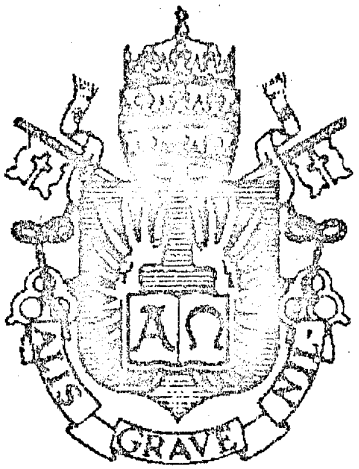

PUC

Monografias em Ciência da Computação
nº 29/91

A Research Agenda on Software Design

Carlos J. P. Lucena
Julio Cesar S. P. Leite
Daniel Schwabe
Hugo Fuks

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453
RIO DE JANEIRO - BRASIL

PUC RIO - DEPARTAMENTO DE INFORMÁTICA

Monografias em Ciência da Computação, Nº 29/91
Editor: Carlos J. P. Lucena

Março, 1991

A Research Agenda on Software Design *

Carlos J. P. Lucena
Julio Cesar S. P. Leite
Daniel Schwabe
Hugo Fuks

* This work has been sponsored by the Secretaria de Ciência e Tecnologia da Presidência da República Federativa do Brasil.

In charge of publications:

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC Rio - Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453 - Rio de Janeiro, RJ
Brasil

Tel.:(021)529-9386 Telex:31078 Fax:(021)511-5645
E-mail:rosane@inf.puc-rio.br

RESUMO

Esta monografia descreve os interesses de pesquisa comuns dos autores, sob a forma de uma agenda cujos tópicos são projetos de pesquisa dos seus alunos de pós-graduação. Seu denominador comum é o *upstream design* de sistemas de *software* que por sua vez requer a representação de uma extensa rede de informação sobre *design*. Esta rede abrange todos os dados relevantes ao processo de *design*, incluindo todo o processo de decisão e suas justificativas, assim como as alternativas cogitadas. A atenção dada ao *upstream* tem um impacto significativo na representação dos requisitos, e as idéias desenvolvidas a respeito de aspectos genéricos do processo de *design* são avaliados experimentalmente em ambientes que suportam diversas representações de *design*. A representação do processo de *design* é, por si só um tópico de pesquisa, dado que esta não pode ser facilmente separada das políticas e dos procedimentos do ambiente na qual se insere. Quatro aspectos do problema de *software design* são enfocados nesta monografia: conhecimento sobre *design*, o produto do *design*, história e justificativas do *design* e *design* cooperativo.

Palavras-chave

Software design, conhecimento sobre *design*, o produto do *design*, *design* cooperativo

ABSTRACT

The present report describes the joint research interests of the authors in the form of a research agenda that refers to topics that are current research projects of their graduate students. Their common investigation centers on the upstream design of software system that requires the representation of a diverse network of design information. This network encompasses all data relevant to the design process including postulated alternatives, their resolution through decisions and rationale that ultimately leads to commitments. Attention to the upstream has a significant impact on representation requirements and the ideas developed about general aspects of the design process are evaluated experimentally in environments that support different design representations. The design process representation is itself a research topic since it cannot be easily decoupled from the policies and procedures within the environment in which it is embedded. Four aspects of the software design problem are addressed by the author's research agenda: knowledge about design, the design product, design history and design justification and cooperative design.

Keywords

Software design, knowledge about design, the design product, cooperative design

A Research Agenda on Software Design

Carlos J.P. Lucena
Julio Cesar S.P. Leite
Daniel Schwabe
Hugo Fuks

Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente 225
Rio de Janeiro 22453 Brasil

March 91

ABSTRACT

The present report describes the joint research interests of the authors in the form of a research agenda that refers to topics that are current research projects of their graduate students. Their common investigation centers on the upstream design of software system that requires the representation of a diverse network of design information. This network encompasses all data relevant to the design process including postulated alternatives, their resolution through decisions and rationale that ultimately leads to commitments. Attention to the upstream has a significant impact on representation requirements and the ideas developed about general aspects of the design process are evaluated experimentally in environments that support different design representations. The design process representation is itself a research topic since it cannot be easily decoupled from the policies and procedures within the environment in which it is embedded. Four aspects of the software design problem are addressed by the author's research agenda: knowledge about design, the design product, design history and design justification and cooperative design.

I. INTRODUCTION

The main purpose of the research outlined in the following agenda is to generate new knowledge about the process of designing software artifacts. Software, as an artificial artifact is strongly dependent on a good design. We see software design as the process taking a functional specification and a set of nonfunctional constraints and producing a description of an implementation from which source code can be developed [RUGA 90].

We follow Simon's objective view [SIMO 73] by which "*design is concerned with devising artifacts to attain goals*". Therefore we see design as a problem solving activity that aims at proposing a solution that satisfies previously established goals.

In the process of software construction two metaphors are often used to characterize the design process. One of them consists of comparing software design to a search space in such a way that the process of software design is a search for a solution that satisfies pre-established goals [SIMO 81], e.g., [ANDE 90]. The other metaphor characterizes software process as a chain of decisions.

Software design is essential not only for the correct implementation of an artifact but also for its future maintenance. It is a known fact that recording a software design is a difficult task. In the software design activity not only it is difficult to formalize the software design product as it is hard to formalize the design process and its rationale. One reason design information is lost is that commonly used design representations are not expressive enough. While they are adequate for describing the cumulative results of a set of decisions, particularly about the structure of components and how they interact, they do not try to represent the incremental changes that come with individual design decisions [RUGA 90].

Our research agenda comprises basic research aspects as well as experimentation aspects. Basic research is oriented towards the formalization of the software design process including situations in which it takes place in a cooperative setting and the characterization of the semantics of reutilization of designs. We believe it is easier to reuse or adapt a generalized component than a restricted one. Experimentation centers on the evaluation of formalized concepts of the upstream design in environments that support specific design representations.

The main foci of the research agenda are:

- knowledge about design
- the design product
- design history and justification
- cooperative design

2. KNOWLEDGE ABOUT DESIGN

In order to develop genuinely useful decision support systems for designers, one must understand what designers do and the types of information they employ. Because design problems are usually ill-defined [SIMO 73], problem solving may require that much more attention be devoted to recognition/classification than is common for the well-defined problems typically studied in problem solving research [ROUS 86]. In other words, problem formulation often is the central issue in design, much more so than in most other forms of problem solving.

For the purpose of the present research agenda it is useful to describe the design process in terms of four stages [ROUS 86]:

1. Formulation (or goal elaboration)
2. Synthesis (or design generation)
3. Analysis (or design evaluation)
4. Optimization

The design process involves a great interplay among the above stages and, in particular, involves synthesis and analysis at multiple levels of detail and abstraction.

Although Software Engineering cannot be compared with more established engineering disciplines because of its general lack of solid foundations, it is already possible to identify a number of established principles to guide effective software designs. Knowledge about the design of software artifacts exists both with respect to specific implementation techniques and in connection with some general design principles which have been observed in practice [FREE 83, PARN 86]. The available knowledge needs to be represented adequately for the purpose of storage and retrieval in design environments.

Our goal is to explore the above research aspect both from the point of view of knowledge based system (KBS) as well as some more specific aspect of the problem such as reutilization. As far as KBS's are concerned we shall be investigating knowledge representation which are adequate for design artifacts and exploring at the same time the use of hipertexts for the storage and retrieval of design related information.

Hypertext has been frequently used to support software design [CONK 88] POTT 88, SCAC 90]. The usual approach is to define special types of nodes and links, which reflect particular language for representing designs; this language will vary depending on the level of design being modelled.

The Hypertext Design Model (HDM) methodology [GARZ 90a,b, SCHW 90] provides a notation in which to define schemas of hypertext applications. In particular, it is possible to define node and links types, as well as the browsing semantics associated to hypertext applications built as instances of these schemas. HDM also allows the (partial) specification of hybrid (or semi-formal [LAI 88]) systems, where it is possible to mix formal and informal representations.

For these reasons, HDM may be used to specify and implement hypertextual support for design. By using such a model, it will be possible to describe different design methodologies, possibly oriented towards different levels of abstraction, using the same notation. It should also be possible to provide coherent interfaces to environments supporting more than one methodology. Furthermore, the generated hypertext applications will be more easily integrated with knowledge based support for particular design methodologies.

As far as domain oriented reutilization [ARAN 89, NEIG 89] is concerned we are investigating the organization and retrieval of software design following the DRACO [FREE 87, NEIG 84] approach. Following this approach components are organized by application domain, with several implementations associated to each domain, and are organized in a component's library for the purpose of reutilization. The various implementations represent various design alternatives for the same specification.

The formalization of the process of retrieval and adaptation of software design is being developed having as a basis a formal theory of metaphors [LUCE 88]. The theory is associated to a representation in which designs are expressed as plans to allow for the choice of good candidates for reutilization, given a specification. For the process of adaptation of design in situations where perfect matches are found, a formal treatment is being developed based on the notion of interpretations between theories.

Design decisions are not made in isolation. Often, a solution is best expressed through several inter-related decisions. Unless the interdependencies are explicitly documented, the unwary design maintainer will fail to notice all the implications of a proposed change.

A model of specification for multiple participants treats the development of a specification as a dialogue in which the participants negotiate, establish responsibilities and cooperatively construct an overall specification [FINK 89]. The term dialogue, as generally used, refers to a conversation or spoken interaction between two or more participants. The logical framework developed for the cooperative development of specifications is being extended to deal with the important new notion of cooperative design (participants producing a design in a cooperative environment). The theoretical results produced in association with this research aspect of upstream design will be incorporated in experiments that use different design representations as the nucleus of cooperative design environments.

3. THE DESIGN PRODUCT

When dealing with the design product itself it is important to recognize that it constitutes an extremely complex web of interconnections. Therefore, the conceptual model of a KBS that attempts to represent the software design product has to pay special attention to the configuration management problem. In [FINK 90], this problem is treated using the aforementioned multiple-participant dialogic model.

To formalize the planning and management of configurations of software components, a modal action logic is being used [ALEN 91]. The adopted logical formalism allows the description and reasoning about changes of the structure (design changes), of the module interdependencies viewed as dependencies between the interfaces of the communicating modules (interface changes) and of the functional aspects of a high-level software system description. A modal action logic is used to capture the change aspects of the high-level software descriptions treated as theories in the logical framework. Reasoning about functional aspects of these descriptions is achieved through the interaction of a theorem prover for the proposed metalogic formalism with a theorem prover for reasoning about sequential programs.

Results in the formalization of configuration management will be used in the research aspects related to KBS's (section 2). It will also influence the representation scheme used for components within the DRACO paradigm, that is, we want to investigate how a formal system for dealing with configurations may effectively help in the set-up of domain networks and also influence the data model for a components' library with multiple implementation alternatives.

4. DESIGN HISTORY AND JUSTIFICATION

As pointed out by many researchers [POLL 88, GOLD 90, BAXT 90, FEAT 89, LEIT 91, PARN 86] the availability of a final representation of a Design Product is not sufficient. Design history together with its justification are indispensable not only for the complete understanding of the artifact to be implemented as well as for its future reuse and maintenance. Therefore, we would like to be able to represent the design process that led to the elaboration of a given design product.

Our research related to the formalization of the design process concentrates in two aspects. One aspect has to do with the definition of a meta-language for the formalization of the

software process [POTT 89]. The second aspect is concerned with a study related to the DRACO paradigm which treats the meta-knowledge related to the design process as tactics [NEIG 84, BAXT 90].

The research in a language to describe processes concentrates in the production of a rigorous description of such a language together with its application in case studies. The possibility of modeling software process and in particular the design process will help characterizing history of designs. Some believe that reutilization is only possible if dealt with at this level [GOLD 90, SEPP 90]. Extensions to the language will allow the recording of justifications (as in [POTT 88]).

As far as the DRACO paradigm is concerned the process of transformation of representations, either through vertical transformations (refinements) or through horizontal transformations (transformations) captures the meta-knowledge involved in the choice of alternatives through the notion of tactics [NEIG 89, BAXT 90]. These tactics make use of a very simple language based in production systems. Following the directives contained in the tactics it is possible to transform representations with the assistance of an automated environment. These directives characterize the design process itself.

The application of a language for process description used as a basis for the application of tactics enables greater flexibility for modeling and implementing practical environments driven by the design process.

5. COOPERATIVE DESIGN

Cooperative software design [MARC 90] may be carried out in at least two settings:

- Given a design methodology, add to it collaboration protocols to allow multiple designers to interact and collectively carry out the design;

- Develop a design methodology that incorporates (as built-in primitives) the collaboration process.

In either case, very little work has been done up to now in investigating the protocols used in cooperative design; that is still an open research area.

Regardless of the form of these protocols, the hypertextual support for this process must be able to accommodate them. In fact, there seems to be some degree of similarity, as far as primitives are concerned, between such support and support for interactive multimedia applications - both must accommodate many agents, accessing simultaneously the hyperdocument, in coordinated fashion.

The logical framework proposed in [FUKS 91] will be used to extend existing software design methodologies by providing them with primitives that model cooperative design.

6. SPECIFIC RESEARCH TOPICS

a) Implementation of the Dialogue Reasoning System

The implementation of a Dialogue Reasoning System based on the notions of commitment [FUKS 89], commitment calculus and commitment axioms - and legality - legality axioms.

b) Extension of the Dialogue System to N-Party Dialogues

The current Dialogue Reasoning System is capable of dealing with two participants. The extension of this system to a N-Party one is not a trivial step because of a variety of side effects. We intend to use techniques originated from distributed systems specification methods to bring about the envisaged extensions.

c) Research in Cooperative Dialogue (to support cooperative design)

First we will look at the type of dialogue - interlocutors and scripts - that occurs in a cooperative working environment for software design. Then, we will adapt and use the Dialogue Reasoning System to represent and to reason with and about these cooperative dialogues.

A second line is to investigate the interactions with a multimedia hypertextual application as an instance of a cooperative dialogue.

d) Formalization of Design Products for Configuration Management

Software system architectures need to be specified, together with interconnection and interface definitions and the collection of families of multi-version modules that belong to the system. The module interconnection aspects of this high level software descriptions are used in the system architectural design and configuration. The other aspects are used to help the maintenance and management of families of multi-version modules. The whole high-level software system description should also reflect the different granularity of the objects used for configuration management, such as procedures, functions, data types, modules, subsystems and systems.

In this context the problem of maintaining configurations of evolving software systems is a relevant one. Here it should be important to provide a formal basis for the existent notions of systems architecture, formal notions of software systems integrity and mechanisms to control the evolution of software systems structural and functional descriptions. The solution proposed adopts a logical formalism that allows us to describe and reason about changes of structure (architectural changes), the module interdependencies viewed as dependencies between the interfaces of communicating modules (interface changes) and the functional aspects of a high-level software system description. A modal action logic is used to capture the change aspects of the high-level software descriptions treated as theories in the logical framework. Reasoning about functional aspects of these descriptions is achieved through an interaction of a theorem prover for the proposed metalogic formalism and a theorem prover for reasoning about sequential programs [ALEN 91].

e) Interface Design and Application Software Design Produced From Interfaces as Specifications

The User Interface Development System called Midas (for Merging Interface Development with Application Specifications which allows interface/systems designers to develop an application-specific user interface interactively, in a prototyping-oriented environment, while refining the specification of the intended application itself. The interface/systems designer receives expert advice on both interface and application software design principles, emerging from MIDAS' knowledge base, and can also animate the intended user dialogue with the interface being designed via an extensive set of visual programming aids.The generated interface can be further customized by the end-user, by flexibly allowing the default appearance of the dialogue scenarios. Furthermore, the application-specific end-user interface is also knowledge based. Its domain knowledge covers user modeling and the application domain, in order to adapt itself dynamically to different degrees of user familiarity with the application. from novice to experienced.

Both the interface code and the programming-in-the-large of the application code are developed within an object-oriented framework. A proposal for a software life cycle model based on the rapid prototyping of user interfaces as a means to refining the specification of the application all the way down to the import-export list and module semantics specification for each and every application module is also presented. The lifecycle model is rule-encoded in MIDAS' knowledge base. The interface/systems designer is guided by the interpretation of those rules. MIDAS aims to provide a testbed for new ideas in human-computer interfaces, knowledge-based support of design activities and life cycle models based on rapid prototyping of user interfaces [CABR 90]

f) An Approach to Software Reuse Based on a Formal Theory of Metaphors

This research topic examines the relationship between software reuse and a formal theory of metaphors and analogies. We use a formal theory of metaphors as a metaphor for the problem of software reusability. Metaphors use symbols belonging to a domain, called the source domain, to refer to objects that belong to a possibly different domain, called the target domain. We demonstrate by exemplification the viability of reuse when a systematic design method is applied in both the source and the target domain [LUCE 88].

g) Formalization of the Software Design Process and its Application to Methodologies for Software Design.

The idea of this new research topic is to investigate how upstream design considerations such as design justification, the recording of design histories and its reutilization can be associated to various formal and semi-formal design methodologies (e.g. [SMIT 90].

A major problem in the design process is the loss of rationale and process information [DUBO 87]. The recording of design decisions and their justifications requires a supporting technology to lessen the burden on the designer. We will study, using real cases, the applicability of hypertext and knowledge bases for collecting, organizing and recording justification and rationale. A candidate approach is the use of hybrid systems [GARZ 90b], combining knowledge-based systems and hypertext. These types of systems open up the possibility of using multi-media representations, that should be able to capture more "informal" aspects of design.

h) Draco's Redesign

The Draco prototype as built by Neighbors [Neighbors,84] is having its design recovered [LEIT 91]. As of now the Parse, the Transformation, and the Refinement subsystems were recovered and the Parse subsystem is going through a redesign process.

The Transformation library and the Tactics subsystem are the parts of Draco where design is a major issue. Both the transformations (horizontal transformations) and the tactics are knowledge sources for the design process within Draco. A transformation library contains operational knowledge for use in the design process, mainly recording optimizations operations. The Tactics subsystem encompass a production system like architecture where the rules are meta-level design knowledge. The Tactics subsystem is used in Draco to drive the choice of which components to use in the process of horizontal transformation (refinement).

A major redesign of the tactics subsystem, incorporating a tactics language [Goldberg 90, Baxter 90], is the principal objective of Draco's Redesign with respect to design issues. The tactics language should be useable both for transformations and for refinements.

i) Experimental Design Components Library

In order to support our research with exemplars, we are planning to populate a prototype library of design components. This library should contain different components in different representations. A faceted classification scheme [GIRA 90] will be utilized for store and retrieval of components. If possible, extra information about the design rationale [DUBO 87] would be kept together with the components.

The expected outcome of this line of action, besides the exemplars organized as library of components, is the seed for a design knowledge base. This knowledge base should be both domain specific as related to applications and broad with respect to meta level knowledge of design (general tactics).

7. REFERENCES

[ALEN 91] Alencar, P.S.C.; Lucena, C.J.P.; "A Logical Framework for Evolving Software Systems", to appear.

[ALEN 88] Alencar, P.S.; Lucena, C.J.P. *Métodos Formais para o Desenvolvimento de Software*. Buenos Aires, Kapeluss, 1988.

[ANDE 89] Anderson, J.S., Fickas, S., "A Proposed Perspective Shift: Viewing Specification Design as a Planning Problem", 5th IWSSD, ACM SIGSOFT, vol.14, 3, May 1989.

[ARAN 89] Arango, G., "Domain Analysis From Art Form to Engineering Discipline", Fifth IWSSD, ACM SIGSOFT Engineering Notes, vol. 14, n* 3, May 1989.

[BAXT 90] Baxter, I. *Transformational Maintenance by Reuse of Design Histories*, Ph. D. thesis, University of California, Irvine. Dept. of Information and Computer Science. Tech. Report 90-36.

[CABR 90] Cabral, R.H.B.; Campos, I.M.; Cowan, D.D.; Lucena, C.J.P.; "Interfaces as Specifications in the MIDAS User Interface Development System", ACM Software Engineering Notes, Vol.15, 4 , Apr.1990.

[CONK 88] Conklin, J.; Begeman, M. L.; "gIBIS: A Hypertext Tool for Exploratory Policy Discussion", ACM Trans. Office Information Systems 6 (1988) 303-331

[DIAZ 87] Prieto-Diaz, R., Freeman, P., "Classifying Software For Reusability", IEEE Software, Jan. 1987.

[DUBO 87] Dubois, E.; van Lamsweerdee, A. , "Making Specification Processes Explicit", 4th IWSSD, April 3-4 1987, Monterey, California, USA, Computer Society Press.

[FEAT 89] Feather, M.S., "Detecting Interference When Merging Specification Evolutions", 5* IWSSD, ACM SIGSOFT, vol.14, n* 3, May 1989.

[FINK 89] Finkelstein,A.; Fuks H.; "Multi-party Specification", Proceedings 5th International Workshop on Software Specification & Design, pp 185-195, IEEE CS Press, 1989.

[FINK 90] Finkelstein, A.; Fuks, H.,"Conversation Analysis and Specification", Computers and Conversation, ed. P. Luff, N. Gilbert & D. Frohlich, pp 173-186, Academic Press.

[FREE 83] Freeman, P., Fundamentals of Design. *Tutorial on Software Design Techniques* 4th ed. Peter Freeman and Anthony Wasserman (editors), IEEE Computer Socienty, 1983.

[FREE 87] Freeman, P,. Conceptual Analysis of the Draco Approach to Constructing Software Systems. *IEEE Trans. on Soft. Eng.*, SE-13.7, Jul 19897.

[FUKS 89] Fuks, H.; Ryan M. & Sadler, M. ,"Outline of a Commitment Logic for Legal Reasoning, Proceedings 3rd International Conference on Logics, Informatics and Law, Florence 2-5 Nov. 1989.

[FUKS 91] Fuks, H.; "Negotiation using Commitment and Dialogue", PhD. Thesis, Department of Computing, Imperial College, University of London, Feb. 1991.

[GARZ 90a] Garzotto F., Schwabe,D.; Paolini P.;, "HDM - A Model Based Approach to Hypermedia Application Design", submitted to ACM - TOIS, November 1990. Also available as Technical Report 90-75, Dipartimento di Elettronica, Politecnico di Milano, Nov. 1990.

[GARZ 90b] Garzotto F., Schwabe,D.; Paolini P.; "New Perspectives For Hypertext Using Model-based Application Design", submitted to Hypermedia. Also available as Technical Report 90-76 Dipartimento di Elettronica, Politecnico di Milano, Nov. 1990.

[GIRA 90] Girardi, M.del R.; Price, R.T., "Especificação de uma Ferramenta de Apoio a Reutilização de Software no Desenvolvimento Orientado a Objetos", Anais do 4º Simpósio Brasileiro de Engenharia de Software, Sociedade Brasileira de Computação, São Paulo, Out. 1990.

[GOLD 90] Goldberg A., "Reusing Software Developments", IV Symposium on Software Development Environments, ACM SIGSOFT, vol. 15, n* 6, Dec. 1990.

[LAI 88] Lai,K.Y.; Malone, T.W.; "Object Lens: A "Spreadsheet" for cooperative work:, Proceedings of the ACM Conference on Computer Supported Cooperative Work, Portland, Oregon, 1988.

[LEIT 91] Leite, J.C.S.P., Investigação e Exploração do Paradigma Draco. *Monografias em Ciências da Computação, Dept. de Informática, PUC-RIO* (to appear), 1991.

[LEIT 91] Leite, J.C.S.P.; Prado, A., Design Recovery: A Multi-Paradigm Approach, *Monografias em Ciências da Computação, Dept. de Informática, PUC-RIO* (to appear), 1991.

[LUCE 87] Lucena, C.J.P. *Inteligência Artificial e Engenharia de Software*, J. Zahar. Rio de Janeiro, 1987.

[LUCE 88] Lucena C.J.P.; Silva, J.R.; "An Approach to Software Reuse Based on a Formal Theory of Metaphors", Tech.Rep. 14/88, Computer Science Dept., PUC-Rio, 1988.

[MARC 89] Marca, D.A., "Specifying Coordinators: Guideline for Groupware Developers", 5* IWSSD, ACM SIGSOFT, vol. 14, n* 3, May 1989.

[MINS 90] Minsky, N.; Rozenshtein, D., "Configuration Management by Consensus: An Application of Low Governed Systems", ACM Symposium on software Development Environments, ACM SIGSOFT, vol.15, n* 6, Dec. 1990.

[NEIG 84] Neighbors, J., The Draco Approach to Constructing Software from Reusable Components. *IEEE Trans. on Software Engineering*, SE-10 (Sep. 1984), 564-573.

[NEIG 89] Neighbors, J.M., "DRACO: A Method for Engineering Reusable Software Systems", in software Reusability, vol I, ed. Ted J. Biggers Kaft and Alan J. Pelis, ACM Press, 1989.

[PARN 86] Parnas, D. ; Clements, P. A Rational Design Process: How and Why to Fake It. *IEEE Trans. on Soft. Eng.*, Feb. 1986.

[POTT 88] Potts, C.; Bruns,G.; "Recording the Reasons for Design Decisions", Proceedings of the Xth International Conference on Software Engineering, 1988.

[POTT 89] Potts, C., "A Generic Model for Representing Design Methods", XI International Conference on Software Engineering, IEEE Computer Society, Pittsburgh, USA, May 1989.

[ROUS 86] Rouse, W.B.; "On the Value of Information is System Design: A Framework for Understanding and Aiding Designers", Information Processing and Management, Vol.2, n* , 1986.

[RUGA 90] Rugaber, S.; Ornburn,S.B.; Leblanc,R.J.; "Recognizing Design Decisions in Programs, IEEE Software, Jan.1990.

[SCAC 90] Scacchi, W.; Garg, P.K., "A Hypertext System to Manage Software Lifecycle Documents", IEEE Software, Vol. 7, 3, May 1990.

[SCHW 90] Schwabe, D.; Caloini, A.; Garzotto, F.; Paolini, P., "Hypertext development using a model-based approach", Technical Report 90-74, Dipartimento di Elettronica, Politecnico di Milano, Nov. 90. Submitted for publication to Software Practice & Experience.

[SEPP 90] Seppänen, V. *Acquisition and reuse of knowledge to design embedded software*, Technical Research Centre of Finland, VTT-Publications 66, 1990.

[SILV 88] Silva, J.R.; Lucena, C.J.P. Um Novo Paradigma para o Problema de Reutilização de Software, *Monografias em Ciências da Computação, Dept. de Informática, PUC-RIO* , MCC 14/88, 1988.

[SIMO 81] Simon, H.A., "The Sciences of the Artificial", 2nd ed., Cambridge, MA:MIT Press 1981.

[SIMO 73] Simon, H.A.; "The Structure of Ill Structured Problems", Arificial Intelligence 4, 1973.

[SMITH 90] Smith, D.R., "KIDS: A Semi-Automatic Program Development System", IEEE Transactions on Software Engineering, vol. 16, 9, Sept. 1990.