ELSEVIER

Information and Software Technology 38 (1996) 275–280

INFORMATION AND SOFTWARE TECHNOLOGY

Method engineering: engineering of information systems development methods and tools

Sjaak Brinkkemper*

Department of Computer Science, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands

Abstract

This paper proposes the term method engineering for the research field of the construction of information systems development methods and tools. Some research issues in method engineering are identified. One major research topic in method engineering is discussed in depth: situational methods, i.e. the configuration of a project approach that is tuned to the project at hand. A language and support tool for the engineering of situational methods are discussed.

Keywords: Information systems research; Method engineering; Information systems development methods; Situational methods

1. Introduction

The everyday practice of information systems development is very diverse. Application domains, analysis and design techniques, programming languages, development paradigms, and project strategies can all vary over different spectra. For instance, the application domain can be transaction processing, real-time process control, or decision support, which have their own specification formalisms and systems development methods. The usefulness of the emerging paradigms for systems development, such as process networks and object-orientation, is debated between the practitioners and the theoreticians. The research world of information systems development is dispersed over many areas. We make the following observations:

- The syntactical structures (grammar, meta-model) of the various specification formalisms are very similar. The semantics of the formalisms, i.e. the precise interpretation of the concepts and interrelationships, can be very distinct.
- The application of information systems development methods makes no sense without a proper automated support tool. We see a further amalgamation of methods and tools: functionality of tools is extended with engineering process support features, complex consistency rules are automatically checked and guarded.
- Little research is performed on real-life information systems development projects. The problems of large-scale systems development coping with all sorts of intricate project constraints are hardly subject to investigations.

This leaves both the practitioner and the researcher in an immature, difficult professional situation. Some structure in this chaos would benefit to the deeper understanding of systems development as an engineering phenomenon that has, and will have, substantial impact on society. We therefore aim to clarify this by providing a research framework for methods and tools for information systems development, baptized with the name method engineering. In order to establish a good starting point for the discussion we start with the definition of the major terms method, technique, tool, and methodology in the next section. Thereafter we introduce the notion of method engineering and discuss several research issues. The remainder of the paper is devoted to situational methods, which are methods configured specifically for the project at hand. The meta-method for the configuration of a situational method is presented, along with a discussion of a method engineering language to describe methods and tools, and with a tool to support the method engineering process.

2. Basic terms

For decades the information systems world has been struggling with its terminology. This is due to its young age as well as to commercial influences. In order to establish a proper scientific basis we need to agree on good terminology. It is essential to relate the terms in accordance with other branches of science that have similar methodical development approaches, such as organizational sciences and mechanical engineering. We give the definitions of the four central notions in method engineering.

2.1. Definition 1: Method

A method is an approach to perform a systems development project, based on a specific way of thinking,

*email:brinkkemper@cs.utwente.nl

SSDI 0950-5849(95)01059-9

consisting of directions and rules, structured in a systematic way in development activities with corresponding development products.

The word 'method' comes from the Greek 'methodos', which means way of investigation. Examples of methods for information systems development are Information Engineering, SSADM, and Jackson Systems Development. Recently, several methods with object-orientation as the central way of thinking were introduced, such as OMT of Rumbaugh et al. [1], and Objectory of Jacobson [2]. Methods are usually described in textbooks and manuals giving the step-wise structuring of the development activities and the structural requirements for the products, also called deliverables. As we are dealing with methods for information systems, we will in the sequel refer to information systems development methods as ISDMs.

2.2. Definition 2: Technique

A technique is a procedure, possibly with a prescribed notation, to perform a development activity.

Commonly, only notations are referred to as techniques. But, similarly as, for instance, electrical engineering is more than drawing electronic circuits using a standardized notation for transistors, resistances and the like, a systems developer employs his professional skills by applying certain notations for design or programming in some structured plan. We therefore claim that a technique should not only embody the representational aspects of development, but also the procedural aspects. Examples of techniques are data modelling with entity-relationship diagrams, interviewing with plain natural language, pseudo-coding with action-diagrams. Techniques can be classified in several ways: related to the degree of formality of the notation (e.g. natural language, structured graphics, or Z), or related to the type of development activity it supports (e.g. data modelling, process modelling, interaction design).

2.3. Definition 3: Tool

A tool is a possibly automated means to support a part of a development process.

The spectrum of tools for systems development is very varied. CASE tool, Integrated Project Support Environments (IPSE), Analysts Workbenches are popular names for types of tools. Some tools just support a couple of different notations, whereas others provide assistance to the whole development life-cycle.

2.4. Definition 4: Methodology of information systems development

The methodology of information systems development is the systematic description, explanation and evaluation of all aspects of methodical information systems development.

This definition implies that we restrict the term methodology to scientific theory building about methodical information systems development. The misuse of the term methodology standing for method is a sign of the immaturity of our field, and should consequently be abandoned. Observe furthermore, from this definition, that there is just one methodology of information systems development and that all research activities in this field contribute to this methodology. Nevertheless, some methodological schools can be distinguished: the software engineering world with its roots in the programming traditions, the management information systems (MIS) arena originating from business schools, and the socio-technical approaches.

3. Method engineering

The area of methods and tools is lacking a proper framework for research. Methods and tools are being developed and employed over years, but a structure to take stock, generalize, and evaluate is needed. We therefore introduce the term method engineering to provide such a structure.

3.1. Definition 5: Method engineering

Method engineering is the engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems.

Similarly as software engineering is concerned with all aspects of software production, so is method engineering dealing with all engineering activities related to methods, techniques and tools. The term method engineering was already introduced in mechanical engineering to describe the construction of working methods in factories. Kumar and Welke coined the term methodology engineering in their paper on situational methods [3], but as explained in the previous section we prefer the proper term method engineering.

It must be obvious that the area of method engineering has links with a lot of other research areas. We mention project management, software configuration management, software engineering environments, software process modelling, and computer supported cooperative work.

Given the present status of the field there exists a multitude of open research questions. We have selected four of them to be presented shortly.

(1) **Meta-modelling techniques.** The design and evaluation of methods and tools require special purpose specification techniques, called meta-modelling techniques, for describing their procedural and representational capabilities. Issues are: what are the proper constructs for meta-modelling; what perspectives of meta-models

should be distinguished; is there a most optimal technique for meta-modelling, or is the adequacy of the technique related to the purpose of the investigation?

(2) **Tool interoperability.** As indicated, there exist lots of tools that only cover part of the development life-cycle. So the ISD practice is confronted with the proper integration of the tools at hand, called interoperability of tools. Open problems are related to the overall architecture of the integrated tools. Should this be based on the storage structure (i.e. the repository) in a data-integration architecture, or on a communication structure between the functional components in a control-integration architecture?

(3) **Situational methods.** As all projects are different, they cannot be properly supported by a standard method in a textbook or manual. How can proper methodical guidance and corresponding tool support be provided to system developers? Construction principles for methods and techniques need further investigation. In the remainder of the paper we will discuss the first research results to some of the questions being raised due to situational methods.

(4) **Comparative review of methods and tools.** How can the quality of a method or of a tool be expressed in order to compare them in a sound, scientifically verifiable way? Quality of methods comprises aspects as completeness, expressiveness, understandability, effectiveness of resources, and efficiency. Efforts in meta-modelling of methods and tools show the advantages of an objective, unbiased description for comparative review [4–6].

4. Situational methods

A situational method is an information systems development method tuned to the situation of the project at hand [7]. Engineering a situational method requires standardized building blocks and guide-lines, so-called meta-methods, to assemble these building blocks.

The importance of situational methods was already recognized by Olle et al. [8]. This 'scenario philosophy' has been further elaborated by Kumar and Welke, who introduced methodology engineering, being an approach to develop and implement methods [3]. A method representation model

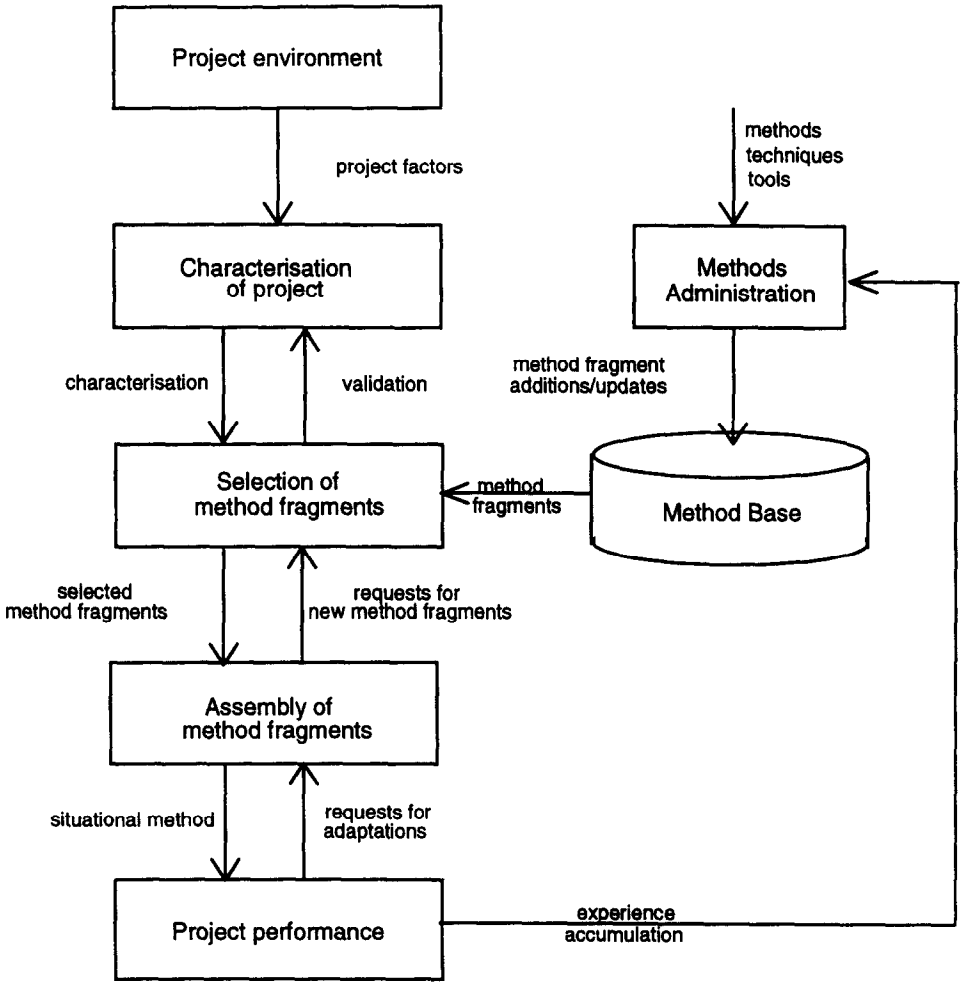

Fig. 1. The configuration process for situational methods.

providing ISDM concepts and a technique to analyse and compare existing methods was presented by Heym and Österle [9]. Saeki et al. also developed a method representation model, as well as a data base called 'method base' from which several complete ISDMs can be selected [10]. Hidding et al. [11] introduce the notion of task package, being a part of the process perspective of methods. Van Slooten et al. outline the construction process of situational methods, emphasizing the determination of the project characterization [12]. Harmsen et al. [7] present the structure of a method based to be filled with parts of existing ISDMs, called method fragments.

Critical to the support of engineering situational methods is the provision of standardized method building blocks that are stored and retrievable from a so-called method base. Furthermore, a configuration process should be set up that guides the assembly of these building blocks into a situational method. This configuration process is shown in Fig. 1. The building blocks, called *method fragments*, are defined as coherent pieces of IS development methods. We distinguish product fragments and process fragments. Product fragments model the structures of the products (deliverables, diagrams, tables, models) of a systems development method. Process fragments are models of the development process. Process fragments can be either high-level project strategies, called method outlines, or more detailed procedures to support the application of specification techniques. We are currently developing a method engineering language, (MEL), that allows to describe and manipulate method fragments. We give a short introduction to MEL below under 'A method engineering language'.

Every project is different, so it is essential in the method configuration process to characterize the project according to a list of contingency factors. This project characterization is input to the selection process, where method fragments from the method base are retrieved. Experienced method engineers may also work the other way round, i.e. start with the selection of method fragments and validate this choice against the project characterization. The unrelated method fragments are then assembled into a situational method. As the consistency and completeness of the method may require additional method fragments, the selection and validation processes could be repeated. Finally, the situational method is forwarded to the systems developers in the project. As the project may not be definitely clear at the start, a further elaboration of the situational method can be performed during the course of the project. Similarly, drastic changes in the project require to change the

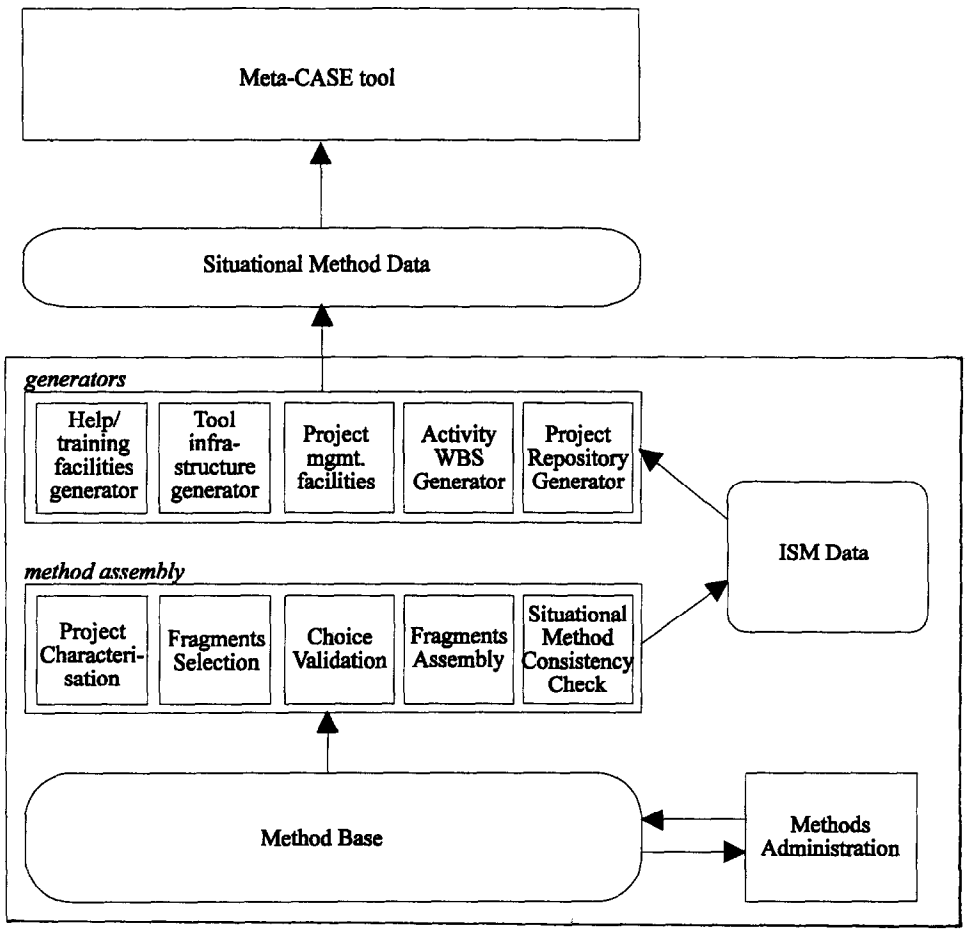

Fig. 2. The architecture of Decamerone.

situational method by the removal of inappropriate fragments followed by the insertion of suitable ones.

5. Tools to support method engineering

Currently, we are developing Decamerone, a Computer Aided Method Engineering (CAME) tool that is based on and used in conjunction with the meta-CASE tool Maestro II [13]. This meta-CASE tool is a fully adaptable CASE tool providing support for the systems development process as well as for the adaptation of the various diagram editors available. The architecture of Decamerone is depicted in Fig. 2. The three functional components in the tool, the method administration tool, the method assembly tool and the generators, provide complete support for the method configuration process outlined in the previous section. Output of the CAME tool is the situational method data that enables to parameterize the meta-CASE tool with the situational method resulting into a situational CASE tool. See the paper of Harmsen et al. [14] for more details.

6. A method engineering language

For description, administration, selection, and assembly of method fragments, we are developing the language MEL. This language provides methodology-dedicated concepts and operators, which apply to both higher level method fragments, like stages and deliverables, and low level method fragments, such as concepts and their relationships. MEL descriptions can be represented graphically, but also in textual or tabular form. We show only the textual representation.

Method fragments are described by listing their components, and by specifying relationships with other method fragments. For process fragments, optionality, alternative steps, repeated steps, and parallelism can be specified. For product fragments, only optionality can be indicated. A large number of method fragment properties, such as *goal*, *purpose*, *creator*, *source method*, *application domain*, are keywords in MEL, possessing predetermined value domains for ease of specification. To cope with method fragments derived from other method fragments (such as 'logical data model' being derived from 'data model'), and to enable multiple views on essentially the same method fragment (such as the manager's and the analyst's views on ERD), an inheritance mechanism is introduced in MEL. MEL recognizes certain verbs, such as 'Create' or 'Make', and nouns, such as 'Description' and 'Diagram', accommodating various kinds of consistency checks. For instance, a 'Diagram' has to be specified by concepts and associations, whereas a 'Description' has not. Fig. 3 shows three examples of method fragments described in MEL descriptions.

Note, that for the first product fragment, only two properties are defined: source and purpose. Components of this fragment are *concepts*, between which a number of associations in the method base exist. The associations that should be taken into account are indicated between the brackets. Process fragments usually require one or more product fragments. Components of process fragments are either activity descriptions, decisions, or other process fragments, structured by constructs to model iteration, parallelism, optionality, and choices. In the example, the process fragment Create Entity-relationship Diagram consists of activity descriptions, meaning that they are not further specified by a process fragment. A process fragment usually yields one or more deliverables. The last example illustrates the inheritance mechanism to accommodate different views on one method fragment. A managerial view on the first product fragment Entity-relationship Diagram could be an ER-diagram where the attributes are hidden.

Besides its method fragment description ability, MEL can be used for the administration of method fragments, by offering operations to change the underlying concept structure of the method base, the Methodology Data Model, or the internal structure of method fragments. Furthermore, constructs are provided for *method fragment selection*, by offering query operations, and for method *assembly*, by offering operations to combine or disconnect method fragments. Fig. 4 depicts an example of each type of operation.

7. Summary and conclusion

We have introduced method engineering as a research framework for information systems development methods and tools. The basic terms for method engineering: method, technique, tool and methodology, have been defined to aid the future scientific debate. The research in the area of method engineering has been exemplified with a discussion of the first results of situational methods: a configuration procedure for situational methods, a CAME tool, and a method engineering language.

Essential to the future development of the field is to keep our eyes open for the needs of the development practice. The research agenda should be set with the needs from industry in mind. Further detailing of research priorities should guide the academic and industrial researchers involved in method engineering projects. We are convinced that method engineering is a promising research field.

Acknowledgements

We wish to thank members of the Design Methodology research group of the Department of Computer Science, University of Twente for their valuable contributions to the overall research project of method engineering.

References

[1] Rumbaugh, J, Blaha, M, Premerlani, W, Eddy, F and Lorenson, W *Object-oriented Modeling and Design* Prentice-Hall (1991)

```
PRODUCT Entity-relationship Diagram:
SOURCE UTMethod;
PURPOSE Data modelling;
(
    CONCEPT Entity (ALL);
    CONCEPT Relationship (Involves Entity);
    CONCEPT Attribute (Describes Entity);
).

PROCESS Create Entity-relationship Diagram:
# simplified version; for demonstration purposes #
REQUIRED Function List;
GOAL Data modelling;
SOURCE UTMethod;
(
   - Determine provional Attribute List;
   - Determine Entity List;
   REPEAT
   - Create global Entity-relationship Diagram;
   - Check global Entity-relationship Diagram;
   UNTIL global Entity-relationship Diagram supports whole Function List;
)
DELIVERABLES {Entity List, Entity-relationship Diagram}.

PRODUCT Entity-relationship Diagram(Manager):
FOR Manager;
(
   INHERITS FROM Entity-relationship Diagram {SOURCE UTMethod, GOAL Data modelling}
   HIDE CONCEPT Attribute
).
```

Fig. 3. MEL descriptions of method fragments.

```
Delete from Entity-relationship Diagram CONCEPT Attribute.

Select PRODUCT Where SOURCE = SSADM

Join Entity-relationship Diagram With Data-flow Diagram
     Through (Entity Describes Data store; Entity Describes Data flow).
```

Fig. 4. Example of an administration, a selection, and an assembly operation.

[2] Jacobson, I, Christerson, M, Jonsson, P and Overgaard, G *Object-oriented Software Engineering* Addison-Wesley (1992)

[3] Kumar, K and Welke, R J 'Methodology engineering: a proposal for situation-specific methodology construction' in Cotterman, W W, Senn J A (eds) *Challenges and Strategies for Research in Systems Development* John Wiley (1992)

[4] Song, X and Osterweil, L J 'Toward objective, systematic design-method comparisons', *IEEE Software* (May 1992) pp 43–53

[5] Hong, S, vd Goor, G and Brinkkemper, S 'A comparison of object-oriented analysis and design methodologies', *Proc. 26th Hawaiian Conf. on System Sciences (HICSS-26)* IEEE Computer Science Press, Vol IV (1993) pp 689–698

[6] Marttiin, P, Rossi, M, Tahvainanen, V-P and Lyytinen, K 'A comparative review of CASE shells—a preliminary framework and research outcomes', in *Information and Management* Vol 25 No 1 (1993) pp 11–31

[7] Harmsen, F, Brinkkemper, S and Oei, H 'Situational method engineering for information system project approaches', in Verrijn Stuart, A A and Olle T W (eds), 'Methods and associated tools for the information systems life cycle'. *Proc. of the IFIP WG8.1 Working Conference CRIS'94* Maastricht, September 1994, North-Holland, Amsterdam, pp 169–194

[8] Olle, T W, Hagelstein, J, MacDonald, I G, Rolland, C, Sol, H G, van Assche, F J M and Verrijn-Stuart, A A *Information Systems Methodologies—a Framework for Understanding* (2nd edn) Addison-Wesley (1991)

[9] Heym, M and Österle, H 'Computer-aided methodology engineering', in *Inf. and Soft. Technol.* Vol 35 No 6/7 (1993) pp 345–354

[10] Saeki, M, Iguchi, K, Wen-Yin, K and Shinohara, M 'A meta-model for representing software specification & design methods' in Prakash, N, Rolland, C and Pernici, B (eds), *Proc. of the IFIP WG8.1 Conference on Information Systems Development Process* Como (1993)

[11] Hidding, G J, Freund, G M and Joseph, J K 'Modeling large processes with task packages', *Workshop on Modeling in the Large, AAAI Conference*, Washington, DC (1993)

[12] Slooten, K van, and Brinkkemper, S 'A method engineering approach to information systems development', in Prakash, N, Rolland, C and Pernici, B (eds), *Proc. of the IFIP WG8.1 Conference on Information Systems Development Process* Como (1993)

[13] Merbeth, G 'Maestro II—the integrated CASE system of Softlab (in German: Maestro II—das integrierte CASE-System von Softlab), in Balzert, H (ed), *CASE Systeme und Werkzeuge* 3e Auflage, BI Wissenschaftsverlag (1991)

[14] Harmsen, F, Brinkkemper, S and Oei, H 'A language and tool for the engineering of situational methods for information systems development', in Zupancic, J and Wrycza, S (eds), *Proc. of the ISD'94 Conference*, Bled, Slovenia September 1994. pp 206–214