2 Estado da Arte

Existem três conceitos importantes que serão abordados durante essa dissertação: geração automática de casos de teste, tabelas de decisão e geração automática de dados de teste. Foi realizada uma pesquisa para cada um desses tópicos. Na última seção, será apresentada a diferença do que já foi realizado para o trabalho que está sendo desenvolvido.

2.1.Geração automática de casos de teste

Os casos de teste podem ser extraídos de diversos documentos ou diagramas e de diferentes formas. Os artigos [HEUMANN, 2001] e [WOOD, 2007] utilizam a especificação de caso de uso para geração dos casos de teste. Os dois explicam a metodologia para geração de casos de teste usando os casos de uso, mas não têm como objetivo a geração automática.

O diagrama de caso de uso acrescido dos contratos (pré e pós condições) é utilizado com o artefato inicial para o processo descrito no artigo [NEBUT, 2006] para gerar um seqüenciamento entre os casos de uso. Como nem todas as informações importantes para a geração estão descritas no diagrama de casos de uso, esse artigo também utiliza o diagrama de seqüência. Após gerar o seqüenciamento entre os casos de uso, é gerado um modelo de simulação através dos valores descritos nos dois diagramas. O resultado final do artigo são todos os cenários de teste.

O artigo anterior foca nos relacionamentos entre os casos de uso, já o artigo [AHLOWALIA, 2002] foca em cada caso de uso e apresenta a técnica de análise de caminho que é realizada de forma manual. A análise de caminho (path analysis) constrói caminhos pelo caso de uso que podem percorrer o fluxo básico, alternativos e exceção e o relacionamento entre os casos de uso (extend e include). Ele apresenta cada caso de teste como um caminho e as diferentes formas de executar o mesmo caminho são descritos através dos dados de teste, diminuindo

PUC-Rio - Certificação Digital Nº 0812585/CA

assim a quantidade de casos de teste. O processo é dividido em quatro etapas: desenhar uma maquina de estados de cada caso de uso, determinar todos os possíveis caminhos, analisar e ranquear os caminhos, decidir quais caminhos serão usados para os testes. As vantagens dessa técnica são: melhor cobertura dos testes, pois inclui tanto os testes de sucesso quanto os de falha e diminui a quantidade de casos de teste. Por casos de teste de sucesso entende-se casos de teste que realizam operações que não ocorrem erros na funcionalidade, e os casos de teste de falha são exatamente o contrário, são os casos de teste que ocasionam erros. Uma maquina de estado pode ser facilmente transformada em uma tabela de decisão, logo todos os benefícios descritos por essa técnica também são aplicáveis ao processo desenvolvido.

O artigo [Fröhlich, 2000] utiliza a especificação dos casos de uso e gera o diagrama de estado. A especificação do caso de uso é descrito de uma forma pré-definida pelo autor. O artigo é dividido em duas partes: a transformação da especificação dos casos de uso para o diagrama de estado e a transformação do diagrama de estado para os testes suítes.

Outra forma de gerar os casos de teste é utilizando o processamento de linguagem natural. O artigo [BODDU, 2004] constrói uma ferramenta que recupera os requisitos, classifica de acordo com a complexidade, refina, checa consistência e gera casos de teste.

Em outro artigo [FARCHI, 2002], foi desenvolvido um gerador de teste baseado em uma máquina de estados finita chamada GOTCHA-TCBeans. Esse artigo tem como foco os testes baseados na especificação, ou seja, testes de caixa preta. Um grafo projetado é derivado da máquina de estados que representa a especificação pelo designer de teste. Durante essa derivação é obtido uma cobertura estrutural do grafo que possibilita que o projetista do teste especifique uma suíte de teste completa ao invés de um teste específico. O teste abstrato é transformado nas suítes de teste utilizando uma tabela de tradução escrita pelo testador.

Um conceito muito interessante que é utilizado pelo artigo [HSIA, 1997] é o "*Behaviour-Based Development*". A técnica de análise de cenário é baseada em cenários desenvolvidos pelo usuário que descrevem como ele utiliza o sistema. O cenário é constituído de passos entre o sistema e o usuário. O [HSIA,1994] já criou um método formal para identificar, gerar e verificar um conjunto de

PUC-Rio - Certificação Digital Nº 0812585/CA

cenários. Após a geração dos cenários com os usuários, cada cenário é transformado em uma árvore. Cada arvore é convertida em uma gramática regular que irá auxiliar para sua transformação em maquina de estado (etapa de *scenario generation*). A partir desse ponto, já existem diversas técnicas que transformam a máquina de estado em casos de teste.

O artigo [MOGYORODI, 2003] apresenta um processo de teste baseado em requisitos (RBT) que é composto de duas fases: revisões de ambigüidade e gráficos de causa-efeito. A ambigüidade é uma análise técnica para identificar ambigüidades em requisitos funcionais para melhorar a qualidade desses requisitos. Gráfico de causa-efeito [MYERS,2004] é uma técnica que obtém o número mínimo de casos teste para cobrir 100 por cento dos requisitos funcionais. Os gráficos de causa-efeito podem ser facilmente transformados em uma tabela de decisão, logo o processo utilizado neste trabalho também contém a propriedade descrita anteriormente. As etapas de teste que esse processo irá cobrir são: definição dos critérios de completeza dos testes, desenvolvimento dos casos de teste (esses testes são definidos por quatro características: o estado inicial do sistema antes do teste, os dados, os inputs e os resultados esperados) e verificação da cobertura dos testes. Esse último artigo é o mais interessante para a dissertação já que ele tem como objetivo transformar os requisitos em um grafo de causa e efeito que poderia ser transformado em uma tabela de decisão conforme mostrado em [MYERS, 2004]. O livro de Myers apresenta no capítulo 4 que a utilização de gráfico de causa-efeito pode ser importante para explorar combinações de dados de input. Isso tornaria o processo de teste completo, começando da especificação até a execução dos testes.

PUC-Rio - Certificação Digital Nº 0812585/CA

2.2.Tabelas de Decisão

Tabela de decisão é uma ferramenta que foi muito utilizada na década de 70-80 e voltou a ser usada depois dos anos 90.

Na década de 70 tabelas de decisão eram muito utilizadas para geração de código fonte. O artigo [SHUMACHER, 1976] apresenta uma metodologia de geração de código ótimo através de tabelas e árvores de decisão. Outros artigos como [HUMBY, 1973] e [POOCH, 1974] geravam código, mas utilizando um método de decomposição diferentemente do primeiro artigo. Nessa época, tabelas

de decisão eram utilizadas para evitar problemas de compilação, por causa da garantia de não ambigüidade e o artigo [RAJARAMAN, 1970] explora a geração de código por esse motivo.

Tabelas de decisão têm sido utilizadas em diversas áreas do desenvolvimento de software. Uma das áreas encontradas foi o de sistemas baseados em conhecimento. O benefício que essa ferramenta traz é evitar problemas como falta de validação, verificação e metodologias de *design*. A ferramenta PROLOGA é usada como exemplo para mostrar que existe uma melhora na especificação e implementação desses sistemas [VANTHIENEN, 1993].

Outra área que utiliza essa ferramenta é a área de regras de negócio. As regras de negócio têm um ciclo de vida, dividido por: orientação, especificação, implementação, aplicação e controle. As tabelas de decisão são utilizadas como ferramenta para a etapa de especificação de regras de negócio [REUSCH, 2007]. Outro artigo [PAWLAK, 1997] também utiliza a tabela de decisão com regras de negócio, mas nesse caso as tabelas são utilizadas como auxílio para sistemas de suporte a decisão.

Existe um artigo [HOOVER, 1995] que apresenta uma ferramenta para fazer a edição de tabelas de decisão. A ferramenta desenvolvida foi a *Tablewise* para edição de tabelas de decisão. Essa ferramenta tem funções como: interpretação de algumas informações contidas na tabela como, por exemplo, para quais *inputs* é indicado mais de um curso de ações, geração de código Ada para implementação da tabela e geração de documentação texto. Foram encontrados mais dois sistemas que têm como principal objetivo fazer a edição de tabelas de decisão e validar as suas propriedades intrínsecas.: *LoginGem* [LOGICGEM] e *Prologa* [PROLOGA].

Já na área de teste, que é o foco dessa pesquisa, existem poucos experimentos com a utilização de tabelas de decisão como ferramenta auxiliar. O artigo [FERRIDAY, 2007] mostra que tabela de decisão é um artefato muito importante para o desenvolvimento dos testes por já possuir os casos de testes estruturados dentro dele. O artigo mostra como a tabela de decisão deve ser construída e como checar as propriedades de completeza, redundância e inconsistência. As alternativas citadas na seção anterior, podem ser quase todas transformadas em uma tabela de decisão, como por exemplo maquina de estados e gráfico de maquina de estados. O livro de Myers [MYERS,2004] apresenta a

PUC-Rio - Certificação Digital Nº 0812585/CA

tabela de decisão como uma ferramenta auxiliar à representação do gráfico de causa e efeito que é utilizado para a área de teste. As colunas da tabela de decisão gerada são correspondentes à estrutura de um caso de teste. Entretanto, não explora como os testes podem ser gerados automaticamente a partir das tabelas. Existem ainda algumas considerações visando facilitar o uso de tabelas de decisão [JORGENSEN, 1995]. Elas seriam utilizadas para identificar os casos de testes, as condições são interpretadas como inputs e as ações como outputs. A inclusão de algumas características nas tabelas de decisão a tornaria mais útil para testadores. Uma dessas inclusões é sempre ter a ação "Impossível", já que a combinação de algumas condições nunca poderá acontecer. A segunda inclusão seria utilizar as entradas "Não importa" representado por "-", para quando aquela condição não tem importância para uma determinada regra.

PUC-Rio - Certificação Digital Nº 0812585/CA

2.3.Geração automática de dados de teste

Existem três formas de geração automática de testes: geração baseada em programa (*program-based generation*), geração baseada em interface gráfica (*Gui-based generation*) e geração baseada em sintaxe (*syntax-based generation*).

Para a geração de teste baseado em programa, são necessárias três grandes estruturas, o analisador de programa, a escolha do caminho e o gerador de dados de teste. O gerador receberá o caminho que deve ser testado e nesse caminho para cada transição de blocos de código (representa com uma "entidade") existe uma condição. A partir de cada uma das transições passadas pelo caminho é construído um predicado do caminho, que é um conjunto de equações ou inequações. Para gerar o sistema de equações, é possível fazer geração simbólica ou geração atual, onde na simbólica é feita a execução de expressões de variáveis do programa ao longo do caminho escolhido e na atual é realmente executado o código. A solução desse sistema pode ser realizada de várias formas, como por exemplo: programação linear, *simulated anealing*, algoritmos genéticos, diversas heurísticas [TRACEY, 1998] e algoritmos evolucionários [WEGENER, 2002] .

O artigo [MCMINN, 2004] apresenta detalhadamente como heurísticas podem auxiliar na geração automática de dados de teste, para teste de caixa branca, caixa preta e de caixa cinza. Para os testes de caixa branca, é apresentada geração de dados de teste através de estrutura estática utilizando duas técnicas:

execução simbólica e restrição de domínio. A execução simbólica já foi apresenta anteriormente e a restrição de domínio foi inicialmente desenvolvida no artigo [DEMILLO, 1991] sobre teste baseado em *constraints*. A execução simbólica usa *constraints* para as variáveis de input e a redução de domínio pretende apresentar uma solução para as *constraints*. Para fazer essa redução é necessário o domínio de cada uma das variáveis de input, que podem ser recuperado através do tipo da variável ou da especificação.

Para construção do gerador existem em três métodos: randômico, orientado a caminho e orientado a objetivo. A geração randômica é a mais simples das técnicas, ela irá gerar valores para todas as transições. Na geração orientada a objetivo, são gerados valores de input apenas para as transições que pertenceram ao algum predicado de caminho. Existe um sistema que implementa essa geração chamado TESTGEN [FERGUSON, 1996]. A última forma, orientado a caminho, é a forma mais forte já que ela produz inputs apenas para um caminho especifico. Além do sistema TESTGEN, o sistema CASEGEN [RAMAMOORTHY, 1976] utiliza essa técnica. No artigo [FERGUSON, 1996] é utilizada a abordagem de "encadeamento" (*chaining*) que usa o conceito de seqüência de eventos como uma forma de decidir o caminho que será executado e é uma versão mais avançada do que a orientada a objetivo. O processo dessa técnica inicia colocando a nó inicial e o nó final. Depois para cada dois eventos adjacentes, existe uma condição e serão procurados dados para que seja seguido o caminho especificado inicialmente.

O artigo [EDVARDSSON, 1999] apresenta alguns problemas encontrados nessa geração de dados de teste que incluem: módulos, *constraints* e oráculos. Esse último problema será abordado por essa dissertação.

No caso de teste de caixa preta [MCMINN, 2004], é verificado metaheurísticas que podem auxiliar o comportamento lógico do sistema. São apresentadas duas técnicas extração de dados de teste a partir de uma especificação Z [JONES, 1995] e a partir de especificação formal [TRACEY, 1998]. Na especificação Z, é mostrando um esquema como o apresentado abaixo das condições e resultados apresentados.

$$\begin{aligned} Triangle ::= \quad & (Triangle0 \wedge EquiTri) \vee (Triangle0 \wedge IsosTri) \vee \\ & (Triangle0 \wedge ScalTri) \vee (Triangle0 \wedge RightTri) \vee \\ & NumError \vee TriangleError \end{aligned}$$

Figura 2 – Especificação Z

PUC-Rio - Certificação Digital Nº 0812585/CA

Cada uma das disjunções é considerada um caminho pelo sistema. Algoritmos genéticos são utilizados para gerar dados de teste para satisfazer esse caminho. Na especificação formal são redigidas as pré e pós condições e, a partir delas, é definido se a implementação está de acordo com a especificação, ou seja, se está correto.

2.4. Diferenças da Dissertação

Por esse levantamento foi percebido que a tabela de decisão é uma ferramenta que poderia trazer diversos benefícios na área de teste. No entanto, não existe nenhum trabalho que utilize essa tabela como ferramenta auxiliar em um processo de geração automática de teste. A utilização dessa ferramenta auxiliar vai garantir que a quantidade de casos de teste gerados será mínimo para testar todas as condições descritas na tabela de decisão.

Para a complementação do processo, é necessário realizar a geração dos dados para cada caso de teste. É importante frisar que os casos de testes gerados representam tanto os casos de teste de falha quando os casos de teste de sucesso. Esses artigos da geração de dados apresentam a geração automática, só que o local de onde são retiradas as informações para a geração de dados fica um pouco obscuras. A forma como serão gerados os dados é utilizando a idéia de domínio apresentada no artigo [MCMINN, 2004], só que o local de onde são extraídas as informações de domínio é a tabela de decisão.

PUC-Rio - Certificação Digital Nº 0812585/CA