Towards Flexible Process Support with a CASE Shell

Pentti Marttiin

e-mail: ptma@jyu.fi

Department of Computer Science and Information Systems

University of Jyväskylä

Finland

Abstract. CASE technology for improving information systems development (ISD) is mostly based on the creation and verification of IS models using a fixed set of techniques. However, ISD is a complex activity, which requires well selected and suited methodologies and development practices for different situations. This calls for CASE shells (metaCASE environments) in which the methodologies can be tailored. Further, the quality of produced deliverables (e.g. specifications and models) is dependent on the development process. The focus of this paper is on integrating a flexible process support into a CASE shell. The ISD process is specified using a graphical process model, the purposes of which are the guidance and coordination of various activities, and the management of the IS deliverables produced during the development. In this paper process modeling requirements are discussed, and the methodology engineering — especially the process modeling — process using a CASE shell is described.

Keywords:

information systems development, methodology engineering, metaCASE, process modeling

1. Introduction

Many organizations invest heavily in CASE (Computer Aided Systems Engineering) technology. One reason for this is continuous problems in the fields of software engineering (SE) and information systems development (ISD)[1], which has been called the software crisis [7, 9]. The promises of CASE are often summarized as improving productivity in development processes and quality in development

[1] Software Engineering is systematic approach to design, implement, maintain and, re-engineering software [18]. It includes engineering a software for IS field. IS development is a process of systems improvement, where a system is transformed from its current state to new improved one [25]. It is less systematic because IS projects are often more user driven and the requirements are less concrete and change during ISD.

products as in [9, 31]. However, new technological innovations themselves can not remove those problems. As pointed out by Jeffery [23] although new technology has potential for improving productivity, without the correct form of management those improvements can not be realized. Also, it is noticed in [34, 43] that the success in the adaptation and use of CASE technology requires different contingencies to be satisfied. These include management processes, learning courses in tools and methodologies, and technical staff for maintenance as examples.

Huff [16] has examined the cost of CASE and noted that it is considerable. Therefore, there should be some return value — productivity and/or quality — for the investments in CASE. Current CASE environments are used mainly for verification of IS specification and documentation [49]. Other benefits are capability to integrate various representation techniques, automate routine tasks such as consistency checking of a diagram and automatically generate new specifications and code. The productivity problems are mostly considered as time delays and cost overruns[2]. The surveys made some years ago [2, 49] show that there is no clear evidence (either theoretical or empirical) that CASE leads to better productivity. Quality of IS/SW models can be described as a sum of various quality aspects including correctness, verifiability, validity, understandability, propriety, reusability, reliability, and robustness. CASE functionality is focused on improving and satisfying only some of these.

Two aspects having potential impacts on an ISD/SE product's validity are discussed. First, every methodology and technique takes a different viewpoint on the problem domain. The selected techniques affect what information is captured during the ISD process. Current CASE environments provide too fixed a variety of techniques. This calls for a metaCASE environment (i.e. a CASE shell) in which we can freely specify techniques and integrate them. CASE shells are described in detail in [29, 44]. Second, the problems of ISD/SE have been said to be due to undisciplined development processes and practices [7, 17]. The SE community has a wide consensus that the quality of products depends on the process through these are produced [1, 17]. Current CASE environments only provide practices to create IS models in normative, encouraging, or free ways [47]. Most of the process supporting tools are integrated to programming environments such as *IStar* [13] and *Marvel* [24]. Only a few tools provide process support for IS analysis and design, such as *HyperCASE* [12] and decision oriented *ConceptBase* [22]. Process supporting tools are discussed more detail in [1, 11].

The goal of this paper is to find a way to integrate a flexible process management environment into the flexible CASE environment. Two basic principles need to be satisfied here. First, the CASE environment should provide a variety of integrated techniques. In this paper we focus on an existing CASE shell — *MetaEdit* [41]. Because it is a graphical tool, the interest is on graphical modeling techniques. Second, the methodology support in CASE environments needs to expanded to facilitate different kind of development and management activities. Therefore, the focus is on flexible modeling of the ISD/SE process.

---

[2] In [43] the productivity effects are further divided to individual productivity to produce specifications, life-cycle effects speeding up single activities and reducing error rates, and down-stream effects that are realized in the long term.

The paper is structured as follows. Section 2 introduces the process modeling requirements. Section 3 describes the MetaEdit+ environment, and its methodology engineering principles. An example of how flexible process support is designed within MetaEdit+ is shown in Section 4. Finally, Section 5 summarizes the results and outlines future directions.

2. Process modeling requirements for flexible CASE environments

CASE environments provide computer support for ISD/SE. ISD can be seen as a composition of various engineering, dialogue, and learning aspects [26], whereas SE is mostly based on engineering. As an engineering process, ISD can be described as complex and ill-structured problem solving activity [42, 45, 25]. It is complex because of its abstract nature and large variety of system components and their relationships [45]. It is ill-structured because design problems contain a great number — sometimes an infinite set — of alternatives and solutions. Further, there is no definite criteria for testing solutions and mechanize process to apply those solutions [37]. As a dialogue process, cooperation among humans plays the fundamental role. Design problems are often called "wicked" problems [35] i.e. design involves compromises between parties with different views and conflicting objectives. As a learning process, ISD is based on the incremental outgrowth of knowledge. Due to new experiences and accumulated knowledge, solutions, views and objectives may change.

The software process is defined by Humphrey [17, pp. 249] to be "the set of activities, method(ologie)s, practices, that are used in the production and evaluation of software". Further, these are fitted to varying organizational and project based practices. A successful CASE environment needs to be powerful enough to manage these diversified needs. Several life cycle strategies, process standards and maturity models, and development methodologies have been introduced to improve ISD/SE.

Life cycle strategies for ISD/SE describe the idealized structure of development activities. These include the waterfall model [4], prototyping [8], spiral models [5, 19], and object-oriented strategies [14, 20]. Earlier waterfall models focus heavily on the engineering aspects and describe a process as a set of sequential development activities such as analysis, design and implementation. The current trend is to describe a process as a complex aggregate of activities including a set of iterative, overlapped, and interlinked activities. Also, alternative approaches based on contracts [13] and decisions [22] instead of activities are introduced.

Software development standards (e.g. *ISO9001* [21]) and maturity models (e.g. SEI's *CMM* [17]) focus on managerial and organizational issues for repeated production. The goal is improve practices by process measurements, monitoring and assessments. CMM maturity levels also take into consideration the learning effects due to the improved SE process. In contrast to the disciplined approaches above, chaos theories indicate that predictive modeling of the ISD process is impossible [3].

There exist a great number of methodologies[3] for improving ISD/SE. The core of methodologies is in the collection of techniques and guidelines to use these techniques based on the underlying life cycle strategy. The proposed process is often illustrated as a list of activities. Although methodologies is said to "standardize development rituals" c.f. [25], they focus on what techniques are used and how these are used rather than how to actually carry out work.

Finding a single best methodology suitable for all development situations seems to be a hopeless task [42]: no one methodology is superior to others if it is compared without taking into consideration the system to be created or changed. The alternative approach is situational methodology engineering. This means that every time the project starts the experience and wisdom about earlier successful and unsuccessful projects are accumulated [25]. Methodologies are contingent upon different development situations, tools available, skill levels of developers and users, complexity of systems to be built, and values of stakeholders.

As noticed above, designing process support for ISD/SE is overall a problematic issue. It needs a process model, which is an abstract description of an actual or proposed process. If we want to build computer support for ISD/SE process at least the following questions arise: what is the purpose of the process model, and what kind of process model would one like to follow.

The purposes of process models is discussed by Curtis et al. [11]. These include facilitating human understanding and communication, improving project and process management, facilitating automatic guidance in performing processes, and supporting automatic execution. In our approach, which is discussed in Section 3, the main purposes of process model are understanding by providing the guidance for the activities, and the management of the evolution and changes of ISD deliverables produced.

Because we demand *flexibility* in process models the following requirements for the ideal process model and the supporting CASE environment can be categorized based on the earlier discussion.

- *Support for life cycle strategies.* ISD process should not be forced to follow only one life cycle strategy. The basic elements (e.g. activities, decisions, or contracts) and the structure (e.g. waterfall or cyclic) of process model needs to be in some degree tailorable.
- *Support for varied methodology processes.* The need of computer support for situation specific methodologies includes techniques as well as processes. One possible strategy for a CASE environment is to provide process models described in methodology handbooks as templates. Projects can then modify them to suit their needs.
- *Management of products evolution.* We can assume that due to the complexity of ISD, user requirements, solution candidates and chosen models may change all the time. According to Baker [3, p. 260] "the CASE environment must be able to store all the alternative branches, [and] provide intuitive navigation mechanisms through alternatives". Tools for handling versions and variants of products, and for navigating between them, are needed in CASE environments.

---

[3] We use the term *methodology* as in CRIS literature [32] to denote e.g. Yourdon's SA [51].

Technically the problem of product management is closely related to problem of version control in the repository.

- *Support for managerial activities.* Methodologies and standards provides variety of managerial metrics. Information about design rationale [10] for clarifying the changes in product evolution sounds tempting. Also, information about project failures and successes is important for laying the foundation for further methodology engineering [25]. We are not always aware of what information we need to gather during the development process.
- *Process unpredictability.* We can not predict all future activities. Further, we do not know the precise *order* of activities (if there is such a feature). The process model structure and activities should be modifiable during the actual process.

3. Overview of the research environment

During the years 1989–93 the SYTI project (and further the MCC company) developed a CASE shell called *MetaEdit*™ . The principles of the tool are reported in [41]. Methodology support in MetaEdit™ means using only one technique at a time. These were specified using the *OPRR* data model (acronym of **o**bject, **p**roperty, **r**elationship and **r**ole) [39, 48]. As a further development for supporting methodologies, we are adding the ability to integrate several techniques [40] and support for ISD process [27]. We call the new design prototype *MetaEdit+*.

The three following features outline MetaEdit+.

- MetaEdit+ (and also MetaEdit™) is based on three levels of abstraction: the *ISD level* is the level where ISD takes place i.e. IS descriptions are developed by a development group; the *ME level (methodology engineering level)* is the level where a ME group specifies methodologies using a MetaEdit+; and the *ISD meta-metalevel,* which contains a set of primitive types (GOPRR, activity types and agent types) which are needed as a language to specify methodologies. The levels are shown in Figure 1.
- The division between products, activities, and agents are discussed in [11, 27]. The specifiable aspects of a methodology are shown in the model level containing the IS models, the ISD process, and the development group. These can be specified using three integrated models: *the meta-datamodel, the activity model* and *the agent model.* These models are further based on GOPRR, *the meta-activity model* and *the meta-agent model.* All models are shown in Figure 1.
- For all parts of the meta-metamodel and methodology specification we have separated a *conceptual* and a *representational* part. The details of this division are described in [41]. The benefit is that mere representational modifications can be done without touching any concepts. Also, one concept can appear in different representations for different techniques of the methodology. An example is a *data flow* concept represented as a line in *DFD*-model and as a cross in certain matrices.

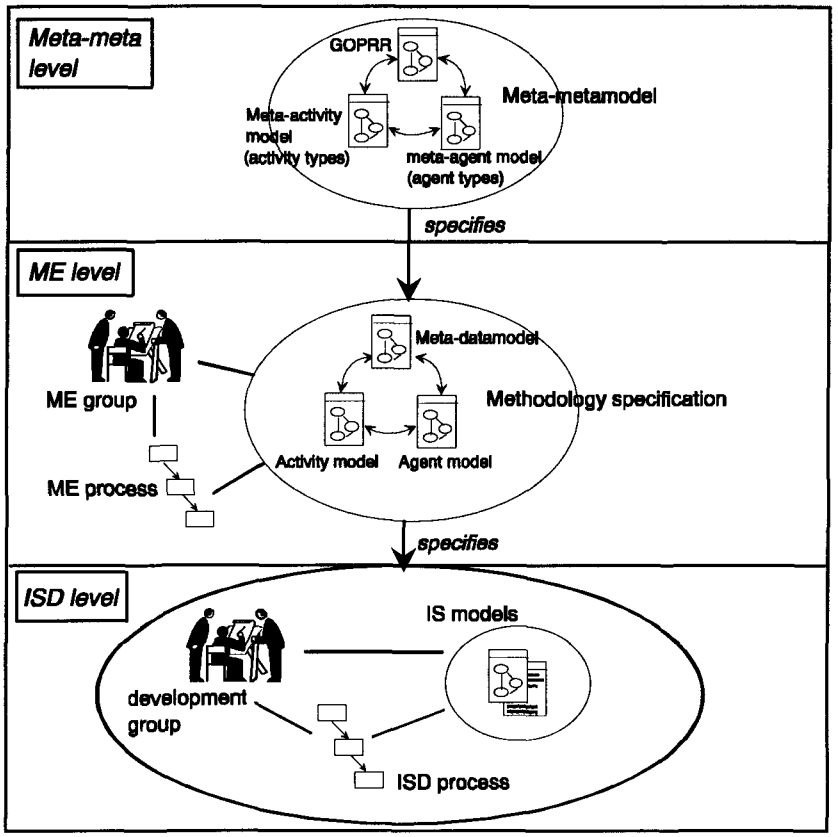

Fig. 1. Three levels of MetaEdit+.

Methodology engineering in MetaEdit+ means creation of a *methodology specification*, which MetaEdit+ uses for specifying ISD techniques, process and group/tool environment. A methodology specification consists of *a meta-datamodel*, *an activity model*, and *an agent model*.

The meta-datamodel specifies techniques and integration of these. It is modelled using the GOPRR types: graph, object, relationship, role and property types. The issues of the meta-datamodel are described in more detail in [40].

An activity model specifies the ISD process. The basic elements of the activity model are activity and deliverable types (Figure 2). The two main purposes of the activity model are:

- to manage different kinds of deliverables (e.g. IS models, specifications, documents), and tools to produce them (e.g. checking tools)
- to provide guidance for the ISD process using on-line helps and pre-defined descriptions.

Agent models define various human agents (e.g. user, project) and user roles (e.g. designer, programmer). These act as electronic notebooks where the information of agent profiles, policies, and strategies is collected. Users get their rights to use the CASE environment through the user roles. Also, technical agents (e.g. checking tool) are defined and linked to techniques through activity types.

Methodology specifications are available as "templates", which are constructed from the information taken from the methodology handbooks. A ME group (Figure 1) selects the techniques, transformations and checkings and links them to the ISD process (activity model template). A similar approach is described in [38] where suitable fragments and route maps based on project characteristics are selected and integrated. Afterwards, an ISD project may want to change or modify the IS techniques and ISD process to better fit its needs[4]. The ME group can make modifications based on the incremental learning and situational changes of the ISD project cf. [25, 38] as follows. When an ISD project has learned to use a methodology it may want to change some parts of it. For example, an ISD project may want to change the life cycle structure of an activity model or improve the techniques. Also, during the life-cycle they may learn how to improve the process by collecting measurement information for specific assessments. Situational changes may happen when the project changes and new user roles are created.

4. Flexible process support in MetaEdit+

Here we focus on one aspect of methodology engineering: process modeling. First, in Section 4.1., we look at the language for creating activity models — the meta-activity model. After that, in Section 4.2., the process of *Yourdon Structured Analysis (SA)* is used as an example of an activity model and potential modifications to it are introduced.

4.1. Meta-activity model

The most covered discussions of process languages and formalisms for SE are represented in [1, 11]. These include systems analysis and design techniques (such as DFD, SADT); data and object modeling (e.g. ER-diagrams, class structures), automata approaches including Petri-nets, AI techniques (rules, pre/post conditions), programming languages, and grammars. We base our work on the object modeling using the concept *activity*, which is the basic concept in most of the life cycle models [1, 4, 14], process modeling approaches [11, 17], and methodology processes [51]. Various types of activities creates a class structure (Figure 2). An activity is any ISD development or managerial task: it uses or produces a deliverable (including checkings and measurements), or acts as a composition of other ISD tasks (e.g. life cycle phases), or a managerial event (e.g. decision or milestone tasks, starting and finishing of phases).

Activities hold a set of user roles for defining the reading and editing rights for the deliverables the activity produces or the rules the activity holds. Activities are managed by starting date. We pay attention only to the starting point so that preceding activities need not be finished before their successors start. In some cases an activity may require a deliverable (i.e. a deliverable is used by an activity) before it can be started.

Basic activity types are a compositional activity *Phase* (Stage), and *Task* (Step) referring to a single task. A *Phase* can contain any number of sub-activities. In

---

[4] MetaEdit 1.0™ contains a method upgrade, which means that techniques can be extended conceptually or changed representationally during the CASE work.

Figure 2 *Tasks* are further specified into *Transformation, Checking, Review, Decision, Milestone,* and other "managerial tasks" *Start* and *Finish*. Most of these types are found in the reference model for ISD [15]. Here the following meanings and extra properties are given. *Transformation* means producing a report, or another deliverable from a deliverable. It includes a transformation model, which contains a set of transformation rules. *Transformation* calls a transformation tool to create and maintain these rules. *Checking* provides correctness and consistency checking for IS specifications by calling the rules attached to GOPRR primitives (e.g. constraint and verification rules in [33]), and/or offers product metrics (e.g. size metrics of the deliverables [36]). Further, it calls a checking tool. *Review* is an adjustment activity directed at deliverables and performed by human agents. *Decision* is used when we have to decide between alternatives and want a solution to be produced. *Milestone* is used when we want to coordinate or (using a technical word) synchronize work, and finish earlier tasks. It can contain a decision to release the "completed" deliverables. Further, every phase can contain informative, managerial *Start*, and *Finish* activities, which act as triggers, and record the starting and finishing of the phase.

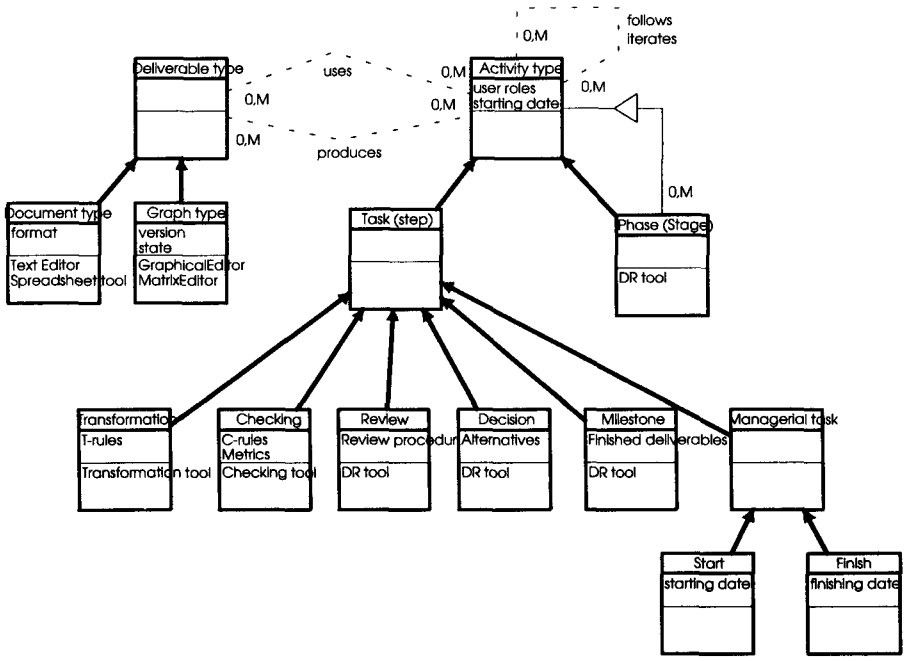

Fig. 2. Basic activity and deliverable types.

The argumentation during the *Review*, *Decision* and *Milestone* can be maintained using the Design rationale tool (DR tool)[5]. *Transformation rules*, *checking rules* and *metrics* are attached to menus of the development tools

[5] The information is structured as *questions* and responding personal *answers* with pro and con *arguments*. The finished discussion results can be taken up to a property of an activity such as *result*, and *goal*.

(Graphical Editor, Matrix Editor). How are these automatically attached to menus? The sphere of influence according to rules and metrics can be set to a phase like *Analysis* and *Design*, which is added as a property to the transformation or the checking tasks. The necessary rules and metrics are available when opening a tool from a deliverable node in the activity model. In the other case, a phase need be selected when opening a development tool. The user roles can affect the use of the rules and metrics.

How are the activities linked together? We place demands on simplicity in the structure of the activity model. Our approach is motivated by the simplicity of *Task structures* [6, 50], which focus on ordering the activities and decisions. Various activities are connected together using the *follows* relationship, which in pre-defined templates can contain conditions and alternative paths to be followed. Activities may also be connected using the *iteration* relationship. This shows the critical path of the changes: if we want to change an deliverable released earlier we must also adjust other related deliverables.

Because our approach is product centered, various deliverables are attached to activities with *uses* and *produces* relationships. Deliverables collect information for initializing the development tools, which can be opened straight from the deliverable node. An initial division is made between graph types and document types. In the activity model, graph type handles *version names* (e.g. initial DFD or checked DFD) and the following *states*: transient (private and locked for owner), working (public, which can be modified by the owner and copied by someone else), or released (the final "frozen" form). A document is a link to any other document, made using e.g. a text editor, in a form like *Checking report* or *Data dictionary format.*

Changes in the meta-activity model can be made by generalizing/specializing activity and deliverable types. These allows the possibility of using different graphical symbols, or collect specific information by attaching properties to various activity types. Examples of the properties might be actual or planned start/finish time, duration, entry/exit criteria, participants, goals, or arguments.

4.2. Activity models

We selected Yourdon's Structured Analysis (SA) [51] as an example methodology. The process of SA is modeled using the meta-activity model more detail in [27]. The main difference to the *Task structure* approach by Verhoef [46] is explicit deliverables in activity models. All the high level activities such as *Analysis* and *Design* are modelled as phases. These act as compositions of other activities, and are used to store the guidance information, i.e. descriptions of the phases (similarly to *HyperSRM tool* [30]), and history of argumentation related to phases.

Figure 3 shows the more detailed description of the task of *Constructing the environmental model.* It contains activities of types task, review, milestone, and transformation. During the task *Construct the event list* one produces a document *Event list*, and in the task *Construct the context diagram* a DFD specification *Initial context diagram*, which is in a working state. The project can select whether it will start with the event list or context diagram. The initial deliverables above (*Event list* and *Initial context diagram*) are used in the reviewing activity *Interrelate event list and context diagram*, which reviews both deliverables and produces the checked

ones. *Release context diagram* is a managerial activity added to Yourdon's example. In this milestone decisions can be made to release the final deliverables. It describes an iteration using two iteration lines to get deliverables completed. As a result, the released *Context diagrams* are collected behind their own specification node. *Produce an initial data dictionary* is a transformation. It implies the use of a transformation model to produce the data dictionary syntax (DD form) described in Yourdon's book. Finally, one can produce an ER model of external stores.

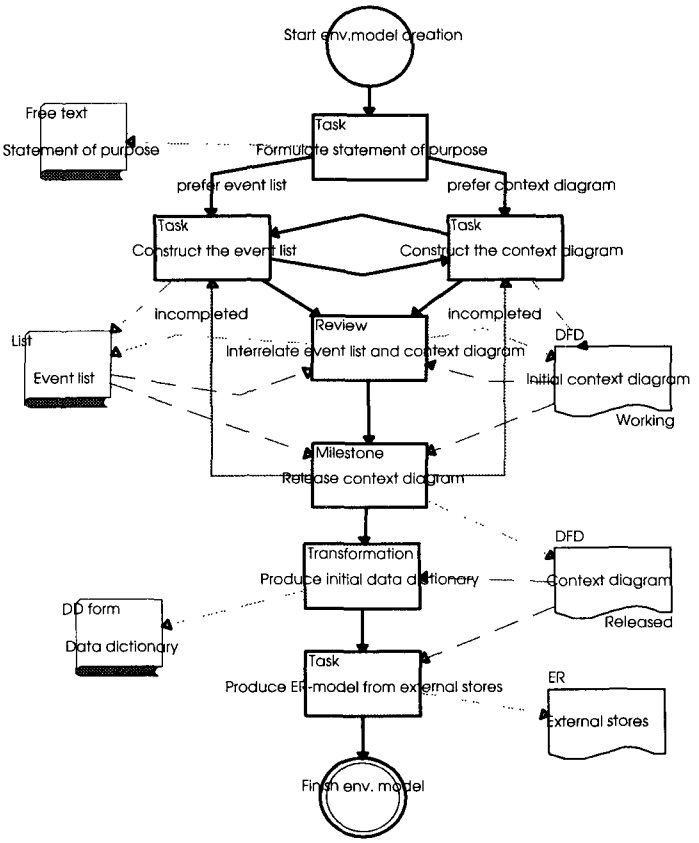

Fig. 3. 'Constructing the environmental model' in Yourdon's SA.

We could continue further to detailed activity diagrams to show how to construct the event list or context diagram. On this level graphical diagrams are powerful tools for guidance. Verhoef [46] describe tasks on the level of *Add Object Type* and *Add Relationship*, and attach operations create_concept(Object) and create_concept (Relationship) to tasks. In MetaEdit GOPRR already handles the semantics of creations, updates and removals. Our aim is to provide better notes and guidelines on how to use the technique: in what order different elements (e.g. object types) can be created, and what information stores (e.g. manuals, reports) can be used to aid methods' use.

SA process is described first as a template, which is the level a metaCASE tool vendor can offer for organizations. The ME group needs to analyze the suitability of templates for the ISD group and change them. The possible changes to the template on the level of *Construction of the Environmental model* can be the following ones. The ME group can first change in the ME level the graphical representations e.g. change the milestone representation to a circle. Also, it can replace the *review* activity with the *checking* activity, giving a second change where the checking contains an automatic checking tool and operations to produce a checking report. Other changes are done in ISD level by the ISD group. First, the ISD group can modify names of the activities. Second, planned starting dates are attached to all activities. Third, the activity *Produce ER-model for external stores*, along with the related ER-diagram, can be moved to later phases. Fourth, the group can replace the *iteration* relationships by a new milestone *Assess context diagram* and a new task *Change event list and context diagram*. So, this project trusts its capability to complete a *Context diagram* using one assessment.

5. Concluding Remarks and Future Directions

This paper describes process modeling support for a product based CASE shell. ISD needs flexibility and extendibility in activities throughout. For the use of various ISD projects the process models should be easy to understand. The graphical metamodeling approach has proved to be suitable for describing ISD techniques. Here we have tested how it can be used in modeling the ISD process.

What are the benefits of an activity model in CASE shell? First, a graphical activity model provides a very expressive tool for process management and improvement. Process management is supported by creating the project-tailored activity model, providing measurement points, and incorporation of tools to a process. As a graphical browsing tool it reduces the maintenance difficulties of various deliverables. Process improvement is facilitated by the reusable templates, and the continuous evolution of a process. Second, the graphical activity model is easy to understand and provides a basis for handling the communication and co-operation aspects of the ISD group. A design rationale tool attached to activities maintains the discussions. Further, the similarity of modeling IS models and ISD processes, as well as modeling techniques and activity types, facilitates the understanding of the tool and its operation on the ME and ISD levels.

The following aspects will be studied in the future. First, the definition of meta-activity models, and the use of activity models have been tested using MetaEdit's capabilities to model techniques. We will implement the designed process support into MetaEdit+. According to this, a version control system to support the versions and states of deliverables will be build into the repository. Second, structuring the design rationale during ISD is one of the ongoing studies. Third, this paper does not address the rule language and mechanism by which the rules are attached to IS models through the activity model. Fourth, possible viewing mechanisms of the activity model filtered by human agents and user roles are not introduced here. These allow developers to use their own subset of the activity model. Fifth, hypertext links empowering the navigation capabilities between deliverables need to be studied.

Acknowledgements

I would like to thank the other members of the MetaPHOR project for fruitful discussions. Special thanks are given to Kalle Lyytinen, Sjaak Brinkkemper, Mauri Leppänen, and Steven Kelly for the improvements of this paper.

References:

1. Armenise, P., Bandinelli, S., Ghezzi, C., Morzenti, A., "A survey and assessment of software process representation formalisms", *International Journal of Software Engineering And Knowledge Engineering*, 3, 3, 1993, pp. 410-426.
2. Aaen, I., Siltanen, A., Sørensen, C., Tahvanainen, V.-P., "A Tale of Two Countries - CASE Experiences and Expectations", *The Impact of Computer Supported Technologies on Information Systems Development* (Eds. K.E. Kendall, K. Lyytinen and J.I. DeGross), Amsterdam, North-Holland, 1992, pp. 61-93.
3. Baker, J.M., "Project Management Utilizing an Advanced CASE Environment", *International Journal of Software Engineering and Knowledge Engineering*, 2, 2, 1992, pp. 251-261.
4. Boehm, B.W., "Software Engineering", *IEEE Transactions on Computers*, 25, 12, 1976, pp. 1226-1241.
5. Boehm, B.W., "A spiral model of software development and enhancement", *IEEE Computer*, 21, 5, 1988, pp. 61-72.
6. Bots, P.W.G., *An environment to Support Problem Solving,* PhD thesis, Delft University of Technology, Delft, The Netherlands, 1989.
7. Brooks, F.P. Jr., *The Mythical Man-Month.* Addison-Wesley, Reading, Mass., 1975.
8. Budde R., Kautz, K., Kuhlenkamp, K., Züllighoven, H., *Prototyping - An approach to Evolutionary Systems Development*, Springer-Verlag, Berlin, 1992.
9. Charette, R.N., *Software Engineering Environments: Concepts and Technology*, McGraw-Hill, New York, 1986.
10. Conclin, J., Begeman, M. L., "gIBIS: A Hypertext Tool for Explanatory Policy Discussion", *ACM Transactions on Office Information Systems*, 6, 4, 1988, pp. 303-331.
11. Curtis, B., Kellner, M.I., Over, J., "Process modeling", *Communications of the ACM*, 35, 9, September 1992, pp. 75-90.
12. Cybulski, J.L., Reed, K., "A Hypertext Based Software Engineering Environment", *IEEE Software*, March 1992, pp 62-68.
13. Dowson, M., "Integrated Project Support with IStar", *IEEE Software*, November 1987, pp. 6-15.
14. Henderson-Sellers, B., Edwards, J.M., "The Object-oriented Systems Life Cycle", *Communications of the ACM,* 33, 9, 1990.
15. Heym, M., Österle, H., "A reference model of information systems development", *The Impact of Computer Supported Technologies on Information Systems Development* (Eds. K.E. Kendall, K. Lyytinen and J.I. DeGross), Amsterdam, North-Holland, 1992, pp. 215-240.
16. Huff, C.C., "Elements of a Realistic CASE Tool Adoption Budget", *Communications of the ACM*, 35, 4, 1992, pp. 45-53.
17. Humphrey, W.S., *Managing the Software Process*, Addison-Wesley, Reading, MA, 1989.

18. IEEE Glossary of Software Engineering Terminology. IEEE Std. 720, IEEE, New York, 1983.
19. Iivari, J., "Hierarchical Spiral Model for Information System and Software Development", *Information and Software Technology*, 32, 6, 1990, pp.386-399.
20. Iivari, J., "Object-oriented Design of Information Systems: The design process", *Object Oriented Approach in Information Systems*, (Eds. F. Van Assche, B. Moulin and C. Rolland), Elsevier Science Publishers, North-Holland, IFIP, 1991, pp. 61-87.
21. Hall, T.J., "The quality manual — The applications of BS5750 ISO9001 EN29001". John Wiley and Sons, Chichester, 1992.
22. Jarke, M., "Strategies for Integrating CASE Environments", *IEEE Software*, March 1992, pp. 54-61.
23. Ross Jeffery, D., "Software Engineering Productivity Models for Management Information Systems Development", *Critical Issues in Information Systems Research* (Eds. Boland R.J. jr. and Hirschheim R. A.), John Wiley and Sons Ltd. 1987, pp. 113-134.
24. Kaiser, G.E., Feiler, P.H., Popovich, S.S., "Intelligence Assistant for Software Development and Maintenance", *IEEE Software*, May 1988, pp.40-49.
25. Kumar, K., Welke, R.J., "Methodology Engineering$_R$: A proposal for Situation-specific Methodology Engineering", *Challenges and Strategies for Research in Systems Development*, (Eds. W.W Cotterman. and J.A. Senn), John Wiley and Sons Ltd., 1992, pp. 257- 269.
26. Lyytinen, K., "Different Perspectives on Information Systems: Problems and Solutions", *ACM Computing Surveys*, 19, 1, March 1987, pp. 5-46.
27. Marttiin, P., "Methodology Engineering in CASE Shells: Design Issues and Current Practice", Licentiate Thesis, Computer Science and Information Systems Reports, Technical Reports TR-4, University of Jyväskylä, 1994.
28. Marttiin, P., Lyytinen, K., Rossi, M., Smolander, K, Tahvanainen, V.-P., Tolvanen J.-P, "Modeling requirements for future CASE: issues and implementation considerations", *Proceedings of the 13th ICIS*, (Eds. J.I. DeGross, J.D. Becker and J.J. Elam), Dallas, USA, 1992, pp. 9-20.
29. Marttiin, P., Rossi, M., Tahvanainen, V.-P., Lyytinen, K., "A comparative review of CASE Shells: a preliminary framework and research outcomes", *Information and Management*, 25, 1993, pp. 11-31.
30. Oinas-Kukkonen, H., "Intermediary hypertext systems in CASE environments", Licentiate thesis, Research papers SERIES A16, Department of Information Processing Science, University of Oulu, 1993.
31. Osterweil, L.J., "Software processes are software too", *Procs. of the 9th International Conference on Software Engineering*, Monterey, California, 1987, pp. 2-13.
32. Olle, T.W., Sol, H.G., Verrijn-Stuart, A.A. (Eds.), *Information Systems Design Methodologies: A comparative review*, North-Holland, 1982.
33. Persson, U., and Wangler, B., "A Specification of Requirements for an Advanced Information Systems Development Tool.", *Procs. of the workshop on the Next Generation of CASE Tools*, (Eds. S. Brinkkemper and G. Wijers), SERC, Netherlands, 1990.
34. Rai, A., Howard, G.S., "An Organizational Context for CASE Innovation", *Information Resources Management Journal*, 6, 3, 1993, pp. 21-35.
35. Rittel, H.W.J., Webber, M.M., "Planning Problems are Wicked Problems", *Policy Sciences*, 4, 1973, 155-169.

36. Rask, R., Laamanen, P., Lyytinen, K., "A comparison of Abrecht's Function Points and Symons' Mark II Metrics", *Proceedings of the 13th ICIS*, (Eds. DeGross J.I., Becker J.D. and Elam J.J.), Dallas, USA, 1992, pp. 207-221.
37. Simon, H., "The Structure of Ill-structured Problems", *Artificial Intelligence*, 4, 1973, pp. 181-200.
38. van Slooten, K., Brinkkemper, B., "A Method Engineering Approach to Information Systems Development", *Information Systems Development Process*, (Eds. N. Prakash, C.Rolland, B. Pernici), Elsevier Science Publishers, North-Holland, 1993, pp. 167-186.
39. Smolander, K., "OPRR - A Model for Methodology Modeling", *Next Generation of CASE Tools*, (Eds. K. Lyytinen and V.-P. Tahvanainen), Studies in Computer and Communication Systems, IOS press, 1992, pp. 224-239.
40. Smolander, K., "GOPRR - a proposal for a meta level model", MetaPHOR internal technical document, Dept. of Computer Science and Information Systems, University of Jyväskylä, 1993.
41. Smolander, K., Lyytinen, K., Tahvanainen, V.-P., Marttiin P., "MetaEdit - A flexible graphical environment for methodology modelling", *Advanced Information Systems Engineering*, (Eds. R. Andersen, J. Bubenko and A. Sølvberg), LNCS #498, Springer-Verlag, 1991, pp. 168-193.
42. Sol, H.G., "A Feature Analysis of Information Systems Design Methodologies: Methodological Considerations", *Information Systems Design Methodologies: A Feature Analysis*, (Eds. T. W. Olle, H. G. Sol and C. J. Tully), Elsevier Science Publishers, North-Holland, Amsterdam, 1983.
43. Sørensen, C., *Introducing CASE Tools into Software Organizations*, Ph.D. Thesis, Dept. of Mathematics and Computer Science, Institute of Electronic Systems, Aalborg University, Denmark, 1993.
44. Sorenson, P. G., Tremblay, J-P., McAllister, A. J., "The Metaview system for many specification environments." *IEEE Software*, 30, 3, 1988, pp. 30-38.
45. Turner, J.A., "Understanding the Elements of System Design", *Critical Issues in Information Systems Research* (Eds. R.J. Boland jr. and R. A. Hirschheim), John Wiley and Sons Ltd. 1987, pp. 97-112.
46. Verhoef, T.F., "Structuring Yourdon's Modern Structured Analysis", *Proceedings of the Second Workshop on The Next Generations of CASE Tools*, (Eds. V.-P. Tahvanainen and K. Lyytinen), Technical Reports TR-1, Jyväskylä, 1991.
47. Vessey, I., Jarvenpaa, S., Tractinsky, N., "Evaluation of Vendor Products: CASE Tools as Methodology Companions", *Communications of the ACM*, 35, 4, 1992, pp. 90-105.
48. Welke, R.J., "The CASE Repository: More Than Another Database Application", Meta Systems Ltd., Ann Arbor, Michigan, 1988.
49. Wijers, G., van Dort, H., "Experiences with the use of CASE tools in the Netherlands", *Advanced Information Systems Engineering*, (Eds. Steinholz, Sølvberg, Bergman), LNCS#436, Springer-Verlag, 1990, pp. 5-20.
50. Wijers, G., *Modeling Support in Information Systems Development*, Ph.D. Thesis, Thesis publishers, Amsterdam, 1991.
51. Yourdon, E., *Modern Structured Analysis*, Yourdon Press, 1989.