Perfil Operacional – Estratégia essencial ao Teste de Software

ANTONIO MENDES DA SILVA FILHO*

"Learn from yesterday, live for today, hope for tomorrow. The important thing is not to stop questioning."
Albert Einstein

Teste de software é uma das atividades do processo de desenvolvimento de sistema de software que visa executar programa de maneira sistemática com o objetivo de encontrar falhas. Minimizar a ocorrência de falhas em software é essencial e como consequência, tem um aumento na confiabilidade de software. Não há exemplo melhor de liberdade e ser criativo que uma criança. Nesse sentido, este artigo apresenta uma prática de engenharia de software de considerar o perfil operacional de um sistema de modo a melhor alocar recursos para atividades de teste de software objetivando aumentar a confiabilidade de um software. Isso visa tornar eficiente os testes de software. [1], [2], [3] e [4].[1]

Teste de software é uma das atividades do processo de desenvolvimento de sistema de software que visa executar programa de modo sistemático com o objetivo de encontrar falhas. Perceba que teste (de software) não é a última atividade do processo de desenvolvimento de software. Uma visão geral do processo RUP (Rational Unified Process) mostrado na **Figura 1**. Observe que as atividades de teste ocorrem durante todo o processo.

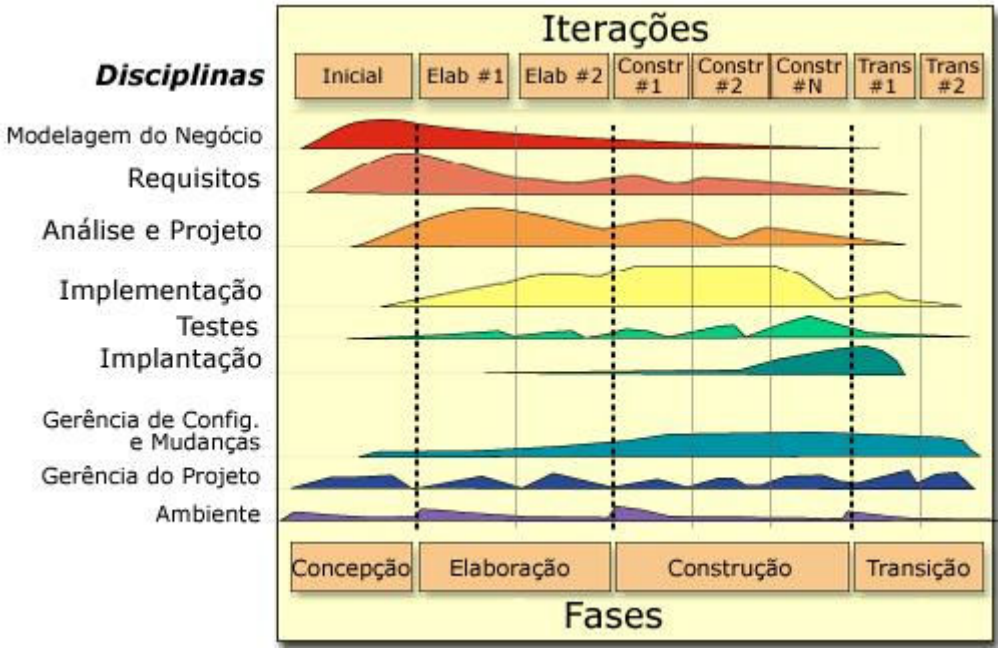

Figura 1. Visão geral do RUP.

(http://www.wthreex.com/rup/portugues/process/ovu_proc.htm)

Agora, como o objetivo básico do teste do software é encontrar falhas, é necessário definir como o software será testado e quanto tempo será gasto com essa atividade. Em outras palavras, torna-se necessário estabelecer quanto de esforço será dedicado aos testes de software de modo a ter resultado satisfatório. Perceba aqui que é preciso ter uma estratégia que torne a atividade de teste de software mais eficiente. Nesse sentido, este artigo apresenta o perfil operacional (ou operational profile) como uma informação estratégica que objetiva auxiliar a atividade de teste de software.

Teste de Software

Software permeia nosso cotidiano e assegurar que ele funcione corretamente, entregando as funcionalidades (para as quais ele foi projetado) de modo confiável é função do engenheiro de software. No entanto, do ponto de vista prático, as questões abaixo surgem:

1. É possível ter um software 'perfeito'?
   a. Se sim, como?
   b. Se não, quando parar de testar?

Antes de responder às questões acima, deve-se lembrar que o componente hardware nos sistemas computacionais é considerado como confiável atualmente. Entretanto, não se pode dizer o mesmo do (componente) software. A confiabilidade de software tem papel relevante nos sistemas atuais e de suma importância nos sistemas críticos.

Ter o software como elemento essencial dos sistemas coloca sobre ele a necessidade de assegurar qualidade e, mais especificamente, sua capacidade de não apresentar falhas quando em uso, isto é, o software tem de ser confiável. A **confiabilidade de software** é definida como a probabilidade do software operar sem ocorrência de

falhas durante um período especificado de tempo em um determinado ambiente.

Com esses conceitos em mente, já podemos responder às questões acima. A resposta para primeira pergunta é não. Para entender isso, você precisa lembrar que faltas podem ser introduzidas em qualquer uma das fases do desenvolvimento de software, ou seja, especificação, projeto, codificação, testes e manutenção. (Vale lembrar que se diz que um sistema possui uma **falta** se para algum dado de entrada, o comportamento do sistema está incorreto, isto é, seu comportamento é diferente daquele incluído na especificação do software. Portanto, uma falta é uma causa identificada da falha de software ou erro interno do sistema, o qual é comumente denominado de **bug**. Entretanto, quando a distinção entre falta (causa da falha) e falha (o efeito observável) não é critica, utiliza-se o termo genérico **defeito** para se referir à falta ou a falha).

Além disso, você também deve lembrar que os requisitos de software evoluem, ou seja, eles estão sujeitos a mudanças. Outro aspecto, que não pode ser esquecido, é que software em sistemas de médio a grande porte possui complexidade (inerente) em termos de código, arquitetura e de funcionalidades.

Perceba que essa dificuldade de se obter um 'software perfeito' deve-se essencialmente à natureza do software.

Respondida a primeira questão, resta a segunda: quando parar de testar?

Aqui, temos mais de uma resposta: podemos parar de testar quando não há mais recursos ou não há mais tempo. Outra razão para encerrar os testes é quando a meta de confiabilidade foi atingida ou uma determinada quantidade de falhas foi encontrada.

Cabe salientar aqui que, além de encontrar falhas, testes objetivam aumentar a confiabilidade de um sistema de software, isto é, aumentar a probabilidade de que um sistema continuará funcionando sem falhas durante período de tempo. Nesse sentido, para que os testes sejam realizados com eficiência, isto é, revelem um maior número de falhas (restando se qualquer um número muito pequeno de falhas) e, portanto, aumentem o nível de confiabilidade de software até a meta desejada, pode-se utilizar o perfil operacional ou operational profile.

Embora seja desejável testar um sistema por completo, deve-se ter em mente que atividade de teste assegura apenas encontrar falhas se ela(s) existirem, mas não asseguram sua ausência. Portanto, as atividades de teste devem ser disciplinadas a fim de identificar maioria dos erros existentes. Para tanto, um informação essencial a ser considerado é o perfil operacional do sistema o qual é tratado na seção seguinte.

Perfil Operacional

A atividade de teste de software demanda tempo e recursos ao longo do ciclo de vida do software. Realizá-la de maneira eficiente implica numa redução de custos e tempo. Uma estratégia que pode contribuir para alcançar esta meta é utilizar informações do perfil operacional do sistema de software considerado.

Teste de software tem duas metas: encontrar falhas (se elas existirem) e, conseqüentemente, aumentar a confiabilidade de software. Uma atividade que pode anteceder a atividade de teste de software é desenvolver o perfil operacional (do sistema em desenvolvimento). O

processo de determinação e avaliação da confiabilidade de software é ilustrado na **Figura 2**.

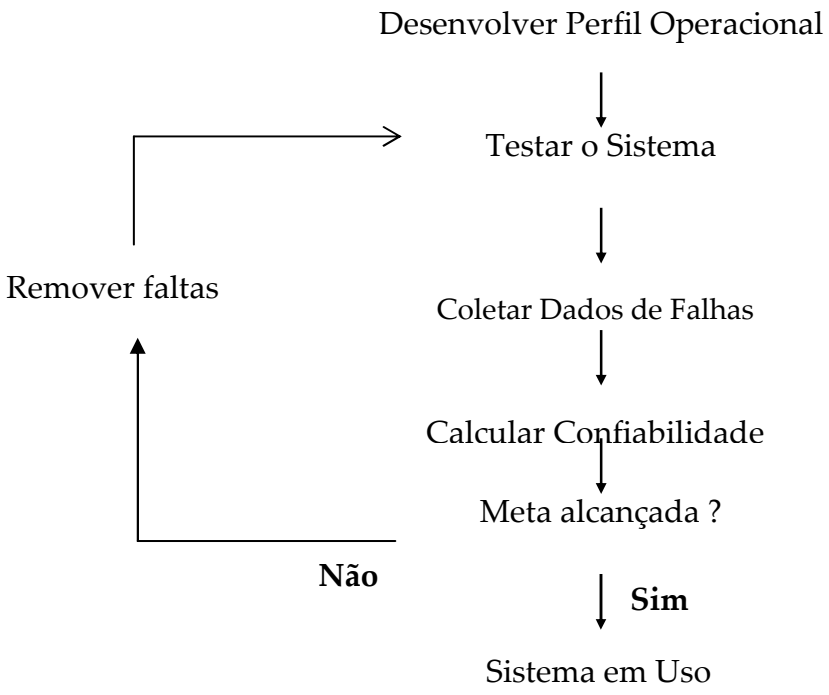

Figura 2. Processo de determinação e avaliação da confiabilidade de software.

Lembre-se que um dos motivos para encerrar os testes de software é alcançar a meta de confiabilidade. Assim, para determinar a confiabilidade, você deve verificar se a confiabilidade alcançada satisfaz ou não a meta definida de confiabilidade. Se o nível de confiabilidade é atendido, então o sistema está pronto para entrar em uso. Caso contrário, mais faltas devem ser removidas, como ilustrado processo acima

Observe que uma atividade essencial desse processo, a qual oferece informações às atividades de testes, é o desenvolvimento de um perfil operacional do sistema. Um perfil operacional é um conjunto completo de operações com suas respectivas probabilidades de ocorrência. Aqui, uma operação representa uma importante tarefa que é inicializada por algum ator a qual terá o controle retornado quando a tarefa tiver sido concluída. Note que a tarefa aqui se trata de uma funcionalidade do sistema. Por exemplo, uma probabilidade de ocorrência de 0,20 para o encaminhamento de uma ligação telefônica significa que 20 de cada 100 operações telefônicas serão para encaminhamento de ligação telefônica.

Você deve utilizar a informação do perfil operacional para guiar as atividades de testes. Para tanto, cinco etapas são necessárias:

1. Identificar atores das operações
2. Criar uma lista de operações

3. Revisar a lista operações
4. Determinar as taxas de ocorrência
5. Determinar as probabilidades de ocorrência

Essas cinco etapas acontecem no início do desenvolvimento do sistema de software e são detalhadas e revisadas durante as demais fases do desenvolvimento.

A primeira etapa consiste em **identificar os possíveis atores responsáveis por inicializar operações**. Esses atores compreendem:

- Usuários que fazem uso do sistema (causando o início de uma operação). Esses usuários podem ser instituições ou pessoas que utilizam o sistema no seu cotidiano.
- Sistemas externos (ou componentes desses sistemas) que iniciam operações sistema em desenvolvimento.

A segunda etapa exige você **criar uma lista de operações** e, para tanto, inicialmente, você deve criar uma lista de operações para cada tipo de ator. Para fazer isso, você consultar toda documentação do projeto, tais como requisitos do sistema, manual do usuário, eventuais protótipos (se existirem) e diagramas do projeto. Com o objetivo de entender melhor as operações, você pode também se reunir com partes interessadas no projeto (isto é, stakeholders), tais como engenheiro de software, projetista de interface, possíveis usuários, dentre outros.

Observe que você deve entender e definir cada operação do ponto de vista do ator (ou seja, aquele que a inicia), sem considerar como ela será implementada. Cabe destacar que até mesmo operações de inicialização de sistema devem ser consideradas.

Cada operação deve ter um caso de teste associado a ela de modo que cada operação seja executada pelo menos uma vez. O propósito de se identificar operações considera que cada operação deveria ter processamento distinto, o que pode motivar um comportamento com falha. Quando você vai realizar testes num software, você tem o objetivo de fazê-lo de forma eficiente e, portanto, de maximizar as chances que seus casos de testes possam revelar faltas (que causarão falhas).

Uma informação empírica levantada e utilizada pela comunidade de Engenharia de Confiabilidade de Software diz que há uma probabilidade de aproximadamente de 0.7 que duas operações terão faltas distintas se elas variam em cerca de 100 linhas de código. Nesse sentido, você deve procurar identificar operações que tenham processamentos significativamente diferentes (isto é, em cerca de 70%), o que poderia motivar comportamento inválido (contendo falha). Lembre-se que em teste de software, a meta é executar cada operação seja executada, pelo menos, uma vez, a menos que ela não seja crítica (ou seja, que tenha baixa probabilidade de ocorrência) e, portanto, não sendo muito utilizada por usuários.

Uma vez que você tenha obtido um conjunto de operações, então você deve **revisar a lista de operações**, o que constitui a terceira etapa no desenvolvimento de um sistema operacional. Para tanto, você não deve se preocupar se sua lista está completa ou não. Observe que cada operação será inicializada por um ator (que pode ser

um usuário, componente do sistema ou algum sistema externo). Em tal situação, é uma boa prática ter um especialista para cada tipo de operação.

Considere, por exemplo, que você é parte de uma equipe que está desenvolvendo um software para uma central telefônica. Num sistema como esse que requer que inúmeros testes sejam realizados para assegurar a confiabilidade de software, você precisará de especialista para parte da central encarregada de tratar o processamento das ligações telefônicas e, possivelmente, outra pessoa especialista na administração de sistema que tem operações sobre adição e remoção de assinantes.

Adicionalmente, você pode ter outras operações tais como redirecionamento de chamadas telefônicas (também conhecido como *call forward*) ou de tarifação e bilhetagem automática (responsável pela observação de dados de tráfego) que possibilite a medição e registros diários, em forma de relatórios específicos para determinação de custos, ocupação dos troncos e ramais, duração de chamadas, avaliação da carga de serviço em períodos específicos.

Perceba que se trata de operações diferentes e, para tanto, é (quase) imprescindível a participação de especialistas para fazer a revisão da lista de operações. A quantidade de operações dependerá do tamanho do sistema e poderá conter de poucas dezenas até centenas de operações.

Lembre-se de que você terá um custo associado que gira em torno de 0,5 hora para cada operação considerada. Outro aspecto a destacar é que o perfil operacional de um sistema evolui à medida que o sistema vai sendo desenvolvido, pois mais informações são obtidas, o que permite seu refinamento. Um exemplo de um conjunto de operações de um software de central telefônica com respectivos atores (responsáveis por inicializá-las) é apresentado na tabela abaixo.

Ator	Operação
Controlador de ligação telefônica	**Processar ligação telefônica completadas**
Controlador de ligação telefônica	**Processar ligação telefônica não completadas**
Gerenciador de recursos	**Controlar recursos p/ completar ligação telefônica**
Assinante	**Iniciar ligação telefônica**
Assinante	**Cancelar ligação telefônica**
Assinante	**Redirecionar ligação telefônica**
Administrador de sistema	**Adicionar assinante**
Administrador de sistema	**Alterar dados de assinante**
Administrador de sistema	**Remover assinante**
Gerenciador da base de dados	**Validar a base de dados de assinantes**

Tabela 1- Exemplo de lista de operações para uma central telefônica.

Observe na tabela acima que as operações listadas compreendem funcionalidades de um sistema exemplo (central telefônica). Embora isso tenha sido utilizado aqui para sistema telefônico, o mesmo procedimento pode ser feito em outros sistemas como um caixa eletrônico, um editor de texto ou mesmo um sistema de informação.

Em um caixa eletrônico, exemplos de operações são *sacar*, *consultar saldo*, enquanto num sistema acadêmico de uma universidade pode-se ter *matricular aluno* como exemplo de operação. De um modo geral, em sistemas orientados a objetos pode-se ter cada operação associada a um ou mais casos de uso (existente na modelagem de sistemas).

Depois de revisar a lista de operações, o próximo passo é **determinar a taxa de ocorrência** para o conjunto de operações levantado. A taxa de ocorrência de uma operação compreende a quantidade de ocorrências da operação dividida pelo tempo que o conjunto de operações está sendo executado. Determinar a taxa de ocorrência de operações é essencial para obtenção do perfil operacional de um sistema de software. No entanto, trata-se de uma tarefa que requer habilidade do engenheiro de software em buscar dados e, para tanto, você pode:

- Primeiramente, buscar por dados numa versão anterior do mesmo sistema (se existir) ou em sistemas similares.
- Buscar por dados no *log* de sistema (se existir).
- Conversar com pessoal da área de Marketing ou quaisquer outras informações relativo às regras de negócio do produto (sistema de software).
- Registrar dados de uso (isto é, dados de campo) do sistema considerado.
- Fazer simulações do sistema e determinar a taxa de ocorrência de eventos que estimulam a execução de operações.
- Fazer estimativas (caso não exista quaisquer informações iniciais) utilizando-se do conhecimento de profissionais experientes (engenheiro de software e/ou engenheiro de sistema) da área ou então aplicar o método Delphi no qual um grupo de especialistas se reúne fazendo estimativas, discutindo e refinando essas estimativas, e por fim, fazendo estimativas refinadas.

Para o conjunto de operações mostrado na Tabela 1, apresentam-se na Tabela 2 exemplos de taxa de ocorrência dessas operações.

Operação	Taxa de ocorrência (operações/hora)
Processar ligação telefônica completadas	**10.000**
Processar ligação telefônica não completadas	**5.000**
Controlar recursos p/ completar ligação telefônica	**8.000**
Iniciar ligação telefônica	**8.000**
Cancelar ligação telefônica	**5.000**
Redirecionar ligação telefônica	**2.800**
Adicionar assinante	**180**
Alterar dados de assinante	**10**
Remover assinante	**10**
Validar a base de dados de assinantes	**1.000**
Total	**40.000**

Tabela 2- Exemplo de taxas de ocorrências para uma central telefônica.

As taxas de ocorrências apresentadas na Tabela 2 visam informar a quantidade de vezes que cada operação é executada no período de uma hora. Em outras palavras, ela nos diz quais operações (ou funcionalidades) são mais requisitadas pelos usuários ou por outros atores do sistema.

É importante também ressaltar que as dez operações apresentadas na Tabela 2 não contemplam todas as operações (ou conjunto de funcionalidades) de um software de central telefônica. Nesse sentido, o desenvolvimento de um perfil operacional de um sistema consiste em criar e utilizar uma organização (ou arquitetura) baseada na operação ao invés de ser baseada em componente. Perceba que as operações (diferentemente de componentes) de software são partes de um produto e, portanto, as operações podem ser utilizadas para planejar a entrega do produto com diferentes releases (ou versões) que suportam operações específicas.

Outro aspecto a destacar é o uso do *Princípio de Pareto* utilizado no desenvolvimento do perfil operacional de um sistema de software. Tipicamente, num software cerca de 5% das operações de um software são responsáveis por oferecer 80% das funcionalidades desejadas pelos usuários.

Uma vez que você tenha determinado a taxa de ocorrência do conjunto de operações do sistema de software, o próximo passo é **determinar as probabilidades de ocorrência**.

Para tanto, você deve dividir a taxa de ocorrência de cada operação pelo somatório (ou total) da taxa de ocorrência das operações do sistema. Em seguida, você deve ordenar as probabilidades obtidas em ordem descendente, geralmente, com o objetivo de destacar aquelas que possuem uma maior probabilidade de ocorrência. Realizando este procedimento na Tabela 2, você irá obter a Tabela 3 das probabilidades de ocorrência do sistema exemplo. Observe que com as informações apresentadas na Tabela 3, você poderá distribuir a quantidade e esforço das

atividades de teste de software de maneira proporcional à probabilidade de ocorrência apresentada no quadro de perfil operacional do sistema que destaca um conjunto de operações que têm maior probabilidade de ocorrência. Perceba ainda que essa informação pode até mesmo ser utilizada nos estágios iniciais de desenvolvimento de software a fim de priorizar quais operações são consideradas como *críticas* e que devem ser tratadas prioritariamente nos primeiros releases (versões) do produto.

Operação	Probabilidade de ocorrência
Processar ligação telefônica completadas	**0.25**
Controlar recursos p/ completar ligação telefônica	**0.20**
Iniciar ligação telefônica	**0.20**
Cancelar ligação telefônica	**0.125**
Processar ligação telefônica não completadas	**0.125**
Redirecionar ligação telefônica	**0.07**
Validar a base de dados de assinantes	**0.025**
Adicionar assinante	**0.0045**
Alterar dados de assinante	**0.00025**
Remover assinante	**0.00025**
Total	**1.0**

Tabela 3- Exemplo das probabilidades de ocorrência para uma central telefônica.

Se você levar em conta o *Princípio de Pareto* e utilizar as informações obtidas no perfil operacional de um sistema de software, você poderá planejar a realização de testes e entrega de versões em três etapas:

1. A primeira versão do software deveria entregar 80% das funcionalidades desejadas pelo cliente, o que corresponde a aproximadamente e 5% das operações do software (pelo *Princípio de Pareto*).
2. A segunda versão, você poderia planejar entregar outros 15% das funcionalidades, o que poderia corresponder a cerca de 25% das operações.
3. Por fim, a terceira versão iria englobar os 5% restantes das funcionalidades, ou seja, aos 70% de operações menos utilizadas.

Se você participa ou está no comando do desenvolvimento do software de um sistema, você deve ter iniciativas que objetivam a tornar eficiente a realização das atividades. Torná-las eficientes

significa entregar o produto no prazo, com qualidade e custo reduzido. Isto requer uso eficiente dos recursos e, para tanto, você deve alocar recursos de desenvolvimento para melhor atender às necessidades do cliente o mais cedo possível. Portanto, adotar um planejamento de entregas em conformidade com o apresentado acima é uma boa prática.

Neste momento, é importante destacar os pontos essenciais de tudo o que foi apresentado. Você recorda a definição de confiabilidade de software?

Confiabilidade de software é a probabilidade de um sistema de software operar sem a ocorrência de falhas em um determinado ambiente operacional durante um período de tempo. Mas, o que compreende esse ambiente operacional? Hardware (computadores e infra-estrutura de apoio), software (sistema operacional, aplicativos e outros utilitários) e o grau de utilização (isto é, o perfil operacional) do sistema. Note que grau de utilização é uma caracterização quantitativa de como o sistema será utilizado.

Essa informação obtida com o perfil operacional de um software pode ser utilizada de duas maneiras:

1. Aumentar a eficiência dos testes a serem realizados;
2. Distribuir, de modo mais eficiente, o esforço de desenvolvimento de um sistema.

De tudo o que foi apresentado acima, uma lição deve ficar em mente: engenheiro de software, responsável pelas atividades de teste de software, deve estar envolvidos durante todo o processo de desenvolvimento de software, pois isto permite a ele:

- Ter um entendimento maior das necessidades do cliente (de como e quanto às funcionalidades serão utilizadas);
- Planejar melhor alocação de esforço e recursos no desenvolvimento do sistema (visando priorizar aquelas operações de maior ocorrência),

Para finalizar, vale ressaltar que as informações oferecidas num perfil operacional servem de apoio às decisões do gerente de projetos tanto sob a perspectiva de redução de custos quanto de identificação de faltas (causadoras de falhas de software) nas funcionalidades mais usadas.

* **ANTONIO MENDES DA SILVA FILHO** é Professor e consultor em área de tecnologia da informação e comunicação e Doutor em Ciência da Computação pela Universidade Federal de Pernambuco.

[1] [1] *Engenharia da Confiabilidade de Software*, disponível http://www.espacoacademico.com.br/027/27amsf.htm

[2] *Gestão de Projetos: Estratégia Essencial às Corporações*, disponível em http://www.espacoacademico.com.br/066/66amsf.htm

[3] *Componente de Software: Componentização no desenvolvimento de software*, disponível em http://www.espacoacademico.com.br/087/87amsf.htm

[4] *Arquitetura de Software – sobre a importância do reuso*, disponível em http://www.espacoacademico.com.br/068/68amsf.htm