Accepted for publication in IAMOT – International Association for Management of Technology (May 2003)

Author's Copy

A User Model Framework to Help the Development of Usability Oriented Systems' Interfaces Associated With Different Users.

Simone Bacellar Leal Ferreira
Associate Professor, Ibmec-RJ
Av. Rio Branco, 108, 5th. floor, Rio de Janeiro, 20040-001, Brazil
lealferreira@ibmecrj.br

Abstract:

When designing a friendly interface, designers must make sure that users feel comfortable and encouraged to use it, and that the interface suits each user's needs and expectations. For this, the interface must have several facilities that should be presented in varying forms to different users. To obtain such interfaces, it is necessary to analyze and know different users well, thus building a "user model" or a "user profile". This paper presents a user model, and, based on the user model presented, a framework has subsequently been developed, allowing the association of different users to different interfaces. This framework helps the requirements' engineer when defining non-functional requirements. He may consult the framework in order to define the requirements necessary to design an interface that may be instantiated for different users or for the same user that has changed his or her expertise level. Such a characteristic reduces the need for users to rethink and remember and offers a less expensive system. The framework facilitates the process reusing internal and external components, behaviors and designs throughout a system, maintaining consistency with purpose.

Keywords: user interface, framework, user Model, user-centered design

Introduction

All software is designed with the purpose of satisfying the expectations of users, so it is essential to know the users (Apple,1992) (Foley, 1990).

User interface is the way in which a user interacts with a system. It must therefore, be well designed in order to be easily used. To be friendly, an interface must suit each user's need and satisfy his/her expectations.

To obtain such interfaces, the designers must make an all out effort to ensure that users feel comfortable and encouraged to use the system. To achieve this end they must be able to communicate with the machine in the most natural way (Foley et al, 1990) and (Pressman, 1992).

In order to design a friendly interface, the process of software development must be "user centered", that is, the designer must analyze and know different users very well, carrying out a user analysis in order to understand what kind of expectations users have. The designer must also find ways to represent how users think, feel, behave, etc.

One way of doing this is to analyze users and sort them into different groups according to their behavioral similarities. With this knowledge, it is possible to build a "user model" or a "user profile".

People, however, do not think of computer systems, applications and user interface in the same way; they have different conceptual models of

systems. Furthermore, as users interact with various applications, their conceptual models change over time.

Since there are several groups, user analysis must also include several user models and emphasize the need for interfaces with distinct features for each user (Hix, 1993), (Collins, 1995) and (Shneiderman, 1998). An interface capable of satisfying several different users must be able to offer varying levels of functionalities and distinct facilities presented in different forms according to user needs.

The present work portrays a user model developed on the basis of certain known cultural models. Based on the user models presented, a framework was then developed with the purpose of allowing the association of a variety of users with several interfaces and functionalities that might be instantiated for different users or for the same user that has changed his or her expertise level.

A prototype of an application based on such a framework has been developed and tested together with a conventional application, similar to the one designed, using at most times fifty graduate students at PUC. Results are described later on in this work.

User Model

The final user must always be the main focus of the designer. All potential users must be analyzed in order to discover user characteristics, their tasks and the environment for which the system is being designed (Souza and Leite, 1995).

It is essential that all users feel comfortable and encouraged to use applications; communication between them must be friendly. Since human beings differ greatly and people are continually undergoing change, the systems must be capable of adapting to different users. In order to achieve this, human and domain factors (problem comprehension) must be studied.

System development is a process that depends on social factors. Software is developed based on a social context and on the client's and the development team's context. Several factors must be considered, such as background, social level, personality and behavior among others (Leite, 1995).

To create a product that can really be used by a group of people, it is necessary to identify and understand the group. Therefore, it is necessary to describe how people perform their tasks, what they think about their work environment and their limits (Apple,1992). An interface may be friendly to a particular group but not so to another.

To design good interfaces, one of the requirements is to know the final users well. During the analysis, the designer discovers what the user expects from the system, that is, he puts together a *user conceptual model* (Collins, 1995), (Hix, 1993), (Pressman, 1992).

The elaboration of a user conceptual model is based on the users´ expectations, objectives and understanding of the system. This will depend on the knowledge and previous experience of each person (Roberts et al, 1998).

Since the perception of the system is influenced by the various experiences a person has had, each user has his own conceptual user model. To design a friendly interface, the designer must know the potential users well.

The software development process must be “user centered”; the interface must be designed bearing in mind the need to attend to the user’s necessities (Norman and Draper, 1986). In this process, activities during the analysis must lead to information about the users, their tasks and about the application domain.

One of the ways to know the users is by elaborating a “*user model*” (or “*user profile*”) that describes the user’s characteristics. But the designer cannot forget that with continual use of the application, the user’s perception of the system changes and as he or she creates a new conceptual model of the system, his/her expectations, understanding and objectives also change. This gives a peculiar characteristic to the users’ model: since it is based on the features, expectations, understanding and objectives of the user, it must also be able to be modified. The users´ analysis must then include several users´ profiles (Hix, 1993), (Collins, 1995) and (Shneiderman, 1998) that allow users to change over to different profiles.

To truly satisfy users’ needs, the application must offer different levels of functionalities according to their needs. But in fact there are many profiles of users, so one system alone can hardly be expected to suit all the necessities of different types of users. What is often seen is that either the application presents an excess of functions for determined people or that it lacks features for others. In practice, it has been verified that a great many users feel frustrated when using a system.

To be accepted, an application must be able to allow users to use all or part of its features in different ways, offering users different interface visions. Customization facilities can be included in order to allow the user to define different sets of specific commands, modify the *default* that it is being offered to him and customize tool bars among other activities. But to do this, the user must first of all learn other tasks related to the customization, which is a difficult procedure for beginners (Murray, 1991).

What is desirable then is the construction of systems that can be instantiated, that is, designed with distinct aspects for each user, with different functionalities, so that it can be used by diverse groups of people (Apple,1992) in different ways. To develop such systems, the present work proposes a framework that has been developed based on a user model.

To build the current user model, it is necessary to analyze people; one must observe their behavior, their thoughts, feelings and actions; in fact, it is necessary to understand the user’s culture once that culture is the behavior that is *learned* and that is formed by thoughts, feeling and actions (Kroeber and Kluckhohn, 1954). Culture is learned and not inherited (Hoft, 1996).

Based on this fact, the proposed *user model* has been defined based on *cultural models.* There are several models available and all of them accede to the fact that most components of culture are invisible, buried in unconscious reality. (Hoft, 1996). Using a cultural model it is possible to analyze the cultural context of a person by comparing the similarities and differences among a group of people using certain variables. A cultural model is useful when building a user model (Ferreira, 1999).

The user model presented herein is based on certain known cultural models: Edward T. Hall’s model (Hall, 1993), David A. Victor’s model (Victor, 1992), *Geert Hofstede’s* model (Hofstede, 1991) and *Fons Trompenaars’* model

(Trompenaars, 1993). These cultural models were carefully studied and the following cultural variables were chosen because they could be useful as user variables.

- *Objective:* the user's objective when using an application.
- *Social-Cultural Conditions*: social-cultural factors.
- *Formation:* background of a person.
- *Function:* position at work.
- *Age.*
- *Geographical Location*: geographical location of the person's home.
- *Time Behavior*: how people deal with time when performing their tasks. People may be *Monochronic* or *Polichronic*. *Monochronic* people do one thing at a time while the *polichronic* may do several things simultaneously.
- *Knowledge*: the user's specific knowledge about computer systems and the application domain.
- *Tolerance with Uncertainty:* refers to how the user deals with unknown situations and mistakes.
- *Psychological Factors:* psychological factors that can influence the performance of a user when using the system.

The Presented Framework

The framework for object-oriented applications introduces promising technology for the refinement of software design and implementation, reducing cost and improving quality (Fayad, 1997).

Originally, object-oriented technology focused on framework as a software component. But framework is more customizable than components and has more complex interfacing. Being more powerful, it reduces the efforts required to develop applications that can be customized (Johnson, 1997).

Frameworks offer standard interfaces that allow for the reuse of existing components, that is, they supply components with a reusable context, providing a standard way for components to treat errors, change data and call operations in each one (Johnson, 1997).

The base of frameworks consists of a library of classes. To construct such a library, it is necessary to find proper abstractions, called abstract classes, for concrete classes (Shlaer and Mellor, 1992). A framework generally includes concrete subclasses that can be used immediately (Gamma et al, 1995) and they provide the default behavior and implementation of the abstract classes (Shlaer and Mellor, 1992). The abstract classes specify the execution flux and can be specialized (Bäumer *et al*. 1997) by means of subclasses.

The applications developed based on such frameworks are designed by means of customization of the classes and are very useful in the development of consistent user interfaces. They reuse their code and design, facilitating extensibility and reuse (Pree, 1995).

In order to develop the proposed framework, it was necessary to built a user model. The study of user modeling became important over the last few years, but little progress was made on how to use non-functional requirements, such as human factors, in the process of interface instantiation (Strachan et al, 1997). To develop the current research, several previous works were analyzed

(Rich, 1979), (Marins, 1987), (Murray, 1991), (Ambrosini,1997), (Benaki and Spyropoulos, 1997), (Gutkauf et al, 1995), (Paranagama and Arnott, 1997), and all of them made clear the fact that if the characteristics of each user are considered, the system will have a greater chance of success.

The proposed framework has the following basic classes (figure 1): *Mediator; Historic; Profile; Interface; Application; Record; Login*

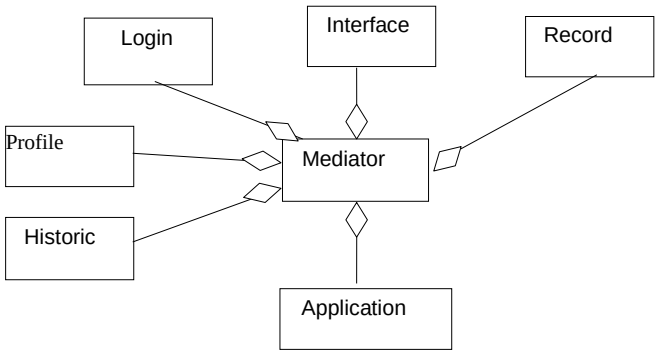

Figure 1: Main classes of the proposed framework

The *Profile* class contains several characteristics that define the profile of the user. It is an abstract class that specifies common attributes and functionalities of users. Classes in this hierarchy collaborate with classes in the interface hierarchy, in order to offer adequate interfaces to users.

The *Interface* class defines several *default* interfaces that are open to each profile of users. This class depends on the *Application* class; therefore it is defined according to the current application.

In order to make it possible to associate a user to one *default* interface there is the *Record* class, responsible for the initial definition of the profile of the user.

The *Application* class is responsible for the functionality of the system.

The *Mediator* class is responsible for the communication and synchronism of the components described above.

The *Login* class is responsible for the first definition of the profile of the user. When the user logs for the first time into the system, the user supplies certain information that will help to define his profile.

Class Profile

The proposed user model considers several defined variables based on certain previously known cultural models. Figure 2 shows an example of a user class profile.

Since no single user can represent the majority of users, what has really happened is that several user *profiles* (*models*) have had to be developed. In the proposed framework, several profiles are represented in a class structure; there is a super-class called *user,* and each user profile is represented by a sub-class of *users*. The attributes of each sub-class are obtained from the user model.

Each user may have his/her state or class changed. Once his/her perception changes a little, he/she changes to another state. Once the changes are very significant, he/she changes to another class, that is, to another profile.

The *Objective* Class portrays the user's objective while using an application. This objective may change as his/her perception changes. At one moment a user may want to use certain software, such as a word processor only as a simple text editor and at the next moment he/she may want to use it as a complete editor, including certain features such as that of equation editor.

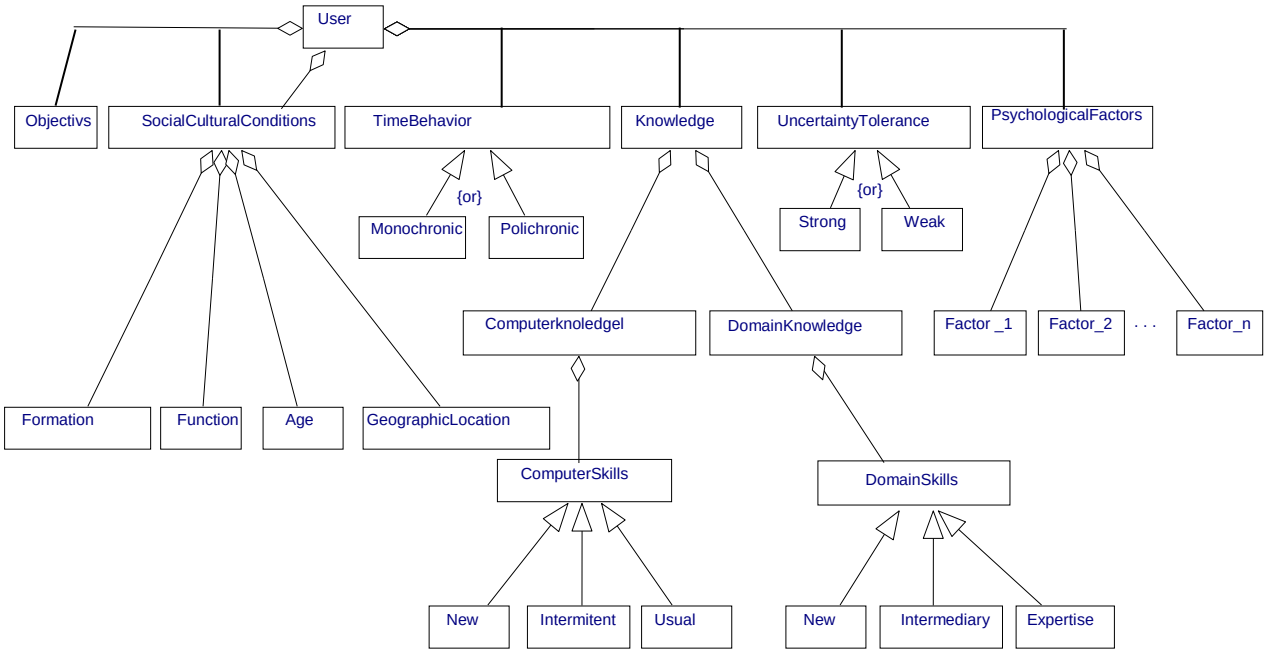

Figure 2: *Profile* Class Structure

The *SocialCulturalConditions* Class is a class that models factors related to the type of life the user leads, that is, the social-cultural factors. This class is formed by the *Formation*, *Function*, *Age and Geographical Location* classes.

The *Formation* class determines the background of a person; it models not only his/her academic level but also his/her field (human sciences, biomedical and others).

The *Function* class is important when designing *software* that will be used in some institutions. Probably people with different functions will access different functionality of the system.

The *Age* class may determine several characteristics of an interface, especially of visual aspects, such as the choice of the *default* color combination (Ferreira et al., 1999).

The *Geographical Location* class models some of the characteristics inherent to users, characteristics determined by the geographical location of his home. Metaphors and colors may be adequately chosen if the geographical location of the user is known.

The *Time Behavior* class shows how people deal with time when performing their tasks. It has two sub-classes: *Monochronic* and *Polichronic*. Monochronic people prefer doing things one right after the other while

polichronic people may do several things at the same time. These classes are, therefore, mutually exclusive, having a disjunction restriction.

The *Knowledge* class refers to a user's specific knowledge about computer systems and the application domain. It has two sub-classes: *ComputerKnowledge* and *DomainKnowledge*. Since a user may have both sets of knowledge, these sub-classes may occur simultaneously with a *superposition* restriction.

The *ComputerKnowledge* sub-class inherits the *Computerskills* sub-class that defines the user's skills when using a computer. The *ComputeKnowledge* sub class models the user's knowledge about the application domain through the *DomainSkills* class.

The *UncertaintyTolerance* class refers to how the user deals with unknown situations and mistakes. It has two sub-classes: *Strong* and *Weak. Strong* refers to people who have no problem with unknown situations and *Weak* to people who can't deal with them easily. These classes have a disjunction restriction. They were considered sub-classes and not states because the present framework intends to define a user's model that better models the user's characteristics. The way someone behaves when dealing with unknown situations is inherent to his/her own personality (Hofstede, 1991); maybe, if necessary, a person may react in a different way to the way that is more natural for him/her, but that would happen only because of dire need; his/her characteristics, though, would still remain the same under normal conditions. The present framework has therefore considered that when adapting interfaces to the nature of a person, it would be better to model this attribute as a sub-class, so as not to make it necessary for a user to change his/her state, which would go against his/her nature.

The *PsychologicalFactors* class models several psychological factors that can influence the performance of a user when using the system. This class depends on the application and its specializations must be defined for each application. But they must be included because they are relevant to the model.

Interface Instantiation

Based on the proposed framework and having a previously-designed system, when a user first accesses the system, the general idea is that he/she must be made to answer several questions with answers that are recognizable by the system in order to discover the profile of the user in question. Then, according to this profile, a proper *default* interface can be opened up. However, if the user so desires, he/she can choose another *default* interface.

As the user uses the application, the system monitors all his/her interactions, tracking all actions, whether they be right or wrong. As a consequence of the continuous use of the application, the user improves his/her ability on the computer and his/her perception of the system naturally changes. Once the user's perception changes, his/her profile also changes and the system then suggests a new *default* interface, more specialized according to the new profile, yet still consistent with the previous one. The user may change to this new one or decide to remain in the one he/she is already in. In the same way, if the user demonstrates that he/she is encountering difficulties with an interface, the system will propose another, less specialized one.

A user may also change his *default* interface at anytime whatsoever without waiting for a suggestion from the system.

Customization facilities can be included in order to allow the user to define different sets of specific commands and change toolbars, thus modifying the *default* that has been offered.

Consistency is essential because at the rate that the user's perception of the system changes, a new *default* interface with more features is offered to him/her. But all interfaces must have the same type of behavior and the same type of layout. They differ only in terms of details. Consistency reduces the need for users to rethink and remember and offers a less expensive system.

To be able to offer different consistent interfaces, the user model must possess a mechanism of data and operation extensibility. In answer to the requirements of the considered model and in order to maintain consistency, an object-oriented approach seems the most indicated. In the object-oriented paradigm, besides the possibility of extensibility for data operations and code reuse, it is possible to encapsulate and protect the definition of data structure and valid operations on this structure.

To make this potentiality possible, object-oriented programming is used. The concept of *classes* allows the implementation of abstract data and *inheritance* facilitates the reuse and adaptation of blocks (Pree, 1995), and makes the system more suitable for different profiles of users (Collins, 1995).

Validation

In order to make a first validation of the proposed framework, a prototype of an application (a word processor) based on framework has been developed. This prototype has been tested and compared with a conventional application, similar to the one designed (word for windows), using in average fifty graduate students at PUC. The students used both applications during a short period of time and later answered certain questions that enabled a comparison to be made of both applications. The great majority of students considered the application based on the developed framework more intuitive, easier to use, less confusing, easier to customize, in short, more friendly (Ferreira, 1999).

Conclusions

To be friendly, an interface must suit each user's needs and satisfy his/her expectations, containing an assortment of facilities to be presented in varying forms to different users. To obtain such interfaces, it is necessary to instantiate them according to each user's characteristics.

An application used by different people that have differing conceptual models of the system must rely on a framework that enables an association between users and software. To be considered friendly the interface must match the conceptual model that each user has.

This work presented a framework that has been developed with an aim towards associating different users to different interfaces. To develop such a framework it has been necessary to build a user model in order to mirror the users' peculiarities.

By consulting this framework the designer is able to define the requirements needed to design an interface that may be instantiated for different users or for the same user that is undergoing changes in his/her expertise level. This facilitates the process in that it reuses internal and external components, behaviors and designs throughout the system, maintaining consistency with purpose throughout.

To make a first evaluation of the framework, a prototype of an application was built and later tested by a group of graduate students. The results showed that using applications based on the framework had a greater likelihood of satisfying different groups as the application was closer to the conceptual models that each group had of the system.

Once the user's perception of the system changes over time, he/she creates a new conceptual model of the system and the user model proposed takes these modifications into account.

The proposed work showed that non-functional requirements, such as human factors, are essential to building friendly interfaces. When they are deliberated and modeled, the designed interface is naturally a friendly one and the system has a better chance of success. This research study may open the doorway to several other studies, such as the development of frameworks for *workgroups*

One of the most important contributions of the present work has been verifying, through testing in a group of students, that if non-functional requirements are well defined, taking into account human-factors, the resulting interfaces are much closer to the real necessities and expectations of users. In order to accomplish this, it is necessary to begin to model the user from the outset of the system design. The proposed framework, by having an embedded user model, allowed systems to adapt to the different perceptions that users had during their interaction with the application.

References

Apple Computer, Inc, (1992). "*Macintosh Human Interface Guidelines"* - Addison-Wesly Company

Ambrosini, L., Cirillo, V. & Micarelli, A. (1997) "*A Hybrid Architecture for User-Adapted Information Filtering on the World Wide Web*" – User Modelling: Proceedings of the Sixth International Conference, UM97, Vienna, New York: Springer Wien New York.

Bäumer, D., Cryczan, G., Knoll, R., Lilienthal, C.,Riehle, D. & Züllighoven, H. (1997) "*Frameworks Development For Large Systems*" – Communications of the ACM – Vol. **40**. Nº. 10. October

Benaki, E., Karkaletsis, V. A.& Spyropoulos, C. D. (1997) "*User Modeling in WWW: the UMIE Prototype*" - Proceedings of the Workshop: "Adaptive Systems and User Modeling on the Word Wide Web" - Sixth International Conference on User Modeling, Chia Laguna, Sardina.

Collins, D. (1995). "*Designing Object-Oriented User Interface*" - Benjamin/Cummings Publishing Company, Inc.

Fayad, M. & Schmidt, D., C. (1997) "*Object-Oriented Application Frameworks*" – Communications of the ACM – Vol. **40**. Nº. 10. Octuber.

Ferreira, S.B.L. (1999) "*Um framework para associar interfaces a usuários*". Doctoral thesis presented to Pontifícia Universidade Católica - 1999

Ferreira, S.B.L.; Carvalho, S.E.R.; Leite, J.C.S.P.; Melo, R.N. (1999) "*Requisitos Não Funcionais para Interfaces com o Usuário - O Uso de* Cores" Anais do 2º Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes Software IDEAS'99

Foley, J. D., Dam, A. V., Feiner, S. K. & Hughes, J. F. (1990) *Computer Graphics - Principles and Practice* - Addison - Wesley Publishing Company.

Gamma, E., Helm, R., Johnson, R. &Vlissides, J. (1995) "*Design Patterns – Elements of Reusable Object-Oriented Software*" - Addison-Wesley Publishing Company.

Gutkauf B., Thies, S. & Domik, G. (1997) "*A User Adaptive Chart Editing System Based on User Modeling and Critiquing*" – User Modelling: Proceedings of the Sixth International Conference, UM97, Vienna, New York: Springer Wien New York .

Hall, E. E.: "The Dance of Life"

Hix, D. & Hartson R. (1993) "*Developing User Interfaces: Ensuring Usability Through Product and Process*" - John Wiley & Sons.

Hofstede, G. (1991) "*Cultures and Organizations: Software of the Mind"* – McGraw Hill

Hoft, N. L. (1996) "*Developing a cultural model*" – publicado em "*International user Interfaces*" editado por Elisa M. del Galdo & Jakob Nielsen

Johnson, R. E. (1997) "*Frameworks (Component Patterns)*" – Communications of the ACM – Vol. **40**. Nº. 10. Octuber.

Kroeber, A. La. & Kluckhohn, C. (1954) "*Culture: A Critical Review of Concepts and Definitions*" – Random house – New Yorl.

Leite, J. C. S. P. (1995) *Engenharia de Requisitos* – Class Notes "*Engenharia de Requisitos*" - Deptº. de Informática da Pontifícia Universidade Católica do Rio de Janeiro.

Marins, C. (1987). *"Interfaces Inteligentes*" – Dissertação de mestrado submetida ao Departamento de Informática da Pontifícia Universidade Católica do Rio de Janeiro.

Murray, D. (1991) *"Modelling for Adaptivity"* – Mental Models and Human-Computer Interaction 2 – M. J. Tauber and D. Ackermann – Elsevier Science Publishers B. V.

Norman, D. A. & Draper, S. W. (1986) "*User Centered Design*" – Hillsdale, NJ: Lawrence Erlbaum.

Paranagama, P., Burstein, F. & Arnott, D. (1997) "*Modelling the Personality of Decision Makers for Active Decision Support*" – User Modelling: Proceedings of the Sixth International Conference, UM97, Vienna, New York: Springer Wien New York.

Pree, W. (1995). "*Design Patterns for Object-Oriented Software Development*" - Addison-Wesley Publishing Company.

Pressman, R. S. (1992) *Software Engineering - A Practitioner's Approach* - 3rd ed., McGraw-Hill, Inc.

Rich. E. (1979) *"User Modeling Via Stereotypes"* – Cognitive Science vol. **3** – 1.

Roberts, D., Berry, D., Isensee, S. & Mullaly J. (1998) "*Designing for the user with OVID: Bridging User Interface Design and Software Engineering"* – MacMillan Technical Publising - Software Engineering Series.
Shlaer, S. & Mellor, S. J. (1992) "*Object Lifecycles - Modeling the World in States*" Yourdon Press.
Shneiderman, B. (1998) "*Designing the User Interface – Strategies for Effective Human-Computer Interaction*" – Addison_Wesley.
Souza, C. S de. & Leite, J.C. (1005) "*Projeto de Interfaces de Usuário*" - Departamento de Informática - PUC-RIO.
Strachan, L., Anderson, J., Sneesby, M. & Evans, M. (1997) "*Pragmatic User Modelling in a Commercial Software System*" – User Modelling: Proceedings of the Sixth International Conference, UM97, Vienna, New York: Springer Wien New York
Trompenaars, F. (1993) "*Riding the Waves of Culture: Understanding Cultural Diversity in Business"* – Nicholas Brealey – 1993
Victor, D. A. (1992) "*International Business Communication",* - Harper Collins