UNIVERSIDADE DE SÃO PAULO

Evaluation of a Reverse Engineering Method through the Application in a Hypermedia System

Valéria Delisandra Feltrim
Renata Pontin de Mattos Fortes

Nº 44

NOTAS

Instituto de Ciências Matemáticas de São Carlos

Instituto de Ciências Matemáticas e de Computação

ISSN - 0103-2577

Evaluation of a Reverse Engineering Method through the Application in a Hypermedia System

Valéria Delisandra Feltrim
Renata Pontin de Mattos Fortes

Nº 44

NOTAS DO ICMC
Série Computação

São Carlos
Junho/1999

Evaluation of a Reverse Engineering Method through the Application in a Hypermedia System

Valéria Delisandra Feltrim[1]
Renata Pontin de Mattos Fortes
{vfeltrim, renata}@icmc.sc.usp.br

Departamento de Ciências da Computação e Estatística
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo

Abstract

The growth of the software market has led to an increasing use of development techniques, which are, sometimes, informal ones. The maintenance of this kind of software is problematic, since its documentation rarely reflects the implemented code. In this context Reverse Engineering of Software can help by means of recovering the project information lost during the development phase and documenting the current software state. This article discusses the issues emerged during the application of the method of reverse engineering named Fusion-RE/I. The described experiment is part of the re-engineering of a prototype hypermedia system, which has, as its main goal, the adaptation to a Software Engineering domain. Since the target is a hypermedia system, the results obtained during the use of Fusion-RE/I can be registered as a hyperdocument. By doing that, it is possible to observe and analyse some relevant issues concerning the method Fusion-RE/I.

1 Introduction

The growth of the software market has led to an increasing use of the development techniques, which are, most of the time, informal ones. Thus, the software documentation is often neglected and does not reflect the implemented code [1]. Besides, the constant modifications and additions of new characteristics to the software lead to unexpected collateral effects, which are not present in the documentation.

In this way, when facing the product maintenance, the software engineer finds an informal and incomplete documentation, which does not reflect the existing software. In this case, the software maintenance is an undoubtedly problematic phase, being responsible for costs of superior proportions regarding the other phases of systems life cycles [2].

---

[1] This work is supported by FAPESP, #97/12999-8.

The maintenance activity consists of three stages: understanding, modification and system revaluation [3]. Particularly, the understanding and modification stages are related to the software information availability, that is, they lean on the correct existence, consistency, completedness and update of the documents, which compose it.

The reverse engineering aims to contribute for the software maintainability increase, giving information firstly used in the software understanding and later in the software modification and revaluation [4]. This happens through the information retrieval from the software documents, aiming to acquire its representation in a higher level of abstraction and in a clearer way.

In its turn, the reverse engineering process, in general way, is not ordinary and demands a high cost of people/time to its accomplishment. This is due to the great volume of information involved during the process development. Besides, to maintain the consistency of the existing relationship among the information retrieved is crucial to the process efficiency and to allow a better understanding of the software under analysis.

The hypertext technology, serving as a more flexible way of making the information available for the application users, is quickly evolving, making possible for new retrieval and presentation ways of information to be examined. In fact, the basis of the hypertext system technology is the information network and has interconnections, which must be easily accessible for users[2]. This information network, where the "nodes" correspond to units of information and the "links" to the_interconnections among them, compose a hyperdocument. In the software engineering, besides a great volume of documents, there are a great variety of types of documents (diagrams, texts, source codes, executables, etc.). Such characteristic, plus the fact that such documents comprise enough_related information, suggests naturally the use of hyperdocuments as a suitable way for the storage and retrieval of this information [5].

One of the aims of this work is the hyperdocuments treatment to give the necessary support to the software reverse engineering process. In this way, a specification of the hyperdocument structure, which would allow the accuracy of the documentation content regarding the software product being documented, was elaborated during an application of the reverse engineering method. Besides, it is desired that the consistency among the hyperdocument parts and the software components be more easily acquired by means of the defined *links*.

In order to accomplish this, a hypertext environment, SASHE (*S*istema de *A*utoria e *S*uporte *H*ipermídia para *E*nsino) [6], was submitted to the reverse engineering, and to the stages of this process, later registered in SASHE itself. Each stage of recovery of the abstraction levels and of the several relations among the implemented software components in SASHE correspond to the definition of the nodes and *links* which document it, in a hypertextual way, into SASHE itself.

This hypertext environment, SASHE, was developed in a project concluded previously. From the first experiments acquired with the creation of hyperdocuments for teaching, it was possible to observe that its characteristics compose a potentially useful resource for other types of applications, and not only for teaching [7], [8]. In this sense, the possible generalization to another application domain, the one of software reverse engineering, was our main motivation for the beginning of this work. In order to complement what is already being developed, the final stage of SASHE

[2] The user of hypertext systems can be reader and author.

system re-engineering will be accomplished, in order to make it suitable to the Software Engineering domain.

The experience of using SASHE in the documentation of SASHE itself, by means of the document registers regarding the reverse engineering process, to which it was submitted, was the fundamental exercise for the requirement investigation that a software engineer should have to use SASHE as a support tool to the documentation process of software reverse engineering. In a certain way, it provided an interactive evaluation of the results acquired during the reverse engineering accomplishment.

Regarding the reverse engineering process accomplished, it was chosen the use of the Fusion-RE/I method [9]. Such choice was mainly due to the fact that the practical experience was the reverse engineering application in a hypertext environment, and this application, in its turn, was registered in the hypertext environment itself. As the use of Fusion-RE/I initially considers the operationality of the user-computer interface of the software, an intense interaction for the accomplished "self-documentation" itself was possible.

Besides, the choice of the Fusion-RE/I method was done due to the fact this leads to the system vision production, that is, it helps in the understanding and recognition of its abstract model, under the Object Oriented (OO) paradigm.

When the reverse engineering task, based on the Fusion-RE/I method and in SASHE environment, was practically finished, to observe positive and critical points in the application of this method was possible, based on the experience acquired during the work performance.

In the following section the Fusion-RE/I method [9] and the products generated in each one of their stages will be presented. Section 3 will show a conceptual modeling of the reverse engineering information domain, having the premises as a basis of the Fusion-RE/I method. As the process final result is a hyperdocument, this modeling was made using the OOHDM method of hypermedia modeling [10], [11], [12]. In Section 4, the products generated during the reverse engineering application and also some relative metrics to this process will be presented. The considerations about the Fusion-RE/I method application, as a way of method evaluation, could be acquired and are described on Section 5. Finally, in Section 6 we will present the conclusions of this work.

2 The Method Used: Fusion-RE/I

The Fusion-RE/I method is a reverse engineering one which, aiming to make the process easier starts from the system interface to the retrieval of useful information for the software understanding. Through this method, the recovery of the system functional and structural visions is possible. The logical considerations are obtained by means of the interface analysis to the recovery of the system functional visions and later we start from these recovered visions and physical considerations acquired through the source code analysis to the recovery of the system structural visions [9]. The Fusion-RE/I method uses the models of the analysis phase of the software development method related to Fusion objects [13] in order to represent the retrieved information. Figure 1 shows a general vision of the method.

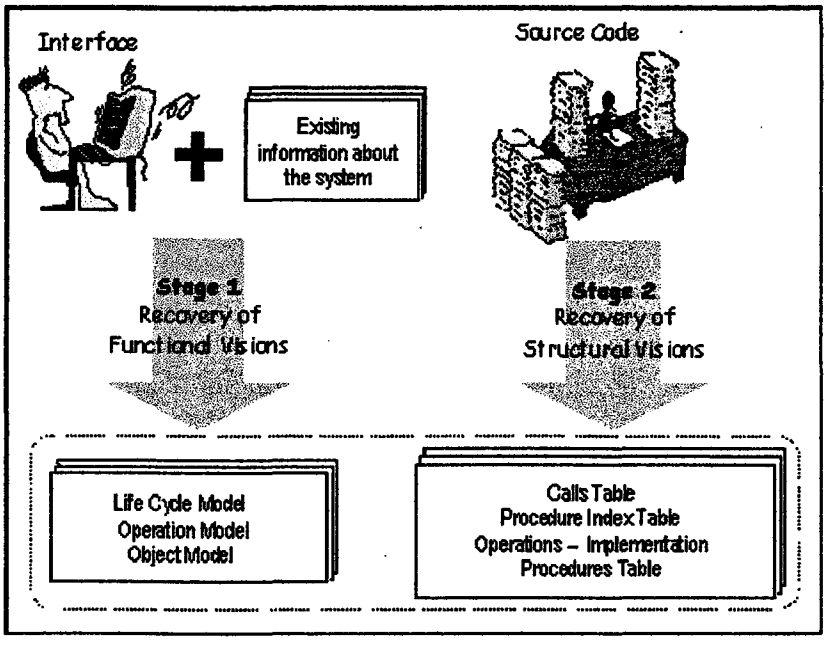

Figure 1. General Vision of the Fusion-RE/I Method

The Fusion-RE/I method arose from another reverse engineering method, the Fusion-RE [14]. The Fusion-RE/I method considers the same approach of its predecessor, the Fusion-RE, which consists of recovering the Fusion analysis models. However, this has distinct characteristics regarding the stages to be accomplished, and this process begins with the system interface analysis for the vision recovery [9]. Next, the two stages of the Fusion-RE/I method and its tasks will be briefly described. At the end of this section, a Table (Table 1), comprising a summary of the prescribed stages by this method, will be presented.

2.1 Recovery of Functional Visions – Stage 1

Aiming to acquire the abstraction of the system functionability, in the first stage of the Fusion-RE/I method, two tasks are accomplished: the acquirement of existing information about the system and the recovery of the system analysis model, which are described below.

Stage1-a. Acquirement of existing information about the system: all the available information about the system being studied is searched for. This involves grouping the existing documentation (manuals, code listing, etc.) about the system and also acquiring other relevant information, such as system domain, implementation language, etc.

Interviews with the user also may be useful since, most of the time, relevant information may be not documented.

After collecting all the existing information, this must be analysed for the identification of information related to the system requirements, for the architectural, data and procedural project, for the environment where the system is executed, for the organization of the file system, etc.

Stage1-b. Recover System Analysis Model: after acquiring all the existing information about the system, there is the second task, which is retrieving the information of the analysis phase. All the information retrieved in this stage is acquired through the investigation of the system interface.

The task of recovering the system analysis model includes, according to [9], the elaboration of the three models of the analysis phase of the Fusion method: the life cycle, operation and object models. It is assumed that this task demands a great effort, because of the system complexity. Next, the procedures for the elaboration of the analysis phase models will be briefly described.

Stage1-b1. Elaboration of the System Life Cycle Model: it is possible define the allowed operation sequence and the input and output events admitted by the system from the system use, the existing documentation study and the interviews with users.

The options presented at the system interface menu compose the main expression of the life cycle model. From the options listed in the main expression, new expressions are constructed, one for each option, showing the allowed operation sequences from that point. For each operation mentioned, a new expression is written identifying the allowed sequence of input events (operation elements) and the respective output events, which appear_preceded by the symbol #.

Stage1-b2. Elaboration of the System Operation Model: this stage starts from the life cycle model acquired previously. This model presents a general vision of the functionability of the system operations, which, for the operation model elaboration, must be further studied through the intensive use of the system, in order to be detailedly specified.

In this way, the execution of each operation is described, starting from the previous life cycle model. The Fusion-RE/I method considers not only the operations and events generated through the interface, but also operations and events, which are not visible in the screen, such as file creation and manipulation. This happens through the observation of the system "work directory" just after the execution of each interface operation, checking the modification or the creation of new files.

Once the operation is understood, its description form can be filled. In this form there are the following items: name, description, read, modify, send, assume and result.

The operation description can be acquired through the user manual. The information in the read item is acquired from the interface observation, identifying the value referent to each element accessed by the operation (input events). Through the interface observation and the operation execution, the information of the modify item is acquired, identifying the value referent to each element accessed and modified by the operation. The information of the send item is acquired through the interface observation, identifying the output events generated by the operation. The information of the assumed item specifies the necessary preconditions for the operation execution, and it is also acquired by the interface observation. The information of the result item can be filled identifying, through operation execution, the relationship between the initial state of the system (before the operation) and the final one (after the operation).

Stage1-b3. Elaboration of the System Object Model: subjects related to the system functionability are defined firstly in the Fusion-RE/I. This subjects are named (*Themes*). To define the themes, an analysis of the information recovered in the previous stages and of the abstractions registered in the operations and life cycle

models are necessary. This is one of the most subjective tasks of the Fusion-RE/I method and has fundamental importance in the method application [9].

Just after defining the themes, the operations are grouped according to the themes to which they refer. Thus, at the end of it there are a list of themes and the theme operations. Themes can be composed of other themes since the subjects are related in a higher level of abstraction.

For each one of the defined themes, an object model is constructed. To accomplish this, the operations are analysed again, searching now to identify components, which compose the object model, that is, classes, relationships, attributes and possible aggregations, specialization and generalizations.

The object model components may include components which are not explicit in the operations, but which are identified by the abstraction and understanding of the functionability of each element and operation.

The construction of the object model is a very subjective task, and even though this is not part of the method, it will probably demand a feedback, after the recovery of the structural visions (code visualization). The necessity of this feedback was so evident during the Fusion-RE/I use, since the main references about this method of reverse engineering do not furnish just one line of direction for the elaboration of the object method. To identify which moment is the best for its elaboration an important investigation was done during the method application. In [9], the Fusion-RE/I is described, but there is not a clear definition of the object model that can be constructed only through the interface observation.

2.2 Recovery of Structural Visions – Stage 2

In this stage, we work with the system source code, aiming to identify the procedures, which insert the system operations specified in the previous stage. Next, the procedures, which must be accomplished in this stage, will be described.

Stage2-a Elaboration of Table of Implementation Procedures: this stage aims to identify each procedure, its functionability and the calling sequence of it. For this, two tables are used: one call table for each system program file and a general index of procedures.

Stage2-a1. Elaboration of the Calls Table: this table must be elaborated for each file of system source code, presenting the procedures comprised in the file, their respective functionability and the used procedure (called by it) and the caller ones (the procedures which call it).

The information of the procedure description functionability can be acquired through the comments in the source code. The used procedures are acquired by the analysis of the code itself, while the caller procedures (procedures which call it) are acquired at same time that the tables are elaborated.

In the end, the procedures of each table are rearranged in alphabetical arrangement.

Stage2-a2. Elaboration of the Procedures Index Table: this table presents all the procedures of the system implementation in alphabetical arrangement, with the respective locations (file and directory). For the elaboration of this table, all the call tables acquired previously are used.

Stage2-b. Elaboration of Operations – Procedures Implementation Table: the procedures which implement the interface operations are identified and, according to their functionability, the procedures are allocated to the interface or to one of the themes defined previously. Then, the *links* among the documents of the "Operation Tables", from the first stage of the method, are identified with the respective codes which implement them.

In the first columns of the table, the menu options and the operations of each option (interface description) are inserted. In the next column, the procedures, which implement each operation, are inserted, according to the call hierarchy described in the call table. The following columns of the table are used to allocate the procedures to the interface or to one of the defined themes. Thus, there is a column for "interface" and another one for each defined theme.

The depth level in which the procedures are detailed in this table depends on the interest [9]. Next, the tasks prescribed for each stage of the method are summarized in the Table 1.

Table 1. Order of the tasks prescribed by the Fusion-RE/I

Stage 1. Recovery of Functional Visions
a. To obtain the existing information about the system
b. Analysis model recover
b1. Life cycle model elaboration
b2. Operation model elaboration
b3. Object model elaboration
Stage 2. Recovery of Structural Visions
a. Implementation procedures table elaboration
a1. Calls table elaboration
a2. Procedures index table elaboration
b. Operations – implementation procedures table elaboration

3 Modeling of the Support Hyperdocument to the Fusion-RE/I

According to the previous section, the reverse engineering process executed in SASHE system composes the exercise to reveal the requirements for the definition of a suitable hyperdocument structure to the reverse engineering. As starting point for the hyperdocument construction, we consider necessary the conceptual modeling of the reverse engineering process through a conceptual modeling method. For that, we choose the OOHDM (*Object Oriented Hypermedia Design Methodology*) [10], [11], [12].

The conceptual or domain modeling is needed for the understanding of the problem domain and for the construction of suitable models from this domain [12], in the case of the OOHDM, identifying objects and organizing them through the identified relationships. Thus, we have as a result, beyond the conceptual modeling of the target domain as a whole, the definition of which objects must be really navigated

in the hyperdocument, and how these objects can be navigated. This defines which links must exist as well as the contexts in which each object can be presented.

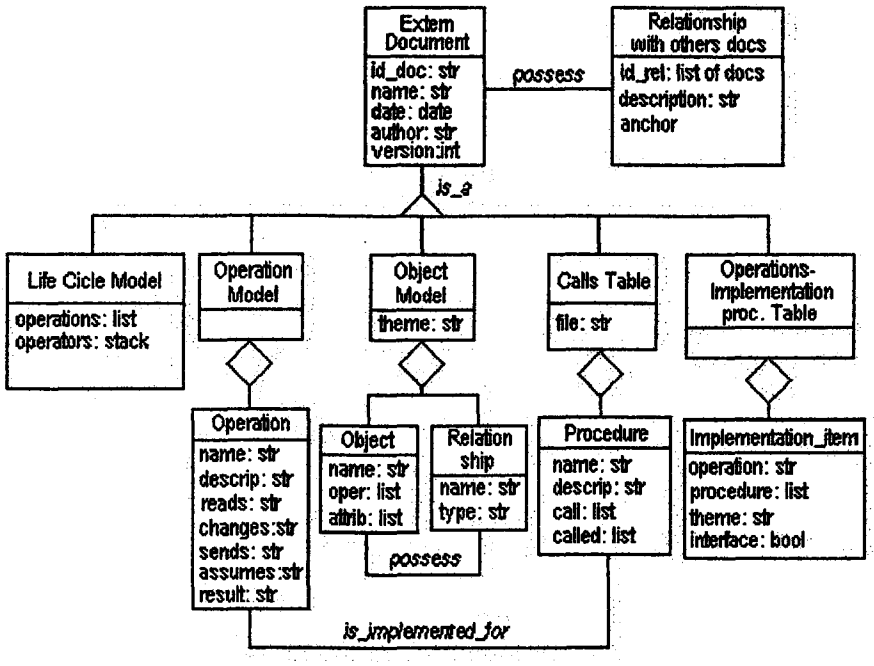

Figure 2. Partial conceptual schema of the reverse engineering process prescribed by the Fusion-RE/I

The process prescribed by the Fusion-RE/I method (Table 1) was the basis for the establishment of the activities involved and of the models generated in the end of the reverse engineering process. All this is reflected in the modeling final result, which is composed by a conceptual schema (object model), a navigational class schema and a context schema [15]. Figure 2 presents part of the conceptual schema of the Fusion-RE/I process, in which the classes of documents generated as result of the relationships among them can be seen. The complete OOHDM modeling of the Fusion-RE/I process is described in [24].

Each process described in the modeling was instanced with the accomplishment of the reverse engineering process. In fact, the results presented in the Section 4, next, are instances of objects of the classes identified in this schema (Figure 2).

4 Results Acquired in the Fusion-RE/I Use

In this section, the products resulting of the Fusion-RE/I method application to SASHE system and also some measures related to the products and the developed process are presented. All the tasks, except the first one (acquirement of existing information about the system), have a document elaboration as final result.

As described in Section 2, the first task to be accomplished is the existing information and documentation collection about the system in question (Stage1-a). In the specific case of SASHE system, all the available information about the system is summarized in two technical reports [6], [16]. Relevant information was acquired by

means of interviews with a person who had participated in the system development and with the system users involved in other projects related to SASHE.

The sequence of operations allowed and the input and output events accepted by the system were defined from the intensive use of the system, the existing documentation study and the interviews with users (Stage1-b1). The life cycle model constructed for SASHE resulted in almost fifty-five (55) sentences consisting of operations allowed in the system, including both modules, which compose the software (Teacher and Student).

In order to make the life cycle sentences checking easier, "State Transition Diagrams" (STD) for each sentence was constructed. The STD set allowed the checking of the life cycle sentences correctitude, since it made the visualization of each interaction clear. The STD was made in a support tool to the software modeling, the Rational Rose [17]. In Figure 3, a small sample (4 sentences) of the life cycle model constructed is shown.

```
life cycle SASHE = ( PROFESSOR | ESTUDANTE | Saída )

PROFESSOR = ( HIPERBASE | JANELAS | AJUDA | b_Criar_hiperbase |
b_ABRIR_HIPERBASE | b_SALVAR_HIPERBASE | b_ROTEIRO | b_GLOSSÁRIO | ELOS
| NÓS )+

HIPERBASE = ( ((Criar | ABRIR) . SALVAR) | Sair )

ABRIR = ( (Nome_do_arquivo . (Pastas | Unidades_de_disco |
Listar_arquivos_do_tipo | Som._Leitura | b_REDE | b_AJUDA)* . ( (b_Ok .
(#msg_não_é_possível_encontrar_este_arquivo...). b_Ok . ABRIR) | (b_OK
. (#msg_este_nome_de_arquivo_tem_muitas_letras) . b_Ok . ABRIR) | b_Ok
| b_Cancelar ) ) | b_Cancelar )
                                    . . .
```

Figure 3. Part of the Life Cycle Model constructed for SASHE system

The high number of sentences of this model indicates the great software interaction and the interface styles (menus, buttons, multiple windows) implemented in the system.

Operation:	Open (**ABRIR**)
Description:	Open an existing hyperbase
Reads:	File name; directory; drive; "read only"; ok button; cancel button
Changes:	
Sends:	Extern agent:{#msg file not found}; Extern agent:{#msg this file name has too many letters}
Assumes:	
Result:	If cancel button, then the operation is cancelled If ok button, then If the file exists, the hyperbase is openned If file not found, then a message of "file not found" is sent, and the operation Open goes on. If the file name has more than 8 letters, a message of "the name has too many letters" is sent, and the operation Open goes on.

Figure 4. Description of the "Open" Operation

The operation model is arising out of the life cycle model. Each operation identified in the life cycle model is individually described through a textual form,

according to what was described in Section 2. Sixty-two (62) operations were identified in the life cycle model constructed for SASHE system, and each one of them were exhaustingly studied in order to be described according to the requirements imposed by the method (Stage1-b2). All the filled forms compose the operation model. Figure 4 presents the description (one of the forms) done for the "Open" operation in the "Hyperbase" menu.

For the elaboration of the object model (Stage1-b3), firstly the topics related to the system functionability were defined: they were named *Themes*. In order to be possible the theme definition, an analysis of the information recovered in the first task (Stage1-a) was necessary. Analysis of the abstractions came from the operation and life cycle models were also necessary. In fact, we proved the subjectivity of the theme definition, what led us to many tries of possible themes, before we decide which would be the definitive themes of the system.

The first try of grouping of the operations identified in the proposed themes revealed that those themes defined were not suitable, since some operations did not fit in any themes and some themes did not comprise any operation. Besides, the themes defined did not evolve to the object models. Thus, through the careful analysis of the operations and its relationships, two themes were proposed for the system: *Authorship* and *Navigation*, which seemed suitable to us. Then, a new grouping of all the operations identified in the operation model in some of these themes was done. The operations were analysed again, aiming to identify components, which compose the object model, that is, classes, relationships, attributes, and possible aggregations, specialization and generalizations. In this way, two object models were constructed, one for the *Authorship* theme and another to the *Navigation* one.

The object model for the *Authorship* theme can be seen in Figure 5. It is important to emphasize the models were proposed based only on the software functional vision, reached through the interaction exhaustive with the software. The object models were constructed in the Rational Rose tool [17], as well as the STD.

With the end of the recovery of the life cycle, operation and object models, the first stage of the Fusion-RE/I method, which recover software functional visions only through the interactions with the information about the system and its interface, is finished. The conclusion of this stage of the method occurred about in the end of 2 months.

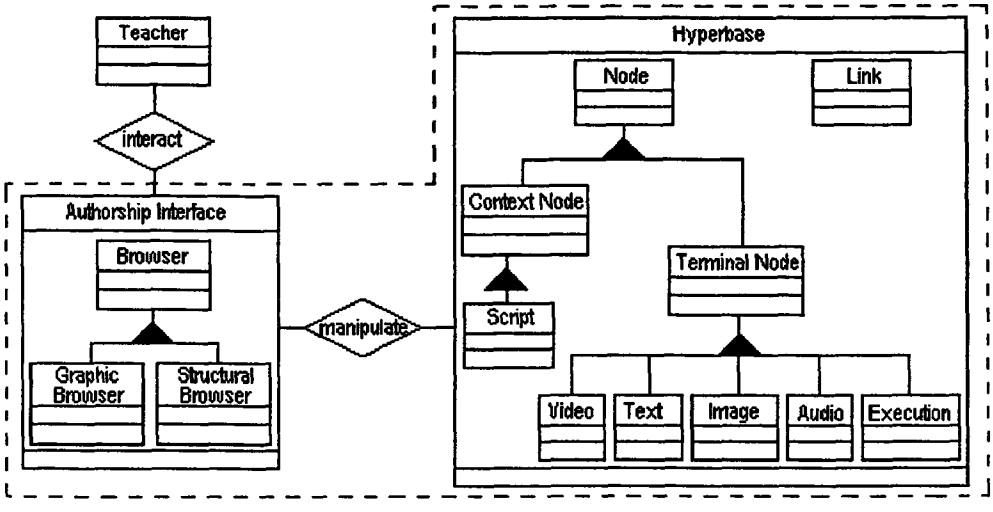

Figure 5. Object model simplified for the *Authorship* theme

This first stage of the Fusion-RE/I method deals with the target system as "black box". In this way, the register support in hyperdocument to track the functional vision evolving revealed extremely important.

The second stage, the stage of retrieval of the structural information through the source code analysis of the software characterized SASHE system with almost ten thousands lines of C++ code, implementing the hyperbase manager (71,5% of the code), and almost four thousands lines of Delphi code, implementing the system interface (28,5% of the code). The interface between the hyperbase manager and the system interface occurs through a DLL, which exports the hyperbase manipulation functions to the interface. According to what was specified by the method and described in Section 2, a call table for each source code file of the system (Stage2-a1) was elaborated. All the tables defined for the C++ code files totalized seven hundred and twenty-two (722) procedures registered, including prototypes and implemented functions, but for one hundred and seventy (170) procedures (23,5%), there were no calls which actived them. This high number of unused procedures occur because some objects implement methods which are not used by the system, however, they complement the concept represented by the object. There are also procedures implemented in previous releases that, even without are used in the analysed release, were not retired from the code by the mantainings.

On the whole, forty-seven (47) classes and five hundred and twenty-nine (529) functions (without the prototypes) were analysed and codified in eighty-four (84) C++ code files. The construction of this table demanded almost one month and half of work only for the C++ files. Part of this task was helped by the use of the *Understand for C/C++* tool [18], of which a demo copy was used, available via Internet.

Regarding the procedures in Delphi, one hundred and seventy-five (165) procedures, defined in twenty-one (21) files. Great part of the procedures analysed did not present any call in the code. Due to the language paradigm, orientated to events, it was already expected that not many calls to the procedures implemented in Delphi were found. The time spent in the analysis of these files was about ten days.

Approximately, five days were necessary for the construction of the procedure index table (Stage2-a2), with the procedures in C++ and Delphi.

Finally, the conclusion of the elaboration of the implementation operation-procedure table, though it is not totally finished yet (Stage2-b), is foreseen for around three weeks. The detail level of this table can be arranged according to the reverse engineering purpose. In this work, the documentation recovered will be used in the system re-engineering for changing the application domain. As the main focus of this re-engineering concentrates on the system interface, the implementation operation-procedure table will be detailed until the level of the operation calls exported by DLL.

5 Evaluation of the Fusion-RE/I

As premise of the Fusion-RE/I, in order to the software engineer can recover the analysis models of a system, first it is necessary to group information about it. The existing documentation, the implementation language, the data acquired by means of interviews with users and the system domain are valuable information sources [19]. From among these, the system domain deserves special attention because it is a very subjective aspect.

The system domain can be understood as a knowledge area, to which a system belongs, that can vary a lot. Certainly, the domain has influence on the investigation information about a determinate system. Since the software engineer has knowledge (familiarity) of the domain to which the system belongs, the chance of its being better understood is bigger. Consequently, the chance of acquiring a better analysis of this is also bigger.

In this way, the familiarity with the system domain has influence on the construction of a mental model of the system. The mental model allows the software engineer to choose a particular way of thought about a problem and to generate a representation of this problem. The more familiar the domain is to the software engineer, better the system representation will be and less errors will appear later [20]. Research already indicated the engineer experience and familiarity in a software domain has directly influence on the stages of the software life cycle [20], [21]. Under the reverse engineering perspective, DeBaud [22] accomplished case studies, in which describes how the domain knowledge can help a reverse engineering process and how the reverse engineering can be used in order to construct a better model of domain of an application.

When a system belongs to a determinate domain, its interface can be mentioned as the element, which better makes clear this domain. It is a long time the interface has been not only one more element of the system; through it the process set, dialogues and actions between users and computer occur, in a way to result in objective mechanisms to acquire the desired functionality.

In the Fusion-RE/I, interaction with the interface is the initial goal to the reverse engineering process. An exhaustive interaction with the interface is accomplished, making the software engineer familiar with the system functionality and operational characteristics. In this way, the Fusion-RE/I makes possible an effective gain of familiarity with the domain.

Although the recovery of the analysis models deals with a task linked to the system manipulation, by means of its interface and in a very intensive way, being in this way a "concrete" task. Great attention and concentration effort on the part of the software engineer (or maintaining) is required, for investigate almost exhaustively all the system interface states.

The operation life cycle models, in spite of their acquirement require great effort, furnish a complete knowledge of the system functionality, revealing the command sequences allowed and the system operation description in an objective and clear way.

A critical point of the method is the recovery of the object model (Stage1-b3). The previous case study accomplished with the Fusion-RE/I [9], [23] was applied to systems not orientated to objects, and with an interface style different from that used in SASHE. The fact of the SASHE system's being orientated to objects allowed us to analyse a series of factors regarding Stage1-b3.

The Fusion-RE/I method prescribes an action sequence for the recovery of the object models – theme definition and elaboration of an object model for each defined theme. The method also prescribes that this task must be accomplished during the recovery of the functional visions, that is, without the analysis of the software code.

A first question to be analysed is the accomplishment validity of this task (Stage1-b3) during the first stage of the method (Recovery of the Functional Visions). In systems whose code is not under the OO paradigm, it makes sense to construct an abstraction of this, by means of object models, through the analysis of their

functionality and performance domain, without the code implemented intervene in the modeling. On the other hand, when the software is implemented under the OO paradigm, the object model structure already exists and is instanced in the software code. Thus, the construction of an object model based on the system functional visions is not valid anymore. The recovery of the object model, without any doubts, is a fundamental task in the software documentation, however, for systems orientated to objects, the accomplishment of this task was considered more suitable in the beginning of the second stage of the method (Recovery of Structural Visions).

In fact, in SASHE experience, when beginning the second stage, the object models had to be fit to the classes implemented.

Other question is the validity of the theme definition for the recovery of the object models for OO systems. From the moment the object model is not an abstraction created from the software functional visions and starts to register its real structure, the theme definition lose the meaning. In this case, the utility of the themes also for OO systems can be discussed, but not in the way proposed by the Fusion-RE/I. A possibility of application of the theme definition would be its use for the organization of the object model extracted from the code. Bigger systems can require that the recovered code structure be divided in many object models. However, in this case, the themes should fit themselves to the implementation, since the model implemented cannot be changed in order to fit itself to the themes proposed according to the functionality observed. Thus, also the theme definition in the second stage of the method would be more suitable.

The tasks for the recovery of the structural visions, second stage of the method, were suitable, however, very difficult, as checked through the numbers presented in Section 4. It was proved that part of these tasks could be successfully automated.
According to the Fusion-RE/I, the first task of structural recovery is the Call Table elaboration (Stage2-a1). Part of the elaboration of this table was made automatically, with the help of the *Understand for C++* tool [18], according to what was written in Section 4. However, the procedure description had to be made manually, what demanded hard work for its conclusion. The information described in this table is very significant, since it is possible through this table to track the execution of each procedure, from the moment in which is called (*procedure which call it*) to the procedures which are released in the procedure body (*Call*). Besides, the understanding of the functionality of each procedure is presented in the *Description* item.

Regarding the Index Table (Stage2-a2), its generation can be totally automated, since this table is a data list of all the procedures and their respective locations (file/directory). In this work, this table becomes especially useful, since it will be used in the construction of an index page of the hyperbase comprising links for access to the procedures from other table.

The Implementation Operation-Procedure Table (Stage2-b) finishes the structural documentation recovered according to the Fusion-RE/I method. This table complement the information presented in the call table, relating to each operation identified in the interface the procedures responsible for its implementation. A positive point of the method is the flexibility of detailing the procedures related to each operation in a deepness level, which be according to the reverse engineering final aim, as shown in Section 4.

A point to be discussed in the elaboration of this table (Stage2-b) is the procedure classification in one of the defined themes in the first stage of the method, or as an interface implementation. Again we enter into the question of validity of the theme elaboration for the systems orientated to objects. In the case of systems implemented under other paradigms, the procedure classification in the defined themes is important to have an idea of where (in which class) each procedure would be better implemented. This would be especially useful in case of a re-engineering for changing the paradigm (not OO to OO). Since the code is orientated to objects, this classification is not necessary anymore, because each procedure is already implemented as method in a determinate class, remaining valid just the procedure classification which implement the interface interactions relating to each operation.

In the following section, the conclusions of this work will be presented, emphasizing the results acquired with its development

6 Conclusions

The reverse engineering aims specifically to the construction of some kind of representation, more abstract in general, acquired from an implemented system. In this work, a reverse engineering method, named Fusion-RE/I, was used with the aim of recovering information of a hypermedia system project.

The method described in Section 2 was carefully studied and its process, modeled according to OOHDM. A relevant aspect was that the partial results (documents recovered) of the reverse engineering process were registered in the hypermedia system, to which was submitted the reverse engineering. As result of this experiment, several aspects of the Fusion-RE/I method evaluation were made clear. Among them, we can point out:

- Effective support to the tasks of the functional visions and system structural recovery, with well defined and aimful procedures;
- The suitable initial stage, interaction stage with the system interface, whose goal is furnish knowledge of the system functionality and operational characteristics, and that, besides, provides an effective familiarity with the domain, making a better understanding for the analysis model elaboration viable;
- The question of the accomplishment of the object model recovery in which, regarding a system implemented under the Object Oriented (OO) paradigm, seemed to be more suitable in the beginning of the structural vision recovery (Stage2);
- Regarding a system implemented under the Object Oriented (OO) paradigm, the object model is not an abstraction created from the software functional visions anymore and starts to register its real structure. In this way, the theme definition looses its meaning. A new theme utility for OO systems can be visualized: its definition to help the object model organization extracted from the code itself. Very large systems may require the structure recovered of the code be divided in many object models.

Thus, the evaluation points of the Fusion-RE/I method, applied to an OO system, provide a simple arrangement in the order of some tasks accomplishment. The Fusion-RE/I has not been proposed for being applied to systems originally implemented under

the OO paradigm. In this way, the method flexibility and its convenience to consider this approach is observed.

The work described in this article will be continued with the goal of complete the re-engineering of SASHE system, and adapt it to the Software Engineering domain.

References

1. Saleh, K., Boujarwah, A.: Communications Software Reverse Engineering: A Semi-Automatic approach. Information and Software Technology, n.38. Oxford (1996) 379-390
2. Schach, S.R.: The Economic Impact of Reuse on Maintanance. Journal Software Maintanance: Research and Practice, Vol.6, n.4. (1994) 185-196
3. Schneidewind, N.F.: The State of Software Maitanance. IEEE Trans. on Software Engineering, Vol.13, n.3. (1987) 303-310
4. Costa, R., Iavanroni, G.M., Sanches, R., Maldonado, J.C., Masiero, P.C.: Engenharia Reversa: da Interface do Software aos Modelos de Análise do Método Fusion. In: Simposio en Orientación a Objetos, *Anales*. Buenos Aires (1998) 133-146
5. Meira, S.M., Cabral, R.S.: Id: Um Sistema de Hipertexto Configurável e Orientado a Objetos para Integrar Documentos de Software. In: VIII Simpósio Brasileiro de Engenharia de Software, *Anais*. Curitiba (1994) 283-296
6. Nunes, M.G.V., Hasegawa, R., Vieira, F.M.C., Santos, G.H.R., Fortes, R.P.M.: SASHE: Sistema de Autoria e Suporte Hipermídia para Ensino. Notas, n.33. ICMC–USP. São Carlos (1997)
7. Nunes, M.G.V., Fortes, R.P.M., Nicoletti, M.C.: Flexible Guided-tours in Hypertexts: a way of controlling the user in learning applications. In: World Multiconference on Systemics, Cybernetics and Informatics SCI'97/ISAS'97, *Proceedings*. Caracas (1997)
8. Nunes, M.G.V., Fortes, R.P.M.: Roteiros em Aplicações no Ensino: a Questão do Controle do Leitor. In: III Workshop em Sistemas Multimídia e Hipermídia, *Anais*. São Carlos (1997) 15-27
9. Costa, R.M.: Um método de Engenharia Reversa para Auxiliar a Manutenção de Software. Dissertação de mestrado, ICMC–USP. São Carlos (1997)
10. Schwabe, D., Rossi, G.: The Object-Oriented Hypermedia Design Model. Communications of the ACM, Vol.38, n.8. (1995) 45-46
11. Schwabe, D., Rossi, G., Barbosa, S.D.J.: Systematic Hypermedia Application Design with OOHDM. In: Hypertext'96, Washington DC. *Proceedings*. ACM Press, New York (1996) 116-128
12. Rossi, G.: Um Método Orientado a Objetos para o Projeto de Aplicações Hipermídia. Tese de Doutorado. Departamento de Informática Pontificia Universidade Católica. Rio de Janeiro (1996)
13. Coleman, D. et al.: Desenvolvimento Orientado a Objetos: O Método Fusion. Ed. Campos, Rio de Janeiro (1996)
14. Penteado, R.A.D.: Um método para Engenharia Reversa Orientado a Objetos. Tese de Doutorado, IFSC–USP. São Carlos (1996)

15. Feltrim, V.D., Fortes, R.P.M.: Requisitos de Hiperdocumentos de suporte ao domínio de Engenharia Reversa de Software. In: Workshop de Engenharia de Requisitos, *Anais*. Maringá (1998) 159-167
16. Nunes, M.G.V., Hasegawa, R., Vieira, F.M.C.: Hip/Windows: Um Ambiente de Autoria de Hiperbases Multimídia. Relatório Técnico, n.38, ICMC–USP. São Carlos (1996)
17. Rational Corporation. URL: http://www.rational.com (1999)
18. STI - Scientific Toolworks, Inc. URL http://www.scitools.com (1999)
19. Biggerstaff, T.: Design Recovery for Maintenance and Reuse. IEEE Computer, Vol.22, n.7. (1989) 36-49
20. Adelson, B., Soloway, E.: The Role of Domain Experience in Software Design. IEEE Transactions on Software Engineering, Vol.SE-11, n.11. (1985) 1351-1360
21. Greenbaum, J., Kyng M. (eds): Design at work: Cooperative of Computer Systems. Lawrence Erlbaum, Hillsdale, NJ (1991)
22. DeBaud, J-M., Moopen, B., Rugaber, S.: Domain Analysis and Reverse Engineering. In: International Conference on Software Maintenance. *Proceedings*. Victoria (1994) 326-335
23. Quinaia, M.A.: Diretrizes para Reengenharia de Software com Características de Software Legado. Dissertação de Mestrado, ICMC–USP. São Carlos (1998)
24. Feltrim, V.D., Fortes, R.P.M.: Uma modelagem do domínio de Engenharia Reversa de Software utilizando o método OOHDM. Notas, n. 40, ICMC–USP. São Carlos (1998)

Resumo

O crescimento do mercado de software a cada dia acarreta o aumento do uso de técnicas de desenvolvimento, muitas vezes informais. A manutenção de tais softwares torna-se problemática, uma vez que a documentação associada ao software, na maioria das vezes, não está de acordo com o código implementado. Nesse contexto atua a Engenharia Reversa de Software, com o propósito de recuperar as informações de projeto perdidas durante a fase de desenvolvimento, e de documentar o real estado do software. Este artigo relata as questões envolvidas durante a aplicação do método de engenharia reversa Fusion-RE/I. O experimento descrito é parte da re-engenharia de um sistema protótipo de hipermídia, cujo objetivo é o de adaptá-lo ao domínio de Engenharia de Software. Em função de o sistema alvo ser hipermídia, os resultados obtidos durante a aplicação do método Fusion-RE/I puderam ser registrados como um hiperdocumento no próprio sistema submetido à engenharia reversa. Foi então possível observar aspectos relevantes sobre a validação das etapas propostas no método Fusion-RE/I.

Palavras-chave: Avaliação de um método de engenharia reversa; Suporte hipermídia à engenharia reversa de software

NOTAS DO ICMC

SÉRIE COMPUTAÇÃO

043/99 PANSANATO, L.T.E.; NUNES, M.G.V. – EHDM: Método para projeto de hiperdocumentos para ensino.

042/99 MORABITO, R.; ARENALES, M. – Optimizing the cutting of stock plates in a furniture company.

041/98 SANTOS-MEZA, E.; SANTOS, M.O.; ARENALES, M.N. – Lot sizing and scheduling in na automated foundry.

040/98 FELTRIM, V.D.; FORTES, R.P.M. – Uma modelagem do domínio de engenharia reversa de software utilizando o método OOHDM.

039/98 FERREIRA, V.G.; MELLO, O .D.; OLIVEIRA, J.N.; FORTUNA, A . O . - Tópicos teóricos e computacionais em escoamentos de fluidos.

038/98 CAVICHIA, M.C.; ARENALES, M.N. - Piecewise linear programming via interior points.

037/98 REZENDE, S. O. ; PUGLIESI, J.B. - Aquisição de conhecimento explícito ou manual - versão 1.0

036/97 SOSSOLOTE, C.R.C; RINO, L.H.M; ZAVAGLIA, C.; NUNES, M.G.V. - As manifestações morfossintáticas da linguagem UNL no português do Brasil.

035/97 FRANCO, N.B. - A volterra integral equation arising from the propagation of non-linear waves.

034/97 MARTINS, A.L.; OLIVEIRA, M.C.F.; MINGHIM, R. - Visualização Científica em mecânica dos fluidos.