Revista UNIMAR 19(4):937-953, 1997.

AN OBJECT-ORIENTED METHOD FOR DESIGNING SHARABLE DATA SCHEMAS IN SOFTWARE ENGINEERING ENVIRONMENTS

Itana M. S. Gimenes*, Cristina D. Aguiar Ciferri*, Edicezar L. Nanni+ and Selma B. Sabião+

ABSTRACT. The integration of CASE tools in software engineering environments requires a set of integration mechanisms to allow data sharing, communication and process conformance. Solutions for the sharing and dynamic management of data schemas in these environments comprise specific repositories, such as PCTE, and object-oriented database management systems (OODBMS). This paper presents an extension with object-oriented concepts to Brémeau's method for designing sharable data schemas. The extension applies both to PCTE and OODBMS. An application of the proposed method is shown through a case study which presents and integrates three CASE tools.

Key words: data integration, data schema, software engineering environments.

UM MÉTODO ORIENTADO A OBJETOS PARA PROJETAR ESQUEMAS COMPARTILHÁVEIS DE DADOS EM AMBIENTES DE ENGENHARIA DE *SOFTWARE*

RESUMO. A integração de ferramentas CASE em ambientes de engenharia de *software* requer um conjunto de mecanismos de integração que permitam compartilhamento de dados, comunicação e uniformidade de processos. Soluções para o compartilhamento e o gerenciamento dinâmico de esquemas de dados nesses ambientes envolvem repositórios específicos, tais como PCTE, e sistemas gerenciadores de banco de dados orientados a objetos (OODBMS). Este artigo apresenta uma extensão que introduz conceitos de orientação a objetos ao método de Brémeau para projetar esquemas de dados compartilháveis. A extensão proposta se aplica tanto ao PCTE quanto aos OODBMS. Uma aplicação

* Departamento de Informática, Universidade Estadual de Maringá, Av. Colombo, 5790, Câmpus Universitário, 87020-900, Maringá-Paraná, Brasil. E-mail: itana@din.uem.br

+ Instituto de Computação, Universidade Estadual de Campinas, Campinas-São Paulo, Brasil.

Correspondência para Itana M. S. Gimenes

Data de recebimento: 08/08/97.

Data de aceite: 28/11/97.

do método proposto é descrita por meio de um estudo de caso que apresenta e integra três ferramentas CASE.

Palavras-chave: integração de dados, esquemas de dados, ambientes de engenharia de *software*.

INTRODUCTION

The CASE[1] technology is based on products which help the developers in some areas of the software process such as: organisation and documentation of the process, analysis and design of the software and configuration and version management. In general, one CASE tool does not cover all phases of the software process. This leads developers and managers to use several tools in the same project. These tools are produced by different suppliers which follow neither standard means of communication nor standard data formats. Thus, project developers and managers have been facing the difficult problem of selecting and administrating a completely independent and diverse set of tools.

Several studies have been made to define mechanisms which facilitate the integration of CASE tools (Schefström, 1993; Brown *et al*., 1994). Integration models presented in Wasserman (1990) and Thomas and Nejmeh (1992) distinguished, at least, three independent dimensions through which integration issues can be examined: data, control and presentation. Data integration aims at defining mechanisms to dynamically administrate sharable data schemas. This paper focuses on data integration.

In addition to support mechanisms, data integration requires a systematic design of tool schemas. It is necessary to identify clusters of types with common features such as: versions, stability, and permissions and access control. It is also important to maximize the reuse of existing types by trying to define a uniform semantics for the types used by several tools. Thus, a conventional data modelling method is not sufficient to model the data of CASE tools which are integrated into open Software Engineering Environments (SEE).

This paper presents an object-oriented method for designing sharable data schemas for open SEE repositories. In particular, it focuses on SEE based on Portable Common Tool Environment (PCTE) and Object-Oriented Database Management Systems (OODBMS). An application of

---

[1] Computer Aided Software Engineering.

the proposed method is shown through a case study which presents and integrates three CASE tools.

Section 2 presents PCTE and OODBMS as potential data repositories for SEE. Section 3 describes the proposed method as an extension of Brémeau's method while section 4 presents the case study. Section 5 summarizes most relevant related works and, finally, section 6 presents the conclusions.

DATA REPOSITORIES FOR SEE

Several approaches for data repository have been developed for SEE. The first generation of them was based on Entity-Relationship-Attribute (ERA) or relational modelling extensions. More recently, specialized object bases, such as PCTE, and commercial OODBMS have been consolidated as potential options for SEE. This section discusses the features of PCTE and OODBMS for SEE.

PCTE

PCTE (ISO/IEC 13719) (Wakeman and Jowett, 1993) is a standard which defines a public tool interface to serve as base for the construction of open SEE. PCTE uses an ERA enriched with several features of integration and security which make the modelling of data schema special.

PCTE represents data, attributes and links as objects of specific types. These objects are represented, in the object base, either as individual types or as collections of related types which are called SDS (Schema Definition Set). Each SDS represents part or the whole model of a tool or an activity. A SDS is a composite object of which the components are called *type_in_SDS*. SDSs are used to define specific sets of the data manipulated by the tool into the object base. When new types are defined in PCTE, they can be integrated with the existing SDSs into the object base. PCTE already offers pre-defined SDSs, such as: system, metasds, security and accounting. The general structure of tool integration into PCTE is illustrated in Figure 1. It shows a SDS for a Requirement Analysis Tool integrated with a SDS of a Coding Tool.

PCTE provides a specific view for each tool through the *working schema* mechanism. Each working schema represents a particular domain or view, such as the role of a user (e.g. programmer or manager) or the

view of a tool or activity (e.g. requirement elicitation or coding). In PCTE, every process has its associated working schemas which enable them to access their instances. Thus, it is through these schemas that executable tools recognise the representation of their data models.

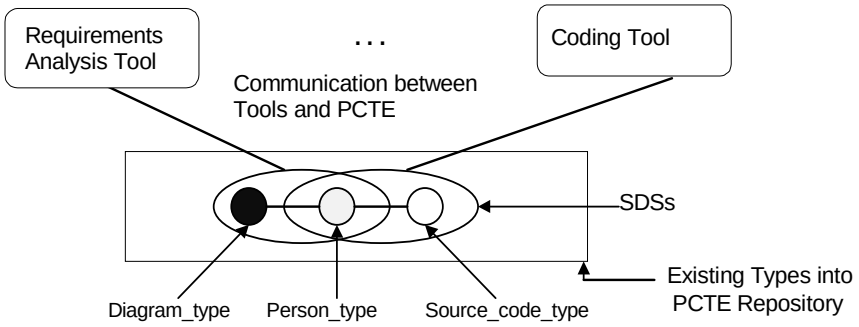

Figure 1. Tool integration in PCTE.

A working schema is a well formed union of types of a sequence of SDSs. This gives origin to a set of *type_in_working_schema*. PCTE provides built-in well defined rules to integrate properties of the defined type into the SDS sequence that composes the working schema. These rules and operations form the mechanisms for dynamic schema management that enable the insertion of tools into open SEE. In addition, they provide a strong basis for reusing the types already defined in the environment base.

In addition to the data management services, PCTE also provides services for process control, definition of access groups, tool enveloping (for external tools) and tool communication via notification.

Although it has become an international standard, PCTE did not achieve the expected commercial success. Few companies have built environments or frameworks based on this standard (Stanford Management Group, 1995; EDS, 1994).

OODBMS

One of the reasons for PCTE difficulties is the parallel improvement of OODBMS which start to offer a competitive set of services for SEE.

The main advantages of using OODBMS are that they encompass all the advantages of the object-oriented approach. This gives flexibility to

define both the characteristics and the behaviour of objects. More modern OODBMS implement mechanisms such as triggers, notifications and invocations, that can be used to verify rules and constraints over classes. Special libraries for software engineering can be acquired from the market which can form an appropriate layer over OODBMS. These library services can be extended by adding new classes with more specific characteristics and behaviours. Thus, more modern and efficient tools, already designed for an object-oriented framework can take advantage of these services to achieve better performance. However, OODBMS do not usually provide services for dynamic definition and manipulation of schemas. They do not usually support the definition of set of types that a tool can view or access, such as the SDSs and working schemas of PCTE. Another shortcoming is the lack of built-in rules for schema integration. Thus, the use of OODBMS as the base of open SEE requires the implementation of a layer to support specific services, as mentioned above.

Therefore, it is not straightforward to decide on the best alternative for data integration. When using OODBMS, additional work is needed to support particular SEE facilities based on simple data abstraction services. On the other hand, PCTE restricts the SEE development to a specific architectural mode.

AN OBJECT-ORIENTED METHOD FOR DESIGNING SHARABLE DATA SCHEMAS

This section describes an extension with object-oriented concepts to Brémeau's method (Brémeau and Thomas, 1993, 1996) for designing and integrating sharable data schemas. This extension makes the design easier both for PCTE and OODBMS. Section 3.1 presents a summary of Brémeau's method, pointing out its shortcomings. Section 3.2 presents the proposed extensions.

Brémeau's Method

This method has four main activities or phases: conceptual design, logical design, definition of SDSs, review and validation (see Figure 2).

The *conceptual design* takes as input data and operation concepts plus software process assumptions and performs the conceptual data modelling of the tool. It offers as output ERA global schemas and

instance diagrams. This phase is decomposed into five steps: define/refine Data Flow Diagram (DFD), define/refine data models, integrate data models, create instance diagrams, and create subset and usage schemas. This phase is completely PCTE-independent. ERA models may even include some n-ary relationships which are not defined by the PCTE OMS[2]. Some of the relevant aspects analyzed in this phase, regarding tool integration, are:

- the groups of users (and their respective roles) which perform operations on data stores as this is important to define the access to data from the tools;
- the effect of operations over related data stores in order to define data clusters of the tools.

The logical design takes as input the ERA global schemas and instance diagrams, produced in the conceptual design, plus existing groups, types, SDSs and tools in the SEE. It offers as output annotated global schemas and instance diagrams with clusters. This phase is decomposed into four steps: identify clusters, refine global schemas, revise instance diagrams and annotate global schemas. This part of the method represents the transition to the PCTE-specific schema design. It takes the instance diagrams and draws cluster boundaries on them, corresponding to the cluster criteria of: versioning, stability, concurrence and discretionary and mandatory access permissions.

The *definition of SDSs* takes as input the set of ERA subset and usage schemas, produced in the conceptual design, plus annotated global schemas and instance diagrams with clusters produced in the logical design. It delivers as output a list of SDSs. It is decomposed into three steps: select PCTE-type properties, evaluate cluster semantic support and allocate PCTE global schemas to SDSs. This phase exploits the annotated schemas to make final decisions on the choice of appropriate PCTE types and links.

The review and validation takes as input: a list of SDSs, produced in the definition of SDSs; the DFD processes, operations and groups, produced in the conceptual design; and the initial operation concepts. The output is the validated list of SDSs. It is decomposed into three steps: analyse performance, check design robustness and validate SDSs.

---

[2] Object Management Systems.

In addition, there are feedbacks between the phases in order to obtain a more precise and integrated modelling. A complete description of Brémeau's method can be found in (Brémeau and Thomas, 1996).

A shortcoming of Brémeau's method is that it is too specific for PCTE. In addition, it is not based on an object-oriented design method.

The Proposed Method

The objective of the proposed method is to incorporate object-oriented concepts in order to make the mapping of the tool's data schemas both onto PCTE and OODBMS easier, thus considering data, behavioural and functional aspects. The main difference between the proposed and Brémeau's method is related to the conceptual design phase. In this phase Brémeau's method is based on (Batini *et al*., 1992) whereas our method is based on OMT (Rumbaugh *et al*., 1991). The differences between the methods are illustrated in Figure 2. In this figure normal fonts represent information of both methods; italic fonts apply only for Brémeau's method; and bold fonts belong to the proposed method.

In the following sections the extensions proposed to each phase of Brémeau's method are presented.

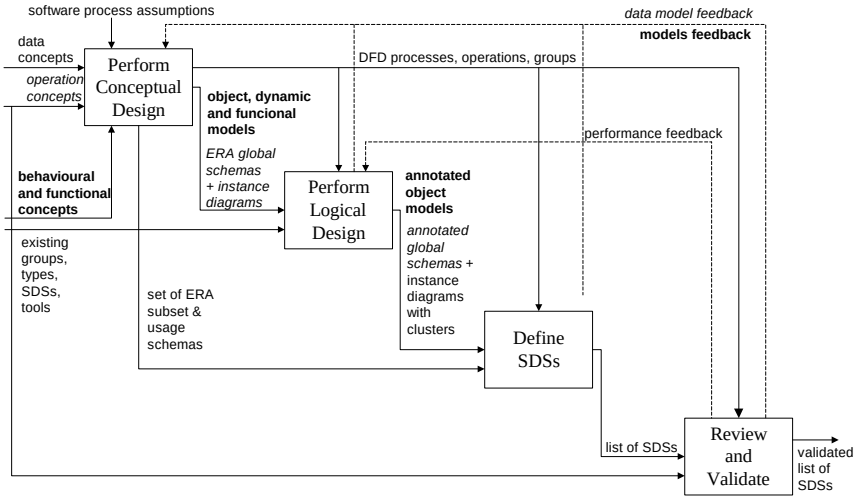

Figure 2. Overview of Brémeau's and proposed method.

Conceptual Design

This phase consists of modelling the tools using OMT. It takes as input data, behavioural and functional concepts of the tool plus software process assumptions and produces the object, dynamic and functional models.

The object model represents the static and structural aspects of the tool's sharable data. This modelling should identify classes of objects, associations and aggregations between objects. It should also consider the object attributes and their relationships. Inheritance should be used to organize and simplify class structuring.

The dynamic model represents the temporal and behavioural aspects of the tool, mainly the sequence of interactions between operations. This model requires the definition of scenarios, and typical and exceptional sessions in which the tool operates. This modelling should: identify external events between the tool and the external world; design a state diagram for each active object, emphasising both the event patterns that it receives and sends, and the actions it executes; and compare events between state diagrams. The final dynamic model is composed of the set of individual state diagrams.

The functional model represents the tool's operations, not considering when or how. In this modelling, a DFD is designed based on the identification of the input and output values of the tool services.

This phase offers the basis for mapping the models either to PCTE or to OODBMS in the following steps of the method. For simplicity, hereafter, we refer to PCTE or OODBMS as object base.

The following phases of the proposed method are very similar to Brémeau's method. One of the differences is that Brémeau's method constructs instance diagrams in the conceptual design phase, based on DFDs, whereas the proposed method derives these diagrams from the logical design, based on the object and dynamic models. Note that we assume that an equivalent concept of SDSs and working schemas will be applied to OODBMS.

Logical Design

This phase consists of the mapping of the OMT models onto the schemas of the object base. It takes as input object, dynamic and functional models and delivers as output annotated object models and instance diagrams with clusters. Three main steps are involved: design of

instance diagrams, identification of clusters and annotation of the object model. In addition, refinements of the OMT models may take place. The instance diagrams are designed based on the object and dynamic models. They describe the behaviour of specific sets of objects before and after each operation. These diagrams are drawn to analyse the boundaries between related objects and makes group those which participate with the same operations into clusters.

The annotation step makes the initial mapping of the associations and attributes of the object model onto the particular properties of the object base. This mapping takes into consideration the pre-defined types of the object base. For instance, PCTE attributes can be mapped onto: boolean, float, integers, natural, string, time, enumeration or contents. New types are only created if an equivalent type is not already defined. In PCTE, associations can be mapped to properties such as: composition, existence, referential integrity and relevance to the origin.

Definition of SDSs

This step produces a list of SDSs from the annotated object model and the instance diagrams with clusters. Three activities are involved: select the final type properties of the object base, evaluate the selected properties and produce the list of SDSs.

The first activity consists of taking decisions to complete the initial mapping of the object model developed in the previous phase. For instance, in PCTE, if an association is annotated as not having the composition property but the referential integrity, the choices of association types are restricted to existence, reference or implicit. There might be cases in which the annotations do not lead to a straightforward mapping onto the object base and feedback to the previous step is necessary to adjust the models.

The evaluation step checks whether the selected types of objects, associations and attributes conform with the requirements of the logical design. For instance, it verifies whether the association properties are appropriate to support the defined clusters.

The list of SDSs should take into account that the defined schemas may be reused by further tools integrated into the environment.

Revision and Validation

This phase is very similar to the equivalent one in Brémeau's method. Thus, it analyzes the performance and robustness of the proposed list of SDSs.

A CASE STUDY USING THE PROPOSED METHOD

This section presents an experiment related to the first step of the proposed method. It consists of modelling three CASE tools and developing their integrated models. It aims at producing integrated models which are more concise and increase data and behavioural sharing. The models were designed at a high level, only considering the main characteristics and most basic operations of the tools. Although we have developed the three OMT models in this experiment, due to space limitations, here we present only the object and dynamic models.

The tools used in the experiment are: a version management, a project management and a bug tracking. This example is an object-oriented version of the model described in (Brown *et al*., 1994), using the ERA model. Figures 3 to 5 show the individual object models of these tools while Figure 6 presents the integrated object model.

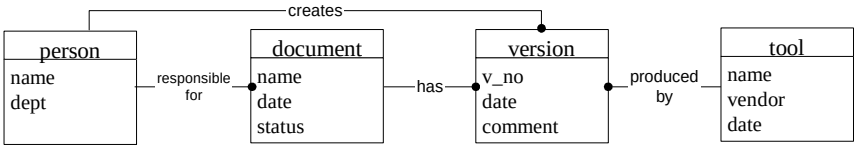

Figure 3. The object model for the version management tool.

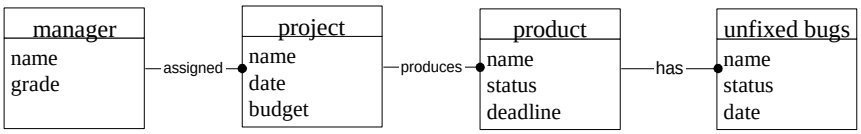

Figure 4. The object model for the project management tool.

The integrated object model was drawn by identifying and solving conflicts amongst similar and different representations of related data of the tools. For instance, *person* (Figure 3), *manager*

(Figure 4) and *developer* (Figure 5) were integrated into the *person* class (Figure 6). Similar correspondences were determined between classes and associations. This can be observed in the integration of the associations *responsible for* (Figure 3) and *assigned* (Figure 4). Another kind of integration was obtained by generalizing the objects *project*, *design*, *program* and *module* to the superclass *document*. This generalization was based on the fact that the objects of both the project management and the bug tracking tool are also versionable. The attribute *status* was not previously part of all the object subclasses but it was maintained in the *document* for semantic reason.

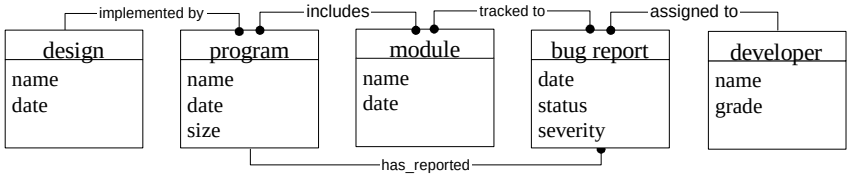

Figure 5. The object model for the bug tracking tool.

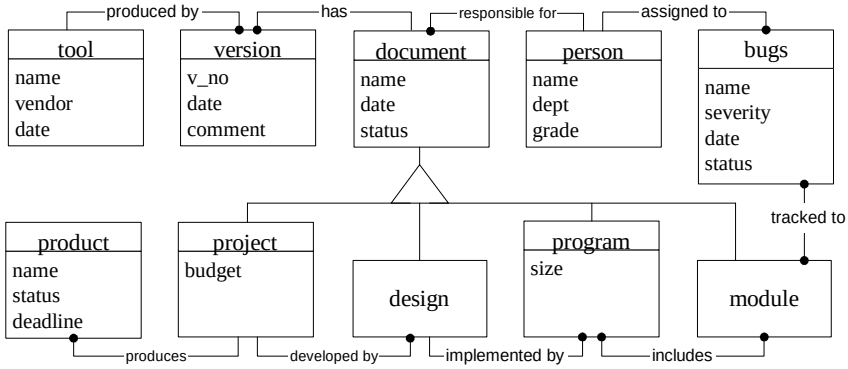

Figure 6. The integrated object model for the tools.

Some associations present in the individual models were eliminated from the integrated model as they became redundant (e.g. *creates* in Figure 3). On the other hand, additional semantic associations were included (e.g. *developed by* in Figure 4).

The dynamic models for each tool are presented in Figures 7 to 9 while the integrated model is shown in Figure 10. The integration of the dynamic models aimed at maintaining the equivalent operations amongst the tools in order to increase behaviour sharing. The integrated dynamic model was developed respecting the integrated object model. For instance, based on the integration of *person*, the operations of the corresponding objects were also integrated. As in the integrated object model the objects related to *document* were generalized, the correspondent operations in the dynamic model were integrated and assigned to this superclass. Further refinements may be necessary to customize the operations of the subclasses.

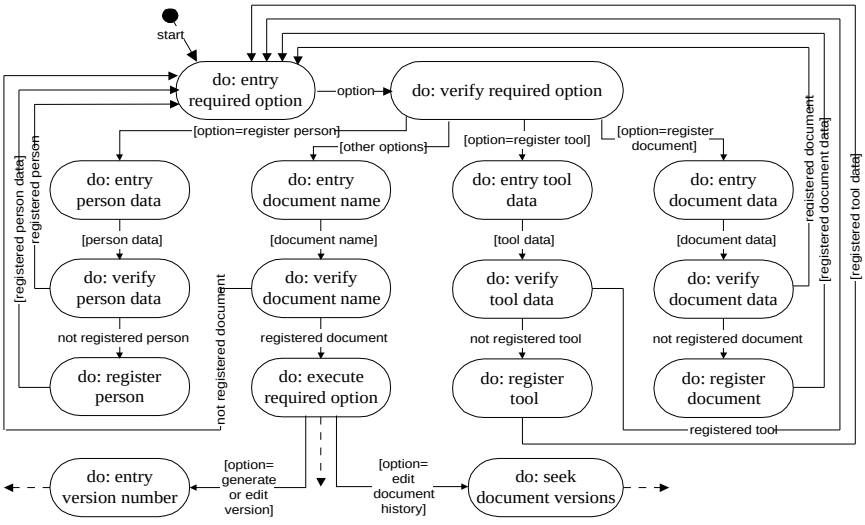

Figure 7. The dynamic model for the version management tool.

The integrated models obtained are concise and clear, and they avoid data and operation redundancies that could occur if the tools worked isolated. We can observe that further refinements are necessary to achieve completeness. In real experiences, this needs to be done taking into account the data and operation requirements of the specific environment and tools being modelled.

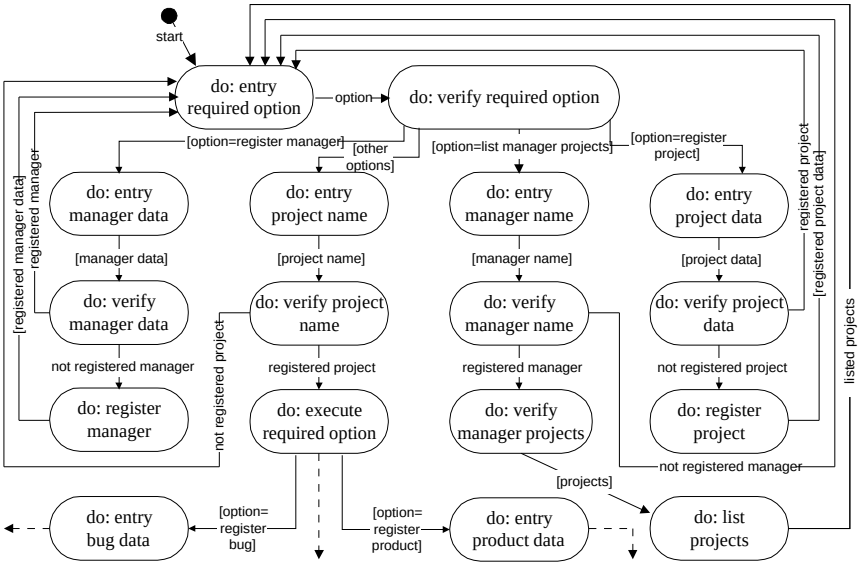

Figure 8. The dynamic model for the project management tool.

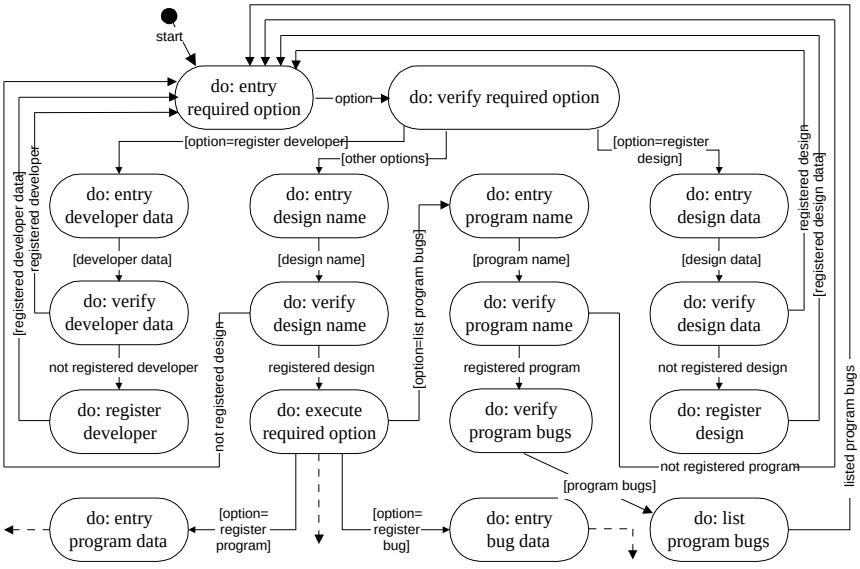

Figure 9. The dynamic model for the bug tracking tool.

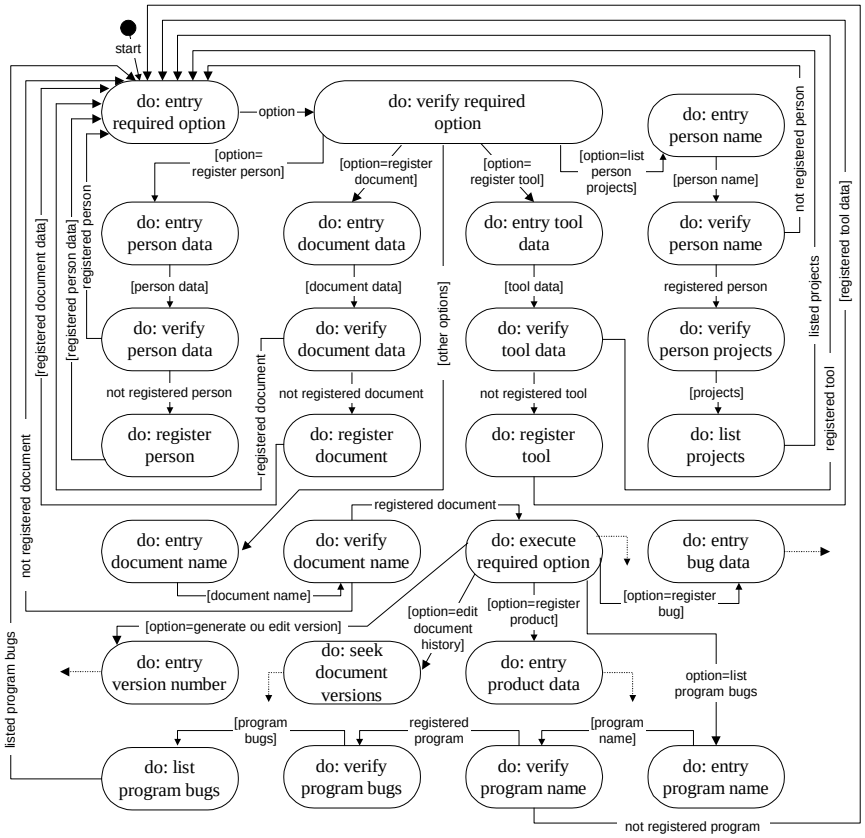

Figure 10. The integrated dynamic model for the tools.

RELATED WORK

Ideas of data integration were introduced since the Ada Programming Environment, which proposed a central repository to state the data shared by its services (Munch *et al.*, 1989). One of the major works in this area is PCTE (Wakeman and Jowett, 1993). This standard establishes advanced services to design and integrate the tool's schemas. There are commercial implementations of this standard (Transtar, 1995; EDS, 1994) and several environments were based on it (Finkelstein *et al.*, 1994). A more recent work by Transtar (Transtar, 1996) brings about a NIM (Neutral Information Model) which provides a meta-model to allow

the mapping of tools to a unified basis. (Reiss, 1996) uses a simplified form of data integration, called fragment integration, (logical unit) which involves creating a simple database by identifying logical portions of files and storing references to these portions as well as additional information to define links and store associated data.

OODBMS have also been used to provide support for SEE (Hudson and King, 1987; Bandinelli *et al*., 1994). However, OODBMS still do not offer some of the special services that PCTE provides, as it was pointed out in this paper. On the other hand, they have been more successful commercially.

Although a lot of work has been done to provide mechanisms for data integration, there are few works on methods to support the suppliers of tools in designing models to achieve data integration. Brémeau's method, which is the basis of our work, is one of them. We have used OMT (Rumbaugh *et al*., 1991) to achieve a design method that deals not only with the data aspects of integration but also with the behavioural ones.

In the database literature theories and methods have been proposed for schema integration (Batini *et al*., 1986; Kim, 1991; Spaccapietra and Parent, 1994). These works consist in identifying similarities and differences between elements of different schemas. In addition, they search for sets of distinct elements which are interrelated by some semantic property. Nevertheless, the main focus of these works is object (entity) integration. Ribeiro *et al*., 1994) have proposed a taxonomy of behavioural conflicts applied to object-oriented heterogeneous databases.

CONCLUSIONS

This paper presents an extension of Brémeau's method which makes the design of sharable data schemas in software engineering environment more systematic and easier. In particular, it focus on PCTE and OODBMS as alternatives data repository for SEE. The proposed method is based on OMT and uses integration rules derived from the PCTE standard. A case study which shows the application of the proposed method to the integration of three CASE tools was described and analyzed.

It was observed that the tool schema integration is a complex task which involves syntactical, semantic and optimization issues. Although

the case study involved only three open tools with few objects and associations, the integration of the object, dynamic and functional models was very laborious. It required a lot of semantic analysis. We can also point out that the instance diagrams did not prove its benefits. Equivalent studies can be carried out using the dynamic and functional models.

Although PCTE is based on a simple modelling approach such as ERA, its integration facilities are difficult to grasp due to the standard extension and complexity. Improvements have been made to the standard, which include dealing with fine grained data and object orientation (Object Management Group, 1995a) (Object Management Group, 1995b)

Additional work needs to be done to formalize the integration rules for the dynamic and functional models, as well as to develop of a tool to support the proposed method.

REFERENCES

BANDINELLI, S., FUGGETTA, A., GHEZZI, C. & LAVAZZA, L. Spade: an environment for software process analysis, design, and enactment, In: FINKELSTEIN, A., KRAMER, J. & NUSEIBEH, B. (ed.). *Software Process Modelling and Technology*. England: John Wiley & Sons, 1994. p. 223-244.

BATINI, C., CERI, S. & NAVATHE, S.B. *Conceptual database design: an entity-relationship approach*. The Benjamin/Cummings Publishing Company, 1992.

BATINI, C., LENZERINI, M. & NAVATHE, S.B.A. Comparative Analysis of Metodologies for Database Schema Integration, *ACM Computing Surveys*, 18(4):323-364, 1986.

BRÉMEAU, C. & THOMAS, I. A schema design method for PCTE, In: *Proceedings of the PCTE'93 Conference*, 1993.

BRÉMEAU, C. & THOMAS, I. *Designing schema for object bases*, SMG Technology Series, 1996.

BROWN, A.W., CARNEY, D.J., MORRIS, E.J., SMITH, D.B. & ZARRELLA, P.F. *Principles of case tool integration*, New York: Oxford University Press, 1994.

EDS DEFENSE LIMITED. *PORTOS - A Technical Overview*, Issue 1.0, Camberley, England, 1994.

FINKELSTEIN, A., KRAMER, J. & NUSEIBEH, B. (ed.). *Software process modelling and technology*, John Wiley & Sons, 1994.

HUDSON, S.E. & KING, R. Object-oriented database support for software environments, *ACM0-89791-236-5/87/0005/0491*, 1987.

KIM, W. Classifying schematic and data heterogenity in multidatabe systems. *Computer*, *24*(12):12-18, 1991.

MUNCK, R., OBERNDORF, P., PLOEREDER, E. & THALL, R. An overview of DOD_STD_1838A (proposed), the common APSE interface set, Revision A. *SIGPLAN. Notices 24*(2):235-247, 1989.

OBJECT MANAGEMENT GROUP PCTE SIG. *FG (Fine Grain Data) Extensions to the PCTE (Portable Common Tool Environment) Standard (ISO/IEC-13719)*, 1995 (draft version).

OBJECT MANAGEMENT GROUP PCTE SIG. *OO (Object-Oriented) Extensions to the PCTE (Portable Common Tool Environment) Standard (ISO/IEC-13719)*, 1995 (draft version).

REISS, S.P. Simplifying data integration: the design of the desert software development environment. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGENEERING, 18, 1996, Berlim. *Proceedings*... Berlim, 1996. p. 25-29.

RIBEIRO, C.F.P., CASTILHO, J.M. & OLIVEIRA, J.P.M. Establishing sameness in object-oriented conceptual modelling, In: CONFERENCE OF THE CHILEAN COMPUTER SOCIETY CONCEPTION, 14, 1994, Chile, *Proceedings*... Chile, 1994, p. 471-482.

RUMBAUGH, J., BLAHA, M., PREMERLANI, W., EDDY, F. & LORENSEN, W. *Object-oriented modelling and design.* Local: Prentice-Hall International, 1991.

SCHEFSTRÖM, VAN DEN BROEK, D. *Tool Integration:* Environments and Frameworks, John Wiley & Sons, England: 1993.

SPACCAPIETRA, S. & PARENT, C. View integration: a step forward in solving structural conflicts. *IEEE Transactions on Knowledge and Data Engineering*, *6*(2):258-274, 1994.

STANFORD MANAGEMENT GROUP. *Dbench technical overview*, Stanford, USA, 1995. (Technical Report).

THOMAS, I. & NEJMEH, B. Definitions of tool integration for environments for environments, *IEEE Software*, *9*(3):29-35, 1992.

TRANSTAR SOFTWARE INC. *Course:* PCTE and the emeraude PCTE framework. Paris, 1995. (Slides).

TRANSTAR SOFTWARE INC. *Transtar Repository:* repository models, Paris, 1996.

WAKEMAN, L. & JOWETT, J. *PCTE: The standard for open repositories*; England: Simon & Schuster International, 1993.

WASSERMAN, A. Tool integration in software engineering environments, In: LONG, F. (ed.), Software engineering environments. Lecture Notes in Computer Science 467:138-150, Berlin: Springer-Verlag, 1990.