Documentation Process in Interactive Systems –
A Case Study to Abstract its Structure

Lisandra C. Fumagalli, Luciano T. E. Pansanato, and Renata P. M. Fortes

Abstract-- The difficulties on leading with a variety of documents during software processes by teams of software developers have been addressed in relevant researches, and Web technologies have emerged bringing new solutions that must be considered. In this context, we report a case study regarding to the process of analyzing and structuring documentation produced by 20 student groups who have developed different projects as a task for an HCI (Human-Computer Interaction) course. Consequently, a DTD could be defined. The result, as a structured set of documents, has being re-written according to XML-DTD specification to be available for improved searching and ex-changing tasks, commonly performed by software developers. Finally, the process of abstracting such structure based on existing documents was concluded to be an interactive activity, corresponding to alternate between bottom-up and top-down approaches.


I. INTRODUCTION

The documentation process is performed simultaneously during the several phases of software development. It assists the project communication and afterwards facilitates the software understanding for future maintenance [10] or reuse [18]. As the documentation process is a flexible activity, documents in different formats, structures and types of content can be produced. However, the whole documentation process is not an easy task. Developers have to deal with an enormous diversity of documents that are different in formats and contents and they usually complain that searching through these documents is too difficult and usually takes a lot of time and attention from the people involved.

Forward [9] has investigated how the software documentation is used in a software project, and has confirmed that software documents favorably contribute for it. In addition, he has also studied how the technologies can improve the use and the utility of such documentation. However, in his work, we cannot observe a concern with a support to allow the exchange of documents during the software documentation process. In this context, we advocate that the Web technology provides an appropriate infrastructure since it incorporates a markup or structuring language for documents, namely XML application [23]. XML tags facilitate the identification of information in the document by an application (parser). By using XML tags it is possible to develop search tools and exchange mechanisms to improve information retrieval for maintenance and reuse of the documents [24].

To recover information from XML documents, they should be structured. The documents can be structured during their production or re-written to conform to a structure. In case of pre-existing documents, literature just offers general guidelines for document analysis and heuristics for document structure definition. It lacks a process document structuring process to address the analysis issues of extensive sets of documents and the involvement of several people in this task. The partial automation of the activities should be also considered to assist ad hoc work.

In this paper we report the process to analyze and to structure documentation produced by 20 groups of students who have developed different projects as a task for an HCI (Human-Computer Interaction) course, given at the Institute of Mathematics Sciences and Computing of the University of São Paulo in São Carlos.

This article is organized as follows: in Section II is presented the work related to the software documentation process, reported in the literature, and which has been essential for the accomplishment of the case study we carried through. In Section III the case study is presented, including the characteristics and volume of the existing information investigated from the registered documents. In Section IV, the process to abstract the HCI documentation structure is presented. Section V briefly describes the related works and how automated tools could assist some process activities. Finally, in Section VI the conclusions of this work are presented as well as the research in progress for recovering of information contained in documentation structured by the presented process.

II. SOFTWARE DOCUMENTATION PROCESS

The software documentation process is indispensable during the software development. The documentation’s role is concerned to register the system evolution, providing information and helping to the use and maintenance of the systems.
Beyond these benefits, the documentation can also facilitate system reuse [18].

Results reported by Forward and Lethbridge [10] show that software engineers consider the documentation important even if it is not up to date (but keeping it up to date is an objective). Developers also consider documents as an important tool for communication, and these documents always have to be used to achieve this purpose.

Although documentation is believed essential during the software development, software engineers consider it an onerous activity. Such difficulty can contribute for an inexact, incomplete, outdated or even inexistent documentation, and this fact can imply in a possible lack of quality of the software, error generation during its development, and an increasing difficulty in its maintenance [6].

Forward [9] has investigated how the documentation is used in a software project, and has confirmed that software documents favorably contribute for the project. In addition, he has also studied how technologies can improve the use and utility of such documentation. However, in his work, we cannot observe a concern with support to allow exchange of documents during the software documentation process. We believe that supplying a structure accordingly to a well-defined and significant markup language to support the document production, along with the use of Web infrastructure, can improve information retrieval for maintenance and reuse of documents.

Since documentation process is an important topic in our work, it has been studied as a basis to establish an appropriate approach consistent with the state of art on software documentation, and to supply an adequate support to a well-defined structuring of such Web documents.

A. Activities of the Software Documentation Process

In the software development process, documentation must reflect and register information related to specification, design, implementation, verification and installation of the software [11]. The ISO-12207 norm [15] establishes four activities for the Software Documentation Process:

1) Process Implementation: the objective of this activity is to study the documentation requirements for the project to be developed, to establish the type of document to be elaborated, as well the technology to be used, the sequence of operations to produce and distribute the documents, and the document language used to index and reference them.

2) Design and Development: the intention is to describe all the aspects demanded for the production of each individual document. Each document, according to its design, must present standards of documentation related to format, content and presentation, among others.

3) Production: in this activity, all the documents are produced. These documents must be produced according to the design defined in the previous activity. The Production activity also includes the preparation of the environment of production and the review, identification and release of documents.

4) Maintenance: this activity deals with the tasks related to the controlled introduction of changes in documents, obsolete document deletion and document storage in files.

Although these activities are established by the ISO-12207 norm, factors as high cost, inaccuracy, and difficulty of manipulation make documents’ creation and updating difficult [4]. These combined factors inevitably produce incomplete, inconsistent and expensive documents [17].

In this context, an alternative to reduce the effect of these factors is to use a well-defined structure to produce the software documents. In fact, many benefits can be obtained from well elaborated documents: less time and effort to the software development, an easier and more efficient use of the software by users, more explicit structures of software, facility to locate and understand the information registered [19].

To identify a structure of common HCI design documents, we have studied the particularities of them. A brief description of the HCI design documentation, used in this work, is presented in the next subsection. In special, the iteration feature is focused during the documentation process of such projects.

B. HCI Design Documentation

To sum up, HCI objectives are developing and improving security, utility, effectiveness, efficiency and usability of all systems. Usability, a keyword to HCI, is concerned in making the systems easier to learn and use [20].

The development of an interactive system is not a trivial task. To be successful, not only should the system be implemented in an effective way but also it must be accepted for a user population. Preece [20] describes the Star model (Fig. 1) to highlight a number of important points for the interactive system development. First, ordering the activities is inappropriate; the development may begin at any stage and may be followed by any of the other stages. The Star model was derived following extensive analysis of actual design practice among HCI designers. Second, the model stresses rapid prototyping and an incremental development of the final product. Third, the model distinguishes between conceptual design and physical design (formal design); conceptual design concerns itself with questions of what is required, while the physical design is concerned with questions of how the requirements can be achieved. Finally, the central activity of the Star model is evaluation; all the aspects of the development are target of constant evaluation by users and designers.

As important as evaluation, the following activities are also considered by Preece [20]: user, work, task and environment analyses; technical analysis; requirement specification; design
and design representation; prototyping and use of other design support tools and techniques; coding and implementation.

Along with the activities of the development process, the representations produced in these activities are also fundamental. Different models (formal and informal) and strategies (checklist, rules and guidelines) are used according to the activity. The resultant products of these representations constitute the project documentation. This documentation, however, differs from the documentation of conventional software systems because the activities of development process of conventional systems and interactive systems also differ [7]. A possible explanation to this difference is the fact that users and usability are essential topics for interactive systems and it is not so relevant in conventional systems [7]. Also, as noted by Preece [20], interactive system documentation design must be flexible to attend the lack of ordering of its activities.

In order to get a well-defined document structure, we have considered document engineering and studied the importance of documentation analysis.

C. Document Engineering

The term document engineering reflects an emerging analogy between issues of digital document creation, development and maintenance, and issues of software development. It has been observed an increasing interest among the digital community in the application of techniques and approaches, which have been proved successful in large software systems to the area of digital documents [28]. In this way, an adequate document structuring is a requirement and the Web is an appropriate environment to make the access to these documents available.

Glushko and McGrath [27] have established a document engineering roadmap by reshaping the classical analyze-design-refine modelling methodology to better fit the domain of documents (Fig. 2). One of the tasks of this roadmap is the documentation analysis. It is a way of removing redundant processes or data, standardizing on one process or rationalizing a document structure. Besides that, document analysis is often conducted with the goal of abstracting a logical model from heterogeneous instances and encoding it as a SGML or XML schema. The design of document structure specification is very rigorous in document engineering. Well-engineered document structures have clear, unambiguous definition of data. In addition, the structure enables the replacement of ad hoc, inconsistent or incomplete formatting with a stylesheet that applies presentation semantics in a consistent fashion to any instance that conforms to the structure [27].

Glushko and McGrath [27] have applied the document engineering process in B2B documents aiming to exchange regular data among Web applications. Thus, they focus on data-centric documents. In our case study, we analyzed a set of HCI project documents focusing on the process of abstracting its structure aiming at later retrieval. As a result we have abstracted a structure to this documentation, thus defining the correspondent markup language (XML DTD).

In the next section is presented the case study we carried through to abstract the structure of a vast set of documents of interface design. This structure will be used later as a base to allow an efficient retrieval of the information registered in the documents.

Fig 2. The Document Engineering Roadmap [27].

III. CASE STUDY

The object of our study consists of the documentation of twenty (20) interface projects developed during a HCI course, given to an undergraduate course and two graduate courses in Computing Science in our institute.

The documentation introduces the specification of a practical interface project that is composed of five phases: 0) Groups and Topic Definition, 1) Understanding the Problem, 2) Design of Alternative Interfaces, 3) Prototyping and Evaluation Plan and 4) Evaluation. The professor of the course supplies a script to the project documentation, with the content list and an orientation for the elaboration of documents related to each phase. The documents must be written in HTML and published in the Web.

The students of the HCI publish their documentation by using CoTeia [2], a CSCW (Computer Supported Cooperative Work) tool for the collaborative authoring of material based on the Web. CoTeia presents the main features of the CoWeb [13] including simplicity and open authoring of documents, and also has other functionalities as access control and concurrent control. Using CoTeia, all the material elaborated by the students during the design phases is available in a Web based repository.

The documentation available for analysis is from 20 projects developed in the period of 2001\(^1\) to 2002\(^2\), totaling 145 pages with 824 links. The detailed data of the projects are graphically presented in Fig. 3. It presents the number of pages and the number of links per page used in the documentation of each interface project developed. In fact, we can see that there is a variety of styles during document authoring. For example, in “MAWA” project the team made use of a lot of links and

\(^1\) http://catuaba.icmc.sc.usp.br:8080/grad1o2001-HCI2

\(^2\) http://coweb.icmc.usp.br/coweb/mostra.php?ident=28.10
only one page was created. On the other hand, “Traffic Aid” project used a large number of pages and few links. This observation leads us to reflect about the fact that even using a pre-defined script to guide the documentation required, the freedom of authoring the hypertext elements (such as links and pages) is evidenced.

Fig. 3. Number of pages and links per pages from HCI course projects documentation.

Software engineering is the area that consists of a series of activities that must be carried through during the software development process and that must be registered [18] [21]. However, questions related to usability are not emphasized; and adding an activity among the existing activities is not enough to solve this problem. Software engineering for the project of interactive systems must include techniques that cross all of the software development [7].

Based on the most important activities cited in ACM Curricula for Human-Computer Interaction [1] to the development process of interactive systems, a documentation script for interface projects was established for the HCI course. This script is presented in Table I.

Although an initial script for the elaboration of the documentation exists, each project encloses a different subject, with specific details that demand the specialization and adaptation of the documentation to the features of each project. Therefore, the produced documentation is similar in their general structure, but is very different when analyzed in detail as we have noticed by the number of links and pages created. The documentation is a rich source for the extraction of a structure that configured a class of documentation for human-computer interface projects. The objective is to use markup languages based on XML, that is XML DTDs, to structure documents on specifications of human-computer interface projects.

IV. PROCESS TO ABSTRACT THE HCI DOCUMENTATION STRUCTURE

Extensible Markup Language (XML) [23] is a meta-language that allows the creation of specific markup languages to structure a document in parts and to identify the parts that are different. Such markup languages (also known as vocabularies or XML applications) are defined with a Document Type Definition (DTD).

Literature about XML shows how to write a DTD, that is which syntax aspects are [14] [24] [29], or how to develop a DTD from scratch, based on general guidelines for the document analysis and heuristics for the document structure definition [14] [26]. In this context, Maler and Andaloussi [25] propose a document developing cycle, and give special attention to document analysis. However, we cannot notice any support for the analysis of extensive document sets and a detailed description of tasks to several people involved in this activity.

In next subsection, the process performed during our case study is described and the markup language defined is presented in appendix.

<table>
<thead>
<tr>
<th>PHASE</th>
<th>REQUIRED AS DOCUMENTATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Groups and Topic Definition (0)</td>
<td>- name, date and a brief project description</td>
</tr>
<tr>
<td>Understanding the Problem (1)</td>
<td>- updated description of the project (optional)</td>
</tr>
<tr>
<td>Project of Alternative Interfaces (2)</td>
<td>- description of technical or social organization</td>
</tr>
<tr>
<td>Prototyping and Evaluation Plan (3)</td>
<td>- users description</td>
</tr>
<tr>
<td>Evaluation (4)</td>
<td>- task analysis of the system under development</td>
</tr>
</tbody>
</table>

A. The Process

Two computing science graduate students carried through
the process. In a first step, they individually abstracted the structure of the hyperdocuments and, in a second step, argued about each element of these structures to get, in a third step, an intermediate structure that was consensus for both. Then, the hyperdocuments were re-written following the intermediate structure. Finally, a new discussion about changes to get the final structure could happen.

The complete process, for \( n \) candidate structures, is represented in Fig. 4. The activity to abstract a candidate structure is graphically represented in Fig. 5 and it is part of the process to abstract the final structure (Fig. 4).

During the individual activity to abstract the structure of the hyperdocuments (Fig 5), the activities of ‘Reading of the selected hyperdocuments’, ‘Identification of elements’, ‘Definition of candidate elements’, ‘Heuristic analysis’ and ‘Evaluation’ were performed. The candidate structure results from a long and recursive activity with backtracking and switching between reading and evaluation. By observation we have concluded that the process is bottom-up.

The ‘Reading of selected documents’ task consisted of reading the hyperdocuments of all interface projects developed by the groups of students of the HCI course in the years of 2001 and 2002 available in the Web via CoTeia tool. The ‘Identification of candidate elements’ task consisted of finding and listing all the possible elements that could be used in the structure and, later, in identifying the fundamental elements, i.e., the elements that were relevant and/or convenient for representing the semantics. These elements were considered candidates to participate of the final structure. After identifying the candidate elements, the task of ‘Definition of candidate elements’ included the definition of which of them would be used effectively.

In the ‘Heuristic analysis’, heuristics were applied to support the definition of a candidate structure. The first time this process was performed, the heuristics had consisted of elementary guidelines for the XML application development [14] [26]. Later, the set of heuristics was enriched with new heuristics identified during the consensus discussions. The previous tasks were repeated if necessary.

The ‘Evaluation’ task consisted of confronting the structure identified so far with the existing hyperdocuments. If the result obtained during the evaluation was not satisfactory, the ‘Reading of selected documents’, ‘Identification of candidate elements’, ‘Definition of candidate elements’ and ‘Heuristic analysis’ tasks were repeated; in an opposite way, the structure was considered a candidate structure (‘Definition of candidate structure’ activity).

At the end of the activities of structure abstraction, two candidate structures had been obtained. The next activity (Fig. 4) was to discuss on each element of these two candidate structures to get a consensus about an intermediate structure. The ‘Consensus discussion’ consisted in the comparison of each element of the two structures defined separately and only after
reaching a consensus, the resulting element of this consensus was added to the intermediate structure. While consensus discussion was being pursued, new heuristics were identified and registered in a repository. The set of heuristics make up the base for the heuristic analysis of the candidate elements performed during the individual activity to abstract the candidate structure.

In “Intermediate DTD definition” activity, an intermediate structure was defined to underlie the activities of structuring the hyperdocuments. The “Hyperdocument structuring” activity consisted of re-writing the structure of existing hyperdocuments to the intermediate structure. Similar to structure abstraction activities, the hyperdocument structuring activities were individually performed. The new hyperdocuments were generated as a result of observation of the intermediate structure and adaptation of existing structure with backtracking and switching between observation and structuring. Through observation we have concluded that the process is top-down.

Again, the “Consensus discussion” activity was necessary to resolve differences between existing structure and intermediate structure. It was performed similarly to prior description. Finally, in the “Final DTD definition” activity, the final structure to documentation of HCI projects was obtained.

V. RELATED WORK

Durán et al. [8] present an approach based on XML for the automatic verification of software requirement documents. Software requirements are represented in XML and the XSLT language is used to verify desired quality properties and to compute a few metrics. Their objective is to apply XML based technologies, during the software documentation process, to verify the productivity quality of software documents. Our approach is to provide means to make a posterior recovering of information based on software documents easier, since we identify the embedded structure and therefore, we emphasize the semantic role of the information, instead of its format.

Badros [3] and Collard et al. [5] propose to change the underlying representation of source code to facilitate the use of more powerful document management methods and tools. They describe an XML application, which is used to add structural information to unstructured source code text files. The text can then be more easily searched, parsed, and transformed with the aid of these tags.

Gionis et al. [12] propose a system for inferring a DTD schema for a database of XML documents. The approach employs two steps, generalization and factorization, to derive a range of general and concise candidate DTDs, and then uses a mechanism for composing near-optimal DTD schema from the set of candidate DTDs generated by the earlier steps. It is worth noting that they work with documents having a pre-defined syntax.

The case study reported in this paper counts only on documents not having any syntax strictness. For this reason, the process required an effective participation of people involved, mainly during the abstraction of candidate structure activity. In addition, we observed that a tool would be very useful to help the analysis task, supporting the consensus discussion activity, in which the candidate structures (in DTD format) are available. To address the consensus discussion issue, related work has been presented in literature. Kuikka et al. [16] have developed a syntax-directed approach to transform documents from one structure to another. The aim is to automate a transformation between two grammars that have common parts, although the grammars and names of elements may differ. They propose a system that can generate a transformation semi-automatically if the user defines a matching among the elements containing the text of the document. The system proposes alternative sub-structures for the user and the user selects an appropriate sub-structure. The proposed approach can be useful to support consensus discussion comparing candidate structures.

VI. CONCLUSION

In general, the software documentation process produces an extensive variety of documents, types of documents, formats and contents. When the activity of recovering information from documentation takes place (face to inevitable changes and evolution), developers usually claim about the difficulty of searching in those documents, since it demands a lot of time and attention from them. Web technology could aid to address this problem, but it is necessary that the documents be structured and, in case of existing documents, literature just offers general guidelines for the document analysis and heuristics for the document structure definition.

In this paper we presented a case study of analyzing and structuring the documentation produced by 20 groups of students who have developed different interactive system projects as a task for a course. The documentation of each group was elaborated following a script, but with freedom to create the contents and hypertext elements (such as links and pages). A great volume and diversity of information was generated making it difficult for future developers or students to access these documents. The extension, complexity and richness of such documentation were enough to allow analysis and extraction of a structure (XML DTD) for HCI projects.

The process considers the analysis of extensive collections of documents and several people involved in this task. The consensus discussion activity is used to converge candidate structures into a single structure. Consensus decisions are taken in account in next heuristic analysis. In “related work” we discussed where automatic tools should be also considered to assist ad hoc work and our proposed process.

We also concluded that a process to abstract such structure based on existing documents should be made through an iterative activity in which concrete task (reading of actual documents to identify their structure and structuring existing hyperdocuments) and abstract task (identifying and defining of structure) take place. The activity corresponds to alternate between bottom-up and top-down approaches and depends on the ability of people involved in the process.

Currently, HCI course is given to two classes, one to undergraduate students and another to graduate students; the produced documentation will be analyzed later. Since the whole
documentation has been structured in a way by which different parts could be identified, we intend to explore the structure to investigate how efficient applications of recovering information based on that structure would be. For that, we count on the inserted semantics of the defined markup tags. One of such applications consists of carrying out scans over documents to extract the best information to combine with the objective of the users (for instance, a software engineer) during the browsing of the documents. Consequently, we will investigate how to combine the documents with other documents representing queries formulated during the browsing. The objective is to provide mechanisms of browsing the documents closer to the goals of users, and then to reduce the cognitive overhead required during navigation on extensive documentation basis.

APPENDIX

The following DTD was the result of process represented in Fig. 4.

```xml
<!DOCTYPE hci_project [title, phases]>  
<!-- Phases -->
<!ELEMENT phases (definition_phase, understanding_phase, project_phase, prototype_phase, evaluation_phase)>  
<!-- Groups and Topic Definition -->
<!ELEMENT definition_phase (project_name, logotype, group_name, group_member+, group_email, project_description)>  
<!-- Understanding the Problem -->
<!ELEMENT understanding_phase (project_description?, organization_description, users_description, tasks_analysis, existing_systems_analysis, usability_guidelines, justification, appendix?)>  
<!ELEMENT tasks_analysis (introduction, characteristics_tasks, characteristics_environment, detailed_analysis+)>  
<!ELEMENT existing_systems_analysis (introduction?, existing_system*)>  
<!ELEMENT existing_systems (system_description, advantages?, disadvantages?)>  
<!ELEMENT usability_guidelines (introduction?, guideline+, analysis)>  
<!ELEMENT justification (bibliography | reference | justification)>  
<!-- Project of Alternative Interfaces -->
<!ELEMENT project_phase (project_description, requirements, design_space, proposal_interfaces, changes, justification)>  
<!ELEMENT requirements (introduction?, requirement+)>  
<!ELEMENT requirement (functional+, non_functional+)>  
<!ELEMENT prototype (project_description+, prototype, plan_evaluation)>  
<!ELEMENT prototype (general_description, scenario+, evaluation)>  
<!-- Evaluation -->
<!ELEMENT evaluation_phase (results, exercise+, recommendations, critics, appendix?)>  

ACKNOWLEDGMENT

Our thanks to students for allowing us to use data they had developed.

REFERENCES


[22] L. Cazassa Fumagalii was born in Ribeirão Preto-SP, Brazil, on January 27, 1977. She received her B.Sc. degree in computer science at Institute of Mathematics Sciences and Computing, University of São Paulo. Her special fields of interest include Software Engineering and Hypermedia. Her current position is as a M.Sc. student at Institute of Mathematics Sciences and Computing, University of São Paulo.


[26] L. Cazassa Fumagalii was born in Ribeirão Preto-SP, Brazil, on January 27, 1977. She received her B.Sc. degree in computer science at Institute of Mathematics Sciences and Computing, University of São Paulo. Her special fields of interest include Software Engineering and Hypermedia. Her current position is as a M.Sc. student at Institute of Mathematics Sciences and Computing, University of São Paulo.


Renata Pontin M. Fortes was born in Campinas-SP, Brazil, on January 19, 1960. She graduated from the Institute of Physics, University of São Paulo, and studied at Institute of Mathematics Sciences and Computing, University of São Paulo.

Her employment experience included the ALCAN Company, the ELEBRA Company, and SERPRO. Her special fields of interest include Software Engineering and Hypermedia.

Her current position is as lecturer and researcher at Institute of Mathematics Sciences and Computing, University of São Paulo, in São Carlos-SP.

Lisandra Cazassa Fumagalii was born in Ribeirão Preto-SP, Brazil, on January 27, 1977. She received her B.Sc. degree in computer science at Institute of Mathematics Sciences and Computing, University of São Paulo. Her special fields of interest include Software Engineering and Hypermedia. Her current position is as a M.Sc. student at Institute of Mathematics Sciences and Computing, University of São Paulo.

Luciano T. E. Pansanato was born in Piraju-SP, Brazil, on April 28, 1968. He is a doctoral student at Institute of Mathematics Sciences and Computing, University of São Paulo, where he received his M.Sc. degree in computer science in 1999. He is a professor in Federal Center of Technological Education of Paraná. His research interests include Web-based Information Systems, Software Engineering and Hypermedia.