UnB – Universidade de Brasilia
FGA – Faculdade do Gama
TPPE – Técnicas de Programação para Plataformas Emergentes

Trabalho Prático 1 – Desenvolvimento com TDD

1 Observações gerais sobre o trabalho:

- Esse trabalho deve ser desenvolvido em grupos de 4 alunos. Será permitido apenas um único grupo de 5 alunos para que todos os alunos da turma possam participar de algum grupo.
- O trabalho visa o desenvolvimento de uma aplicação utilizando as técnicas de *Test-driven Development – TDD*. Os detalhes da aplicação a ser desenvolvida estão descritos em seguida.
- As técnicas falsificação, duplicação e triangulação, bem como as *features* dos frameworks unitários necessariamente deverão ser utilizadas ao longo do desenvolvimento desse trabalho.
- Não há restrições de linguagens e frameworks de testes unitários a serem utilizados. Pelo contrário, o grupo deve se sentir motivado a escolher a linguagem e framework que seja de domínio pelos integrantes do grupo.

2 Cenário do trabalho – Aplicação para evolução de modelos UML:

A UML oferece diversas notações para representar diferentes características de software Orientados a Objetos. Dentre tais notações destaca-se a representação comportamental através dos diagramas UML de Atividade e de Sequência, sendo que cada um desses diagramas tem um determinado nível de abstração. Enquanto o Diagrama de Atividades é geralmente utilizado para representação mais alto nível das funcionalidades de um software ao descrever a ordem em que as etapas são realizadas por um programa, o Diagramas de Seqüência mostra as relações entre os componentes durante a realização de uma tarefa sendo tais realizações representadas pela troca de mensagens entre os componentes. Em resumo, pode-se assumir que cada uma das atividades representadas em um diagrama de atividades tem seu comportamento detalhado através de um Diagrama de Seqüência.

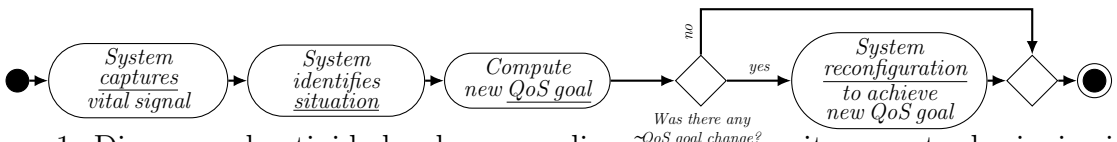

Figura 1: Diagrama de atividades de uma aplicação para monitoramento de sinais vitais.

As figuras 1 e 2 apresentam exemplos de diagramas de atividade e de sequencia, respectivamente, de um programa de monitoramento de sinais vitais de um indivíduo. O Diagrama de Atividades representado na Figura 1 mostra quais as etapas a serem seguidas para esse monitoramento, iniciando-se pela coleta de dados brutos dos sensores corporais e seguida pela análise etapa de análise de informações de saúde. Com base nessas informações a aplicação identifica se há a necessidade de ativaçao ou desativação de sensores (*QoSGoal*), sendo que tal rearranjo de sensores acontece na última etapa (*Reconfiguration*). Apesar de simples, o

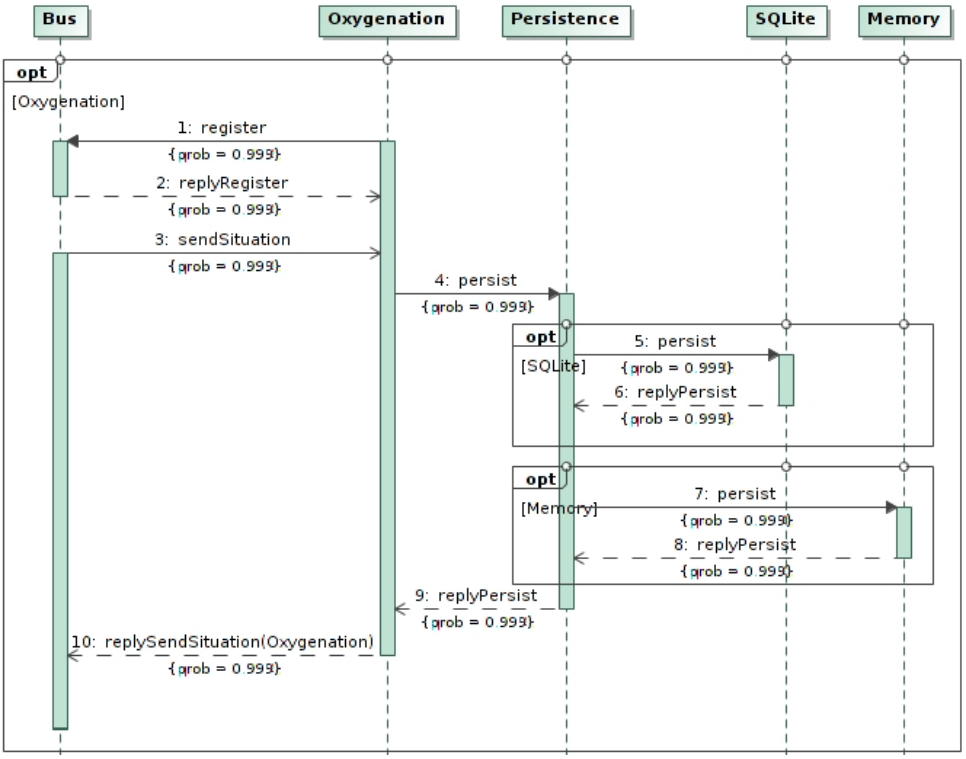

Figura 2: Diagrama de Sequência associado à atividade "*System identifies situation*".

Diagrama de Atividades apresenta todos os elementos de um Diagrama de Atividades a citar os nodos inicial, de atividade, de decisão, de fusão (*Merge*) e final.

A Figura 2 representa o Diagrama de Sequencias que detalha o comportamento da atividade "*System captures Sensor Information*". Nesse diagrama estão representados os componentes (de hardware e software) envolvidos nessa tarefa tais como sensores (*Sensor*), barramento (*Bus*), dentre outros. As setas contínuas com ponta fechada (setas cheias) indicam mensagens síncronas enviadas entre dois componentes, as setas contínuas abertas (setas vazadas) indicam mensagens assíncronas enviadas entre componentes e, por fim, as setas tracejadas e abertas indicam mensagens de retorno (respostas) a uma mensagem síncrona ou assíncrona. Independentemente do tipo, todas as mensagens possuem um valor de probabilidade associado (prob = <valor>, em que <valor> é um número real entre 0 e 1). O Diagrama de Sequencia mostra ainda fragmentos opcionais (representados por um quadrado rotulado com OPT <nome>, em que <nome> é o nome do fragmento opcional). O fragmento opcional possui uma condição de guarda [<cond>] representada entre colchetes [e] em que cond é definido por uma expressão lógica. Quando essa expressão lógica é avaliada como sendo verdadeira (*True*) o comportamento definido pelo fragmento é executado, caso contrário (*False*) não é executado.

3 Descrição da ferramenta a ser implementada

Em resumo a ferramenta deve ser capaz de criar modelos comportamentais UML em memória e persisti-los em arquivos XML. Para a criação dos diagramas de atividades a ferramenta deve ser capaz de:

- criar todos os tipos de elementos de um Diagrama de Atividades, a saber, nodo inicial, atividade, nodos de decisão, nodos de fusão (merge) e nodo final. Cada nodo, independentemente de seu tipo, possui um nome (*String*) associado.

- relacionar os elementos do diagrama de atividades através de associações que representem

o fluxo do Diagrama de Atividades. Por simplificação assume-se que uma atividade é seguida por apenas uma outra atividade. Nodos de decisão podem ter mais de um fluxo de saída. Nodos de fusão pode ter vários fluxos de entrada e apenas um fluxo de saída. Nodo inicial possui apenas um fluxo de saída, nodo final pode ter mais de um fluxo de entrada. Um Diagrama de Atividades tem apenas um nodo inicial e pelo menos um nodo final. No caso de alguma regra dessa for violada, uma exceção do tipo *ActivityDiagramRuleException* deve ser lançada e capturada. Cada associação entre elementos de um diagrama de atividades possui um nodo de origem (*source*), um nodo de destingo (*target*), um nome (*String*) e um valor de probabilidade *prob* associados.

- cada atividade tem que possuir apenas um Diagrama de Sequência associado, para representar o comportamento da atividade em detalhes. No caso de não haver um Diagrama de Sequência associado uma exceção do tipo *ActivityRepresentationException* deve ser lançada e capturada.

- ao persistir um Diagrama de Atividades e seus elementos em XML a estrutura a ser gerada deve ser igual a

```
1  <ActivityDiagram name=‘‘nome do diagrama’’>
2    <ActivityDiagramElements>
3      <StartNode name=‘‘nome do nodo inicial’’/>
4      <Activity name=‘‘nome da atividade’’/>
5      <DecisionNode name=‘‘nome do nodo de decisao’’/>
6      <MergeNode name=‘‘nome do nodo de fusao’’/>
7      <FinalNode name=‘‘nome do nodo final’’/>
8    </ActivityDiagramElements>
9    <ActivityDiagramTransitions>
10     <Transition name=‘‘nome da transicao’’ prob=‘‘valor de
     probabilidade’’/>
11     <Transition name=‘‘nome da transicao’’ prob=‘‘valor de
     probabilidade’’/>
12     . . .
13   </ActivityDiagramTransitions>
14 </ActivityDiagram>
15
```

Para os diagramas de sequência as seguintes regras deverão ser observadas:

- todos os elementos de um Diagrama de Sequência devem ser criados através da ferramenta. Os elementos de um Diagrama de Sequência são os componentes (*Lifelines*); as mensagens síncronas, assíncronas e de resposta (*Reply*) que e, por fim, os fragmentos opcionais;

- Cada diagrama de sequência possui um nome e uma condição de guarda;

- Cada mensagem, independentemente de seu tipo, possui um nome, um valor de probabilidade *prob* associado, uma *lifeline* de origem e uma *lifeline* de destino;

- Uma mensagem pode ter a mesma *lifeline* como origem e destino, de modo a representar uma auto-mensagem;

- Um fragmento opcional possui um Diagrama de Sequência como seu conteúdo, ou seja, o comportamento de um fragmento opcional é determinado por um Diagrama de Seqüência associado a ele.

- Uma mensagem não pode ser criada sem algum dos elementos que a compõem (nome, probabilidade, lifelines de origem e/ou destino). Caso essa situação ocorra uma exceção do tipo *MessageFormatException* deve ser lançada e capturada.

- Um Diagrama de Sequências não pode ter uma condição de guarda nula. Caso não seja atribuída uma expressão lógica para a condição de guarda de uma Diagrama de Sequência (*true/false*) uma exceção do tipo *EmptyGuardConditionException* deve ser lançada e capturada.

- Um fragmento opcional não pode ser criado sem ter um Diagrama de Sequência associado a ele. Caso isso aconteça uma exceção do tipo *EmptyOptionalFragment* deve ser lançada e capturada.

- ao persistir um Diagrama de Atividades e seus elementos em XML a estrutura a ser gerada deve ser igual a

```
<SequenceDiagrams>
  <Lifelines>
    <Lifeline name=‘‘nome da lifeline’’ />
    <Lifeline name=‘‘nome da lifeline’’ />
    <Lifeline name=‘‘nome da lifeline’’ />
    . . .
  </Lifelines>

  <Fragments>
    <Optional name=‘‘nome do fragmento’’ representedBy=‘‘nome do diagrama de sequencia’’/>
    <Optional name=‘‘nome do fragmento’’ representedBy=‘‘nome do diagrama de sequencia’’/>
    <Optional name=‘‘nome do fragmento’’ representedBy=‘‘nome do diagrama de sequencia’’/>
    . . .
  </Fragments>

  <SequenceDiagram name=‘‘nome do diagrama’’>
    <Message name=‘‘nome da mensagem’’ prob=‘‘valor da probabilidade’’ source=‘‘nome da lifeline’’ target=‘‘nome da lifeline’’ />
    <Message name=‘‘nome da mensagem’’ prob=‘‘valor da probabilidade’’ source=‘‘nome da lifeline’’ target=‘‘nome da lifeline’’ />
    <Fragment name=‘‘nome do fragmento’’/>
    <message name=‘‘nome da mensagem’’ prob=‘‘valor da probabilidade’’ source=‘‘nome da lifeline’’ target=‘‘nome da lifeline’’ />
    . . .
  </SequenceDiagram>

  <SequenceDiagram name=‘‘nome do diagrama’’>
    <Message name=‘‘nome da mensagem’’ prob=‘‘valor da probabilidade’’ source=‘‘nome da lifeline’’ target=‘‘nome da lifeline’’ />
    <Message name=‘‘nome da mensagem’’ prob=‘‘valor da probabilidade’’ source=‘‘nome da lifeline’’ target=‘‘nome da lifeline’’ />
    <Fragment name=‘‘nome do fragmento’’/>
    <message name=‘‘nome da mensagem’’ prob=‘‘valor da probabilidade’’ source=‘‘nome da lifeline’’ target=‘‘nome da lifeline’’ />
    . . .
  </SequenceDiagram>
</SequenceDiagrams>

```

Para efeitos de criação dos testes (e da ferramenta), os grupos podem utilizar os diagramas apresentados nas figuras 1 e 2 e recriá-los durante o desenvolvimento do trabalho.