# 7. Conclusões

A qualidade de um software é geralmente atestada pelo usuário, se ele atende adequadamente às suas necessidades. Como suas necessidades são geralmente descritas por uma especificação de requisitos, é fundamental que o software a atenda para obter sucesso.

Para verificar se um software atende aos seus requisitos, geralmente são realizados testes de aceitação, visando verificar se o funcionamento do software corresponde ao que foi acordado. Estes testes de aceitação são muitas vezes realizados através de testes funcionais, pela interface (gráfica) apresentada pelo software.

Sendo a especificação de requisitos definida através de casos de uso, como os casos de uso definem uma expectativa de interação com a interface, é possível automatizar o processo transformando estas expectativas em verificações. Através da simulação da interação do usuário com o software e a verificação se as ações do sistema – percebidas em sua interface – são condizentes com as esperadas, é possível atestar se a implementação construída é aderente à especificação.

Desta forma, esta dissertação de mestrado procurou definir e construir uma solução que permite realizar este trabalho, gerando estes testes, executando-os e analisando seus resultados. Para realizar essa tarefa, fez um considerável levantamento bibliográfico, analisando soluções existentes, definiu detalhes, processos e algoritmos utilizados, e construiu uma ferramenta funcional e capaz de obter ótima eficácia. A viabilidade técnica e a aplicação prática da solução foram demonstradas através de uma avaliação com um software construído por terceiros, sem o conhecimento da ferramenta aqui apresentada. Nele foi possível verificar que a ferramenta é capaz de descobrir falhas no software sob teste, além de atestar se a implementação construída condiz com a expectativa definida na especificação, que, como mencionado, pode ser uma forma de verificar a qualidade do software construído.

## 7.1.Contribuições do trabalho

O presente trabalho introduziu a especificação e uso de regras de negócio para viabilizar a geração dos valores e dos oráculos usados nos testes. Assim, regras sobre como o software deve se comportar podem ser definidas em mais detalhes, habilitando seu teste de forma automatizada.

Nas definições das regras de negócio também foi permitido especificar a origem dos dados por diferentes meios, sendo o uso de banco de dados mais significativo, por permitir o uso de massas de dados que podem estar (ou não) de acordo com a expectativa do software sob teste. Isto habilita o uso de dados com estrutura idêntica aos reais, o que permite simular condições de uso em produção.

A solução apresentada também: gera automaticamente os cenários de um caso de uso, permitindo recorrências (*loops*) entre fluxos; combina automaticamente cenários entre vários casos de uso, através de suas relações e dependências; gera casos de teste semânticos valorados e com oráculos, usando diversas estratégias, que possibilitam explorar valores limítrofes, o uso de valores aleatórios válidos e inválidos, a ausência de valores obrigatórios, etc. Estes processos, totalmente automáticos, simplificam a geração e execução de testes, cobrindo um grande número de casos práticos, que seriam extremamente trabalhosos se feitos manualmente, ou de difícil manutenção se feitos através de ferramentas de *capture-and-replay*.

Os casos de teste semânticos valorados e com oráculos gerados pela solução possuem um formato independente de ferramenta e linguagem de programação, o que permite adaptar a ferramenta para funcionar com diferentes linguagens e *frameworks*. Sua conversão para código-fonte é feita por meio de extensões da aplicação, que ainda executam os testes, pré-analisam e convertem os resultados, para que sejam lidos pela ferramenta num formato igualmente independente.

Assim, além de atestar a conformidade da implementação de um software em relação aos seus requisitos, a solução permite estabelecer um meio de aplicar *Test-Driven Development* no nível de especificação, de modo que a aplicação seja construída para passar nos testes gerados pela especificação construída. Nesse contexto, será de interesse do especificador produzir especificações mais completas e corretas, uma vez que compensará fazê-lo.

## 7.2.Resultados alcançados

Esta seção apresenta alguns resultados comparados às expectativas definidas na seção 1.2.

Apesar de não termos realizados testes que evidenciem todos os fatos, acreditamos que por causa do *processo* empregado em nossa ferramenta, alguns resultados são alcançados **por construção**.

Sobre o **aumento da eficácia dos testes** comparado a ferramentas de *capture-and-replay* e ferramentas que geram testes a partir de casos de uso, acreditamos que os testes gerados pela nossa ferramenta são mais eficazes, uma vez que essas ferramentas não criam oráculos, não verificam regras de negócio e não exploram valores limítrofes (dentre outros discutidos na seção 4.6.1). Assim, a eficácia dos testes gerados por elas *depende exclusivamente da habilidade de seus criadores*, não podendo ser garantida. Já nossa ferramenta apresenta um processo previsível e bem definido, podendo garantir a qualidade e a quantidade dos testes gerados.

Sobre a **diminuição dos fluxos alternativos**, esta pode ser facilmente notada, pelo fato de que não foi necessário criar nenhum fluxo alternativo que verificasse regras de negócio, o que não ocorre com outras ferramentas.

Sobre o **esforço de geração dos testes comparado ao uso de *capture-and-replay***, seria necessário medir os tempos de confecção dos casos de uso e geração dos testes de nossa ferramenta e comparar com o tempo de gravação das sessões de captura das outras ferramentas, para produzir os mesmos testes. É importante lembrar que, como nossa ferramenta gera diversos tipos de teste (discutidos na seção 4.6), seria necessário realizar várias sessões de gravação para obter os mesmos resultados, além da intervenção do testador para definir os oráculos de cada teste, uma vez que estas ferramentas não são capazes de produzí-los. Ademais, enquanto o processo de gravação de sessões serve apenas para gerar testes, a confecção dos casos de uso (de nossa ferramenta) serve para documentar os requisitos do software, sendo útil em sua construção, manutenção e teste.

Sobre o **esforço de geração de testes comparado à criação manual**, poderíamos comparar dois cenários. No primeiro, o testador precisa especificar os requisitos do software, com o uso de nossa ferramenta e sem ela – provavelmente em um documento de texto. Nesse caso, precisaríamos primeiro comparar o tempo

de confecção dos casos de uso e depois o tempo de criação dos testes. Quanto ao tempo de confecção dos casos de uso, acreditamos que o apoio fornecido por nossa ferramenta poderia levar à produção de casos de uso mais corretos e, possivelmente, em menor tempo. Ainda que seja mais demorado produzí-los com nossa ferramenta, é inegável a diferença de tempo de criação dos testes. Enquanto de forma manual seriam necessárias horas para produzir todos os testes, nossa ferramenta os gera em poucos segundos (de acordo com os resultados medidos em sua avaliação, disponíveis nas seções 6.7.6 e 6.8.6). No segundo cenário, em que o testador não precisa especificar os requisitos funcionais para criar os testes de forma manual, seria preciso comparar o tempo gasto na codificação dos casos de teste com o tempo necessário para especificar os casos de uso, uma vez que o tempo de geração dos testes com a ferramenta seria (proporcionalmente) desprezível. Independente de nossa crença – de que o tempo de confecção da especificação de casos de uso seja consideravelmente menor que o de codificação de todos os casos de teste produzidos a partir dela – é relevante observar que, ainda que o tempo de *codificação* fosse menor, nele estariam sendo produzidos apenas testes, deixando de lado a especificação dos requisitos do software.

Sobre a **possibilidade de atestar a conformidade de um software em relação aos seus requisitos**, esta pôde ser observada durante a realização de sua avaliação. Concluímos que a ferramenta é capaz de atestar essa conformidade, tendo encontrado diferenças entre a especificação de requisitos e o SST corretamente, em diversos casos (variando a especificação ou o SST).