to introduce new attributes and relationships only in new extended classes.

ISO/IEC 24744 contains a relationship between producer, task and document, as shown in figure 2,

but this is not enough for defining a direct relationship between them. ISO/IEC 24744 classes will be extended in order to obtain a direct relationships between producer (role) and product (document). The same type of relationship between document, task, technique and tool is required. The following paragraphs present how classes have been extended.

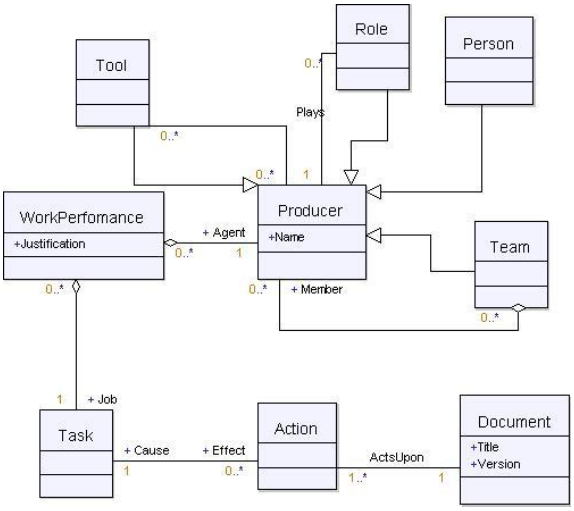

Figure 2: Relationships between producers, tasks and documents in the standard.

Extension of Role class: this is a subclass of Producer. In ISO/IEC 24744 it is possible to define the roles to be played by persons, teams or even tools. There is a indirect relationship between producers and products through Workperformance, Task and Action classes. It is possible to specify which producers have been involved in different products creation, but is not possible state explicitly which participant is involved in the creation of a particular product. In our approach, a direct relationship for setting this participation is needed. At the same time, it is necessary to set the producers role type in relation to the product. For achieving this, a new attribute called Type is created with two possible values: *producer* or *consumer*. The new extended class is called Figure and it is a subclass of the ISO/IEC 24744 class Role. Both extended classes in the methodology domain and in the endeavour domain are shown in figure 3.

Extension of Document Class: As presented in figure 4, Document class and its parent class, WorkProduct are the main classes of our approach. It is important to highlight that a Document could have a parent document. This hierarchy allows a *subdocuments* schema that could be used to support document sections and subsections. This provides the metamodel with a mechanism for defining the methodology documentation at any level of detail, depending on the required document granularity. The Document

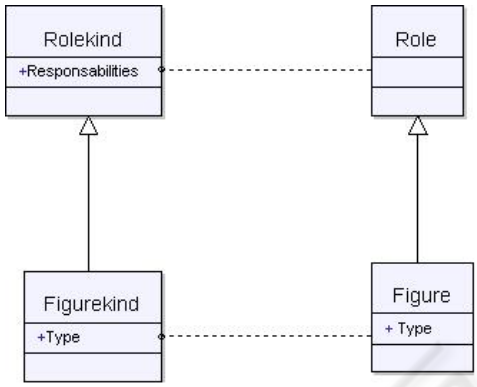

Figure 3: Role Class Extension.

class extension is necessary for setting the following new relationships:

- Relationship with producers, that is, with Figure class, as previously discussed.
- Relationship with tasks, techniques and tools needed for document or subdocument (section) development. In ISO/IEC 24744 there is a direct relationship between tasks and products (documents) through the Action class, but is not possible to state the same type of relationship with the applicable techniques. Although this direct relationship is not necessary in the methodology domain, we consider it essential in the endeavour domain for our approach. The same case occurs with *tools* relationship: there is no direct relationship in ISO/IEC 24744, but we consider it necessary for achieving a well defined document-centric methodology .
- Relationship between documents for setting up constraints among them. In the methodology domain, the different types of documents and the constraints among them should be defined. A document could have constraints that involve other documents that are subdocuments of the first one.

Extension of Constraint Class: The Standard Constraint class is a condition that holds or must hold at certain point in time. Constraints are often used to declaratively characterize the entry and exit conditions of actions. This class applies only in the methodology domain, having no sense in the endeavour level. Initially, it is related with ActionKind class, setting conditions form the execution of its related actions. In our approach, the constraints will be applied on documents: the document production will depend on conditions applied on other documents (or subdocuments). From this, it is necessary an extension of the Constraint class and setting a new relationship.

Extension of TaskTechniqueMapping Class:

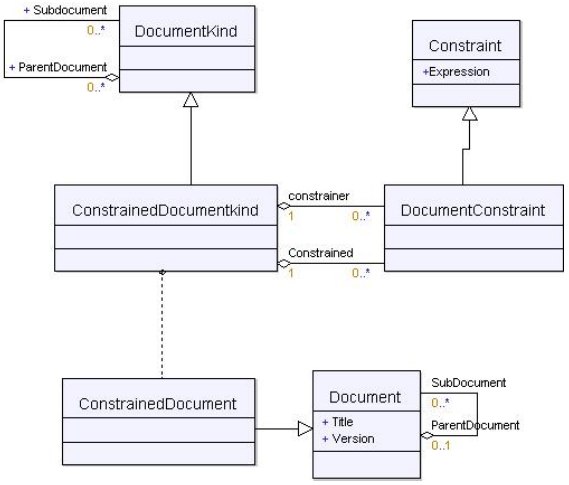

Figure 4: Document Class Extension.

the TaskTechniqueMapping ISO/IEC 24744 class establishes the relationship between tasks and the applicable techniques in both the methodology and the endeavour domains. However, our aim is, as discussed above, providing the method engineer with the possibility of assigning tasks to documents and corresponding techniques with a direct relationship. At the same time, we consider useful the possibility of assigning which tools should be involved in certain tasks execution, following certain techniques, related with these documents. For this, we need to extend the TaskTechniqueMapping ISO/IEC 24744 class in the new DocumentTaskTechniqueToolMapping class, which makes it possible the direct document assignment to the corresponding tasks for completing it, the applicable techniques in each case, and the tools to be used. Figure 5 shows the classes involved in this mapping. In our approach, this mapping is necessary in both methodology and endeavour domains in order to provide support not only for the methodology definition, but the Document-Centric Software Engineer Environments as well.

Extension of Tool Class: ISO/IEC 24744 tool class is a subclass of the Producer class and it is related with other producers that are assisted by it. As Producer subclass, it has the same relationship with products as described for Role class. As previously discussed, in our approach, tools are related in an direct way with tasks and techniques required for developing a document. The relationship between tools and assisted producers is established, in an indirect way, through the document in which they take part. The relationship between Tool class and DocumentTaskTechniqueToolMapping could be achieved by the extension of the Tool class into a new class called AssignedTool.

The metamodel extension, in the methodology do-

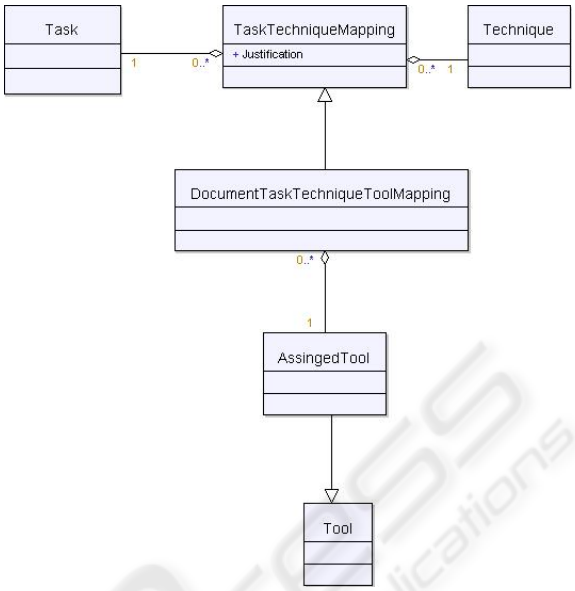

Figure 5: Task, Technique and tool mapping.

main, is shown in figure 6. The ConstrainedDocumentkind is the core of our proposal: Figures are related with documents and will follow the tasks, use the techniques and tools attached to the documents under the corresponding documents constraints.

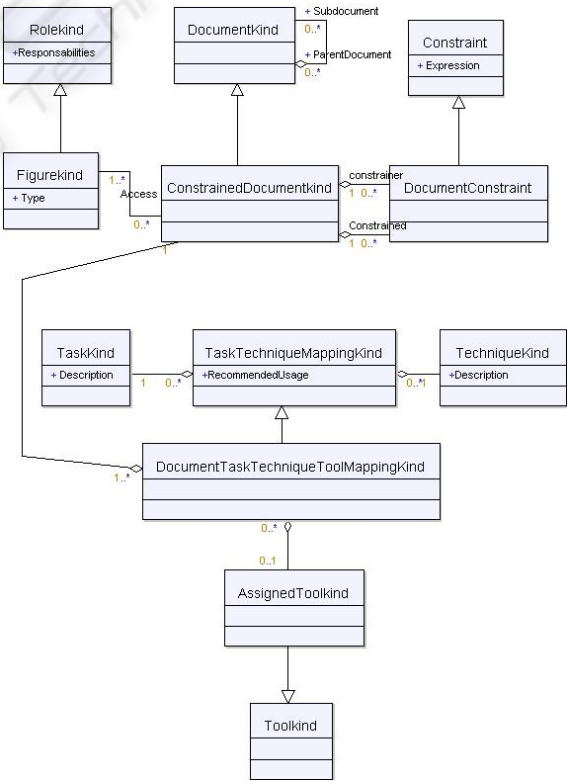

Figure 6: Metamodel extension in the methodology domain.

The metamodel extension in the endeavour domain is shown in figure 7. It is very similar to the corresponding methodology domain point of view. The main difference lies on the lack of constraints in this domain.

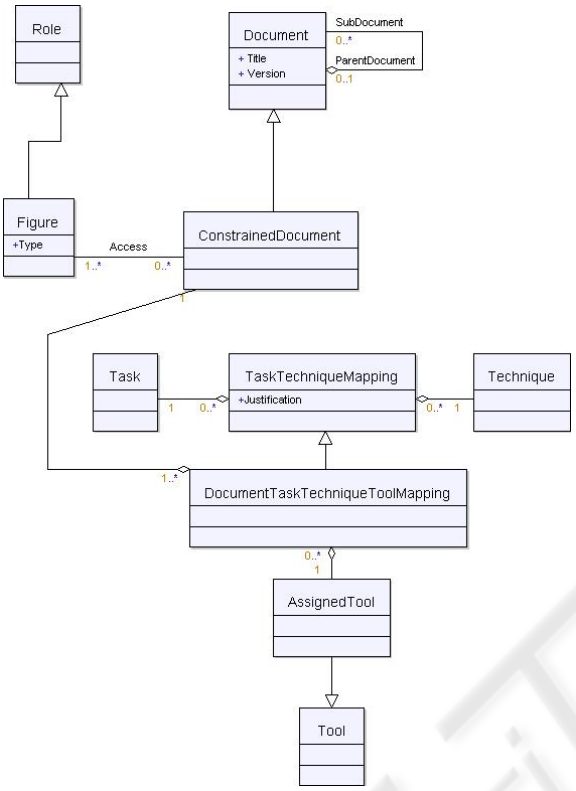

Figure 7: Metamodel extension in the endeavour domain.

4 METAMODEL EXTENSION USAGE

The steps for defining a methodology using this extension are the following:

- Document structure definition. ConstrainedDocumentKind structure in the methodology domain and ConstrainedDocument structure in the endeavour domain. A document kind is composed by a set of subdocuments that could be treated as sections. The detail level of each subdocument depends on the desired detail level in the methodology definition.
- FigureKind definition. For each ConstrainedDocumentKind, FigureKind, an extension of RoleKind class, will be defined. This element could grant access on document kinds to TeamKind or Rolekind elements. At this point, stakeholders will be defined and also the role they will play as document or subdocument producers or consumers. In the endeavour domain, it is possible as well, the definition of concrete persons as Figures.
- Document constraints definition. Using DocumentConstraint class, it is possible the documents (or subdocuments) to define constraints such as creation conditions or documents precedence. For example, a Unit Test Report document could be produced only if a Unit Test Plan document is approved and the first draft of Source Code document is baselined. This constraint could only be defined in the methodology domain. Though working at document level can be considered a coarse granularity, it is necessary to recall that this kind of constraints can equally be defined at any document sub-structure level.
- Document Tasks, techniques and tools assignment. Any document development will involve tasks, techniques and tools that will be carried out by the figures with producer grants on this document. For example, a Unit Test Report document can be assigned to a producer Figure called *Unit Test Team*. This document also have assigned the *Run Unit Test* task, using the *Black Box* technique, with the *Test Case Generator* tool support.

Therefore, tasks precedence, responsibilities distribution and applicable techniques and tool definition are determinated by complete documents definition, and consequently, documents are the drivers of methodology definition and process execution.

5 CONCLUSION AND FUTURE WORK

This paper has presented an approach that is characterised by two issues: first, documents become the key product developers produced, not as by-products and, second, it enables the possibility to define methodologies that use constraints on documents templates to drive software processes. For this, the standard ISO/IEC 24744:2007 Software Engineering – Metamodel for Development Methodologies has been extended in order to provide a modelling baseline to support the approach.

While the approach looks promising on the one side, an identified limitation is related to the need of having detailed document templates. A balance between the approach and the required information for a cost-effective project is required. Another issue is

that software/system models should have to be specified in the form of documents. For instance in the case UML, OCL will facilitate this task obviously.

As future work, it is planned to extend this metamodel in order to support configuration management. This will imply that for any type of document it will be necessary to identify at which level configuration control will be performed. For instance for a software requirement document the configuration item could be each requirement. This will be a methodological decision that should be taken in each case by the method engineer.

Automation may be a key issue to support the approach described within this paper. Therefore it is planned to set up the basis to define a software engineering environment architecture in which documents are the central issue and supports all the concepts introduced within the paper. Documents will be used as the integration mechanism of tools and services using XML schemes to define the document templates. These XML schemes will be the result of applying some mapping rules to the proposed metamodel. A Software Engineering Environment will manage the XML schemes and documents and will trigger the execution of the defined tools for performing the tasks assigned to different documents or subdocuments. A prototype was already developed few years ago as presented in (Alarcón et al., 2004).

ACKNOWLEDGEMENTS

Authors are indebted to Cesar Gonzalez for the support and advice provided while working with ISO/IEC 24744. The work reported herein has been partially supported by Spanish "Ministerio de Educacion, Ciencia y Cultura" within the AGMOD TIC2003-08503 and the OVAL/PM TIC2006-14840, the project VULCANO FIT-340503-2006-3 from "Ministerio de Industria, Turismo y Comercio", and by the DOBERTSEE project, European Space Agency ESA/ESTEC Contract No. 15133/01/NL/ND.

REFERENCES

Alarcón, P. P., Garbajosa, J., Crespo, A., and Magro, B. (2004). Automated integrated support for requirements-area and validation processes related to system development. In *IEEE INDIN*, Los Alamitos, CA, USA. IEEE Computer Society.

Atkinson, C. and Kune, T. (2000). Meta-level independent modelling. In 14th European Conference on Object-Oriented Programming, editor, *International workshop on model engineering*.

Barnes, A. and Gray, J. (2000). Cots, workflow, and software process management: An exploration of software engineering tool development. In *Proceedings of the 2000 Australian Software Engineering Conference*, page 221. IEEE Computer Society.

Beck, K. (1999). *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional.

Boehm, B. W. (2002). Get ready for agile methods, with care. *IEEE Computer*, 35(1):64–69.

Bollain, M., Alarcon, P. P., Garbajosa, J., and Amador, J. (2003). A low-cost document-centric software/system engineering environment. In *Proceedings of the 16th International Conference "Software & Systems Engineering and their Applications" Paris, 2003.*

Cass, A. G., Lerner, B. S., Stanley M. Sutton, J., McCall, E. K., Wise, A., and Osterweil, L. J. (2000). Littlejil/juliette: a process definition language and interpreter. In *ICSE '00: Proceedings of the 22nd international conference on Software engineering*, pages 754–757. ACM Press.

Ciancarini, P. (1995). Modeling the software process using coordination rules. In *WETICE*, pages 46–53.

Devanbu, P., Chen, Y.-F., Gansner, E., Müller, H., and Martin, J. (1999). Chime: customizable hyperlink insertion and maintenance engine for software engineering environments. In *Proceedings of the 21st international conference on Software engineering*, pages 473–482. IEEE Computer Society Press.

Fuggetta, A. (2000). Software process: a roadmap. In *Proceedings of the conference on The future of Software engineering*, pages 25–34. ACM Press.

Gary, K., LindqKuist, T., Koehnemann, H., and Derniame, J. (1998). Component-based software process support. In *13th IEEE International Conference on Automated Software Engineering (ASE'98)*, pages 196 – 199.

Gonzalez-Perez, C. and Henderson-Sellers, B. (2005). Templates and resources in software development methodologies. *Journal of Object Technology, vol. 4, no. 4*.

Gray, J., Scott, L., Liu, A., and Harvey, J. (1999). The first international symposium on constructing software engineering tools (coset '99). In *Proceedings of the 21st international conference on Software engineering*, pages 707–708. IEEE Computer Society Press.

Henrich, A. (1996). Document retrieval facilities for repository-based system development environments. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 101–109. ACM Press.

ISO (1995). *ISO/IEC 12207:1995 Information technology - Software life cycle processes*.

ISO (2007). *ISO/IEC 24744:2007 Software Engineering – Metamodel for Development Methodologies*. International Organization for Standarization.

Kopka, C. and Wellen, U. (2002). Role-based views to approach suitable software process models for the development of multimedia systems. In *ISMSE*, pages 140–147.

Luqi Zhang, L., Berzins, and V. Qiao, Y. (2004). Documentation driven development for complex real-time systems. *Software Engineering, IEEE Transactions on*, pages 936 – 952.

Nguyen, T. N. and Munson, E. V. (2003). The software concordance: a new software document management environment. In *Proceedings of the 21st annual international conference on Documentation*, pages 198–205. ACM Press.

Nguyen, T. N., Munson, E. V., and Boyland, J. T. (2003). Configuration management in a hypermedia-based software development environment. In *Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, pages 194–195. ACM Press.

Oinas-Kukkonen, H. (1999). Flexible case and hypertext. *ACM Comput. Surv.*, 31(4es):7.

OMG (2001). *OMG Unified Modelling Language Specification. Version 1.4*. Object Management Group.

Pohl, K., Weidenhaupt, K., Haumer, P., Jarke, M., and Klamma, R. (1999). PRIME - toward process-integrated modeling environments. *ACM Trans. Softw. Eng. Methodol.*, 8(4):343–410.

Rausch, A., Bartelt, C., Ternité, T., and Kuhrmann, M. (2005). The V-Modell XT Applied Model-Driven and Document-Centric Development. In *3rd World Congress for Software Quality, VOLUME III, Online Supplement*, number 3-9809145-3-4, pages 131 – 138. International Software Quality Institute GmbH. available at http://www.isqi.org/isqi/deu/conf/wcsq/3/proc.php.

Robinson, W. N. and Volkov, V. (1997). A meta-model for restructuring stakeholder requirements. In *ICSE*, pages 140–149.

Sharp, H., Finkelstein, A., and Galal, G. (1999). Stakeholder identification in the requirements engineering process. In *DEXA Workshop*, pages 387–391.

Singh, H. and Han, J. (1997). Increasing concurrency in object-oriented databases for software engineering environments. In *DASFAA*, pages 175–184.

Visconti, M. and Cook, C. R. (1993). Software system documentation process maturity model. In *ACM Conference on Computer Science*, pages 352–357.