Uma Análise Realista da Utilização do Enfoque da Prototipagem Rápida no Desenvolvimento de Sistemas de Informação.

Marcos R. S. Borges

Núcleo de Computação Eletrônica
Universidade Federal do Rio de Janeiro
Caixa Postal 2324
20001 - Rio de Janeiro
BRASIL

RESUMO: Neste artigo discute-se as condições julgadas necessárias para que as vantagens do enfoque da prototipagem rápida se materializem sem incorrer nos problemas desta abordagem. Estas condições são discutidas no plano das características da aplicação a ser desenvolvida, do tipo de usuário, do ambiente de desenvolvimento e das ferramentas utilizadas no projeto. A posição defendida neste artigo é de que a adoção do enfoque da prototipagem rápida, principalmente quando se refere ao desenvolvimento do sistema através de versões sucessivas, necessita de condições favoráveis a sua utilização, pois sem estas, muitas das vantagens da abordagem são perdidas.

PALAVRAS-CHAVE: Engenharia de Software, Metodologia de Desenvolvimento de Sistemas, Ciclo de Vida, Prototipagem Rápida.

Marcos Roberto da Silva Borges, Engenheiro Eletricista (UFRJ 1975), Mestre em Ciência da Computação (COPPE/UFRJ 1981), Doutor em Ciência da Computação (East Anglia 1986). Analista Consultor do Núcleo de Computação Eletrônica da UFRJ e Professor Adjunto do Departamento de Computação da UFRJ. Trabalha na área de computação desde 1972 e atua junto ao grupo de desenvolvimento de sistemas para a administração da universidade como consultor de apoio nas áreas de Banco de Dados e Engenharia de Software.

1 - INTRODUÇÃO

Muito tem-se discutido sobre as vantagens do enfoque da prototipagem rápida como alternativa ao enfoque tradicional baseado no ciclo de vida em fases [3, 6, 7, 8, 9, 10, 13, 16]. O entusiasmo pelo enfoque de prototipagem rápida é justificável pelas vantagens tão propaladas pelos defensores deste enfoque: retorno mais rápido aos usuários, maior chance de sucesso da aplicação e uma evolução do sistema mais condizente com a necessidade e a capacidade de absorção reais do usuário [9, 10, 11].

A verificação destas vantagens na prática, porém, está condicionada a uma série de características que vão desde o tipo de problema a ser resolvido a formação do pessoal de desenvolvimento, passando pela disponibilidade de ferramentas adequadas. Um exemplo disto é o fato que, de uma forma ou de outra, as idéias em que se baseia o enfoque da prototipagem rápida já são utilizadas há bastante tempo no desenvolvimento de sistemas sem que com isso se tenha alcançado as vantagens tão propaladas. Por força muitas vezes de prazos exíguos ou da formação dos responsáveis pelo desenvolvimento da aplicação, é comum encontrar-se projetos que derrespeitam o ciclo de vida tradicional através da simplificação excessiva das fases de análise e projeto. Nestes casos os fundamentos do enfoque da prototipagem rápida foram adotados empiricamente causando os problemas já conhecidos deste enfoque, qual sejam: documentação deficiente, dificuldades para evolução/manutenção do sistema, etc.

Neste artigo discute-se as condições julgadas necessárias para que as vantagens do enfoque da prototipagem rápida se materializem sem incorrer nos problemas desta abordagem. O artigo se concentra na utilização do enfoque da prototipagem em situações que sugiram o desenvolvimento do sistema através da produção de versões sucessivas. Estas condições são discutidas no plano das caracteristicas da aplicação a ser desenvolvida, do tipo de usuário, do ambiente de desenvolvimento (métodos e formação do pessoal de análise) e das ferramentas utilizadas para o desenvolvimento. Pretende-se com isto contribuir para o esclarecimento da premissa de que a adoção do enfoque da prototipagem rápida necessita de condições favoráveis a sua utilização sem as quais muitas das vantagens desta abordagem são perdidas.

2 - O CICLO DE VIDA TRADICIONAL E ALTERNATIVAS

O ciclo de vida tradicional consiste, em geral, da condução do desenvolvimento do sistema pelo seguinte processo evolutivo [6]:

1 - Análise Organizacional;
2 - Avaliação dos Sistemas;
3 - Análise de Viabilidade;
4 - Plano de Projeto;
(passos 1 - 4 compõem o que se chama de Ante-Projeto)
5 - Projeto Lógico;
6 - Projeto Físico;
7 - Definição de Programas;
8 - Implementação;
9 - Operação;
10 - Revisão e Avaliação.

Dentro deste modelo é que se dá o desenvolvimento da maioria dos projetos de sistemas de informação. As etapas se dão de uma forma sequencial, pois, em geral, o resultado gerado por uma fase é usado como entrada para a seguinte.

Os que defendem do ciclo de vida tradicional consideram que este representa um modelo que auxilia o processo de desenvolvimento do projeto, pois marca bem as etapas que vão sendo cumpridas, e facilita o acompanhamento da evolução deste [8].

Um outro argumento forte em defesa do ciclo de vida tradicional é que a divisão do trabalho nas etapas que o compõe está associada muito fortemente ao nível de detalhe de cada fase [14].

Por outro lado, há vários autores que apontam deficiencias no enfoque tradicional, pois este não induz uma participação maior do usuário no processo de desenvolvimento além de gerar um intervalo muito grande entre a definição do problema e o uso efetivo da solução computacional. Isto provoca um risco do sistema não ser concluido ou, se o for, não estar dentro das reais necessidades dos usuários.

Uma amostragem de dados sobre o desenvolvimento de projetos para o Departamento de Administração do governo americano durante o ano de 1979 mostra que somente cerca de 3% do total gasto com o desenvolvimento de sistemas de aplicação de 6,5 milhões de dólares foram efetivamente aproveitados. Isto demonstra bem os problemas que a má definição de requerimentos provoca.

Várias tentativas foram ou estão sendo feitas no sentido de formalizar melhor o processo de desenvolvimento através do enfoque do ciclo de vida tradicional. Neste

enfoque é que se pode categorizar os trabalhos sobre análise e projeto estruturado de sistemas e os de metodologias para uma ou mais fases do ciclo, como o SADT e o SAMM.

Há cerca de seis anos entretanto, começaram a surgir artigos que vieram em defesa de enfoques alternativos para o processo de desenvolvimento de sistemas de informação. Um destes enfoques é o da Prototipagem Rápida.

3 - PROTOTIPAGEM RÁPIDA: VANTAGENS E PROBLEMAS

Em várias ramos da Engenharia o enfoque de prototipagem é largamente empregado como uma ferramenta poderosa e essencial. Entretanto, no processo de desenvolvimento de software a prototipagem é raramente empregada [18]. Recentemente, um movimento na direção de fazer um maior uso de protótipos foi iniciado e vários artigos e experiencias foram ou estão sendo publicadas.

De um modo geral, o uso de protótipos no processo de desenvolvimento de software se dá em três modalidades:

A primeira é como auxílio na especificação de requerimentos do software. O uso neste caso é recomendado quando o usuário tem pouca experiencia com sistemas computadorizados ou quando se trata de um software totalmente novo do qual, mesmo o usuário mais experiente tem pouco conhecimento. O uso do protótipo viria então ajudar a levantar os requerimentos através da utilização efetiva de uma representação do sistema.

A segunda modalidade visa a utilização do protótipo para efetuar um estudo de viabilidade de parte ou da totalidade do sistema. Neste caso o protótipo é utilizado para avaliar e experimentar soluções para um determinado problema, em geral associado a questão de performance, por exemplo, o tempo de acesso que uma certa organização de dados proporciona.

Finalmente, a terceira modalidade prevê que o protótipo evolua efetivamente para o produto final. Ao contrário das duas situações anteriores, onde o protótipo é descartado após a obtenção dos resultados esperados, nesta terceira modalidade cada versão gerada visa atender as necessidades imediatas do usuário e serve como base as versões seguintes. Desta forma, se dá um processo que é denominado de desenvolvimento através de versões sucessivas.

As vantagens e desvantagens do enfoque variam de acordo com a modalidade em que este é utilizado. Na primeira há a potencial melhoria da especificação do sistema pois os requerimentos são obtidos a partir do uso efetivo de um

modelo que se propõe a executar as funções reais do sistema. Além disso, induz o envolvimento por parte do usuário na fase de definição do projeto, que nem sempre é conseguido pelo processo convencional de entrevistas e discussões sobre modelos hipotéticos. Se alcançados, estes efeitos irão resultar em um sistema de melhor qualidade e de menor custo de manutenção, visto que os requerimentos foram mais consolidados.

Como desvantagem há o custo e o tempo extra dispendido na construção do protótipo. Ao contrário do que se esperaria, o enfoque de prototipagem nesta modalidade aumenta ao invés de reduzir o tempo total de desenvolvimento do sistema.

A segunda modalidade apresenta de uma maneira geral as mesmas vantagens e desvantagens da primeira modalidade. A diferença está na fase do projeto em que o protótipo é utilizado e o que se objetiva com o uso deste.

A terceira modalidade difere das duas anteriores na maioria dos aspectos, pois cada versão do protótipo objetiva o produto final e não somente prover subsídios para um projeto de melhor qualidade. As vantagens do enfoque de versões sucessivas é que este propicia ao usuário uma resposta mais rápida às suas necessidades mais prementes. Como não há necessidade da completeza na elaboração dos requerimentos, é possível através de ferramentas adequadas prover o usuário de uma versão parcial mas real do sistema de forma mais rápida.

A desvantagem é que este processo em geral é bastante dispendioso porque pode ser necessário que uma parte substancial do código seja gerado várias vezes para que este se adapte aos novos requerimentos. Desta forma, embora se possa alcançar resultados parciais em curto espaço de tempo, o produto final pode levar muito mais tempo e ser muito mais caro do que se fosse desenvolvido pelo método tradicional.

Entretanto, há ainda uma outra vantagem que nem sempre é visível para aqueles que não lidam com o problema diretamente. É que muitas vezes, na tentativa de evitar omissões que mais tarde poderão lhe causar problemas, o usuário tende a ser mais abrangente em seus requerimentos do que o necessário. Isto acarreta em muitos casos que dados ou relatórios jamais sejam utilizados pois não fazem parte das necessidades da aplicação. Com o enfoque de versões sucessivas este problema é minimizado pois o projeto evoluiu em função das necessidades reais do usuário, reduzindo desta forma o tamanho do sistema.

Neste artigo, somente o uso do enfoque na terceira modalidade será abordado.

4 - CONDIÇÕES PARA UTILIZAÇÃO DO ENFOQUE

A disciplina Engenharia de Software mais do que em outras áreas da informática, possui um caráter muito prático. Por esta razão, os artigos que exploram os diversos aspectos da disciplina são em geral classificados de especulativos e práticos. Os de caráter especulativo são em geral escritos por profissionais que se abstraem dos aspectos práticos do problema e muitas vêzes propoem soluções julgadas inviáveis de serem utilizadas na prática. Por outro lado, os artigos de caráter puramente prático não raramente carregam em suas propostas os vícios adquiridos no exercício da atividade e não conseguem vislumbrar soluções alternativas fora do modo de proceder do autor. Este artigo, embora possua o caráter especulativo, é baseado em algumas experiências práticas vividas pelo autor.

4.1 - Conscientização do usuário e da equipe de desenvolvimento

Antes de iniciar um projeto baseado no enfoque de versões sucessivas é necessário que tanto o usuário quanto a equipe de desenvolvimento estejam conscientes das características deste enfoque.

Entre as equipes de desenvolvimento é comum encontrar-se pessoas que não tendo entendido o enfoque da prototipagem, acabam por transformá-lo em prototipagem lenta, com todas as desvantagens deste enfoque sem necessariamente as vantagens.

O projetista deve estar consciente, por exemplo, de que os aspectos de eficiência são menos importantes nas primeiras versões para evitar que seja dispendido um tempo excessivo na procura de soluções para problemas ainda não totalmente definidos.

Ao usuário é preciso esclarecer como se dará o desenvolvimento para evitar uma expectativa maior do que a desejável com relação as novas versões. é esperado que a inclusão dos novos requerimentos torne cada vez mais demorada a geração de novas versões.

é preciso, sobretudo, induzir o usuário a participar efetivamente no processo de desenvolvimento, de modo a forçar o uso das primeiras versões do protótipo e da produção de comentários e sugestões que irão influenciar as novas versões.

Deve ser estabelecido um tempo mínimo (1 a 2 meses, dependendo da aplicação) entre a implantação de novas versões, a fim de forçar que os pedidos de modificações sejam bem substanciados e baseados no uso efetivo do protótipo.

4.2 - Características da aplicação

Não é todo tipo de aplicação que sugere o uso do enfoque da prototipagem. ésta equivocado aquele que imagina que o enfoque irá produzir soluções para todos os tipos de problemas.

O enfoque deve ser utilizado principalmente quando não hà maior conhecimento sobre o conjunto de requerimentos que definirão o sistema a ser projetado. Isto pode ser decorrencia da pouca experiência do usuário em sistemas automatizados, ou do pioneirismo do sistema.

O enfoque não deve ser utilizado em situações conhecidas, para as quais, por exemplo, já existem soluções bem difundidas. Não é aconselhável também, a utilização do enfoque em problemas muito grandes, a não ser na primeira ou segunda modalidades, pois o processo evolutivo pode causar muitos problemas quando se começa de uma versão muito reduzida do sistema para se alcançar a solução global. Nestes casos é aconselhável a adoção do enfoque tradicional baseado no ciclo de vida em fases.

4.3 - Características das ferramentas

A primeira característica importante do enfoque da Prototipagem Rápida é que este seja capaz, como o próprio nome sugere, de produzir soluções rápidas, embora parciais e nem sempre as melhores, para os problemas.

Para que isto seja possível os modelos que compõem a ferramenta devem funcionar quase que como um gerador de aplicações onde alguns componentes como relatórios e telas de entrada e saída são gerados a partir de especificações simples fornecidas pelo projetista. Para isto, a linguagem de definição destes componentes deve ser preferencialmente declarativa, não só para facilitar a documentação mas também para possibilitar uma definição mais rápida.

Com relação a estrutura de dados de suporte ao armazenamento das informações (o banco de dados), esta deverá ser capaz de armazenar informações semanticas de modo a facilitar a interação com o usuário na definição dos dados.

Tanto a facilidade de programação, quanto as facilidades para uma organização de arquivos eficiente devem ser proteladas para as fases finais (últimas versões) do processo quando os requerimentos da aplicação estarão mais estabilizados. Tanto a programação como a organização mais adequada dos arquivos visa o aumento da eficiência do sistema, mas consomem um tempo que pode ser totalmente perdido no caso de mudança de requerimentos que afetem a solução. Além disso, a programação dificulta a geração de novas versões.

Obviamente, para se evitar o uso de programação a ferramenta deve incorporar facilidades que permitam este procedimento. Como isto nem sempre é verdade, a programação deve se restringir a umas poucas rotinas onde esta é essencial.

A ferramenta deve possuir um suporte de boa qualidade para documentação para permitir que todos os aspectos que orientaram as decisões de projeto fiquem bem registradas e acessível a todos os envolvidos: projetistas e usuário. É uma boa idéia permitir a documentação "on-line" para permitir que sugestões e reclamações sejam incluidos durante o uso do sistema, assim que estes percebam algum problema na sua utilização. Isto irá simplificar o processo de comunicação entre a equipe de desenvolvimento e os usuários.

O banco de dados deve funcionar também como um repositório das decisões que nortearam a definição das telas de entrada, relatórios, etc, além, é claro, dos dados propriamente ditos. Isto irá possibilitar que o acesso a estas informações se dê de uma maneira uniforme.

Facilidades para auxílio ao uso do sistema (help) devem ser providas pela ferramenta, tanto com relação aos comandos como também com relação aos dados e sua estrutura de armazenamento. Isto possibilitará que o treinamento do usuário seja simplificado já que este contará com um bom esquema de suporte por parte do próprio sistema

5 - CONCLUSÕES

Neste artigo foram sugeridas algumas diretrizes sobre a condução de um projeto baseado no enfoque da prototipagem rápida na terceira modalidade, qual seja, o desenvolvimento do sistema através da geração de versões sucessivas.

Este conjunto de diretrizes visam a tornar mais viável o uso do enfoque no processo de desenvolvimento de aplicações e nortear a definição de uma metodologia baseado neste enfoque.

6 - BIBLIOGRAFIA

[1] Carvalho, A.S.L. Quem desemvolve esta aplicação: O usuario ou o departamento de informatica? Anais do XIX Congresso Nacional de Informatica, pp. 231-237, Rio de Janeiro, 1986.

[2] Silva Neto, A.M. et al. Ambiente de desenvolvimento automatizado. Anais do XIX Congresso Nacional de Informatica, pp. 239-244, Rio de Janeiro, 1986.

[3] Medeiros, G. & Maia, L.C.L. Prototipação de aplicação - Presente ou futuro do desenvolvimento de sistemas de informação? Anais do XIX Congresso Nacional de Informatica, pp. 291-296, Rio de Janeiro, 1986.

[4] Blum, B.I. Still more about rapid prototyping. ACM SIGSOEI Software Engineering Notes Vol 8 No 3, July 1983, pp. 9-11.

[5] Glass, R.L. Recommended: A minimum standard software toolset. ACM SIGSOEI Software Engineering Notes Vol 7 No 4, October 1982, pp. 3-13.

[6] McCracken, D.D. & Jackson M.A. Life cycle considered harmful. ACM SIGSOEI Software Engineering Notes Vol 7 No 2, April 1982, pp. 29-32.

[7] Gladden, G.R. Stop the life-cycle, I want to get off. ACM SIGSOEI Software Engineering Notes Vol 7 No 2, April 1982, pp. 35-39.

[8] Hall, P.A.V. In defence of life cycles. ACM SIGSOEI Software Engineering Notes Vol 7 No 3, July 1982, pp. 23.

[9] Blum, B.I. Rapid prototyping of information management systems. ACM SIGSOEI Software Engineering Notes Vol 7 No 5, December 1982, pp. 35-38.

[10] Keus, H.E. Prototyping: A more reasonable approach to system development. ACM SIGSOEI Software Engineering Notes Vol 7 No 5, December 1982, pp. 94-95.

[11] Taylor, T. & Standish, T.A. Initial thoughts on rapid prototyping techniques. ACM SIGSOFT Software Engineering Notes Vol 7 No 5, December 1982, pp. 160-166.

[12] Mason, R.E.A. & Carey, T.T. Prototyping interactive information systems. Communications of ACM Vol 26 No 5 May 1983, pp. 347-354.

[13] Gomma, H. The impact of rapid prototyping on specifying user requirements. ACM SIGSOFT Software Engineering Notes Vol 8 No 2, April 1983, pp. 17-28.

[14] Zahniser, R.A. Levels of abstraction in the system life cycle. ACM SIGSOFT Software Engineering Notes Vol 8 No 1, January 1983, pp. 6-12.

[15] Lawrence, J.L. Why software is always late? ACM SIGSOFT Software Engineering Notes Vol 10 No 1, January 1985, pp. 19-30.

[16] Alavi, M. An assesment of the prototyping approach to information system development. Communications of ACM Vol 27 No 6 June 1984, pp. 556-563.

[17] Blum, B. Interactive Development of Information Systems: A Case Sudy. Software Practice & Experience, Vol 6 No 6, June 1986, pp. 503-515.

[18] Gomaa, H. The impact of rapid prototyping on specifying user requirements. ACM Software Engineering Notes Vol 8 No 2, April 1983, pp. 17-21.