5CONTECSI June 04 - 06, 2008 São Paulo, Brazil

5th International Conference on Information Systems and Technology Management
5º Congresso Internacional de Gestão da Tecnologia e Sistema de Informação
De 04 a 06 de Junho de 2008 - São Paulo - Brasil

**RF-875**

# USE OF INTERFACE PROTOTYPES AS AN IDIOM DURING REQUIREMENTS VALIDATION – A SEMIOTIC ANALYSIS

Carlos Eduardo Marquioni (Universidade Tuiuti do Paraná [UTP], Paraná, Brazil) – cemarquioni@uol.com.br; marquioni@rhealeza.com.br

This work suggests the use of interface prototypes as an idiom to establish communication between software technical professionals and non technical users, during the execution of the Validation process (of the Requirements Engineering), instead of using Use Case Diagrams to execute this process. The article uses the semiotic (peircean) as the main academic theory reference to analyze the required repertory to read and to understand these two kinds of artifact as the reason to this suggestion.

Keywords: Interface Design, Prototype, Requirements Engineering, Use Case Diagram, Validation Process.

# USO DE PROTÓTIPOS DE INTERFACE COMO IDIOMA DURANTE A VALIDAÇÃO DE REQUISITOS – UMA ANÁLISE SEMIÓTICA

Este trabalho propõe o uso de protótipos de interface como idioma para estabelecer comunicação entre técnicos e usuários não técnicos durante a execução do processo de Validação (da Engenharia de Requisitos), como alternativa à utilização dos Diagramas de Casos de Uso para executar este processo. O artigo utiliza a semiótica (peirceana) como referencial teórico principal para analisar o repertório requerido na leitura e compreensão desses dois tipos de artefato como justificativa para a sugestão apresentada.

Palavras-chave: Design de interfaces, Prototipação, Engenharia de Requisitos, Diagrama de Casos de Uso, Processo de Validação.

## INTRODUÇÃO[1]

Em meados dos anos 1990, foi difundido mundialmente um conjunto de notações de diagramas para representação gráfica de requisitos de software nomeado UML[2], que unificou as formas de modelagem da Orientação a Objetos de três teóricos de software (Ivar Jacobson, Grady Booch e James Rumbaugh). A UML é então uma

> [...] linguagem de modelagem [...] (basicamente gráfica) [...] [e] é certamente o aspecto chave para comunicação. Se você [técnico de software] deseja discutir seu design [a solução de software proposta] com alguém [inclusive usuários não técnicos], é a linguagem de modelagem que ambos necessitam entender[3] (Fowler & Scott, 1999: 01).

Assim, a notação UML definiu o conteúdo gráfico, os metadados e a sintaxe a utilizar para criar um conjunto de diagramas (também chamados de artefatos). Um destes artefatos é o Diagrama de Casos de Uso – ou *Use Case Diagram*, que é apresentado como um elemento para auxiliar "o cliente, os usuários e os desenvolvedores [técnicos de software] a obterem acordo no modo de uso do sistema[4]" (Jacobson; Booch & Rumbaugh, 2001: 40): este diagrama usualmente é a primeira transcrição técnica gráfica dos requisitos. Apesar dos casos de uso serem uma espécie de idioma, quando este tipo de diagrama é analisado sob uma perspectiva teórica comunicacional (particularmente a partir de conceitos de semiótica e repertório), podem ser originadas reflexões que indicam provável dificuldade no estalecimento de diálogo entre técnicos e usuários.

Outra forma de linguagem que é utilizada para desenvolvimento de software (embora em escala inferior) para comunicação de requisitos é a prototipação de telas: o uso em menor escala é justificado pois a criação de protótipos costuma ser apenas recomendada, o que talvez induza a indústria de software a considerar este tipo de artefato como *opcional* para o desenvolvimento de software.

Este trabalho propõe algumas reflexões acerca do uso de diagramas use case e protótipos como idiomas – particularmente para comunicação entre técnicos e usuários durante a execução do processo de Validação, que compõe a Engenharia de Requisitos. Para realização dos debates semióticos (ainda que em alto nível) são utilizadas formulações elaboradas pelo lógico norte-americano Charles Sanders Peirce[5] (1839-1914), e conceituação básica de *design*.

A condução das análises e reflexões ocorre ao longo de 3 seções principais, além desta breve introdução e das considerações finais: a primeira seção apresenta brevemente os processos da Engenharia de Requisitos (Kotonya, Sommerville) para contextualizar o processo de Validação e sua relevância no desenvolvimento de software. A segunda seção

---

[1] Todas as traduções apresentadas no texto são de minha autoria.
[2] *Unified Modeling Language* (Linguagem de Modelagem Unificada).
[3] Original: "modeling language […] (mainly graphical) […] It is certainly the key part for communication. If you want to discuss your design with someone, it is the modeling language that both of you need to understand".
[4] Original: "the customer, users, and developers agree on how to use the system".
[5] As referências a Charles Sanders Peirce utilizam o formato padrão de citação a este teórico, através dos chamados *Collected Papers* (CP) quanto a seus volumes e parágrafos. Assim, quando apresentado, por exemplo, CP 2.228, trata-se de uma citação extraída dos *Collected Papers,* volume 2 da edição americana, parágrafo 228.

apresenta conceitos gerais de casos de uso (Jacobson, Booch, Rumbaugh) e realiza uma breve análise de sentido semiótico e de repertório (Peirce, Ramsdell, Pignatari, Netto) em relação a estes diagramas. A terceira seção analisa o *design* de interfaces (Nadin, Galitz, Stone) como ferramenta comunicacional. As considerações finais resumem as principais análises realizadas e procuram apontar desdobramentos para este trabalho.

## 1 A ENGENHARIA DE REQUISITOS

Os requisitos de um produto de software correspondem a "descrições do comportamento do sistema[6], informações do domínio da aplicação[7], regras da operação do sistema ou especificações de uma propriedade ou atributo de um sistema[8]" (Kotonya & Sommerville, 1998: 6).

A Engenharia de Requisitos é a parte da Engenharia de Software através da qual os requisitos são abordados de forma sistematizada, englobando os processos de Levantamento (ou Elicitação), Análise, Especificação, Validação e Gerenciamento.

O Levantamento de requisitos é o processo executado para identificar, junto aos usuários, o problema a ser resolvido, o desempenho desejado do produto, restrições de equipamentos etc. É importante esclarecer que o Levantamento de requisitos

> [...] não envolve apenas perguntar aos usuários o que eles querem; ele requer uma cuidadosa análise organizacional, do domínio da aplicação e dos processos de negócio nos quais o sistema será utilizado[9] (Kotonya & Sommerville, 1998: 53).

Além de não ser um mero questionamento aos usuários, é possível afirmar que o Levantamento corresponde a uma atividade complexa em termos comunicacionais, pois indivíduos tipicamente com conhecimentos distintos procuram entendimento – mais exatamente, o técnico de software necessita compreender o discurso, o jargão do usuário para, a partir deste discurso, derivar os requisitos desejados do software.

No processo de Análise, o discurso dos usuários apresentado durante o Levantamento passa por considerações e avaliações técnicas. Ao longo da execução desse processo (Kotonya & Sommerville, 1998: 77) há categorização e organização dos requisitos segundo perspectiva técnica, explorando o relacionamento de cada requisito com todos os demais, examinando consistência, omissões e ambigüidades. A Especificação caracteriza o momento no qual a compreensão/interpretação dos requisitos que o técnico de software elaborou durante a Análise é formalizada tecnicamente. Dentre os idiomas para formalização podem ser citados diagramas técnicos (como os casos de uso), assim como os protótipos de interface.

A formalização criada para os requisitos durante a especificação deve ser validada (Kotonya & Sommerville, 1998: 87-88) para determinar se a compreensão do técnico acerca do discurso do usuário (e os requisitos resultantes) está correta. Para realizar esta revisão, o usuário é convidado à Validação dos requisitos em vários momentos: um desses momentos é após a confecção dos diagramas técnicos e, quando criados, dos protótipos.

---

[6] Nota do autor: o termo sistema é utilizado como sinônimo de software neste trabalho.

[7] Nota do autor: o domínio da aplicação corresponde à compreensão do contexto geral de negócio no qual o software será utilizado. Para detalhes, consulte (Kotonya & Sommerville, 1998: 54).

[8] Original: "[…] descriptions of how the system should behave, applications domain information, constraints on the system's operation, or specifications of a system property or attribute".

[9] Original: "[…] doesn't just involve asking people what they want; it requires a careful analysis of the organization, the application domain and business processes where the system will be used".

Se durante a execução do processo de Levantamento o técnico de software necessita derivar requisitos a partir do discurso do usuário, durante a Validação o usuário necessita compreender seu próprio discurso transcrito como requisito de acordo com o idioma *falado* pelo técnico – quando se trata de validar diagramas, o indivíduo que vai *ler* os requisitos transcritos desta forma necessita conhecer a notação em questão – novamente há execução de um processo complexo em termos comunicacionais. A relevância do processo de Validação fica evidente quando observado que é a partir do aceite, da aprovação dos requisitos pelo usuário durante a Validação, que os programas serão construídos. Com isso, se não for estabelecida comunicação eficiente, os requisitos do produto de software podem se tornar voláteis (ou instáveis), com solicitações de mudança frequentes[10].

As reflexões propostas neste artigo analisam o aspecto comunicacional envolvido na *leitura* e compreensão dos requisitos pelos usuários durante a execução do processo de Validação, quando estes requisitos estão descritos através da *linguagem* de diagramas de casos de uso e da *linguagem* de protótipos de interface.

## 2 DIAGRAMAS DE CASOS DE USO E COMUNICAÇÃO

Em projetos de software que formalizam requisitos utilizando a técnica da orientação a objetos com notação UML, o diagrama de casos de uso é elaborado com o objetivo de modelar as interações, "as ações de nossos usuários, e as respostas associadas do sistema [...] [, uma vez que] o que nós necessitamos que o software faça depende de como os usuários o acessam e o que esses usuários tentam fazer[11]" (Rosemberg & Scott, 2001: 35). Conceitualmente, "casos de uso são as funções que um sistema provê para agregar valor a seus usuários. Analisando sob a perspectiva de cada tipo de usuário, nós podemos capturar os casos de uso que eles necessitam para executar seu trabalho[12]" (Jacobson; Booch & Rumbaugh, 2001: 37).

Para representar o comportamento do produto e estabelecer comunicação, o *idioma* diagrama de casos de uso utiliza um conjunto de convenções que representa atores[13], casos de uso[14] e associações de comunicação[15]. Um exemplo de um diagrama de casos de uso é apresentado na figura 1.

---

[10] Para reflexões acerca da volatilidade de requisitos consulte (Marquioni, 2007a).

[11] Original: "our users' actions, and the associated system responses [...] what we need the software to do depends on how the users are accessing it and what the users are trying to do".

[12] Original: "use cases are the functions a system provides to add value to its users. By taking the perspective of each type of user, we can capture the use case that they need to do their work".

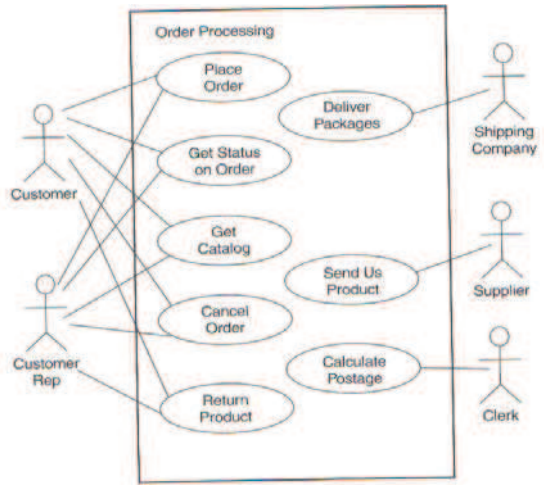
[13] Um ator é "qualquer coisa que tenha interface com seu sistema – por exemplo, pessoas, outros softwares, dispositivos de hardware, depósitos de dados ou redes. Cada ator define um papel específico. Cada entidade externa a seu sistema pode ser representada por um ou mais atores" (Schneider & Winters, 2001: 12). Original: "anything that interfaces with your system – for example, people, other software, hardware devices, data stores, or networks. Each actor defines a particular role. Each entity outside your system may be represented by one or more actors".

[14] Um caso de uso é representado graficamente por uma elipse e corresponde a "um comportamento do sistema que produz um resultado de valor mensurável para um ator. Casos de uso descrevem as coisas que os atores querem que o sistema faça [...] [e] deve ser a uma tarefa completa, segundo a perspectiva do ator. [...] O conjunto de todos os casos de uso irá descrever a funcionalidade completa do sistema sob o ponto de vista dos usuários" [grifo meu] (Schneider & Winters, 2001: 14). Original: a behavior of the system that produces a measurable result of value to an actor. Use cases describe the things actor want the system to do […] should be a complete task from the actor's perspective. […] All of the use case together will describe the complete functionality of the system from the user's point of view".

[15] A representação gráfica das interações entre os atores e os casos de uso é convencionada pela UML através de linhas chamadas associações de comunicação, ou simplesmente associações: "associações mostram com

Além da representação gráfica, há metadados no formato de textos explicativos associados aos casos de uso – este trabalho não avalia semioticamente os metadados que descrevem textualmente os casos de uso.

Figura 1 – Exemplo de Diagrama de Casos de Uso




Fonte: Schneider & Winters, 2001: 16

Os autores da UML afirmam que

> [...] há muitas razões pelas quais os casos de uso são bons, se tornaram populares e universalmente adotados [...] [e uma dessas razões seria o fato que] eles oferecem um meio sistemático e **intuitivo** [grifo meu] de capturar os **requisitos funcionais** [grifo no original] [...] com foco no valor agregado ao usuário[16] (Jacobson; Booch & Rumbaugh, 2001: 37).

Este suposto caráter intuitivo seria reforçado, ainda segundo os autores, pelo argumento que "usuários e clientes não necessitam aprender uma notação complexa[17]" (Jacobson; Booch & Rumbaugh, 2001: 37). Aqui merece destaque a não manutenção de uma morfologia na representação gráfica de requisitos[18]: desde o final da década de 1970 foram definidas, difundidas e utilizadas mundialmente pelo menos três notações para representar graficamente os requisitos, procurando estabelecer diálogo entre técnicos e usuários: DFD[19] segundo Gane e Sarson[20], DFD segundo Yourdon[21] e DeMarco[22], *Use*

---

quais atores o caso de uso se comunica [...]. A associação deve sempre ser binária, implicando um diálogo entre o ator e o sistema" (Eriksson *et al*, 2004: 66). Original: "associations show which actors the use case communicates with […]. The association should always be binary, implying a dialog between actor and system".

[16] Original: "there are several reasons why use cases are good and have become popular and universally adopted. […] they offer a systematic and intuitive means of capturing **functional requirements** […] with a focus on value added to the user".

[17] Original: "users and customers do not have to learn complex notation".

[18] Para reflexões acerca da não manutenção de uma morfologia para representação de requisitos, consulte (Marquioni, 2007a).

[19] Diagrama de Fluxo de Dados.

[20] Consulte (Gane & Sarson, [1977] 1983).

[21] Consulte (Yourdon, 1989).

[22] Consulte (DeMarco, [1978] 1989).

*Case Diagram* segundo Jacobson, Booch e Rumbaugh[23]. Uma vez que os diagramas para especificar requisitos podem ser considerados um idioma, é possível realizar uma comparação entre estas variações de representação com uma situação hipotética na qual as regras gramaticais da Língua Portuguesa mudassem uma vez a cada década: provavelmente haveria dificuldade para as pessoas *aprenderem* as novas regras em função das mudanças frequentes. No caso da produção de software, a dificuldade principal com as mudanças é do usuário não técnico. Em outros termos: a possibilidade de aprendizagem do diagrama, e sua suposta intuitividade para leitura podem ter sido comprometidas pelas constantes mudanças de notação.

Embora indubitavelmente os diagramas de casos de uso possibilitem uma visão **técnica** detalhada do comportamento do software, sua eficiência no estabelecimento de diálogo entre técnicos e usuários pode ser objeto de várias análises[24] e reflexões uma vez que, em termos semióticos, o leitor *entende* a partir do momento que possui códigos para entender, pois "toda cognição é mediada por idéias ou representações[25]" (Ransdell, 2005: s/p). O leitor também *entende* quando *deseja* entender pois "o *input* visual é fortemente afetado pelo tipo de necessidade que motiva a investigação visual, e também pelo estado mental ou humor do sujeito. **Vemos aquilo que precisamos ver**" [grifo meu] (Dondis, 2003: 133).

Ainda em termos semióticos, "a comunicação pressupõe a existência de um repertório[26] e de um código comuns a transmissor e receptor. Todo signo[27] novo, externo ao código, é 'ininteligível'" (Pignatari, 2003: 65). Encontrar, portanto, um mecanismo de representação adequado é um aspecto chave para que ocorra comunicação que minimize o risco que "aqueles que compreendem a notação não vão tirar *nenhuma informação falsa*" (Gombrich, [1959] 1995: 96). Para evidenciar que a intuitividade comunicacional não é tão fácil de ser obtida quanto se pode imaginar, pois a compreensão efetiva nem sempre é aquela desejada, Umberto Eco relata, em relação à construção civil (logo,em relação a um produto mais tangível que software),

> [...] um exemplo divertido mas conclusivo dado por Koenig a propósito de certas casas fabricadas para populações rurais pela 'Casa del Mezzogiorno' [segundo a tradutora da obra, Pérola de Carvalho, trata-se de estabelecimento de crédito e poupança para o Sul da Itália]. Postas em situação de dispor de casas modernas dotadas de banheiro com privada, as populações locais, habituadas a fazer suas necessidades no campo e despreparadas ante a chegada misteriosa das bacias sanitárias, usavam-nas como caixas de limpeza para azeitonas: suspendendo uma redezinha onde eram postas as azeitonas, davam a descarga e procediam assim à limpeza dos vegetais (Eco, [1968] 2005: 200).

---

[23] Consulte (Jacobson; Booch; Rumbaugh, [1999] 2001).

[24] Outras análises acerca da utilização de diagramas use case como idioma para comunicação podem ser encontradas em (Marquioni, 2007b).

[25] Original: "all cognition is mediated by ideas or representations"

[26] Nota do autor: "Entende-se por repertório uma espécie de vocabulário, de estoque de signos conhecidos e utilizados por um indivíduo [...] [Com isso] uma mensagem será ou não significativa (produzirá ou não mudanças de comportamento) conforme o repertório dessa mensagem pertencer ou não ao repertório do receptor" (Netto, 2003: 123).

[27] Nota do autor: para os objetivos deste trabalho, signo corresponde a "toda e qualquer coisa que substitua ou represente outra, em certa medida e para certos efeitos. Ou, melhor: toda e qualquer coisa que se organize ou tenda a organizar-se sob a forma de linguagem, verbal ou não" (Pignatari, 2004: 15). Assim, requisitos são signos uma vez que substituem as necessidades do usuário – por exemplo, através de diagramas de casos de uso.

Evidentemente o repertório é crítico, e no caso de software talvez seja necessário utilizar uma representação para os requisitos com a qual um usuário não técnico esteja minimamente familiarizado. Este trabalho considera a hipótese de a tela gráfica caracterizar este objeto mais familiar ao usuário, por ser através deste tipo de artefato que se estabelecem as interações com produtos de software no dia-a-dia dos usuários não técnicos (quando utiliza um *home-banking* ou a interface de um caixa bancário eletrônico, quando interage com um *game* de computador ou com algum sistema de informação no ambiente de trabalho etc); este contato cotidiano com interfaces talvez faça com que o usuário tenha repertório para *ler* e compreender melhor os requisitos quando estes estão descritos na forma de protótipos. Esta familiaridade remete ao conceito de modelos mentais:

> Como um resultado de nossas experiências e cultura, nós desenvolvemos modelos mentais de coisas e pessoas com as quais interagimos. Um modelo mental é simplesmente uma representação interna do entendimento atual de uma pessoa em relação a algo. [...] Modelos mentais são desenvolvidos gradualmente a fim de entender e explicar as coisas, tomar decisões, executar ações ou interagir com outra pessoa[28] (Galitz, 2002: 70).
>
> Ao confrontar um novo sistema computacional, usuários – geralmente de forma inconsciente – 'rodam' [executam] seu modelo mental existente para interpretar a nova situação, explicá-la, e predizer os resultados de suas interações com o sistema. O modelo mental ajuda os usuários a responderem questões como 'O que eu estou vendo agora?' 'O que o sistema acabou de fazer?' e 'O que eu fiz para que ele [o sistema] fizesse isso?'[29] (Stone *et al.*, 2005: 197).

Uma vez que

> [...] há um sentido real no qual o objeto – por exemplo, o território representado pelo mapa – pode ser dito como imediatamente perceptível, no sentido que, **se a representação é verdadeira, as características relevantes do objeto representado são também apresentadas imediatamente na consciência como sendo as características do objeto representado**[30] [grifo meu] (Ransdell, 2005: s/p)

, a hipótese desenvolvida em seguida é que a representação verdadeira, para software, talvez seja a interface, a tela através da qual o usuário interage com o produto. Com isso, um protótipo caracterizaria para as atividades executadas no ambiente de negócios (que constam do discurso do usuário durante a execução do processo de Levantamento de requisitos) o equivalente ao mapa para o território representado, e o efeito cognitivo em relação ao requisito seria formado para o usuário através do contato com esse protótipo.

---

[28] Original: "As a result of our experiences and culture, we develop mental models of things and people we interact with. A mental model is simply an internal representation of a person's current understanding of something. […] Mental models are gradually developed in order to understand something, explain things, make decisions, do something, or interact with another person".

[29] Original: "When confronted with a new computer system, users – often unconsciously – 'run' their existing mental model to interpret the new situation, explain it, and make predictions about the outcome of their interaction with the system. The mental model helps the users to answer questions such as 'What am I now seeing?' 'What did the system just do?' and 'What have I done to make it do that?'".

[30] Original: "there is a real sense in which even the other object – for example, the territory represented by the map – can be said to be immediately perceived, in the sense that, if the representation is a true one, the relevant features of the represented object are just as immediately present in consciousness as are the features of the representing object".

## 3 DESIGN DE INTERFACES E COMUNICAÇÃO COM USUÁRIOS

> "[...] nós percebemos o que fomos ajustados para interpretar[31]" (CP 5.185).

Apresentado usualmente como complementar aos diagramas técnicos – ou mesmo como uma alternativa de ciclo de vida de desenvolvimento –, a prototipação[32] não é novidade na indústria de software. A criação de protótipos de interface costuma ser justificada pelo fato que eles "possibilitam ao usuário entender como a interação homem-máquina irá ocorrer[33]" (Pressman, 2000: 277). O que esta seção do artigo pretende evidenciar é que talvez os protótipos devam ser elaborados com maior frequência que o usual, e eventualmente em um momento diferente daquele recomendado pelos autores da UML.

Talvez devido ao suposto caráter intuitivo que atribuem aos casos de uso, Jacobson, Booch e Rumbaugh propõem a criação do que chamaram "design lógico da interface-usuária[34]" (2001: 161), no qual recomendam que a prototipação seja realizada **após** a elaboração e validação dos casos de uso. Com isso, os requisitos formalizados graficamente através dos diagramas de casos de uso seriam apresentados aos usuários para validação e, uma vez com estes diagramas aprovados, poderia ser elaborado o protótipo de telas do produto de software.

A argumentação do repertório requerido para leitura dos casos de uso e a eventual familiaridade com interfaces apresentada anteriormente permite apontar como sugestão a inversão desta ordem de elaboração. Além disso, a não necessidade de realismo, precisão e legibilidade do protótipo (Galitz, 2002: 703-704) pode levar a supor que eventualmente a prototipação possa ser aplicada em níveis diferentes, em função do estágio no qual se encontra o desenvolvimento do produto de software.

Considerando que "Se há uma ciência da interface (interface de computadores ou qualquer outro tipo), então esta ciência é a semiótica, e a semiótica [...] estabelecida por Peirce parece apropriada [...][35]" (Nadin, 1988: 272), o trabalho apresenta a seguir conceitos de *design* que parecem adequados para a prototipação visando a validação dos requisitos pelos usuários: o que se pretende é apontar uma possibilidade preliminar, que deve ser refinada, para identificar como o protótipo pode ser sistematicamente elaborado (e não apenas sugerido) em conjunto com os diagramas técnicos, mas em ordem distinta daquela proposta por Jacobson, Booch e Rumbaugh.

Vale ainda destacar que não são considerarados a seguir alguns aspectos indubitavelmente relevantes do *design* de interfaces (aqueles associados a ergonomia, por exemplo), pois eles seriam tratados em um momento posterior, uma vez que haveria – segundo a proposta a seguir – níveis de prototipação. No contexto proposto neste trabalho,

---

[31] Original: "we perceive what we are adjusted for interpreting".

[32] "Um protótipo é primariamente um veículo para exploração, comunicação e avaliação. Sua principal função é o papel comunicacional que executa, e não a exatidão ou perfeição. [...] Um protótipo não necessita ser perfeitamente realístico, mas deve ser razoavelmente preciso e legível" (Galitz, 2002: 703-704). Original: "A prototype is primarily a vehicle for exploration, communication, and evaluation. […] Its major function is the communicative role it plays, not accuracy or thoroughness. […] A prototype need not be perfectly realistic, but it must be reasonably accurate and legible".

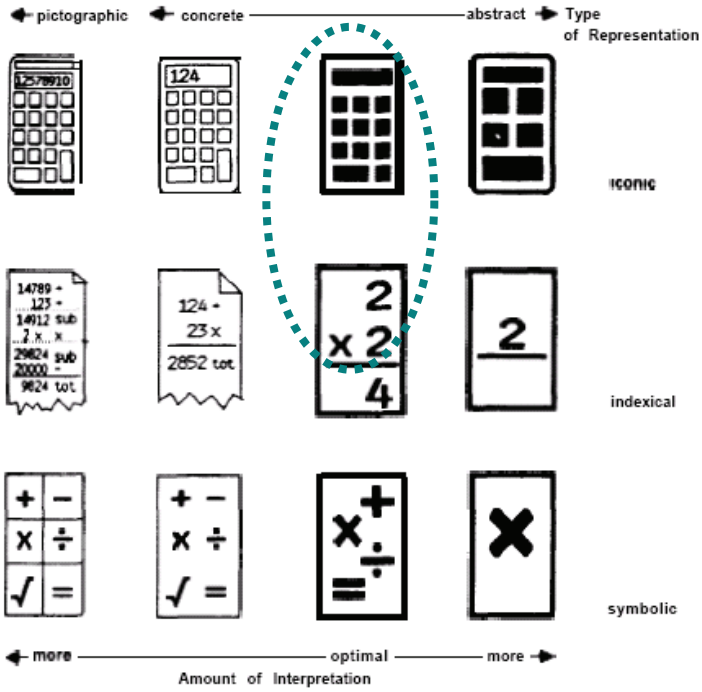
[33] Original: "enable a user to understand how human-machine interaction will occur".

[34] Original: "logical user-interface design".

[35] Original: "If there is a science of interface (computer interface or any other kind), then this science is semiotics, and the […] semiotics established by Peirce seems appropriate".

o que se deseja é procurar meios para facilitar que os usuários reconheçam as ações que executam no dia-a-dia através das telas sugeridas, bem como as inter-relações entre estas ações, possibilitando tratar os protótipos como um idioma para validação dos requisitos. A sugestão apresentada é elaborada a partir da figura 2.

Figura 2: Níveis de representação de um objeto




Fonte: [adaptação[36] a partir de] Nadin, 1988: 281

A análise da figura 2 permite observar que, à medida que a representação se torna mais pictográfica e icônica, ela se aproxima mais do objeto representado (possibilita melhor *reconhecimento* do objeto representado). Utilizando esta classificação proposta por Nadin baseada no modelo triádico da semiótica proposta por Charles Sanders Peirce, é possível supor que os diagramas técnicos de software (casos de uso, particularmente) talvez sejam de difícil compreensão por usuários não técnicos por serem demasiadamente simbólicos, com representação muito abstrata. Esta *distância* entre o objeto reconhecível e a representação, associada às mudanças de notação comentadas anteriormente podem dificultar o processo comunicacional.

Um trabalho de *design* de interface, quando empregado para subsidiar aspectos de ergonomia, provavelmente estará mais próximo da representação icônica e pictográfica. Para os objetivos deste trabalho, parecem adequados e suficientes os tipos de representação associados ao *design* de interface destacados na figura 2 para tornar os requisitos passíveis de validação pelo usuário, uma vez que

[36] O destaque (círculo pontilhado) não existe na figura original apresentada por Nadin. Este círculo foi adicionado pelo autor do trabalho e é explicado na continuidade do texto.

> Representações pictográficas são muito concretas, às vezes tão concretas que se o contexto muda e o usuário é apresentado a um pictograma diferente [...], ele terá dificuldades em 'utilizar' o signo. **O nível semiótico é alcançado quando a convenção do signo se torna evidente**[37] [grifo meu] (Nadin, 1988: 283-284).

Assim, nos estágios iniciais de um projeto, seria suficiente um protótipo preliminar (eventualmente apenas textual, mas com convenção estabelecida de forma que seja possível ao usuário não técnico estabelecer modelos mentais que remetam aos objetos de uma interface gráfica reconhecível). Em um segundo momento, o desenvolvimento conjunto de protótipos e diagramas técnicos poderia auxiliar na formalização do comportamento esperado do produto, apoiando o debate para validação dos requisitos, mas sem que estes diagramas sejam o alvo principal da Validação. Em um terceiro momento, estando os requisitos validados pelo usuário, poderia ser iniciada uma prototipação para avaliação de usabilidade e ergonomia, com utilização dos conceitos de *design* de interfaces para avaliar facilidade de uso, simular *navegação* entre interfaces etc. Em comum a estes três estágios estaria a utilização dos protótipos como idioma, como um "conjunto de signos ótimo para a interação entre duas entidades [o técnico de software e o usuário não técnico][38]" (Nadin, 1988: 273).

Destaque-se que o que se espera a partir da confecção do protótipo "não são valores verdadeiros ou falsos, como na lógica formal, mas ***significado***[39]" [grifo no original] (Nadin, 1988: 276): o protótipo passa a ser complementar aos diagramas de software no sentido que enquanto estes fazem o mapeamento **técnico** gráfico e textual (via metadados – não debatidos neste artigo) do comportamento das interfaces do produto de software para leitura por técnicos, o desenho da tela formaliza estes diagramas de maneira compreensível por leitores não técnicos. Este uso integrado permitiria que os casos de uso e sua especificação fossem utilizados pelos técnicos como subsídio adicional para explicar ou validar junto aos usuários o comportamento esperado da interface, mas estes artefatos técnicos deixam de ser o foco principal, e passam a ser um mecanismo de apoio à validação. Com esta abordagem, o diálogo tende a ser estabelecido a partir de um objeto que é percebido por um não técnico de forma *talvez mais natural* que um diagrama elaborado segundo uma notação técnica. Destaque-se que não se sugere aqui que esta *naturalidade perceptiva* esteja associada à intuição[40], mas a uma convenção que tende a ser relativamente fácil de estabelecer caso os usuários utilizem produtos de software (ou, em outros termos, estejam familiarizados com metáforas[41] de software – como a do *desktop*[42], por exemplo).

---

[37] Original: "Pictographic representations are very concrete, almost so concrete that if the context changes and the user is presented with a different pictogram […], he will have difficulties in 'using' the sign. The semiotic level is reached when the conventionality of the sign becomes evident".
[38] Original: "the optimal set of signs for the interaction between two entities".
[39] Original: "is not the value of true or false, as in formal logic, but ***meaning***".
[40] Vale ressaltar ainda que "criar e compreender mensagens visuais é natural até certo ponto, mas a eficácia [...] só pode ser alcançada através do estudo" (Dondis, 2003: 16). Isto porque "ver é um fato natural do organismo humano [...] [, mas] o fato de ver não garante a ninguém a capacidade de tornar compreensíveis [...] manifestações visuais" (Dondis, 2003: 137).
[41] Para informações sobre metáforas de software consulte (Galitz, 2002 e Stone *et al.*, 2005).
[42] Para informações sobre a metáfora do desktop, consulte (Johnson, 2001: 38-40).

## 4 CONSIDERAÇÕES FINAIS

Estabelecer comunicação entre técnicos de software e usuários não técnicos não é uma tarefa simples. Apesar de os teóricos de software afirmarem que a leitura da notação de casos de uso seja fácil (ou mesmo intuitiva), há fatores de ordem comunicacional e não técnica (associados a repertório, basicamente) que devem ser considerados.

É importante destacar que este trabalho não defende que os diagramas de casos de uso não sejam artefatos úteis para o desenvolvimento de software, mas sugere que talvez este tipo de diagrama não seja tão eficiente quanto divulgado quando se trata de comunicação entre técnicos e usuários não técnicos, devido ao fato de a notação remeter a uma representação demasiadamente simbólica e abstrata, que poderia dificultar que usuários não técnicos relacionem a representação a suas ações cotidianas. Contudo, quando se trata de comunicação entre técnicos de software (por exemplo entre analistas de sistemas, entre analistas e testadores etc), a comunicação através de casos de uso *pode* ser eficiente – desde que a convenção da notação seja estabelecida e aceita pelos participantes do diálogo.

Parece necessário aprofundar as reflexões acerca do uso de protótipos como idioma para validação de requisitos, particularmente estabelecendo uma forma de especificação que possibilite elaborar modelos mentais adequados, eventualmente de forma gradual. Para isso, o *design* de interfaces, associado à semiótica peirceana constituem referenciais teóricos fundamentais, uma vez que é importante considerar que a comunicação eficiente em software não depende da intenção dos autores de uma notação para criação de diagramas, mas de complexos processos comunicacionais que ultrapassam as fronteiras técnicas de software.

### Referências

DEMARCO, Tom. **Análise Estruturada e Especificação de Sistema**. Rio de Janeiro: Campus, 1989.

DONDIS, Donis A. **Sintaxe da linguagem visual**. São Paulo: Martins Fontes, 2003.

ECO, Umberto. **A estrutura ausente**. São Paulo: Editora Perspectiva, 2005.

ERIKSSON, Hans-Erik *et al*. **UML 2 Toolkit**. Indianapolis: Wiley Publishing Inc, 2004.

FOWLER, Martin; SCOTT, Kendall. **UML Distilled – Applying the standard object modeling language**. Boston: Addison-Wesley, 1999.

GALITZ, Wilbert O. **The essential guide to user interface design**. New York: John Wiley & Sons Inc, 2002.

GANE, Chris; SARSON, Trish. **Análise Estruturada de Sistemas**. Rio de Janeiro: LTC – Livros Técnicos e Científicos, 1983.

GOMBRICH, E. H. **Arte e Ilusão – Um estudo da psicologia da representação pictórica**. São Paulo: Martins Fontes Editora, 1995.

JACOBSON, Ivar; BOOCH, Grady; RUMBAUGH, James. **The Unified Software Development Process**. Boston: Addison-Wesley, 2001.

JOHNSON, Steven. **Cultura da Interface – Como o Computador Transforma Nossa Maneira de Criar e Comunicar**. Rio de Janeiro: Jorge Zahar Editor, 2001.

KOTONYA, Gerald; SOMMERVILLE, Ian. **Requirements Engineering – Processes and Techniques**. New York: John Wiley & Sons Inc, 1998.

MARQUIONI, Carlos Eduardo. **Requisitos de Software Instáveis – Uma análise conceitual de um problema prático**. In: CONTECSI (Congresso Internacional de Gestão de Tecnologia e Sistemas de Informação), 4º, 2007, São Paulo (USP). Anais. São Paulo: CD-ROM, 2007a.

MARQUIONI, Carlos Eduardo. **Comunicação através de Diagramas de Casos de Uso no Desenvolvimento de Software – Uma breve análise de sentido**. In: CONECO (Congresso de Estudantes de Pós-Graduação em Comunicação), 2º, 2007, Rio de Janeiro (PUC-Rio). Anais. Rio de Janeiro: CD-ROM, 2007b.

NADIN, Mihai. Interface design: a semiotic paradigm. In:___**Semiotica 69-3/4**. Amsterdam: Mouton-deGruyter, 1988, pp. 269-302. Disponível em: http://www.cs.ucsd.edu/users/goguen/courses/nadin.pdf. Acesso em: 03 fev. 2007.

NETTO, José Teixeira Coelho. **Semiótica, Informação e Comunicação**. São Paulo: Editora Perspectiva, 2003.

PIGNATARI, Décio. **Informação Linguagem Comunicação**. Cotia: Ateliê Editorial, 2003.

PRESSMAN, Roger. **Software Engineering – A Practitioner's Approach – European Adaptation**. London: McGraw Hill International Limited, 2000.

RANSDELL, Joseph. The epistemic function of iconicity in perception – version 2.0 (12-24-2005). **Arisbe**, Lubbock, s/n, dez. 2005. Disponível em: http://www.marketleadersgroup.com/cspeirce/menu/library/aboutcsp/ransdell/epistemic.htm. Acesso em: 03 fev. 2007.

ROSENBERG, Doug; SCOTT, Kendall. **Applying use case driven object modeling with UML – An annotated e-commerce example**. Boston: Addison-Wesley, 2001.

SCHNEIDER, Geri; WINTERS, Jason P. **Applying Use Cases – A practical guide**. Boston: Addison-Wesley, 2001.

STONE, Debbie; *et al*. **User interface design and evaluation**. San Francisco: Morgan Kaufmann Publishers, 2005.

YOURDON, Edward. **Modern Structured Analysis**. New Jersey: Prentice Hall, 1989.