# Uma Abordagem de Análise do Tempo de Resposta para Teste de Desempenho em Aplicações Web

**Priscila Guarienti, Maicon Bernardino, Avelino F. Zorzo e Flávio M. Oliveira**

[1]Programa de Pós-Graduação em Ciência da Computação
Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Caixa Postal 1.429 – 90.619-900 – Porto Alegre – RS – Brasil

priscila.guarienti@acad.pucrs.br, bernardino@acm.org,

{avelino.zorzo, flavio.oliveira}@pucrs.br

***Abstract.*** *The testing activity is usually very expensive and time consuming, since most of the time it is executed manually. To improve that, Model-Based Testing (MBT) has been used lately to generate testing artifacts automatically. The main contribution of this work is to investigate how MBT can be applied for performance testing, i.e. how to include timing constraints in UML diagrams and based on them to execute the tests automatically, and verify whether the constraints are met or not. Another contribution is the setting of service level agreements, i.e. set parameters to verify whether a system delivers answers based on pre-agreed values. We applied our whole strategy to a complete case study and measured its execution time, as well as discuss some of our results.*

***Resumo.*** *A atividade de teste é geralmente bastante onerosa, isso ocorre porque boa parte dessas atividades são executadas manualmente. Para melhorar isto, Model-Based Testing (MBT) tem sido utilizado para gerar artefatos de teste automaticamente. A principal contribuição deste trabalho é investigar como MBT pode ser aplicado em teste de desempenho, i.e. como incluir restrições de tempo em diagramas UML e com base nelas executar os testes e verificar se elas são satisfeitas ou não. Outra contribuição é a definição dos acordos de níveis de serviço, i.e. definir parâmetros para verificar se um sistema processa os tempos de respostas com base em valores pré-determinados. A abordagem proposta é aplicada a um estudo de caso real.*

## 1. Introdução

A evolução e o incremento da complexidade dos sistemas computacionais existentes têm tornado o processo de teste tão ou mais complexo que o processo de implementação do *software*. Para contornar este problema e aumentar a eficiência do processo de teste foram desenvolvidas diversas ferramentas com o objetivo de automatizar as atividades de verificação e validação (V&V).

Uma das abordagens que auxiliam o processo de teste de *software* é o teste baseado em modelos (*Model Based Testing - MBT*) [Apfelbaum and Doyle 1997], que tem o intuito de criar artefatos de teste que descrevem o comportamento do sistema. Neste modelo está incluso em sua especificação as características que serão testadas [El-Far and Whittaker 2001], com o objetivo de automatizar o processo de teste de

*software*. Esta abordagem consiste na geração dos cenários, casos e *scripts* de teste com base nos modelos do *software*. Além disso, o uso de MBT facilita as atividades desempenhadas pelos testadores, proporcionando maior eficácia e qualidade no processo de teste, reduzindo a probabilidade de má interpretação dos requisitos do sistema e/ou redução do tempo do teste.

Outro fator importante relacionado com a qualidade de *software* é o desempenho. Do ponto de vista do usuário final, um *software* possui bom desempenho quando apresenta um tempo de resposta aceitável, mesmo quando submetido a um volume de processamento próximo de situações reais [Meier et al. 2007]. Com o intuito de melhorar a qualidade do produto de *software*, a Engenharia de Desempenho de Software (*Software Performance Engineering - SPE*) [Woodside et al. 2007] aplica seus esforços para melhorar o desempenho de duas formas: no ciclo inicial baseado em modelos preditivos ou no ciclo final baseado em medições. Além disso, o teste de desempenho abrange atividades como: configurar o ambiente e o cenário de teste; identificar critérios de aceitação de desempenho; planejar e projetar os testes; executar os testes e analisar os resultados [Meier et al. 2007].

Neste contexto, a fim de aplicá-lo ao teste de desempenho, a abordagem proposta neste artigo enfatiza o MBT para avaliação de desempenho, como uma forma uniforme e formal de associar informações de tempo aos modelos UML. Posteriormente, com a execução dos casos de teste será possível coletar e analisar os resultados obtidos, além de comparar com os requisitos de tempo inicialmente estimados nos diagramas UML. Isso possibilita ter estimativas de tempo mais concretas e mensuráveis além de verificar a aceitabilidade do usuário em relação aos critérios de tempo estabelecidos nos acordos de níveis de serviço.

Além das contribuições nas áreas de MBT e teste de desempenho, a proposta deste trabalho contribui no desenvolvimento de uma Linha de Produto de *Software* (*Software Product Line - SPL*) [Clements and Northrop 2001] para geração de ferramentas de teste que usam MBT, chamada PLeTs (*Product Line for Model Based-testing tools*) [Rodrigues et al. 2010], [Silveira et al. 2011], [Costa et al. 2012]. De acordo com [Software Engineering Institute (SEI) 2014], SPL é importante e aplicável ao desenvolvimento de *software*, pois proporciona diversos benefícios durante o processo de desenvolvimento: menor custo, melhor qualidade, maior produtividade e maior velocidade de entrada em novos mercados. Isso é possível devido a flexibilidade que os componentes de *software* possuem na arquitetura da família possibilitando assim a reusabilidade.

Desta forma, este trabalho tem dois objetivos principais: auxiliar durante a fase de teste de *software*, prevendo situações e cenários que possam comprometer o desempenho de determinada aplicação e permitir a análise do tempo de resposta dos casos de teste. Para descrever estes aspectos foram definidos alguns estereótipos UML representando informações de tempo e que serão utilizados no processo de modelagem, geração e execução dos *scripts* de teste utilizando a PLeTs Perf [Silveira et al. 2011] que é uma ferramenta derivada da linha de produto PLeTs.

Este artigo está estruturado da seguinte forma. A Seção 2 faz uma breve introdução dos conceitos de teste de desempenho, SPL para MBT, técnica de avaliação e revisão de programas e acordo de nível de serviço. A Seção 3 apresenta detalhadamente a abor-

dagem deste trabalho. A Seção 4 apresenta um estudo de caso, bem como a análise e interpretação dos resultados. A Seção 5 discute alguns trabalhos relacionados. Ao final, a Seção 6 apresenta as considerações finais e trabalhos futuros.

## 2. Fundamentação Teórica

O teste de desempenho [Meier et al. 2007] tem por objetivo avaliar o comportamento do sistema submetido a uma determinada carga em um ambiente de teste específico. Ele fornece indicadores de desempenho que são utilizados para avaliar o quanto um sistema ou componente de um sistema é capaz de cumprir os requisitos de desempenho, tais como tempo de resposta ou *throughput*, além de identificar os possíveis gargalos que podem estar degradando o desempenho do sistema [Smith 2002].

Atualmente, está em desenvolvimento no Centro de Pesquisa em Engenharia de *Software* (CePES)[1] na PUCRS uma linha de produto de *software* denominada PLeTs [Rodrigues et al. 2010], que busca facilitar a derivação de ferramentas MBT, com as quais é possível executar casos de teste de forma automatizada. A relação da PLeTs com este trabalho diz respeito ao fato de que um dos produtos gerados por esta SPL refere-se a PLeTs Perf, o qual por meio da definição de transações e associação de tempo aos modelos UML serão definidos Acordos de Nível de Serviço (*Service Level Agreements - SLAs*) com o intuito de avaliar o desempenho das aplicações.

Um SLA caracteriza-se como um conjunto de procedimentos e objetivos formalmente acordados entre as entidades envolvidas com a finalidade de manter a qualidade de um serviço. Além de estipular métricas associadas para que o desempenho dos serviços e das aplicações possam ser medidos e avaliados [Blokdijk and Menken 2008].

Com base nas estimativas de tempo relacionadas as transações e por meio da Técnica de Avaliação e Revisão de Programas (*Program Evaluation and Review Technique - PERT*) [Stilian 1967] serão calculados os SLAs das transações. De acordo com Stilian a precisão das estimativas do tempo de execução de determinadas atividades podem ser aperfeiçoadas considerando as incertezas das estimativas e os riscos.

Neste contexto, esta técnica usa três estimativas de tempo para definir uma faixa aproximada para a execução de determinada atividade: I) Pessimista ($T_P$) - o tempo da atividade é com base na análise do pior cenário para a atividade; II) Mais Provável ($T_M$) - a duração da atividade, dados os prováveis recursos a serem designados, sua produtividade, expectativas realistas de disponibilidade para executar a atividade, dependências de outros participantes e interrupções; III) Otimista ($T_O$) - o tempo da atividade é com base na análise do melhor cenário para a atividade.

Com as estimativas de tempo estabelecidas a etapa seguinte consiste em calcular o tempo esperado ($T_E$) de execução de determinada atividade ou sequência de atividades. Este cálculo é obtido por meio de uma média ponderada por meio da seguinte fórmula:

$$T_E = \frac{(Pessimista + (4 * Mais\ Provavel) + Otimista)}{6}$$

Essa técnica é muito utilizada para planejar, avaliar e controlar o tempo de execução de programas e projetos. Após calcular o SLA, com base no cálculo do tempo

[1]Homepage CePES: http://www.cepes.pucrs.br

esperado, serão gerados os cenários e *scripts* de teste de desempenho que serão executados pela ferramenta LoadRunner [Hewlett Packard – HP 2014].

## 3. Abordagem Proposta

Em trabalhos anteriores [Silveira et al. 2011] [Costa et al. 2012] modelos e métodos para geração de sequências de teste de desempenho e teste estrutural foram investigados. Neste trabalho, será apresentada a aplicação de uma abordagem que tem por objetivo enfatizar o MBT, associando informações de tempo aos diagramas UML com o intuito de medir o tempo de execução das sequências do teste, bem como automatizar o processo de verificação dos requisitos de desempenho, *e.g.* tempo de resposta por meio dos SLAs.

A abordagem proposta de teste de desempenho é apresentada na Figura 1. Ela tem início com a divisão do processo de acordo com as ações que cada participante ou ferramenta envolvida realizará, *i.e.* `Tester`, `PLeTs` e `LoadRunner`. Este processo inicia com a atividade `Modelar os Diagramas UML`, desenvolvidos pelo `Tester` na especificação dos cenários de teste de desempenho, representado por diagramas de casos de uso (*Use Case Diagram - UC*) e por diagramas de atividades (*Activity Diagram - AD*). Com base no perfil UML SPT (*Schedulability, Performance and Time*) [Meier et al. 2007] são adicionadas aos modelos informações de teste relacionadas aos cenários de desempenho, cargas de trabalho (*workload*) e perfis dos usuários, representada pela atividade `Adicionar Dados de Teste de Desempenho`.

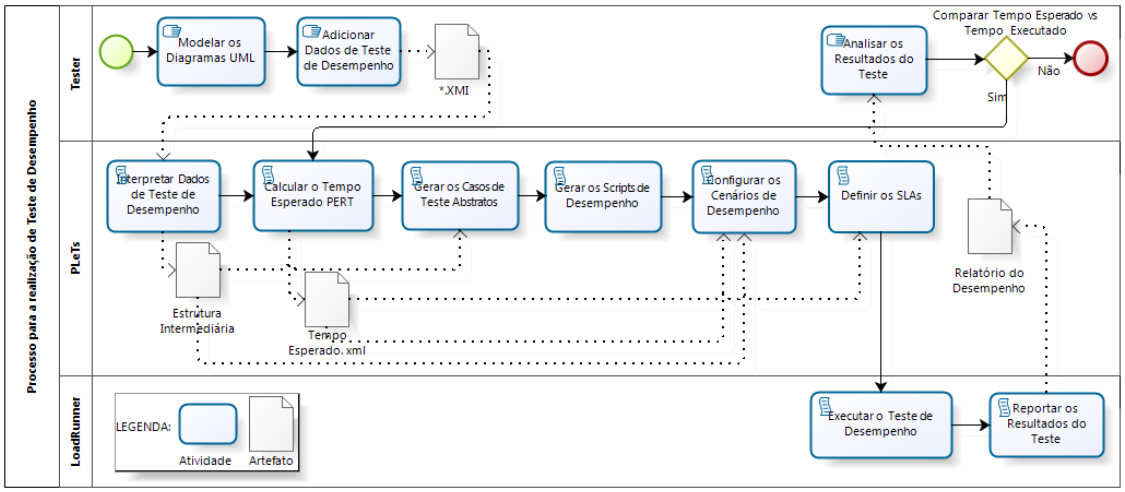


**Figura 1. Processo da abordagem de teste de desempenho baseado em modelos**

Visando a automação dos cenários e *scripts* de teste de desempenho pela `PLeTs`, o modelo UML gerado é exportado em um arquivo XMI (*XML Metadata Interchange*) que por sua vez é submetido ao *parser* da `PLeTs`, a fim de `Interpretar Dados de Teste de Desempenho` necessárias para a geração de uma estrutura de dados abstrata em memória (`Estrutura Intermediária`) que foram adicionadas por meio dos estereótipos.

Com o cenário de teste de desempenho especificado, cada caso de uso é decomposto em um AD, o qual demonstra o fluxo de ações executadas pelo usuário para realizar determinada tarefa no sistema. É nesta etapa que são definidas as transações que controlam as atividades e que serão necessárias para definir os SLAs e avaliar o tempo de

resposta dos *scripts* de teste de desempenho. É importante salientar que estimar tempo e esforço é ainda uma das atividades mais difíceis durante a fase de planejamento, isso inclui também as atividades de teste de *software*. No entanto, de alguma forma é necessário medir o esforço despendido para a realização dos casos de teste e avaliar se o resultado obtido está de acordo com o que foi estimado. Neste contexto e diante da dificuldade das equipes em estabelecer estimativas de tempo para a execução de testes de desempenho, os três pontos presentes na técnica PERT [Stilian 1967] (ver Seção 2), possibilitam uma faixa de variabilidade, sendo possível trabalhar com intervalos de tempo. A estimativa dos tempos diante das três perspectivas (pessimista, mais provável e otimista) estabelecidas pelo PERT é definida inicialmente a partir da execução de um *smoke test*[2]. Com base nesta técnica, a atividade seguinte é responsável por `Calcular o Tempo Esperado PERT` para cada uma das transações mapeados nos diagramas UML interpretados.

Em seguida, os diagramas UML com as informações de teste são convertidos em uma MEF (Máquina de Estados Finitos) e por meio da aplicação do método HSI (*Harmonized State Identification*) [Luo et al. 1995], são geradas as sequências de teste a serem executadas, representado pela atividade `Gerar os Casos de Teste Abstratos`.

As informações referentes a tecnologia que será usada são fornecidas nas etapas `Gerar Scripts de Desempenho` e `Configurar os Cenários de Desempenho`, quando o usuário deve fornecer todos os dados necessários para gerar os *scripts* e configurar os cenários de teste de desempenho usando uma ferramenta de teste específica, *i.e.* gerador de carga de trabalho. Na etapa a seguir é responsável por `Definir os SLAs` de acordo com as transações pré-definidas nos *scripts* de teste de desempenho gerados. Os cenários de teste de desempenho gerados serão usados posteriormente na atividade `Executar o Teste de Desempenho` pela ferramenta `LoadRunner` [Hewlett Packard – HP 2014].

Durante a execução dos *scripts* de teste de desempenho, o `LoadRunner` coleta os dados a fim de (`Reportar os Resultados do Teste`), que são gravadas em um arquivo XML denominado (`Relatório do Desempenho`) e reportados para a `PLeTs`. Desta forma é possível o `Tester` verificar as transações que cumpriram o SLA e as transações que não cumpriram, além de `Analisar os Resultados do Teste`, comparando o tempo mínimo, médio e máximo de resposta de cada transação, *i.e.* os tempos executados contra os tempos esperados.

As informações sobre cenários de teste de desempenho, carga de trabalho e perfil dos usuários são inseridas nos diagramas UC e AD por meio de estereótipos e rótulos (*tags*), que são necessários para geração dos *scripts* de teste de desempenho, bem como para avaliar a escalabilidade da aplicação em cenários com maiores requisições e concorrência. Neste contexto foram utilizados os estereótipos de desempenho conforme [de Oliveira et al. 2007] [Rodrigues et al. 2010] [Silveira et al. 2011] [Costa et al. 2012]: a) `TDtime`: determina o tempo total de execução do teste; b) `TDpopulation`: representa o número de usuários virtuais que irão executar a aplicação; c) `TDhost`: define o endereço (caminho) para executar a aplicação; d) `TDaction`: define o endereço do *link* acessado pelo ator; e) `TDmethod`: define o método de requisição HTTP utilizado pelo servidor; f) `TDparameters`: representa duas informações: nome e valor, que deve ser

[2]Processo no qual um produto de *software* é submetido, para verificar sua funcionalidade básica. Neste caso, será realizado um *smoke test* para coletar as estimativas de tempo que irão popular o modelo de teste.

preenchido para prosseguir, por exemplo, uma tela de login, ou um formulário de pesquisa; g) `TDprob`: indica a probabilidade de distribuição dos usuários virtuais em cada caso de uso. Esta informação deve estar presente em todas as associações *Actor-UseCase* e a soma de todos os rótulos `TDprob` anotados para um mesmo ator devem ser igual a 1 (ou seja, 100%). É importante ressaltar que o rótulo `TDprob` pode também ser informado nas transições existentes nos ADs com o objetivo de distribuir os usuários na execução de determinadas atividades da aplicação, como por exemplo em situações de paralelismo; h) `TDthinkTime`: denota o tempo existente entre a disponibilidade de execução da atividade até o início de sua real execução pelo usuário. Pode ser exemplificado pelo preenchimento de um formulário até sua submissão.

Estes estereótipos, bem como seus rótulos são apresentados nos exemplos nas Figuras 3 e 5. Como a maior contribuição deste estudo é gerar *scripts* de teste de desempenho considerando o tempo de resposta usando modelos UML, foi adicionado um novo estereótipo `TDexpTime`. Este estereótipo está presente nas transações do AD, e nele são informadas as estimativas de tempo referente à execução de cada transação. Caso esta informação não esteja presente no modelo, é executado um *smoke test* com o intuito de coletar os dados de teste de desempenho e gravar no arquivo XML `Tempo Esperado`.

Já a modelagem dos ADs é composta por duas etapas principais. A primeira etapa consiste na criação de um AD de alto nível, que na segunda etapa será expandido em outros ADs mais detalhados. Isto possibilita ter uma visão macro do processo de execução, além de permitir referenciar a um mesmo diagrama várias vezes, possibilitando, assim, o reuso dos modelos e dos *scripts* na execução dos testes de desempenho.

É importante ressaltar que o teste de desempenho também é composto por transações (*Transactions*) e requisições (*Requests*). Cada transação especifica um conjunto de requisições contidas em um *script* de teste de desempenho e são definidos com o intuito de isolar o processamento em diferentes pontos. Isto possibilita identificar os gargalos do sistema, além de mensurar as Transações por Segundo (TPS) e verificar se estas satisfazem os critérios de aceitação de acordo com as métricas estabelecidas no SLA.

O processo para definição dos SLAs inicia na criação dos modelos UML, em que são estabelecidas as transações e anotadas as estimativas de tempo para sua execução por meio do estereótipo `TDexpTime`. A partir do modelo UML criado um arquivo XMI é exportado. Em seguida, este arquivo XMI é interpretado e calculado o tempo de resposta esperado para execução de cada transação, por meio da técnica PERT. As informações do teste de desempenho, referente a criação do SLA, como nome da transação e estimativas do tempo ficam armazenadas em um arquivo XML, chamado `Tempo Esperado`, bem como o valor referente ao tempo de resposta esperado. Após calcular o valor esperado, tendo como referência os valores que estão presentes no arquivo XML, são gerados e executados os *scripts* de teste de desempenho, incluindo a criação dos SLAs no LoadRunner.

## 4. Estudo de Caso: Gerência de Portfólios e Projetos

Com o intuito de avaliar a abordagem proposta para teste de desempenho na perspectiva de inclusão de tempo nos modelos UML, esta seção detalha a realização de um estudo de caso, apresentando a atividade de modelagem, a geração dos cenários e *scripts* de teste de desempenho e sua instrumentalização para a ferramenta LoadRunner, juntamente com a definição dos SLAs, bem como análise dos resultados do teste. Para isso será

utilizada uma aplicação que tem por objetivo a gerência de portfólio de produtos e projetos multidisciplinares.

O cenário de testes de desempenho (Figura 2) é composto por quatro servidores da aplicação (*App Servers*), dois servidores *web* (*Web Servers*) e três servidores de banco de dados (*Database Servers*). Por meio da figura apresentada, também é possível verificar os aspectos de configuração de infraestrutura das máquinas, tais como: CPUs, memória e sistema operacional.

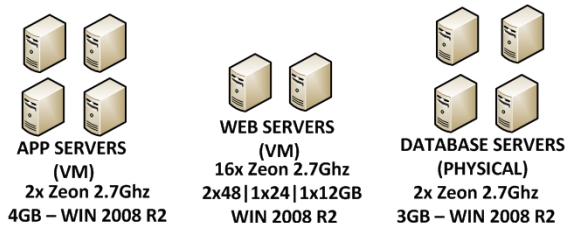


**Figura 2. Ambiente do teste de desempenho**

## 4.1. Modelos UML

O processo de modelagem tem início com a criação do cenário de teste de desempenho representado pelo UC (Figura 3). Nele é mostrada a interação do ator com os casos de uso com que ele está associado, tendo como base as principais funcionalidades da aplicação.

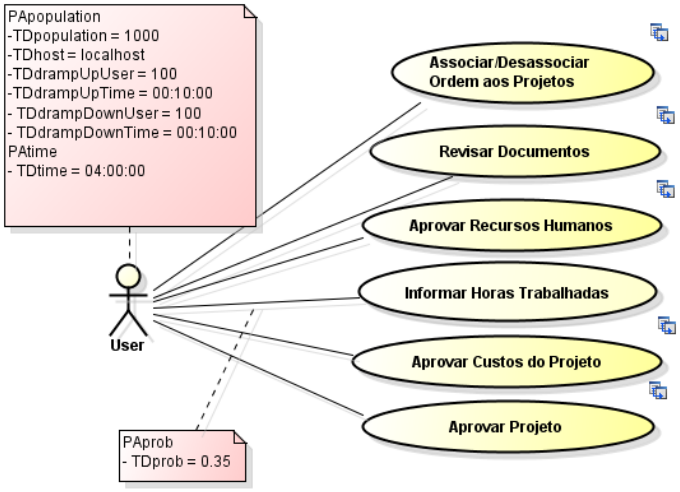


**Figura 3. Diagrama de casos de uso da aplicação**

Para a execução dos testes foi modelado um cenário de carga de trabalho composto por 1000 usuários virtuais, sendo que durante a inicialização do teste são acrescentados 100 usuários (`TDrampUpUser`) a cada 10 minutos (`TDrampUpTime`), totalizando 1h40min para a inicialização total do teste. A finalização do teste (`TDrampDownUser` e `TDrampDownTime`) também foi configurada com os mesmos parâmetros. Totalizando 3h20min entre a inicialização e finalização do teste, resultando em 40min de execução do teste com os 1000 usuários virtuais concorrentes para uma duração total de execução do teste de 4h (`TDtime`).

Após definido o cenário de teste, a próxima etapa de elaboração do modelo de teste é decompor cada caso de uso em um AD. Sendo assim, foram construídos ADs correspondentes aos casos de uso apresentados na Figura 3. Outra característica importante é o parâmetro `TDprob` que define a probabilidade de distribuição dos usuários virtuais em cada caso de uso relacionado a ele. Para representar este processo será exemplificado o caso de uso `Informar Horas Trabalhadas`[3]. A probabilidade do ator em executar este caso de uso é equivalente a `TDprob` = 35%, o qual representa 350 usuários virtuais dos 1000 concorrentes, conforme também é apresentado na Tabela 1, assim como os demais casos de uso que somados totalizam a carga de trabalho dos usuários virtuais no cenário de desempenho avaliado.

**Tabela 1. Distribuição da carga no cenário de teste de desempenho da aplicação**

| Ator | Caso de Uso | Probabilidade | Usuários Virtuais |
|---|---|---|---|
| User | Associar/Desassociar Ordem aos Projetos | 20% | 200 |
| | Informar Horas Trabalhadas | 35% | 350 |
| | Revisar Documentos | 20% | 200 |
| | Aprovar Recursos Humanos | 10% | 100 |
| | Aprovar Custos do Projeto | 10% | 100 |
| | Aprovar Projeto | 5% | 50 |
| | | | **1000** |

Nesta etapa inicialmente é criado um AD alto nível, identificando todas as atividades que serão necessárias para a criação e execução do *script* de teste de desempenho. Conforme observa-se na Figura 4, o AD é composto por três atividades principais, sendo que a primeira e a terceira são referentes as atividades de `login` e `logout`, e a do meio é referente a atividade `Informar Horas Trabalhadas`. É importante salientar que neste diagrama não existem estereótipos e informações nas atividades e nas transições, ele serve apenas para demonstrar a sequência de execução dos demais diagramas que compõem o cenário de teste e auxiliar no gerenciamento e reuso dos modelos.

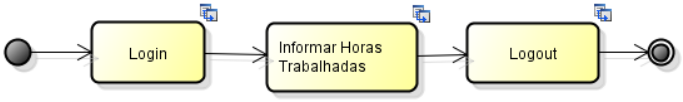


**Figura 4. Diagrama de atividades do caso de uso `Informar Horas Trabalhadas`**

Em seguida, cada uma das três atividades é expandida em um novo diagrama, que possuirá as informações de teste, anotadas por meio dos estereótipos apresentados na Seção 3. Já as estimativas de tempo diante das três perspectivas (pessimista, mais provável e otimista) são anotadas nas transações existentes no diagrama, conforme é possível observar na Figura 5.

O diagrama mostrado na Figura 5 representa a interação do usuário no sistema no momento em que o mesmo irá informar o tempo consumido em cada tarefa relacionada ao projeto. Esta ação é composta por três transações: na transação `Acessar Folha Ponto` estão presentes as ações de acesso à "Página Inicial" do sistema, selecionando em seguida as opções "Aplicação" e "Folha Ponto"; na transação `Folha Ponto Projeto` são

[3]Todos os diagramas UML modelados para o estudo de caso Gerência de Portfólios e Projetos estão disponíveis no endereço http: //www.cepes.pucrs.br/performance_time

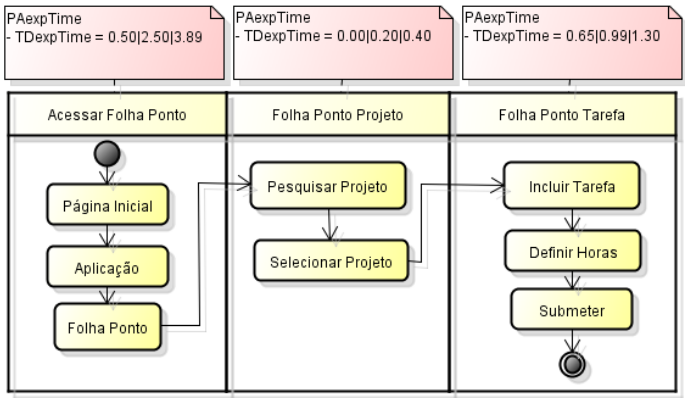


**Figura 5. Subdiagrama de atividades da atividade `Informar Horas Trabalhadas`**

executadas as ações "Pesquisar Projeto" e "Selecionar Projeto"; finalizando, na transação `Folha Ponto Tarefa` é composta pelas atividades "Incluir Tarefa", "Definir Horas" e "Submeter" os dados. Por meio do estereótipo `TDexptime`, anotado em cada transação, é possível verificar os tempos estimados para execução de cada sequência de teste. Por exemplo, na transação `Acessar Folha Ponto` foram estimados os seguintes tempos de execução: mínimo: 0.50, médio: 2.50 e máximo: 3.89. É importante salientar que este processo foi definido para os ADs da atividade `Login` e `Logout`, presentes na Figura 4, como também para os demais casos de uso representados pela Figura 3. Por meio das transações definidas nos ADs apresentados e das estimativas de tempo de resposta informadas é que serão definidos os SLAs para cada transação. Posteriormente, estas transações serão interpretadas pela ferramenta LoadRunner na execução dos cenários e *scripts* de teste de desempenho, conforme descrito na Seção 3.

### 4.2. Geração dos cenários e *scripts* de teste de desempenho

Os diagramas UML modelados na Seção 4.1 são representados por um arquivo XMI com as informações de teste. Essas informações por sua vez são submetidas a PLeTs a fim de convertê-las em um modelo formal, *e.g.*, Máquinas de Estados Finitos (MEFs).

Primeiramente, o modelo é analisado e convertido em uma estrutura de dados correspondente aos diagramas UML. A próxima etapa consiste em converter este modelo UML em uma nova estrutura genérica, chamada de estrutura intermediária que fica armazenada em memória, a qual será usada como entrada pelos diferentes modelos formais, *e.g.*, MEFs. Posteriormente, os modelos UML são convertidos em MEFs, e por meio da aplicação do método de geração de sequências de teste HSI [Luo et al. 1995] [Costa et al. 2012], são gerados os casos de teste abstratos. Finalmente, estes casos de teste abstratos são instanciados para a ferramenta LoadRunner a fim de que sejam gerados os cenários e *scripts* de teste de desempenho.

A Figura 6, apresenta um trecho do código XML, referente ao *script* de teste de desempenho gerado pela PLeTsPerf. Conforme pode-se observar, ele é composto pelas diversas requisições (*requests*), referente as ações realizadas pelo usuário no momento de incluir uma tarefa no projeto. Com base nos tempos estimados nos modelos é calculado o tempo esperado em segundos (*Threshold* = 0.985), que definirá o SLA dentro do arquivo de configuração do cenário de teste de desempenho. Por limitações de espaço foi apresentado somente um trecho do *script* LoadRunner, no entanto, nele estão presentes

todas as transações e atividades definidas no diagrama. Vale destacar, que com base nas transações presente no *script* e no tempo de resposta esperado configurado no cenário, neste momento também são criadas os SLAs.

```
1           lr_start_transaction("Folha Ponto Tarefa")
2             lr_start_sub_transaction("Incluir Tarefas")
3               web_url("Incluir Tarefas",
4               "URL=http://{servidor}/folha_ponto/projeto/tarefa/incluir_tarefa",
5               "Method=GET",
6               "Resource=0",
7               "Referer=http://{servidor}/folha_ponto/projeto/tarefa/incluir_tarefa",
8               "Mode=HTTP",
9               ITEMDATA,
10                "Name=PROJETO", "Value={{LISTA_PROJETOS}}", ENDITEM,
11                "Name=TAREFA", "Value={{LISTA_TAREFAS}}", ENDITEM,
12                "Name=UNIDADE_NEGOCIO", "Value={{LISTA_NEGOCIOS}}", ENDITEM,
13              LAST);
14            lr_end_sub_transaction("Incluir Tarefas",LR_AUTO)
15            lr_think_time(5);
16          lr_end_transaction("Folha Ponto Tarefa",LR_AUTO);
```

**Figura 6. Trecho de Script LoadRunner da transação** `Folha Ponto Tarefa`

Ao término da execução, o LoadRunner apresenta um resumo, demonstrando os tempos de resposta de cada transação, além do *status* de quais as transações que passaram no teste e quais as transações que falharam de acordo com o SLA definido, possibilitando assim, comparar e analisar os resultados obtidos com a realização do teste. É importante ressaltar que as informações referente a execução dos testes, bem como as transações e estimativas de tempo também são armazenadas em um arquivo XML chamado de `Expected Time`, conforme descrito na Seção 3 e são reportadas para a PLeTsPerf. Sendo assim, o usuário poderá além de analisar os resultados, interagir com a aplicação calibrando o modelos de acordo com o cenário e a carga que deseja testar, tendo como base os próprios resultados de execução dos testes.

### 4.3. Análise e Interpretação dos Resultados

Conforme é possível observar no gráfico apresentado na Figura 7 em que destaca a evolução do tempo de resposta em relação a cada uma das cargas de trabalho submetidas, essa transação compreende o conjunto de atividades apresentados na Figura 5 que faz referência ao caso de uso `Informar Horas Trabalhadas`. A figura apresenta os tempos de resposta para cada carga de trabalho, os tempos estimados de acordo com a técnica PERT (Pessimista, Mais Provável e Otimista), bem como o SLA definido pelo tempo esperado, e ainda os tempos executados do teste de desempenho, destacando o tempo Mínimo, Médio e Máximo coletados pelo LoadRunner.

Analisando o gráfico da Figura 7 pode-se evidenciar que a carga de 100 usuários obteve os tempos médios e mínimos muito inferiores às demais cargas, inclusive com um desvio superior ao SLA estimado. Entretanto, as demais cargas obtiveram tempos médios próximos aos SLAs definidos. Vale destacar que a carga de 300 usuários virtuais em que o tempo do SLA estimado foi de 1.67s contra o tempo médio executado pelo teste de desempenho de 1.60s, essa foi a menor diferença entre as estimativas de tempo (SLA) e o tempo executado, variando apenas 4.2%.

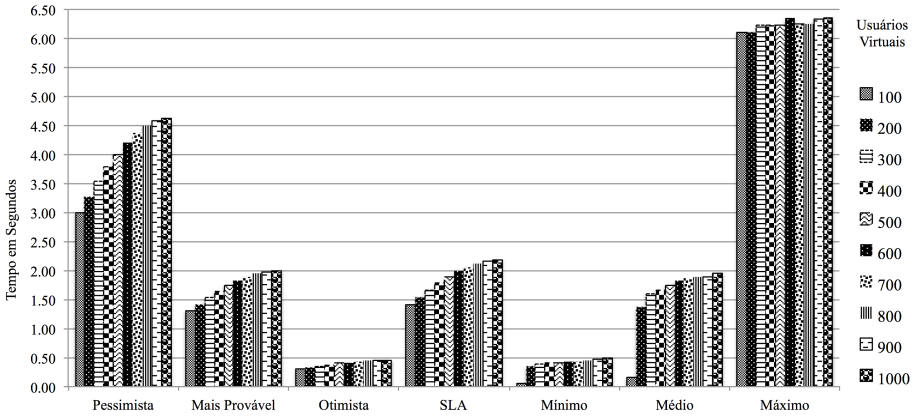


**Figura 7. Gráfico de análise do tempo de resposta da transação Informar Horas Trabalhadas do diagrama de atividades da Figura 4**

Com a realização do teste de desempenho, pode-se observar também que dependendo do aumento dos usuários virtuais que acessam o sistema, o SLA deverá ser revisto. Pois, à medida que mais usuários acessavam a aplicação, o tempo de resposta das sequências de teste tendem a aumentar e as transações que inicialmente cumpriam com o valor estabelecido no SLA passam a não mais cumprir, exceto nos casos em que a solução algorítmica seja linear. Outro fator que também deve ser considerado quando o SLA é definido é o ambiente em que a aplicação será executada, além dos históricos de execuções que permitam verificar o tempo de resposta das transações, permitindo assim, definir estimativas mais concretas e próximas ao ambiente real de execução dos testes de desempenho. Isso pode ser observado na Figura 7 apresentado, uma vez que a medida que novas execuções eram realizadas o valor estimado para o SLA ficava mais próximo do tempo médio, o que não acontece com a carga de 100 usuários, em que os tempos são estimados sem considerar execuções anteriores.

## 5. Trabalhos Relacionados

Atualmente, há um interesse crescente relacionado à área de teste de desempenho. Neste contexto, diversos autores têm apresentado abordagens e metodologias para testar o desempenho das aplicações utilizando técnicas de teste aplicadas no desenvolvimento de *software*. Em particular, teste de desempenho com modelos UML é um dos tópicos mais abordados pelos pesquisadores. Diversos trabalhos apresentam abordagens e metodologias que derivam cenários e *scripts* de teste de desempenho a partir de modelos UML adaptados para representar informações de desempenho, *e.g.*, UCs, ADs, etc. Alguns dos principais trabalhos nesta área são apresentados em [Kalita et al. 2011], [Geetha et al. 2011], [Vasar et al. 2012] e [Weiss et al. 2013].

Na abordagem proposta em [Kalita et al. 2011], é apresentada uma ferramenta chamada PReWebN (*Prototype Research Web application in Netbeans platform*) com o objetivo de avaliar o desempenho de aplicações com base em tecnologia Java. A ferramenta realiza uma análise estatística a partir dos dados coletados pela ferramenta de geração de carga de trabalho para diferentes métricas de desempenho, entre elas o tempo de resposta (*response time*). No entanto, a abordagem proposta pela autora não se baseia na abordagem MBT. Portanto, tal abordagem não se beneficia das vantagens da adoção

de MBT, como a validação de requisitos de desempenho por meio de modelos de teste, geração sistemática e automação dos SLAs.

Em [Geetha et al. 2011] um *framework* híbrido para um modelo de processo de predição de desempenho é apresentado, a fim de avaliar o desempenho de aplicações já na fase de estudo de viabilidade do projeto de *software*. Os autores propõem uma abordagem de engenharia de desempenho com base em diagramas de casos de uso UML, oferecendo flexibilidade para integrar o processo de previsão de desempenho com processos de engenharia de *software*. Apesar do estudo ser pautado na abordagem MBT, o *framework* proposto não leva em consideração a geração automática e análise dos SLAs.

Um *framework* usando uma heurística sensível ao tempo de resposta é apresentado em [Vasar et al. 2012] . Este *framework* serve como base para suportar a alocação dinâmica de servidores em ambiente de computação em nuvem, dada a carga de processamento exigido pela aplicação. Apesar do estudo levar em consideração a automação dos SLAs, os mesmos não são previamente definidos com base em modelos (MBT) e integrados a ferramentas de desempenho, tal como a abordagem proposta neste artigo, mas sim avaliados por meio da integração de ferramentas de gerenciamento da escalabilidade em ambientes de computação em nuvem. Por fim, em [Weiss et al. 2013] uma abordagem para avaliação sistemática de desempenho com base em aplicações de referência (*benchmark*) adaptadas é descrita. Assim como os demais trabalhos apresentados nesta seção, os autores também não contemplam em sua abordagem a automação dos SLAs como proposto pelo presente artigo.

Embora estes trabalhos apresentem alternativas que contribuem para minimizar o esforço na atividade de teste de desempenho, a maioria foca somente em avaliar as métricas de desempenho. Em geral, a maioria destas pesquisas deixa muitas lacunas quando se referem a automação dos SLAs de desempenho. Por outro lado, a abordagem proposta neste artigo também busca minimizar este esforço por meio da abordagem MBT para geração e avaliação sistemática do tempo de resposta usando SLAs.

## 6. Considerações Finais

Este trabalho apresentou uma abordagem de "Análise do tempo de resposta para teste de desempenho em aplicações *Web*", tendo como principal objetivo medir o tempo de execução das sequências de teste. Inicialmente, o processo de modelagem apresenta as informações necessárias para a realização de teste de desempenho que servem como entrada para as ferramentas de teste baseada em modelos, que são geradas a partir da linha de produto - PLeTs. Com base neste modelo as ferramentas MBT geram casos e *scripts* de teste de desempenho para um gerador de carga, *e.g.* LoadRunner. Para demonstração da abordagem proposta no trabalho foi desenvolvido um estudo de caso, em que foram criados cenários e *scripts* de teste de desempenho para a ferramenta LoadRunner, medindo o tempo de execução de cada transação em relação aos tempos definidos no SLA.

Após criar o SLA, o teste foi executado, e com os resultados obtidos com a execução foi realizada a etapa de análise dos resultados. Nesta etapa, foi possível identificar, de acordo com o cenário criado e os tempos definidos nos SLAs quais as transações que passaram e quais falharam no teste, além de comparar os tempos estimados em relação ao tempo executado por cada transação. Outro aspecto importante a ser considerado, é que uma vez criados os modelos, os tempos estimados que são informados, podem ser

ajustados de acordo com o cenário de desempenho que se deseja testar, tendo como base os históricos obtidos com as execuções anteriores. Também é importante ressaltar que os *scripts* de teste de desempenho gerados foram executados pelo LoadRunner. No entanto, a abordagem proposta foi desenvolvida com o objetivo de ser estendida para outras ferramentas de desempenho, como por exemplo, o VisualStudio. Da mesma forma que os produtos gerados pela PLeTs não compreendem um módulo de análise dos resultados que até então foram analisados manualmente, sendo essa uma das atividades de trabalhos futuros, permitindo uma integração maior com as tecnologias geradoras de carga de trabalho.

Diante da abordagem proposta, e na realização de um estudo de caso baseado em uma aplicação de gerenciamento de portfólios e projetos, foi possível demonstrar todas as etapas realizadas para a execução de teste de desempenho. Descrevendo desde a fase de modelagem UML até a definição dos SLAs e instrumentalização dos *scripts* de teste de desempenho gerados para a ferramenta LoadRunner. Além disso, outro aspecto significativo na realização do trabalho é que os dados obtidos com a execução dos testes são referentes a uma aplicação amplamente utilizada pela indústria, o que proporciona um cenário e um ambiente de teste muito similar a situações reais.

## 7. Agradecimentos

Avelino Zorzo, Flávio Oliveira, Maicon Bernardino e Priscila Guarienti são pesquisadores do Centro de Competência em Teste de Desempenho da PUCRS, uma parceria entre a Dell Computadores do Brasil Ltda e PUCRS. Estudo desenvolvido pelo Grupo de Pesquisa do PDTI 001/2014, financiado pela Dell Computadores do Brasil Ltda com recursos da Lei 8.248/91. Agradecemos também aos demais membros do grupo de pesquisa do CEPES, Elder de Macedo Rodrigues e Leandro Teodoro Costa pelas discussões sobre a PLeTs.

## Referências

Apfelbaum, L. and Doyle, J. (1997). Model Based Testing. In *Proceedings of the Software Quality Week Conference*, pages 296–300.

Blokdijk, G. and Menken, I. (2008). *Service Level Management Best Practice Handbook: Building, Running and Managing Effective Service Level Management SLAs - Ready to Use Supporting Documents Bringing ITIL Theory into Practice*. Emereo Pty Ltd, London, UK.

Clements, P. and Northrop, L. (2001). *Software Product Lines: Practices and Patterns*. Addison-Wesley Longman Publishing, 3rd edition.

Costa, L. T., Czekster, R. M., de Oliveira, F. M., Rodrigues, E. M., Silveira, M. B., and Zorzo, A. F. (2012). Generating Performance Test Scripts and Scenarios Based on Abstract Intermediate Models. In *Proceedings of the 24th International Conference on Software Engineering and Knowledge Engineering*, pages 112–117.

de Oliveira, F. M., Menna, R. d. S., Vieira, H. V., and Ruiz, D. D. A. (2007). Performance Testing from UML Models with Resource Descriptions. *I Brazilian Workshop on Systematic and Automated Software Testing*, pages 47–51.

El-Far, I. K. and Whittaker, J. A. (2001). *Model-based Software Testing*, pages 825–837. Wiley, New York, USA.

Geetha, D. E., Kumar, T. S., and Kanth, K. R. (2011). Framework for Hybrid Performance Prediction Process Model: Use Case Performance Engineering Approach. *SIGSOFT Software Engineering Notes*, 36(3):1–15.

Hewlett Packard – HP (2014). Software HP LoadRunner. Disponível em: https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp.

Kalita, M., Khanikar, S., and Bezboruah, T. (2011). Investigation on performance testing and evaluation of PReWebN: a java technique for implementing web application. *Institution of Engineering and Technology Software*, 5(5):434–444.

Luo, G., Petrenko, A., and v. Bochmann, G. (1995). Selecting test sequences for partially-specified nondeterministic finite state machines. In *7th IFIP WG 6.1 International Workshop on Protocol Test Systems*, pages 95–110.

Meier, J., Farre, C., Bansode, P., Barber, S., and Rea, D. (2007). *Performance testing guidance for web applications: patterns & practices*. Microsoft Press, Redmond, USA.

Rodrigues, E. M., Viccari, L. D., Zorzo, A. F., and de Souza Gimenes, I. M. (2010). PLeTs-Test Automation using Software Product Lines and Model Based Testing. In *Proceedings of the 22th International Conference on Software Engineering and Knowledge Engineering*, pages 483–488.

Silveira, M. B., Rodrigues, E. M., Zorzo, A. F., Costa, L. T., Vieira, H. V., and de Oliveira, F. M. (2011). Generation of Scripts for Performance Testing Based on UML Models. In *Proceedings of the 23rd International Conference on Software Engineering and Knowledge Engineering*, pages 258–263.

Smith, C. U. (2002). *Software Performance Engineering*. John Wiley & Sons, New York, USA.

Software Engineering Institute (SEI) (2014). Software Product Lines (SPL). Disponível em: http://www.sei.cmu.edu/productlines/.

Stilian, G. (1967). *PERT: un nuevo instrumento de planificación y control*. Deusto.

Vasar, M., Srirama, S. N., and Dumas, M. (2012). Framework for Monitoring and Testing Web Application Scalability on the Cloud. In *Proceedings of the WICSA/ECSA Companion Volume*, pages 53–60.

Weiss, C., Westermann, D., Heger, C., and Moser, M. (2013). Systematic Performance Evaluation Based on Tailored Benchmark Applications. In *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, pages 411–420.

Woodside, M., Franks, G., and Petriu, D. C. (2007). The Future of Software Performance Engineering. In *Proceedings of the Future of Software Engineering*, pages 171–187.