Metodologias Ágeis: Um Novo Paradigma de Desenvolvimento de Software

Renata Bastos Ferreira[1], Francisco de Paula Antunes Lima[2]

[1]Departamento de RH – ATAN Ciência da Informação
Tel.: (31) 3289.7769 – Belo Horizonte – MG – Brasil

[2]Departamento de Engenharia de Produção – UFMG
Tel.: (31) 3499.4895 – Belo Horizonte – MG – Brasil

renata.bastos@atan.com.br, fpalima@dep.ufmg.br

Abstract: *The Software Engineering for a long time has been facing problems related to the delay in the delivery of projects, the extrapolated budget, the costumers and users discontentment, besides the conflicts and detritions among annalists and costumers. Aiming at better results, the IT companies are adopting software development methodologies more flexible and propitious to the frequent changes, beyond more interaction during the whole project between the users and the system itself. These methodologies are considered agile in contraposition to the heavy methodologies that, traditionally, prevailed in the area but have shown themselves ineffective and unproductive.*

Resumo: *A Engenharia de Software há muito vem enfrentando problemas relativos a atraso na entrega de projetos, orçamento extrapolado, insatisfação de clientes e usuários, além de conflitos e desgastes entre analistas e clientes. Visando a melhores resultados, as empresas de TI estão adotando metodologias de desenvolvimento de software mais flexíveis e propícias às freqüentes mudanças, além de mais interação durante todo o projeto entre os usuários e o próprio sistema. Estas metodologias são chamadas de ágeis em contraposição às metodologias pesadas que, tradicionalmente, predominaram na área, mas que se mostraram ineficientes e improdutivas.*

1. Introdução: definição do escopo de projeto

Um problema freqüente da concepção de sistemas informatizados é a dificuldade de se definir, de início, o escopo do projeto (funcionalidades, requisitos etc.). No momento da definição do escopo, o cliente e os diversos usuários do sistema geralmente não sabem o que querem, isto é, parecem ser incapazes de definir detalhadamente as funções e procedimentos operacionais do sistema, o que comumente é atribuído ao desconhecimento, por parte do cliente, (1) da tecnologia de informática (linguagem técnica e possibilidades das aplicações), (2) de características do próprio processo a ser informatizado ou (3) de suas próprias necessidades, sempre cambiantes ao longo do desenvolvimento do sistema. Em conseqüência, os programas desenvolvidos não atendem ou atendem mal às necessidades dos clientes, gerando frustração, perda de tempo em retrabalho e, inevitavelmente, conflitos e desgaste das relações entre os contratantes. Os estudos de avaliação das promessas da informática colocam em dúvida se os ganhos de produtividade são efetivos ou tão significativos como se pretende (sobre

isso, ver LOJKINE, 1996; DERTOUZOS, 1997 e vários estudos em KLING, 1996).
De modo geral, a tendência atual de diversificação de produtos para atender mercados fragmentados, mediante uma maior customização, depende do reconhecimento preciso das necessidades reais dos usuários. Essa adequação pode não ocorrer devido a problemas de comunicação, quando se adota um processo de desenvolvimento seqüencial, separando a fase de concepção da operação em escala real. As conseqüências são bem conhecidas: «canibalismo» entre produtos semelhantes, não atendimento das necessidades dos clientes, retrabalho, atrasos e perda de produtividade em geral. Em seus estudos de caso, Wheelwright e Clark (1992) revelaram o modo reativo de resolução de problemas adotado pelos gerentes de projeto (*«after-the-fact» problem solving*), ao invés de prevenção de problemas por meio de pré-projetos e planejamento mais efetivos. Esse comportamento reativo diminui o poder dos gerentes para influenciar as soluções de projeto.

O problema do escopo mal definido acontece em todos os tipos de projeto, de bens de consumo às instalações e equipamentos industriais, de natureza mais técnica. Por exemplo, a análise dos conflitos entre clientes e fornecedores de equipamentos automáticos na França revelou o papel crucial das especificações («*cahier des charges*»), que estavam sujeitas a múltiplas interpretações. De um lado, os clientes estavam insatisfeitos com equipamentos que não atendiam suas necessidades, de outros os construtores reclamavam da imprecisão dos requisitos das encomendas. Após estudos conduzidos por um consórcio entre clientes, fornecedores, entidades de classe e pesquisadores, chegou-se à conclusão que a especificação não é uma questão puramente técnica (a ser resolvida, por exemplo, com um *check list* aperfeiçoado), uma vez que os problemas decorriam, sobretudo, das relações sociais entre cliente e fornecedor (TIGER, 1990). Essa tendência de associação dos clientes e dos usuários finais ao processo de desenvolvimento de produtos em geral, e de softwares em particular, surge como uma alternativa para se melhoras as especificações, na forma de uma co-responsabilização.

No caso dos projetos de automação estudados por nós, um dos técnicos associa os prejuízos sofridos pela empresa à precariedade da especificação dos requisitos de projeto: *"O foco principal do prejuízo não está no desenvolvimento técnico do projeto, pois este é muito bom, mas no pouco contato com o cliente e na especificação rápida"* (programador). O foco de análise deve, portanto, ser redirecionado para as relações entre fornecedores e clientes. Apesar dos métodos de desenvolvimento de softwares preverem a avaliação permanente dos clientes, os procedimentos e as técnicas de participação ainda estão distantes de se conseguir uma associação efetiva. As recomendações, por vezes, se limitam a aspectos comportamentais, notadamente na capacidade comunicativa: uma publicação já antiga, nos ensina que o analista tem que ter "talentos combinados de líder, professor e psicólogo (...), saber entender as queixas dos clientes, reclamações e opiniões do usuário" (CARVALHO, 1988, p. 175) e,na fase de implementação, detectar "desvios, erros, atitudes perigosas"(*ibid*. p. 201). Mais recentemente, o surgimento das metodologias ágeis se apresenta como uma possível solução para integrar o usuário ao processo de desenvolvimento de softwares (OLIVEIRA, 2003), às vezes combinadas com metodologias top-down (BOEHM e TURNER, 2004).

2. Quadro de referência: associando abordagens descendentes e ascendentes

Os métodos tradicionais de engenharia possuem em comum o fato de serem um processo descendente (*top-down*), que parte de uma lista de requisitos ou do «conceito» de um produto, progressivamente concretizado ao longo do processo de desenvolvimento do produto (PDP). Isso torna o PDP, predominantemente, seqüencial e linear. Mesmo os reajustes, nos momentos de iteração e de avaliação, são correções de percurso em função de um objetivo pré-determinado, não influenciando de modo significativo o escopo inicial. A distância entre o produto oferecido e as necessidades reais dos clientes e usuários, verificada na prática, coloca em questão a eficácia desse processo seqüencial. Ainda que esta seqüência seja quebrada em vários ciclos iterativos, o procedimento permanece *top-down*, partindo do modelo conceitual para os detalhes das situações de uso. "*Uma outra abordagem, que pode ser qualificada de ascendente ou* bottom-up, *parte do princípio de que a consideração das condições de realização da atividade de trabalho, desde as etapas iniciais de um projeto, pode ajudar a esclarecer as escolhas a serem feitas em relação à concepção dos sistemas técnicos e dos postos de trabalho. Trata-se de uma abordagem complementar à abordagem descendente ou* top-down" (DUARTE, 2000).

O princípio desta abordagem é o inverso da anterior: as fases subseqüentes influenciam nas decisões tomadas em fases iniciais, o que instaura uma temporalidade específica ao PDP, dando mais força às linhas de retroação. Assim, a possibilidade de (re)definir o escopo durante o PDP depende das diversas combinações entre estratégias de projeto descendentes e ascendentes, onde se criam condições de antecipação e de correção, sem necessidade de um grande retrabalho. Os esquemas abaixo permitem compreender melhor as possibilidades de combinação entre métodos descendentes e ascendentes, configurando-se espaços de manobra diferentes.

A Figura 1 representa as duas abordagens e o espaço possível para a consideração de determinantes das situações futuras de utilização do produto (operação, manutenção...). A área sob a curvas indica o espaço de manobra para a equipe de projeto efetivar alterações e encontrar soluções alternativas, o que depende da capacidade de antecipação dos compromissos a serem estabelecidos no decorrer do PDP.

Na Figura 2 estão representadas diferentes possibilidades de articulação entre as abordagens ascendentes e descendentes. A **linha A** é típica do PDP tradicional, quando as informações sobre as situações reais de utilização aparecem apenas no final do projeto ou após a partida (*start-up*). O espaço de manobra para se estabelecerem compromissos favoráveis é reduzido, com efeitos negativos sobre as condições de trabalho e sobre a produtividade. A **linha B** representa um PDP em que as abordagens ascendentes e descendentes, apesar de iniciarem simultaneamente, só se encontram ao final do projeto, sem interações durante o desenvolvimento. Seus inconvenientes são os mesmos da situação **A**: "*as margens de manobra para mudanças nos projetos são muito reduzidas, pois as principais decisões já foram tomadas*" (DUARTE, 2000).

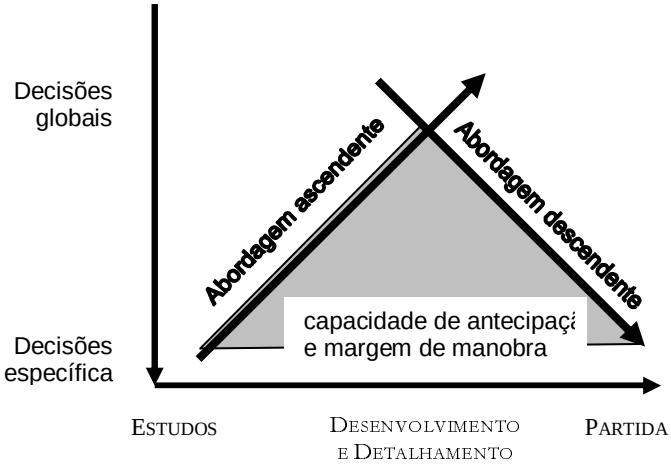

FIGURA 1: Articulação entre as abordagens descendentes e ascendentes. Adaptado de Duarte, 2000.

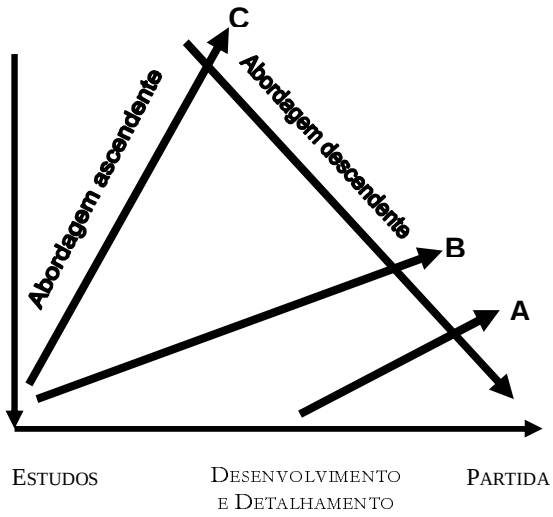

FIGURA 2: Diferentes articulações entre abordagens descendentes e ascendentes. Adaptado

Finalmente, na abordagem ascendente representada pela **linha C**, torna-se possível a reflexão entre os agentes envolvidos no projeto (gerentes, técnicos, clientes e usuários) a respeito de suas opções principais. "*Essa combinação das abordagens descendentes e ascendentes permite a descrição e compreensão das inter-relações entre os diferentes componentes do projeto, ampliando a capacidade de antecipação e reduzindo, ao longo do processo de concepção, as incertezas relativas à eficácia do funcionamento futuro."* (DUARTE, 2000) Assim, gere-se melhor os efeitos de irreversibilidade implicados em qualquer decisão.

Evidentemente, a simples enunciação desse princípio não cria as condições necessárias para a antecipação, apenas assinala a necessidade de que isso ocorra. Se viabilizado, problemas clássicos em desenvolvimento de softwares, como a prototipagem, em especial a tendência a transformar um protótipo que consome grande parte dos recursos em produto (PREECE, ROGERS e SHARP, 2005), poderiam ser resolvidos ou amenizados. A questão se resolve na medida em que as avaliações se tornam mais realistas e que a experiência dos usuários diretos é incorporada na especificação dos softwares. Vinck et al. (1999) mostram, em uma análise comparativa, como a incorporação da experiência dos técnicos de chão-de-fábrica permite desenvolver um sistema CAD-CAM (aplicado em projetos de matrizes de extrusão de perfis de alumínio) bem mais eficiente do que o sistema desenvolvido apenas com os conhecimentos de engenharia. O primeiro software não definia por si só as saídas, mas deixava espaço para ajustes feitos pelos técnicos e supervisores de produção que amenizavam, por exemplo, um ângulo da matriz "demasiadamente agudo".

3. Como a metodologia influência no sucesso ou no fracasso do projeto

3.1 Metodologias tradicionais ou pesadas

Dentre as abordagens descendentes, encontram-se as chamadas metodologias tradicionais ou pesadas, devido ao seu tamanho e à dificuldade de serem implementadas na íntegra (OLIVEIRA, 2004). Historicamente, elas predominaram e ainda são muito utilizadas nos projetos de desenvolvimento de software (OLIVEIRA, 2004). O que melhor caracteriza estas metodologias é a separação bem rígida das fases de projeto, que consistem em: Levantamento de Requisitos, Análise, Desenho, Implementação, Testes e Implantação. Cada fase tem suas especificidades e possuem entre si interdependência, isto é, a próxima fase só começa quando a anterior estiver pronta. Isto significa que o processo é seqüencial e linear, no qual cada fase deve ser concluída antes de passar para a próxima etapa.

Estas metodologias seguem o modelo tradicional da engenharia como a Engenharia Civil ou Elétrica, onde o desenvolvimento do sistema é dividido em duas fases distintas: o sistema é primeiramente planejado e, depois do planejamento pronto, o sistema é construído. A fase de concepção do projeto ou planejamento é realizada por uma equipe de analistas que, mediante reuniões e discussões com os clientes ou usuários, estrutura como o sistema vai ser, quais serão suas funcionalidades e características. Nesta fase é necessário grande trabalho de documentação, pois este será o alicerce do projeto; com isso muito tempo é gasto para que haja um levantamento de requisitos completo e claro, a fim de evitar o aparecimento de novos requisitos ou funcionalidades durante a fase de desenvolvimento do projeto. Por este motivo é recomendável que os analistas de sistema tenham como característica de personalidade a facilidade de comunicação. *"É indispensável ao trabalho dos analistas de sistemas uma boa aptidão para o diálogo, para a expressão, comunicação verbal, escuta e para a interação, respeitosa e profunda, com pessoas que vivem e trabalham no meio ambiente onde a fábrica de informação vai ser instalada"* (CARVALHO, 1988, p. 143). Como seu trabalho é uma espécie de ponte entre as necessidades e problemas humanos e a computação, "*o analista tem de saber entender as queixas, reclamações e opiniões do usuário, filtrando-as e sintetizando-as em termos de necessidades efetivas e economicamente justificáveis*" (*ibid*., p. 175).

Os analistas de sistemas consideram esta fase de definição do escopo a mais importante do projeto, pois dependendo da sua qualidade, o projeto terá sucesso ou não. Quanto mais imprevistos surgirem durante o desenvolvimento, maior será o retrabalho e conseqüentemente o tempo de realização do projeto, uma vez que uma modificação na base do sistema, isto é, no escopo do projeto, acarreta mudanças estruturais muitas vezes complexas e impactantes para o desenvolvimento do sistema, comprometendo assim os prazos, o preço e qualidade do serviço. É como se um prédio já tivesse todo pronto e, de repente, descobre-se um problema na sua sustentação, obrigando os engenheiros a desconstruir boa parte do trabalho e refazê-lo novamente. Além dos prejuízos com o grande retrabalho e os atrasos, outra característica marcante destas metodologias preditivas é a punição com faturamentos de extra-escopo quando o conjunto inicial de requisitos é modificado, o que leva a conflitos entre clientes e analistas.

Mas pior que o aumento do orçamento originado pelo retrabalho ou pelo

aumento do escopo devido a requisitos não especificados inicialmente, é a ineficiência do sistema que não atende às necessidades dos clientes, situação esta comum de acontecer devido à estrutura seqüencial e linear do projeto. Os clientes ou usuários só participam da fase inicial do projeto (concepção) e da final, que é a implantação do sistema. Durante o desenvolvimento propriamente dito não há interação alguma entre os usuários e o sistema, impedindo assim as correções graduais e as implementações de novas funcionalidades correspondentes às necessidades dos usuários. Quando o projeto está todo pronto é que os usuários terão a oportunidade de experimentá-lo, e, caso percebam problemas, há muita resistência dos analistas em corrigi-los, deixando de fora muitas necessidades dos usuários. O relato de um operador, usuário final de um sistema de automação, retrata muito bem este problema: *"quando eles vão embora fica um monte de problema pra trás, a gente tem que ficar arrumando e burlando o sistema, senão a gente não trabalha e a produção não sai como deveria"* (operador de fábrica).

Devido a esses problemas, a Engenharia de Software tem se empenhado na construção de novas metodologias de desenvolvimento, dentre elas as metodologias ágeis, que parte de premissas contrárias àquelas das metodologias pesadas.

3.2. Metodologias ágeis

Uma premissa fundamental das metodologias ágeis é o reconhecimento da dificuldade do usuário saber de antemão as funcionalidades que gostaria que o sistema tivesse. Por isso, essas metodologias adotam a abordagem ascendente, isto é, criam condições favoráveis para as interações e as retro-alimentações entre usuários e o sistema durante todo o projeto, uma vez que são as necessidades reais dos usuários, e não o "conceito" do sistema ideal, o ponto chave do sucesso do projeto. As necessidades dos usuários são, por outro lado, sempre mutáveis, isto é, não são definidas (nem definíveis) *a priori*, mas vão-se desenhando ao longo do projeto. Donde a necessidade de uma abordagem que rompa com a temporalidade linear do projeto organizado em fases seqüenciais (ver, por exemplo, a metodologia do «objetos intermediários» (CAMPOS, 2002; VINCK, 1999), que aperfeiçoa as simulações e modelos convencionais de prototipagem).

> "*A percepção que os usuários têm de suas necessidades também evolui à medida que eles conhecem o sistema. É difícil compreender o valor de uma determinada funcionalidade até que ela seja efetivamente usada, principalmente porque não se pode requerer de um usuário comum a mesma capacidade de abstração que um desenvolvedor possui ao olhar um conjunto de requisitos"* (OLIVEIRA, 2003, p. 16).

Com isso, as metodologias ágeis são estruturadas de modo a atender a natureza mutável e dinâmica do processo de concepção do sistema.

Diferentemente das metodologias pesadas que possuem fases bem separadas e delimitadas, nas metodologias ágeis, as fases de concepção e desenvolvimento interagem durante todo o projeto, possibilitando desse modo uma interação constante entre os usuários e os analistas, ou seja, entre os profissionais da concepção e da operação, como ocorre nos projetos na "produção enxuta": *"as equipes de projeto japonesas permanecem no local até bem depois da implantação e operam contínuas mudanças, acumulando, assim, novos conhecimentos, que serão transferidos aos sistemas através das modificações que fazem"* (LOJKINE, 1995, p. 248). Tem-se, assim, um modelo não-linear, que enfatiza a retroação de fases posteriores sobre as

fases anteriores e interação concepção/produção.

As metodologias ágeis propõem que os projetos devam ser conduzidos de forma adaptativa, isto é, feito através de desenvolvimento iterativo e interativo. A idéia central é trabalhar com iterações curtas. Cada iteração entrega ao seu final um produto completo e pronto para ser usado, que contém a implementação de um novo subconjunto de características. O uso de iterações curtas permite aos usuários e clientes fazerem uma avaliação do sistema logo que uma versão inicial é colocada em produção. Neste momento, usuários, clientes e desenvolvedores decidem sobre quais características devem ser adicionadas, quais devem ser modificadas, e até, quais devem ser retiradas do sistema. O sistema é desenvolvido da forma mais iterativa possível.

O escopo de cada iteração é pequeno e contempla somente as funcionalidades que aquela iteração deverá possuir, deixando para iterações futuras o restante dos requisitos. O orçamento segue a lógica das iterações, isto é, cada iteração terá um custo definido e pago após a sua conclusão. Obtém-se com isso facilidade na negociação, tendo em vista que o aparecimento de novas funcionalidades será negociado na próxima iteração. Para o cliente esses procedimentos são também vantajosos, pois ele terá maior transparência e visibilidade do processo, poderá acompanhar seu desenvolvimento e seus investimentos. Mas nem sempre foi fácil convencer o cliente deste método, que começa a ser posto em prática em projetos de automação industrial em uma empresa brasileira, antes acostumados aos projetos baseados na metodologia pesada. A dificuldade era convencer o cliente a pagar parcelado, ao invés de comprar o sistema no valor global de 1000,00 reais (valor fictício), ele pagaria 10 iterações de 100,00 reais. Por mais interessante que possa parecer, causou muita estranheza no início. Atualmente os clientes que participaram de projetos baseados nas metodologias ágeis estão satisfeitos e demandando sempre por novas iterações, o que tem gerado contratos permanentes. O sucesso desta metodologia está relacionado aos sistemas mais adaptados às necessidades dos usuários, ao cumprimento dos prazos e do orçamento, uma vez que as correções “esperadas” de cada iteração não impactam mais em grandes retrabalhos. Pelo contrário, as modificações aqui são desejadas, tanto que as iterações servem para estimular o contato do usuário com o sistema justamente para possibilitar o máximo de correções, evitando-se assim que, ao final, ele venha a apresentar ineficiências. Quanto mais cedo os problemas aparecerem, melhores são as chances de se obter um sistema eficiente, no tempo estimado e no preço acordado. Problemas clássicos na informática (apesar de ser uma prática relativamente nova), como não cumprimento de prazos, má qualidade dos softwares e orçamentos estourados (CARVALHO, 1988), são contornados com esta nova metodologia.

4. Conclusão

A solução dos problemas relatados aqui é fundamental para evitar o retrabalho decorrente de informações lacunares ou inadequadas, obtidas durante o projeto de controle automático. Podemos resumir o princípio geral de uma abordagem alternativa na gestão da irreversibilidade das decisões tomadas ao longo do PDP. A possibilidade de (re)definir o escopo depende das diversas combinações entre estratégias de projeto descendentes e ascendentes, onde se criam condições de antecipação e de correção, minimizando o retrabalho. Isto implica uma nova temporalidade do PDP, gastando-se mais tempo na especificação, para se economizar no retrabalho e no *start-up*, e a realimentação mais eficaz dos momentos de decisão com informações de campo

(métodos ascendentes). Isto é possível, em parte, com o aperfeiçoamento dos métodos de obtenção e validação das informações de campo e dos testes intermediários (testes de plataforma, objetos intermediários...).

O PDP é uma criação coletiva. No entanto, a mistura de papéis pode ser prejudicial, transformando o fornecedor de softwares em um mero executante. A gestão dessa relação é complexa precisamente porque exige uma definição precisa dos papéis de cada agente, assim como do conhecimento especializado que cada um possui enquanto especialista em um certo campo, ao mesmo tempo em que esses diversos especialistas devem cooperar entre si, levando em conta o ponto de vista do outro e traduzindo suas necessidades em exigências de projeto.

Propostas específicas sobre uma nova abordagem foram detalhadas em Ferreira (2004). Ressaltamos, aqui, apenas o princípio geral: as especificações técnicas devem se inserir em um «contrato social» mais amplo, cujo objetivo é definir as condições de relacionamento dos agentes, em especial entre cliente e fornecedor. Nesse sentido, as metodologias ágeis vêm favorecer a confrontação de lógicas diferentes entre usuários e analistas durante toda a fase de desenvolvimento do projeto, de modo a manter o mais aberta possível, o máximo de tempo possível, a definição dos objetivos e dos requisitos do projeto (TIGER, 1991). Seja pela antecipação, seja pela retroalimentação, o que vai acontecer interfere sobre o já acontecido, tornando o PDP menos determinado por decisões irreversíveis e criando condições mais favoráveis para «se projetar certo da primeira vez».

Referências

BAINBRIDGE, L. (1999). “Verbal reports as evidence of the process operator's knowledge”, In: Int. Human-Computer Studies, n.51, p.213-238.

BECK, K. (2000). “Extreme programming explained: embrace change”, Boston: Addison-Wesley.

BOEHM, B. E TURNER, R. (2004). “Balacing agility and discipline: evaluating and integrating agile and plan-driven methods”, Proceedings of the 26th Int. Conf. On Soft. Eng. (ICSE'04).

CAMPOS, N. (2002). “Equipes multifuncionais de projeto”, Dissertação Mestrado. UFMG, Belo Horizonte.

CARVALHO, L.C. (1988). “Análise de sistemas”, Rio de Janeiro: Livros Técnicos e Científicos.

COLLINS, H.M. (1992). “Artificials experts”, Seuil: MIT Press.

DERTOUZOS, M. (1997). “O que será? Como o novo mundo da informação transformará nossas vidas”, São Paulo, Cia das Letras.

DREYFUS, H.; DREYFUS, S (1986). “Mind over machine”, New York: Free Press.

DUARTE, F. (2000). “Complementaridade entre ergonomia e engenharia em projetos industriais”, In: DUARTE, F. (org.) Ergonomia e projeto na indústria de processo contínuo. Rio de Janeiro, Lucerna, pp. 11-21.

FERREIRA, R. B. (2004). “Diálogo de surdos: a difícil explicitação do saber entre programadores de software e operadores de fábrica”, Dissertação de mestrado pela

Engenharia de Produção: UFMG.

KLING, R. (1996). “Computerization and Controversy”, Academic Press

LOJKINE, J. (1996). “A revolução informacional”, São Paulo: Cortez.

MCGRAW, K.L.; HARBISON-BRIGGS, K. (1989). “Knowledge acquisition”, Englewood Cliffs: Prentice Hall.

PREECE, J.; RIGERS, Y.; SHARP. H. (2005). “Design de interação”, Porto Alegre, Bookman.

OLIVEIRA, E. S. (2003). “Uso de Metodologias Ágeis no Desenvolvimento de Software”, Monografia apresentada no Programa de Pós-Graduação em Engenharia de Software da UFMG.

SILVA, C. & LIMA, F. (2000). “A objetivação do saber prático em sistemas especialistas”, In: DUARTE, op. cit.

TIGER, H. (1991). “L'établissement du Cahier des Charges des équipements automatiques”, Colloque A.R.R.P., MRT, Janvier 1991.

VINCK, D. (1999). “Ingénieurs au quotidien”. Grenoble, PUG.

WHEELWRIGHT, S. C. & CLARK, K. B. (1992). “Revolutioning product development”, New York, Free Press.