CLASSIFICAÇÃO DOS REQUISITOS DE QUALIDADE APLICADO AOS REQUISITOS NÃO-FUNCIONAIS

11.5.1 FUNCIONALIDADE

A funcionalidade diz respeito àquilo que o software faz quando solicitado pelo usuário, como, por exemplo: imprimir um relatório, apresentar dados na tela ou registrar uma informação em uma base de dados. A característica se refere à capacidade para o cumprimento de tarefas; em outros termos, se uma dada função foi implementada ou não no programa. A maneira como essa função é executada é algo que pode ser avaliado em função de outras características.

Pode-se dizer que esta característica é idêntica aos “requisitos funcionais” definidos por Sommerville [2003]: “Os requisitos funcionais para um sistema descrevem a funcionalidade ou os serviços que se espera que o sistema forneça.”

A ISO/IEC 9126 determina que seja estabelecido um escopo quando se faz a definição de funcionalidades de um programa. Um exemplo simples é o caso de gravação de informações em um arquivo: o escopo aqui representa espaço suficiente em disco para que a operação seja bem-sucedida. Outras funções podem apresentar requisitos menos óbvios e que devem ser explicitados no momento de elencar os requisitos. Diz-se que é preciso definir um contexto de uso para avaliar ou definir a funcionalidade.

A subcaracterística de adequabilidade é representada no software pela capacidade de prover funções corretamente adaptadas às necessidades do usuário. Por exemplo: um programa de controle de hotel pode oferecer um cadastro de clientes exigindo preenchimento de diversos dados. Para um determinado usuário, isto pode ser inadequado, pois caso se deseje efetuar um *check-in* rapidamente apenas com o nome do hóspede, o programa não permitirá. A função de cadastro existe, mas não está bem adaptada às necessidades do hotel.

A acurácia pode ser exemplificada em programas envolvendo cálculos, como softwares para engenharia. Uma vez que o computador trabalha com precisão limitada, o resultado de um cálculo é, em geral, aproximado. Pode-se estabelecer, então, um valor máximo para o erro admissível no programa ou o número de dígitos (casas após a vírgula) desejados nas respostas.

A interoperabilidade é a capacidade de o software interagir com outros componentes de um sistema. Exemplos dessa subcaraterística são recursos de OLE (Object Linking and Embedding) e DDE (Dynamic Data Exchange).

A subcaracterística segurança foi traduzida do inglês security e não deve ser confundida com safety. Security, traduzido como segurança, diz respeito à proteção das informações contra acessos não autorizados. Safety, em português também traduzido como segurança, relaciona-se à ausência de riscos de operação – por exemplo, no caso de um software de controle de uma usina nuclear. Enquanto security é uma subcaracterística de funcionalidade, safety é uma característica de qualidade em uso. Outra interpretação para segurança que não faz parte da característica “funcionalidade” diz respeito à perda de informações.

11.5.2 MANUTENIBILIDADE

A característica de manutenibilidade está relacionada à facilidade de modificação de um produto de software. Esta característica é de interesse especialmente para desenvolvedores e não deve ser confundida com a possibilidade de configurar o software. Uma modificação consiste, por exemplo, em uma correção do produto ou de adaptação a mudanças, de requisitos, como mudanças de legislação.

A subcaracterística de modificabilidade refere-se à implementação do software e traduz-se, em geral, por alterações de código. Há alguns fatores que podem melhorar a modificabilidade:

- documentação da estrutura interna do produto;
- arquitetura adequada do software;
- clareza do código-fonte.

Dois exemplos podem ilustrar como é possível interferir na manutenibilidade de um software.

O primeiro é a construção de um compilador, que pode ser feita por meio das ferramentas LEX/YACC, ou com um conjunto de rotinas recursivas. A implementação usando LEX/YACC exigirá do programador que estude e se familiarize com essas ferramentas, enquanto a implementação recursiva pode ser feita diretamente. Contudo, o esforço empregado no primeiro caso se traduz em um produto de manutenção mais fácil.

Um segundo exemplo, menos acadêmico, é o uso de arquivos de recursos (resource files) utilizados na implementação de aplicações no sistema operacional Windows. Se mensagens de erro e outros itens de interface –

eventualmente telas inteiras – são agrupados em um único arquivo, modificar o produto torna-se uma tarefa muito mais simples do que percorrer e alterar vários pontos de código-fonte.

A anasabilidade de um software depende da facilidade com que se pode identificar deficiências ou, se houver falhas, os defeitos que as causaram. Um exemplo de uso prático dessa característica poderia ser a disponibilização de informações sobre as operações internas que o software realiza. No caso de um programa de cálculo, a exatidão dos valores intermediários permite aumentar a confiabilidade nos resultados finais e descartar uma eventual hipótese de cancelamento de erros.

A estabilidade do produto é uma forma especial de robustez: na definição da 9126, é a capacidade de o produto evitar que modificações levem a efeitos inesperados. Uma das formas de conseguir isso é empregando algoritmos redundantes.

Por fim, a testabilidade segundo a 9126 é a capacidade de o software permitir que, uma vez modificado, seja validado.

11.5.3 USABILIDADE

A usabilidade representa basicamente o quão fácil é usar o produto. Esta é provavelmente a característica mais difícil de tratar, tanto durante a definição de requisitos quanto durante os estágios do ciclo de vida, na verificação e na validação do produto. O motivo para isso é que a característica de usabilidade depende, sobretudo, da interface com o usuário. Por esse motivo, está é a característica que envolve a maior carga de fatores subjetivos durante a análise. Segundo Nielsen [1993]:

> Embora não esperemos que auditores se tornem especialistas em psicologia cognitiva, queremos que estejam cientes de que grande porcentagem do desempenho de uma tarefa ocorre de maneira invisível na cabeça do usuário.

Durante uma avaliação de usabilidade, é preciso distinguir a influência do software e a influência do avaliador em um teste. Fatores como o nível de atenção, habilidades cognitivas – e mesmo motoras – podem mudar completamente os resultados. Um exemplo seria utilizar dois avaliadores para julgar uma interface particularmente complexa de um produto: uma secretária e uma programadora. Embora ambas utilizem computadores com freqüência, o conhecimento da segunda a respeito da construção interna de softwares pode auxiliá-la a compreender mais depressa a operação da interface. Além disso, é

praticamente impossível determinar o efeito de fatores como ruídos ambientes, desconforto ou talvez uma preocupação pessoal sobre o nível de atenção e de desempenho.

A usabilidade é dividida em quatro subcaracterísticas.

A operabilidade do software representa a possibilidade de ele ser controlado pelo usuário. À primeira vista, isto pode parecer um requisito óbvio e até mesmo automaticamente atingido: se uma função existe no programa, então pode ser acessada por meio de interface, entretanto, uma vez mais é preciso não confundir o conceito clássico de requisito funcional com o modelo de qualidade 9126. A ausência de certas operações pode denotar a falta de operabilidade. Exemplificando, se uma operação iniciada por um usuário – como imprimir uma listagem ou realizar uma cópia de segurança dos dados – não puder ser interrompida, isto corresponderá à falta de controle e de operabilidade.

A definição de compreensibilidade da 9126 é bastante ampla e traduz-se na capacidade de o programa permitir que o usuário compreenda – ou decida – se ele é apropriado ou não às suas tarefas. Para que o usuário possa efetuar esse julgamento, é natural que ele possa interagir com todo o programa, fornecendo entradas e interpretando os resultados fornecidos. Elementos como a maneira de apresentar informações nas interfaces, seqüenciar as tarefas e até o texto das mensagens impressas fazem parte da avaliação dessa subcaracterística. Um vocabulário inapropriado, como, por exemplo, um software para uso médico que utiliza termos técnicos incorretos, é um exemplo de fator que pode tornar mais difícil o uso do programa.

A subcaracterística de apreensibilidade representa a facilidade para aprendizado de uso do programa. Pode-se materializar por meio de interfaces intuitivas, que utilizem elementos de comunicação já conhecidos pelo usuário ou que provavelmente ele possa deduzir. Outra possibilidade são os avatares – personagens (virtuais) – que são apresentados na interface e que interagem com o usuário para ajudá-lo a definir dúvidas e realizar as tarefas. Programas que novos usuários possam aprender a operar com facilidade possuem menores custos de treinamento e podem representar maior produtividade.

Atratividade foi uma subcaracterística que inicialmente gerou certo debate nos comitês que cuidavam da norma 9126. O fato de um software ser “bonito” ou “feio” não é um dado muito comum em uma análise de qualidade. Algumas aplicações, entretanto, tornam claro como tal aspecto do produto pode ser importante, como software educativo para crianças e jogos de computador.

A subcaracterística de atratividade não se reduz à “beleza”: deve-se avaliar a capacidade da interface de atrair e manter a atenção do usuário [Sutcliffe, 2002]. Novamente, técnicas de psicologia cognitiva combinadas à estatística

podem ser empregadas para esclarecer o grau de satisfação desse requisito no prosuto.

11.5.4 CONFIABILIDADE

Habitualmente, diz-se que um produto é confiável quando não falha. E em software, sabemos que a ocorrência de falhas é sempre uma possibilidade. Então, o que fazer?

Em primeiro lugar, como foi feito com a característica “funcionalidade”, aqui também é preciso definir um escopo. De acordo com a ISO/IEC 9126, a confiabilidade de um programa se traduz como a capacidade de manter um certo nível de desempenho quando operando em um certo contexto de uso.

Entre as subcaracterísticas de confiabilidade encontram-se requisitos classificados por Sommerville [2003] como funcionais, por se tratarem de operações que o software é programado para realizar. Assim temos a tolerância a falhas e a recuperabilidade, ou seja, recuperação de falhas. Esse fato ocorre também com outras características de qualidade do modelo 9126.

No primeiro caso – tolerância –, espera-se que o programa seja capaz de, dentro de certos limites, continuar a funcionar quando algo inesperado ocorrer. Por exemplo, o espaço em disco necessário à operação torna-se menor do que o especificado nos requisitos do software. Numa situação como essa, seria desejável que o programa contornasse o problema, evitando errou ou, no pior caso, encerrando a execução de uma maneira controlada, mas não abortando repentinamente.

No segundo caso – recuperabilidade –, o comportamento esperado é que o software possa voltar a funcionar corretamente após a ocorrência da falha. Um exemplo muito conhecido é a recuperação de arquivos do editor de textos Word, da Microsoft. Após queda de energia ou outra interrupção externa ao programa, ao ser novamente executado, é capaz de apresentar ao usuário uma versão do documento contendo o texto digitado até alguns momentos antes da falha.

A maturidade é uma característica difícil de garantir e significa, de maneira geral, que o programa é robusto. A definição diz que o software deve ser capaz de evitar falhas em virtude de defeitos. Isto só será possível se o programa contiver, por exemplo, teste para condições de erro pouco prováveis e tratamento de exceções.

É usual também empregar o termo maturidade para referir-se a um produto que já foi extensivamente utilizado por vários usuários e no qual falhas já foram encontradas e corrigidas. Quanto mais o produto, menor a possibilidade de existirem falhas.

11.5.5 EFICIÊNCIA

A norma 9126 estabelece duas dimensões para avaliar a eficiência de um software: o tempo e a utilização de recursos.

Todas as medidas de softwares devem ser feitas em ambiente controlado. Algumas dessas medidas, como as utilizadas para avaliar a subcaracterística de comportamento temporal, mostram de maneira mais evidente a importância desse controle.

A velocidade de operação de um software pode ser afetada por inúmeros fatores: velocidade da CPU, quantidade de memória cachê e memória RAM, desempenho de disco rígido, volume de tráfego de rede, interação com outros softwares e com o sistema operacional, configurações deste último etc. ao especificar o comportamento desejado ou ao medi-lo, o desenvolvedor deve especificar também as condições sob as quais aquela medida será válida (contexto de uso).

Na 9126, a segunda subcaracterística – utilização de recursos – engloba todo recurso que não seja o tempo de CPU: quantidade de memória, carga de CPU, ocupação de disco etc.

Alguns cuidados especiais são necessários para realizar a medição de eficiência de um programa. É provável que em muitos casos seja difícil ou mesmo impossível controlar com precisão o ambiente de execução do software. Se, por exemplo, uma medida de eficiência como “memória ocupada” depende do tráfego de rede, a flutuação natural desse parâmetro levará a resultados diferentes quando a medição for repetida. A solução, nesse caso, consiste em estabelecer os limites mínimo e máximo de operação, assim como um valor médio. No caso de sistemas cujo desempenho dependa da alocação de recursos, uma ferramenta interessante a considerar é a análise quantitativa [Tazza, 1987].

11.5.6 PORTABILIDADE

A palavra “portabilidade” é utilizada geralmente para indicar a possibilidade de um código-fonte ser utilizado em diferentes plataformas de execução. Na norma 9126, a definição foi estendida para abranger a idéia de portar aplicações entre organizações diferentes. Em tese, supõe-se que um programa possa, então, ser elaborado para operar em ambientes com características diferentes.

A subcaracterística de instabilidade não afeta apenas o programa de instalação de um produto: freqüentemente, a primeira execução de um determinado programa exige a definição de uma série de parâmetros pelo usuário.

A subcaracterística de adaptabilidade mede a capacidade de o software operar em ambientes diferentes mediante modificação de configurações que já traz embutidas. A adaptação pode, por exemplo, ser realizada por um usuário final em vês de um técnico especializado.

A co-existência não deve ser confundida com a interoperabilidade. A subcaracterística de co-existência refere-se ao compartilhamento de recursos com outros produtos e não a trocas de dados entre eles.

Por fim, a substitutibilidade pode ser compreendida como compatibilidade. Essa subcaracterística indica que um dado produto pode ser usado em lugar de outro, em um dado ambiente e para execução de uma determinada tarefa.