2. Terminologia

Neste capítulo são apresentados, de forma breve, detalhes sobre alguns dos assuntos abordados neste trabalho.

2.1.Descrição textual de casos de uso

Segundo Cockburn (2000), a descrição textual de casos de uso pode servir como meio de capturar o processo de negócio de empresas e organizações, documentar o comportamento de sistemas de *software* e especificar os requisitos de sistemas de *software*. Para permitir conter tais informações, esta descrição pode conter vários itens, dispostos num formulário, projetado para facilitar e uniformizar seu preenchimento. Diversos autores variam na escolha dos itens contidos neste tipo de formulário, adotando somente alguns itens ou ainda criando outros. Também é comum encontrar estilos de preenchimento diferentes (usando diferentes níveis de abstração), variando de bem detalhados a até bastante vagos.

De forma geral, podemos escolher certos itens que são úteis e podem ser necessários com frequência, como os propostos por Staa (2013a) e mostrados na Tabela 1, a seguir

Tabela 1 - Formulário exemplo para a descrição textual de casos de uso

Caso de uso	Nome (identificação) do caso de uso.
Resumo	Descrição resumida do objetivo principal.
Escopo	O que é abrangido pelo caso de uso.
Ator principal	Nome do ator que ativa o caso de uso.
Interessados	Nome (identificação) dos interessados, contendo opcionalmente a descrição de seus interesses ou objetivos.
Invariantes	Condições que devem ser satisfeitas antes e após efetuar o caso de uso. (Devem envolver somente dados, arquivos, recursos e estados manipulados pelo caso de uso).

Pré-condições	Condições que devem ser satisfeitas antes de efetuar o caso de uso. (Invariantes + Pré-condições devem envolver todos os dados, arquivos, recursos e estados necessários para poder realizar todo o caso de uso).
Acionamento	Condição ou evento que inicia o caso de uso, disparado pelo ator principal.
Fluxo principal	Sequência de ações a serem executadas no caso normal de funcionamento.
Fluxos alternativos	Sequência de ações disparadas por eventos previstos durante a execução de ações do fluxo principal, ou de fluxos alternativos já descritos. (Fluxos alternativos devem indicar onde iniciam e onde terminam).
Pós-condições	Condições que devem ser satisfeitas após efetuar o caso de uso. (Invariantes + Pós-condições devem envolver tudo o que resta ao final do fluxo normal. Pós-condições devem poder ser utilizadas como oráculos de teste).
Garantia mínima	Condições que devem ser satisfeitas sempre que o caso de uso não termine de forma normal.
Requisitos	Requisitos adicionais, tais como requisitos não funcionais e características de interface humano-computador (IHC).
Regras de negócio	Especificação das regras (assertivas) a serem satisfeitas nos passos.
Casos de uso correlatos	Relação dos casos de uso correlacionados ao presente caso de uso (ex.: o conjunto de características que implementam um propósito).
Artefatos correlatos	Relação de documentos, artigos, mensagens, fontes de informação, etc. que contém alguma informação relevante para o caso de uso.

Para que esta especificação se torne verificável, seus itens – como os passos do Fluxo Principal ou as Regras de Negócio – precisam ser descritos com mais formalidade (do que a descrição em linguagem natural e de forma livre) e de forma mais precisa, especificando detalhes sobre os elementos de interface envolvidos. Dessa forma, eles poderão ser usados para a construção de testes funcionais. Essa especificação é detalhada na seção 4.2 e exemplificada na seção 6.7.

2.2.Decomposição de casos de uso

Cada caso de uso deverá ser redigido de modo que corresponda exatamente a uma **única funcionalidade** (característica ou *feature*) de alto nível, do ponto de vista do usuário. Um caso de uso que se encontra num alto nível de abstração deverá ser decomposto em outros, e assim por diante, até que cada um implemente completamente apenas uma funcionalidade e seja verificável. Esta prática viabiliza o Desenvolvimento Guiado por Funcionalidades (*Feature Driven Development*, FDD), proposto por Coad *et al*. (1999). Nele, cada funcionalidade do software é construída na ordem de prioridade atribuída pelo cliente (investidor ou interessado) e tem sua construção realizada num menor intervalo de tempo (por ser um artefato menor), permitindo um melhor acompanhamento e resposta a mudanças.

2.3.Descrição de fluxos de um caso de uso

Os fluxos de um caso de uso devem ser escritos, preferencialmente, em um **vocabulário restrito e semiestruturado**, como o adotado por Días *et al.* (2004) para a língua inglesa ou o adotado por Staa (2013a) para a língua portuguesa. Adicionando esta formalidade no preenchimento do fluxo, reduz-se o número de variações na interpretação da descrição e facilita-se a geração dos testes.

A Tabela 2 apresenta um exemplo de fluxos de um caso de uso. O vocabulário usado nos passos de cada fluxo é limitado e visa descrever suscintamente a ação executada pelo sistema ou pelo ator. A seção 4.2 apresenta detalhes sobre a estrutura dos fluxos usados neste trabalho. A seção 6.7 mostra exemplos de sua descrição, para o software selecionado para a avaliação da ferramenta, incluindo regras de negócio e outros campos relevantes para a geração de casos de teste.

Tabela 2 - Exemplo de fluxos de um caso de uso

Caso de uso	Efetuar Login
Descrição	Restringe o acesso ao sistema, permitindo que somente usuários registrados se autentiquem.
Atores	Administrador, Usuário

Pré-condição	1- Usuário cadastrado.
Acionamento	Ao sistema ser acessado.
Fluxo Principal	1. Sistema apresenta a Tela de Login; 2. Ator fornece Login e Senha; 3. Ator escolhe a opção Entrar; 4. Sistema verifica o Login e a Senha; 5. Sistema fecha a Tela de Login; 6. Sistema concede acesso ao usuário e termina o caso de uso;
Pós-condição do Fluxo Principal	1. Acesso concedido;
Fluxo Alternativo 1: "Cancelar"	Ao passo 3 do Fluxo Principal: 1. Ator escolhe a opção Cancelar; 2. Sistema fecha a Tela de Login; 3. Sistema termina o caso de uso sem conceder acesso ao usuário;
Fluxo Alternativo 2: "Usuário ou senha incorretos"	Ao passo 5 do Fluxo Principal: 1. Sistema detecta que Login ou Senha estão incorretos; 2. Sistema exibe mensagem de erro, com a opção OK; 3. Ator escolhe a opção OK; 4. Volta ao passo 2 do Fluxo Principal;

2.4.Geração de cenários

Um cenário é uma instância de um caso de uso ou um "caminho" completo pelo caso de uso (HEUMANN, 2001). Seguir o fluxo principal, por exemplo, pode ser visto como um cenário. Seguir o fluxo principal mais um fluxo alternativo pode ser visto como outro cenário, e assim por diante. Cada combinação de fluxos, partindo do fluxo principal, pode ser vista como um cenário.

O processo de geração de cenários é detalhado na seção 4.4.

2.5.Combinação de cenários

Um caso de uso pode possuir dependências de *estados* gerados por outros casos de uso, que são informadas através das *pré-condições*. Essas pré-condições geralmente se referem a *pós-condições* geradas por outros casos de uso. Adicionalmente, quando um caso de uso invoca (em um de seus fluxos) outro caso de uso, este também passa a ser uma dependência.

Assim, todos os casos de uso do qual um caso de uso depende devem ter seus cenários gerados anteriormente a ele, para que sejam combinados na ordem de dependência. Não deve haver ciclos (dependência mútua) entre dois casos de uso e a geração de cenários deve ser impedida nesse caso.

A combinação de cenários é discutida em detalhes na seção 4.5.

2.6.Geração de casos de teste

Um caso de teste tem o objetivo de exercitar um caminho particular de um programa ou verificar sua conformidade com um requisito específico (HEUMANN, 2001). Os cenários podem, então, ser usados como ponto de partida para a geração dos casos de teste, já que determinam os possíveis caminhos a se percorrer.

A simulação dos caminhos percorridos pelo usuário mais a verificação das expectativas geradas pela passagem por cada um (ex.: sucesso, falha, etc.) será a base da verificação dos requisitos funcionais. Se o sistema se comportar como foi especificado, sendo possível simular corretamente a passagem pelos cenários e verificar propriedades a respeito da passagem realizada e dos valores utilizados nela, saberemos que ele está em conformidade com os requisitos.

Para verificar diferentes aspectos de um mesmo cenário, podem ser gerados diversos casos de teste para ele. Cada caso de teste poderá exercitar o caminho do cenário usando entradas diferentes, o que poderá gerar saídas diferentes. Logo, cada variação de saída pode ser exercitada usando entradas específicas. Neste aspecto, as regras de negócio são uma boa fonte de informação para saber quais valores devem ser produzidos como entrada. Este trabalho as usa como ponto de partida para a geração dos valores dos testes.

O processo de simulação dos caminhos percorridos corresponde a uma espécie de **máquina de estados**, onde entradas serão geradas visando obter uma saída que leve ao resultado esperado. Então, as saídas são verificadas em certos pontos na cadeia de processamento, visando verificar a corretude do sistema, dada sua especificação.

2.7.Máquina de estados generalizada

Uma máquina de estados finitos generalizada – referenciada neste trabalho apenas como ME – é um modelo computacional que consiste em um conjunto de *estados*, um conjunto de *estados iniciais*, um *alfabeto de entrada,* e uma *função de transição* que relaciona os símbolos de entrada e os estados atuais para o próximo estado (ATALLAH, 2000).

Uma ME pode ser vista como um *grafo dirigido*, onde cada *vértice* é um estado e cada *aresta* é uma condição ou ação. Cada estado pode conter código executável e ser decomposto em máquinas de estado ainda mais detalhadas (STAA, 2013b).

A máquina de estados usada por este trabalho se aproxima de uma **máquina de Turing**,[4] pois possui alguma forma de memória (bases de dados, variáveis-membro de objetos, etc.). Porém, sua estrutura base é a de um **autômato finito**,[5] acrescido de operações genéricas tanto nos estados quanto nas transições.

2.8.Forma de Backus-Naur (BNF)

O presente trabalho usa uma notação estendida de Backus-Naur (*Extended Backus-Naur Form*), baseada na notação de Bray *et al*. (2008), chamada neste trabalho simplesmente de BNF. Esta notação é usada para especificar a sintaxe esperada para alguns formatos com o qual a ferramenta construída trabalha, como os Casos de Teste Semânticos Valorados (CTSVO) ou o Relatório de Execução de Testes.

Cada regra na gramática define um símbolo na forma:

[4] Uma máquina de Turing é essencialmente uma máquina de estados finitos com a habilidade de ler suas entradas mais de uma vez e também de apagar ou substituir os valores de suas entradas; ela também tem uma memória auxiliar ilimitada (GERSTING, 1993)

[5] Um autômato finito é uma máquina de estados finitos limitada a reconhecer uma classe de linguagem específica, usualmente formal.

```
<símbolo> ::= expressão
```

A expressão no lado direito pode conter:

- Cadeias de caracteres (*strings*) literais, entre apóstrofos ou aspas;
- Outro símbolo;
- Tipo de dado admitido pelos valores do símbolo, podendo ser: `string`, `boolean`, `integer`, `double`, `date`, `time`, ou `datetime`.

Cada expressão pode ser combinada para formar outros padrões mais complexos, onde *A* e *B* representam expressões simples, como apresentado na Tabela 3:

Tabela 3 - Significado de algumas expressões usadas neste trabalho

Expressão	Significado
A?	Casa com **A** ou com nada. (**A** é opcional).
A B	Casa com **A** seguida de **B**. Este operador tem maior precedência que uma alternativa, de forma que **A B | C D** é equivalente a **(A B) | (C D)**.
A|B	Casa com **A** ou **B**.
A+	Casa com uma ou mais ocorrências de **A**. Este operador tem maior precedência que uma alternativa, de forma que **A+|B+** é equivalente a **(A+)|(B+)**.
A*	Casa com zero ou mais ocorrências de **A**. Este operador tem maior precedência que uma alternativa, de forma que **A*|B*** é equivalente a **(A*)|(B*)**.