1
Introdução

Realizar testes automatizados em sistemas que possuem componentes predominantemente reativos é usualmente uma tarefa nada trivial, pois as técnicas de automação de testes tradicionais freqüentemente não são suficientemente eficazes para este tipo de sistema. Tais técnicas dificultam a descrição dos comportamentos esperados de um módulo em produção, o que pode resultar em testes incompletos e que não especificam de forma suficientemente compreensível o que será construído. Tais sistemas são muitas vezes de missão crítica, e os altos custos decorrentes de suas falhas justificam a busca por técnicas de testes mais adequadas, principalmente que descubram defeitos nas etapas iniciais do processo de desenvolvimento, onde o custo para removê-los é menor [1].

Testes adequados ajudam a equipe de um projeto de *software* a ter mais confiança no código desenvolvido e também ajudam os gerentes do projeto a obterem medidas de progresso realistas e baseadas em informações sistemáticas [2], pois fornecem uma forma confiável de medir o progresso a partir de artefatos aceitáveis. A especificação de testes em uma linguagem mais adequada ao domínio também poderá apresentar um nível de abstração suficientemente elevado para servir como especificação do que está sendo desenvolvido. Para o desenvolvimento de sistemas reativos, de controle e com requisitos de tempo real, por exemplo, muitas vezes existem especificações com graus elevados de formalidade que são mais eficazes, usualmente através de máquinas de estado [3]. Portanto, utilizar uma linguagem baseada em máquinas de estado, mais próxima do domínio da aplicação, pode gerar ganhos significativos nesses domínios.

As técnicas de desenvolvimento dirigido por testes (DDT, ou no original *test driven development* - TDD) são amplamente utilizadas em diversas organizações [4], sendo especialmente importantes em metodologias ágeis de desenvolvimento de *software* como *extreme programming* [2]. Ao utilizá-las, a equipe do projeto deve escrever os testes dos artefatos em desenvolvimento antes de iniciar a codificação dos mesmos. Dessa maneira, estes passam a fornecer não

apenas uma forma mais prática e eficiente de realizar testes, mas também de refletir e entender melhor os problemas que se deseja resolver. Até certo ponto, isso inclui a idéia de assertivas: prega-se que as mesmas sejam escritas antes mesmo da escrita dos métodos, de forma que o programador seja forçado a pensar a respeito do que faz [5]. Este tipo de comportamento pode ser considerado como uma forma de redundância no processo de desenvolvimento, a redundância de raciocínio [6], onde o programador é obrigado a analisar diversas vezes uma mesma questão sob diferentes pontos de vista, como, por exemplo, pensar sobre o que será construído no momento de planejar os testes e no momento de escrever o código que será testado. Esta forma de redundância tem como vantagem aumentar a qualidade do código que será produzido, pois aumenta a chance de se encontrar defeitos que possam ter passado despercebidos em oportunidades anteriores, e quando realizada nas fases iniciais do processo de desenvolvimento pode ajudar a reduzir os custos do processo de desenvolvimento [1].

As metodologias ágeis têm como uma de suas premissas manter o foco do desenvolvimento em agregar valor para o cliente o mais rápido possível, e para tal adotam práticas como entregas freqüentes, ciclos de desenvolvimento curtos, desenvolvimento apenas do código estritamente necessário para cada funcionalidade, maior flexibilidade com relação a mudanças nos requisitos ao longo do projeto, entre outras [7]. Neste cenário, o desenvolvimento dirigido por testes ajuda a equipe a fornecer código de maior qualidade e medidas de progresso realistas, por possuir critérios sistemáticos para a aceitação dos artefatos.

Uma possível forma de se obter especificações de melhor legibilidade ao realizar o DDT e é a aplicação das técnicas de desenvolvimento dirigido por comportamentos (DDC ou *behaviour driven development* no original - BDD) [4,8,9]. DDC busca auxiliar nesse sentido através da criação de código de testes mais focado nos comportamentos do *software* em desenvolvimento e menos em sua estrutura interna. Esta técnica prevê também a utilização de uma linguagem mais abrangente, que possa ser utilizada em diversos níveis do processo de desenvolvimento, com o intuito de obter uma melhor utilização das especificações de testes como especificação do sistema.

Para a aplicação das técnicas de DDC no domínio de sistemas descrito anteriormente, é proposta uma linguagem de especificação de testes baseada em máquinas de estados. Essa linguagem foi utilizada para especificação de módulos

de um sistema real com as características apresentadas, e para isso foi criada uma ferramenta de automação de testes que possui a capacidade de configuração da linguagem de especificação dos testes. Através das experiências obtidas durante a aplicação da ferramenta, e conseqüentemente da linguagem proposta, espera-se que seja possível concluir sobre:

- a eficácia da utilização das especificações de testes como especificação do *software* que se deseja construir através da utilização de uma linguagem mais próxima do domínio da aplicação;
- a dificuldade, em uma situação prática, de utilização de uma linguagem diferente do convencional, o que muitas vezes exigirá a adaptação de processos e ferramentas, além da necessidade de realizar o treinamento da equipe que a utilizará;
- e a utilidade da ferramenta construída para se auxiliar equipes de desenvolvimento de *software* a alcançarem os objetivos anteriormente descritos, através da aplicação de linguagens específicas para seus domínios.

Este documento está estruturado da seguinte maneira: na seção 1 é feita uma breve introdução do trabalho; na seção 2 são apresentados e discutidos trabalhos relacionados e que serviram de base para esse trabalho; na seção 3 é descrita a ferramenta criada com um exemplo simples de sua utilização; na seção 4 é apresentada a linguagem proposta, baseada em máquinas de estados, com a aplicação da linguagem na ferramenta criada; na seção 5 são descritas as principais tecnologias utilizadas nesse trabalho; na seção 6 é descrita a aplicação da ferramenta a um projeto real; na seção 7 são discutidos os resultados obtidos; na seção 8 é realizada uma conclusão; na seção 9 são propostas seqüências a essa pesquisa; na seção 10 são listadas as referências.