Controle de Qualidade de Software para a Área Científica

Alvaro de Sá Bahia

Programa de Engenharia e Sistemas de Computação - COPPE/UFRJ
Caixa Postal 68511
21945 - Rio de Janeiro - RJ

Petrobrás S. A. - Depex-Disep-Sesup
Av República do Chile, 65/1521
20035 - Rio de Janeiro - RJ

Resumo:

A especificação dos atributos de qualidade de um software, de modo geral, sofre influências devido ao ambiente operacional, ao ambiente de desenvolvimento, à visão do usuário e a fatores circunstanciais. O software científico, em particular, sofre também influência devido à existência de uma fase de pesquisa, anterior à elaboração do produto de software. Serão aqui analisadas as caraterísticas e dificuldades existentes na produção de software científico.

Palavras-chaves:

ambiente de desenvolvimento, software científico, qualidade de software

INTRODUÇÃO

A produção de software científico, tanto quanto em outras áreas, vem enfrentando obstáculos para obter produtos de qualidade, em tempo hábil. A dificuldade, ainda existente, para conceituar e medir certos aspectos relativos à qualidade de software, bem como a inexistência de consenso a respeito da forma mais adequada para a elaboração de software científico ou numérico, contribuem de forma significativa para aumentar tais dificuldades.

Neste trabalho, inicialmente, conceituamos o que compreendemos por qualidade de software e analisamos os diversos fatores que influenciam a percepção e quantificação dos atributos de software a serem atingidos na produção de sistemas. O termo software deverá ser entendido em seu sentido mais amplo, compreendendo o conjunto de programas e toda documentação a ele associado. Neste texto os termos software, produto de software e sistema serão usados indistintamente e se referem à descrição anteriormente citada.

Em seguida, caracterizamos o que entendemos por software científico, identificando as principais características e dificuldades encontradas hoje para a sua produção, com o objetivo de avaliar os atributos de qualidade mais relevantes no desenvolvimento de software científico.

QUALIDADE DE SOFTWARE

Tendo em vista os atuais custos verificados na produção de software, é necessário garantir que os produtos entregues sejam de boa qualidade. Os fatores que caracterizam a qualidade de um software podem variar de sistema a sistema e de programa para programa {Arthur84}. Não se pode embutir qualidade num produto, nem medi-la sem primeiro definir o que se entende por qualidade para o produto em questão {Perry83}. Qualidade é portanto um atributo associado a alguma coisa e deverá ser buscada ao longo do desenvolvimento de cada fase do ciclo de vida. O produto ou documento de cada fase, isto é: especificação de requisitos, especificação de projeto, confecção de programas etc, deverá ter sua qualidade avaliada. Sendo a

qualidade um conceito multidimensional, ela será obtida através da agregação de um conjunto de atributos e características {Rocha87a}.

Para que uma organização possa construir software com um nivel de qualidade pré-determinado, é necessário analisar-se a natureza do produto e identificar quais as caracteristicas consideradas como desejáveis. É necessário que tais caracteristicas possam ser avaliadas e que para elas se possa estabelecer um grau minimo a ser atingido, de forma a que o software possa ter a sua qualidade avaliada {Rocha87b}.

Rocha propôs um método para a avaliação e controle de qualidade que pressupõe a definição de objetivos de qualidade, que são atingidos através de fatores, que podem ser compostos de subfatores, que por sua vez são avaliados através de critérios. Os critérios são atributos de qualidade primitivos, isto é, independentes de qualquer outro e que portanto podem ser medidos {Rocha87a}.

É necessário que todo software produza resultados corretos, sejam eficientes, de uso amigável e de fácil manutenção. Entretanto a determinação do peso que cada fator ou sub-fator exercerá na composição final dos objetivos de qualidade sofrerá influências devido: ao ambiente operacional, ao ambiente de desenvolvimento, à visão do usuário, e a fatores circunstanciais. A seguir será analisada a influência destes elementos sobre alguns dos fatores que determinam a qualidade do software.

a) Influências Devido ao Ambiente Operacional

O hardware onde o software irá operar poderá determinar a relevância de certos fatores. Por exemplo, software produzido para micro-computador poderá sofrer influência devido à limitação de memória, precisão dos resultados e velocidade de processamento. Alguns dos aspectos que são afetados por tais restrições são: modularidade, concisão, necessidade, precisão e eficiência. Por outro lado o público a que se destina o software para micro-computador requer atenção para os seguintes fatores: robustez, segurança, portabilidade {Bargut86}.

O tipo do processamento e a natureza da aplicação também vão determinar a ponderação dos fatores de qualidade de um software. Ferreira e Rocha {Ferreira87}, verificaram que:

i) existem alguns atributos de qualidade de programas que são relacionados com a natureza da aplicação, enquanto que outros são dependentes do tipo do processamento;

ii) existem atributos de qualidade que têm suas maiores dimensões influenciados simultaneamente, tanto pela natureza da aplicação, quanto pelo tipo do processamento

O controle de processos confere ao software um elevado grau de complexidade e, apesar disto, requer sistemas eficientes para que possam responder ao estimulo externo em tempo hábil. A natureza da aplicação por outro lado poderá exigir alta precisão nos cálculos (software cientifico e militar) e/ou elevada robustez e segurança (área militar e nuclear). A manipulação de grande massa de dados, na área comercial, requer programas com alta eficiência para a realização destas tarefas.

b) Influências Devido ao Ambiente de Desenvolvimento

Dependendo de quem desenvolve o produto e qual a finalidade do desenvolvimento, certos fatores assumirão uma menor ou maior importância . O desenvolvimento de um sistema por uma "software house" tem por objetivo comercializá-lo. Desta forma, o produto deverá privilegiar os seguintes fatores de qualidade: uso amigável, eficiência, portabilidade, alta capacidade de adaptação às exigências do mercado (facilidade de manutenção), tudo isto associado a um baixo custo de produção.

Por outro lado, o software produzido por empresas para o seu uso privado, poderá considerar a portabilidade desnecessária, em função da certeza de que não haverá mudanças de hardware a curto e médio prazos. O custo, por sua vez, poderá não ser relevante se o software for considerado como componente estratégico para a empresa.

Por fim, tem-se o software desenvolvido para uso pessoal onde a necessidade de interfaces amigáveis com o usuário já não é tão necessária pois, em geral, será o próprio desenvolvedor quem irá operá-lo.

c) Influências Devido à Visão do Usuário

Certos fatores de qualidade são afetados pela visão do usuário e pelo ângulo com que tais fatores são avaliados. Neste contexto entendemos usuário do software como qualquer pessoa que ao longo do ciclo de vida o manipule, quer desenvolvendo-o, mantendo-o, usando-o ou mesmo produzindo e comercializando-o.

O grupo desenvolvedor por ter prazos e metas mais rigidas a cumprir, tenderá a considerar, onde possivel, niveis minimos aceitáveis de qualidade. Por sua vez, o grupo responsável pela manutenção espera encontrar no produto: um alto grau de clareza, concisão e modularidade; um estilo e estrutura bem definida; documentação atualizada, que permita o rastreamento das funções. Este grupo espera que o software seja manutenivel, isto é, possa ser mantido com facilidade.

Na visão do usuário final, o produto deverá atender suas necessidades reais e deverá ser ameno ao uso, eficiente, portátil, correto, seguro e que responda sem perda de controle às condições de erro.

Há, ainda, a visão do gerente responsável pela confecção do produto, que espera encontrar todas as caracteristicas acima descritas , mas que tem que conciliar as restrições de recursos que lhes são impostas.

d) Influências Devido a Fatores Circunstanciais

Em determinadas ocasiões existem restrições circunstanciais que poderão determinar uma maior atenção para um fator, que em condições normais não seria considerado. Podemos citar, por exemplo, o desenvolvimento de um software que irá operar inicialmente num ambiente de hardware e que a curto prazo será alterado. Tal software terá que, necessariamente, ter uma alta portabilidade para que a migração de um hardware para o outro seja feita sem maiores dificuldades.

SOFTWARE CIENTÍFICO

O software científico, ou numérico, é caracterizado por um conjunto de algoritmos, por sua vez compostos de uma série de algoritmos menores, agrupados normalmente numa estrutura complexa onde a iteração é a principal característica. A quantidade de dados gerados, de um modo geral, pode não ser substancial, mas uma proporção significativa do total é operada um número grande de vezes. A execução do software numérico requer equipamentos com grande memória e alta capacidade computacional.

A produção de software científico requer duas fases distintas. A primeira de pesquisa, é necessária para o equacionamento do problema. Muitas vezes a solução não é conhecida e deve ser procurada. Os algoritmos necessários à solução numérica do problema devem ainda ser estabelecidos e testados. Uma vez tendo-se chegado à concepção da solução do problema proposto, é necessário que se valide a solução através de um protótipo. Verificar-se-á, assim, se o conjunto de algoritmos pode ser implementado de forma satisfatória . Uma vez avaliada a factibilidade da solução, poder-se-á então passar à fase de construção do software.

Entretanto o desenvolvimento de software científico, traz consigo uma série de dificuldades. Em primeiro lugar é necessário conciliar o trabalho do grupo de pesquisa com o de confecção do produto de software.

Os técnicos responsáveis pela fase de pesquisa se veem, via de regra, primeiro como matemáticos, engenheiros ou cientistas e não como analistas de sistema. Apesar de gastarem cerca de 80% do seu tempo projetando, confeccionando e mantendo software] Por outro lado a comunidade de informática tende a perceber o grupo de pesquisa como uma fraternidade extremamente mudana, uma vez que, tais pessoas escrevem programas usando linguagens e estruturas tidas como ultrapassadas, sendo que talvez, a atividade mais substancial em que participam, seja a de pressionar a capacidade numérica de um dado processador ao seu limite {Cross86}.

Outro fator presente na construção de software científico é o de que o esforço requerido para implementar com sucesso determinados algoritmos numéricos é tão substancial, que uma vez terminado, a tendência é deixá-lo do jeito que está, desconsiderando-se que o produto obtido está a nível de protótipo, sem os requisitos necessários para ser considerado um produto de software. Outro problema é que a concepção e projeto iniciais muitas vezes se mostram inadequados para as alternativas, não previstas, de desenvolvimento que o produto eventualmente assume.

Apesar de lidar-se com protótipos, muitas vezes ao chegar-se à implementação, o estado-da-arte na teoria já coloca os algoritmos implementados como algo obsoleto. Decorre daí que boa parte dos software numéricos não possuem uma interface com o usuário adequada. A opção de tornar o software numérico robusto e orientado ao usuário pode fazer com que as técnicas numéricas implementadas possam ficar atrás do estado-de-arte dos algoritmos numéricos em até meia geração. O software numérico tende então a ser técnicamente capaz e poderoso mas com a interface com o usuário prejudicada.

Finalmente, existem hoje um grande número de software orientado para micros que possuem uma interface com o usuário extremamente boa. Entretanto, enquanto tais produtos realizam funções que a poucos anos atrás eram tidas como complexas, se comparadas aos produtos atualmente disponíveis para sistemas de grande porte, se tornam demasiadamente simples.

Situação Atual

Podemos afirmar que, como nas outras áreas, o desenvolvimento de software cientifico se encontra em crise. Sistemas são produzidos sem que atendam adequadamente ao usuário final e são, na maioria das vezes, confeccionados de tal forma que a sua manutenção é dificil e onerosa. Em pesquisa realizada na Inglaterra, por Medes et alli, verificou-se que cerca de 37% dos respondentes não se consideravam usuários de técnicas de programação estruturada {Medes87}. Cross chega a afirmar que a metodologia de desenvolvimento utilizada na maioria das organizacões envolvidas com software numérico é informal. Elas muitas vezes se baseiam em um estilo próprio, que não pode ser facilmente explicado de maneira coerente.

Na pesquisa de Medes observa-se que a linguagem predominante é o FORTRAN, o que seria explicado pelo fato de ser uma linguagem disponivel mundialmente, por ser bastante portátil (compativel) e também pelo fato do mercado dispor de uma grande variedade de bibliotecas de subprogramas numéricos {Bell86}. Outro aspecto interessante, é o fato de que as ferramentas de auxilio ao desenvolvimento mais citadas eram destinadas à fase de testes. Isto corrobora a pesquisa efetuada por Ferreira, onde constatou-se que o controle de qualidade das empresas pesquisadas era, na maioria das vezes, exercida através de testes de programas {Ferreira87}. Acreditamos que a realidade brasileira se aproxima mais da visão de Cross, do que do perfil apresentado pela pesquisa de Medes.

CONCLUSÕES

Não há um consenso a respeito do ambiente de desenvolvimento adequado para a confecção de software cientifico. Não se conseguiu determinar a melhor forma de se passar da etapa de pesquisa, que produz apenas um protótipo, para a fase onde o produto de software é obtido. Todos concordam que deve ser na ultima forma que o sistema deve ser colocado em produção.

Rocha sugere que tal problema pode ser solucionado através de um elemento de ligação entre os dois grupos. Tal pessoa, familiarizada com os dois ambientes facilitaria a migração do protótipo para um produto de software {Rocha87c}. Pereira por sua vez, entende que a melhor abordagem seria uma maior iteração entre os dois grupos. Ou seja, o profissional de engenharia se comprometendo mais com a qualidade final do produto como um todo e não apenas com a correção do algoritmo implementado, e o profissional de informática auxiliando mais na fase de prototipagem {Pereira87}.

Emkin, sem entrar no mérito da passagem da fase de pesquisa para a confecção do produto de software, recomenda que os programas para engenharia satisfaçam requisitos mínimos, para que possam ser considerados e usados como tal {Emkin87}.

Não está claro ainda a forma de obtê-lo, mas todos concordam como deve ser o software científico: confiável, portátil e fácil de usar, modificar e manter. Torna-se, portanto, necessário que se continue avaliando métodos e ambientes de desenvolvimento de software.

Bibliografia

{Arthur84} - Arthur, L. J. - "Measuring Programmer Productivity and Software Quality"; John Wiley & Sons; 1984.

{Bargut86} - Bargut, M. F.; Rocha, A. R. C. - "A Qualidade de Software para Microcumputadores"; COPPE/UFRJ, ES-102/86; junho 1986.

{Beaufond87a} - Beaufond, C. E. C. - "Verificação e Validação de Software na Fase de Especificação de Requisitos"; COPPE/UFRJ M. Sc.; 1987.

{Beaufond87b} - Beaufond, C. E. C. - "Verificação e Validação de Software na Fase de Especificação de Requisitos"; Conf. Latino Americana de Informática; Bogotá, novembro 1987.

{Bell86} - Bell, K. - "Some Thoughts on Design, Development and Maintenance of Engineering Software"; Adv. Eng. Software, n. 2, vol 8; 1986.

{Cross86} - Cross, M.; Moscardini, A. O.; Lewis, B. A. - "Software Engineering Methodologies for Scientific and Engineering Computation"; Appl. Math. Modeling; vol 10, pags 376-385; october 1986.

{Emkin87} - Emkin, L. Z. - "Computers in Structural Engineering Practice - the Issue of Quality"; 1st International Conference on Reliability & Robustness of Enginneering Software; september 1987.

{Fairley85} - Fairley, R. E. - "Software Engineering Concepts"; McGraw Hill; 1985.

{Ferreira87} - Ferreira, R. L.; Rocha, A. R. C. - "A Influência da Natureza da Aplicação na Qualidade de Software"; COPPE/UFRJ ES-130/87; 1987.

{Medes87} - Medes, J. S.; Smith, P.; Newton, M. J. - "A Survey of the Methods Used for Development of Engineering Software in the United Kingdom"; 1st International Conference on Reliability & Robustness of Enginneering Software; september 1987.

{Pereira87} - Pereira, J. C. - "Metodologia para Desenvolvimento de Software para Engenharia"; VIII Congresso Latino Americano e Ibérico; trab. B-3, pags 31-47; novembro 1987.

{Perry83} - Perry, W. - "Effective Methods of EDP Quality Assurance"; Prentice-Hall; 1983.

{Rocha87a} - Rocha, A. R. C. - "Análise e Projeto Estruturado de Sistemas"; Campus; 1987.

{Rocha87b} - Rocha, A. R. C.; Aguiar, T. C.; Blasheck, J. R. - "Ambientes para Desenvolvimento de Software: Definição de Termos"; COPPE/UFRJ ES-137/87; 1987.

{Rocha87c} - Rocha, A. R. C. - "Algoritmos Numéricos x Software Numérico: Uma Visão do Engenheiro de Software"; VIII Congresso Latino Americano e Ibérico; trab. B-4, pags 49-58; novembro 1987.