UNIVERSIDADE FEDERAL DE ALAGOAS
INSTITUTO DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

POLIANA VIEIRA BELO DA SILVA

Casos de teste baseados em modelos formais para aumentar a qualidade de software

Maceió
2017

POLIANA VIEIRA BELO DA SILVA

Casos de teste baseados em modelos formais do sistema para aumentar a qualidade de software

Trabalho de Conclusão de Curso submetido ao Curso de Bacharelado em Ciência da Computação do Instituto de Computação da Universidade Federal de Alagoas como requisito parcial para a obtenção do Grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Leandro Dias da Silva

Maceió
2017

POLIANA VIEIRA BELO DA SILVA

Casos de teste baseados em modelos formais do sistema para aumentar a qualidade de software

Este Trabalho de Conclusão de Curso (TCC) foi julgado adequado para obtenção do Título de Bacharel em Ciência da Computação e aprovado em sua forma final pelo Instituto de Computação da Universidade Federal de Alagoas.
Maceió, 18 de dezembro de 2017.

_____

Prof. Fábio José Coutinho da Silva, Dr.
Coordenador do Curso de Ciência da Computação

Banca Examinadora:

_____

Prof. Leandro Dias da Silva, Dr.
Orientador

_____

Prof. Patrick Henrique da Silva Brito, Dr.
UFAL campus Arapiraca

_____

Prof. Evandro de Barros Costa, Dr.
UFAL campus A. C. Simões

DEDICATÓRIA

Dedico este Trabalho de Conclusão de Curso (TCC) aos trabalhadores e pesquisadores da área de Ciência da Computação que possuem interesse no tema de qualidade de software.

Maceió, 18 de dezembro de 2017.

AGRADECIMENTOS

Agradeço, primeiramente, a minha mãe por me fortalecer e me guiar em todos os momentos.

Agradeço a meu pai e irmãs por todo o carinho e motivação que deram durante a minha vida.

Agradeço ao professor Leandro Dias da Silva, por ser um grande incentivador deste trabalho e da minha vida acadêmica como um todo.

Agradeço a todos aqueles que, de alguma forma, contribuíram para a elevação do meu conhecimento na academia e vida profissional.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A qualidade de software muitas vezes torna-se um desafio no processo de desenvolvimento. A depender do nível de exigência colocado pelos interessados e da complexidade do software em questão, torna-se necessário utilizar técnicas ou ferramentas que colaborem para uma melhor qualidade do produto. De forma a contribuir com este aspecto, é apresentada a proposta de elaboração de casos de teste de um sistema a partir de seu modelo formal construído com o redes de Petri Coloridas.

Partindo de uma especificação de caso de uso do sistema, foi definido o seu modelo formal, possibilitando a simulação do sistema antes mesmo dele ser construído. Este modelo possibilitou investigar e extrair informações sobre o funcionamento e as propriedades do software em desenvolvimento. Algumas das informações utilizadas na modelagem do sistema e, em seguida, na sua execução foram adicionadas nas definição dos casos de teste.

A modelagem formal com redes de Petri Coloridas foi útil para a especificação dos casos de testes. Informações sobre os fluxos de execução ficaram mais evidentes no modelo. Além disso, os dados necessários para a sua simulação, como a introdução de dados de entrada e regras de validação foram aproveitados para os casos de teste elaborados em seguida.

O benefícios obtidos colaboraram para a atividade de testes do processo, e para a qualidade do sistema como um todo. Os testes se tornaram mais seguros, e também foram validados os requisitos do sistema com a execução do modelo formal. As práticas utilizadas serviram como forma de verificar e validar o funcionamento do software em desenvolvimento.

ABSTRACT

Software quality often becomes a challenge in the development process. Depending on the level of demand placed by stakeholders and the complexity of the software in question, it is necessary to use techniques or tools that contribute to a better quality of the product. In order to contribute to this aspect, we propose the elaboration of test cases of a system from its Colored Petri nets formal model. Starting from a system use case specification, we construct its formal model, allowing simulation of the system before it is even built. Using the formal model it is possible to investigate and extract information about the operation and properties of the software being developed. Some of the information used in the system modeling and simulation were added in the definition of the test cases. Formal modeling with Colored Petri nets was useful for specifying the test cases. Information about the execution flows became more evident in the model. In addition, the data required for their simulation, such as the input data and validation rules, were used for the test cases elaborated. The benefits obtained contributed to the testing activity and to the quality of the system as a whole. With these activities the tests became more secure, and also the system requirements were validated with the execution of the model. The results served as a way to verify and validate the functioning of the software under development.

1. Introdução

É possível identificar nos dias atuais um número cada vez maior de sistemas que fazem parte do cotidiano da sociedade. Os sistemas computacionais estão incorporados na rotina da população, atendendo a uma demanda considerável de atividades. O rápido crescimento do uso de software e, consequentemente, das indústrias deste setor, impulsionou o avanço dos processos de desenvolvimento da Engenharia de Software.

À medida que a utilização de sistemas se tornou mais comum, aumentou também a exigência por qualidade nos produtos. Tais aspectos incentivaram a constante evolução dos processos de desenvolvimento de software, fazendo com que técnicas e atividades fundamentais, utilizadas na criação de sistemas, fossem frequentemente aprimoradas para atender a esta demanda. Existe um grande esforço e dedicação dos profissionais da área para padronizar a criação de software, e, ao mesmo tempo, oferecer práticas flexíveis, que possibilitem responder com qualidade a este tipo de produção quase sempre diverso e dinâmico.

A criação de um sistema computacional pode ser realizada de diferentes formas, a depender da organização responsável e do produto de software em questão. No geral, o desenvolvimento é composto por uma série de atividades relacionadas, que visam garantir que os objetivos dos usuários com aquele produto sejam alcançados com qualidade.

O aspecto da qualidade é um dos grandes desafios encontrados na produção de sistemas. De acordo com a norma internacional ISO/IEC 25010 [5], a qualidade de software é "A totalidade de características de um produto de software que lhe confere a capacidade de satisfazer necessidades explícitas e implícitas". Assim, para garantir um bom produto, é necessário atender aos aspectos especificados pelo usuário e também aos aspectos que, apesar de nem sempre apresentados, são necessários para utilizar o sistema. Para atender a essa demanda, existem práticas e técnicas, comumente recomendadas, que podem ser aplicadas durante o processo de desenvolvimento de software, ajudando a prover um sistema que satisfaça às necessidades de seus futuros usuários.

De acordo com Sommerville [9], as atividades que compõem o processo de desenvolvimento de software são: a Engenharia de Requisitos, Projeto e Implementação, Validação e Evolução de software. A Engenharia de Requisitos, uma das fases centrais consideradas neste trabalho, é usada para descrever e especificar o sistema em desenvolvimento, e, também, seus objetivos e restrições. É possível dizer que especificar um sistema é uma atividade crítica dentro do processo

de desenvolvimento de software, pois, além de ser realizada nas fases iniciais do processo, é utilizada como base para as demais fases do desenvolvimento de software. As ações realizadas na Engenharia de Requisitos servem para a descoberta e padronização dos requisitos do sistema, considerando inclusive a necessidade de validar se os requisitos foram descritos da forma correta. O resultado final desta etapa pode ser apresentado a partir de documentos ou modelos do sistema, e representa a concordância sobre o que será desenvolvido entre as diferentes partes envolvidas no processo.

A técnica de modelagem, na Engenharia de Requisitos, é recomendada para facilitar o entendimento e descrição dos requisitos de um sistema. Modelos do sistema auxiliam a comunicação entre as partes envolvidas, além de também serem úteis nas demais fases do processo de software. Quanto mais fiel for a modelagem criada, expressando corretamente as necessidades do usuário, mais seguras se tornam outras atividades do desenvolvimento, e mais satisfeito fica o cliente.

A atividade de teste é um procedimento de qualidade que faz parte da Verificação e Validação do sistema, ajudando a encontrar e prevenir erros no software desenvolvido. Esta prática depende de etapas anteriores para sua realização. Para definir o que será testado, torna-se necessária a especificação resultante da Engenharia de Requisitos, mesmo que existam outros fatores influenciáveis, como o nível de criticidade do software e exigência do cliente, por exemplo. Algumas atividades que compõem o procedimento de testes são: planejamento de testes, definição dos casos de testes, execução e avaliação dos resultados [9]. Os casos de teste são resultado direto da especificação do sistema, e são fundamentais para a atividade de teste. Com eles são definidas as entradas e ações realizadas pelo testador, assim como as respostas que se espera do sistema.

A cobertura do funcionamento do sistema, com uma quantidade eficiente de casos de testes, apesar de não garantir a ausência de erros, agrega mais segurança ao software. O grande número de cenários e entradas possíveis na execução de um software acaba dificultando este trabalho. É possível, com os modelos elaborados na Engenharia de Requisitos, auxiliar a definição dos casos de testes. Em conjunto, a modelagem do sistema e os testes de software podem elevar a qualidade do sistema desenvolvido [8].

Neste trabalho é apresentada a técnica de modelagem formal Redes de Petri Coloridas [6] para modelar os requisitos de um projeto, e definir os seus casos de teste. Esta técnica é indicada para representar sistemas dinâmicos, e que apresentam propriedades como concorrência, controle ou compartilhamento. Com Redes de Petri Coloridas é possível tornar mais prática a representação de sistemas. Ao utilizá-la é possível simular sistemas dinâmicos, descrevendo os diferentes estados alcançados e os eventos que fazem o sistema mudar de estado. Desta forma, torna-se

mais prático analisar situações do comportamento do software, e depurar com mais rigor suas propriedades que são aspectos centrais para a atividade de teste.

1.1 Metodologia

O objetivo principal neste trabalho é ressaltar os benefícios trazidos ao se utilizar uma técnica de modelagem formal para a definição dos casos de testes de um sistema. Para isto, é apresentado inicialmente um estudo sobre o processo software com foco nas atividades de verificação e validação, onde são definidos os casos testes de um sistema. Em seguida são apresentadas as bases teóricas das redes de Petri Coloridas, descrevendo as vantagens em se utilizar esta técnica formal de modelagem.

Apresentadas as bases teóricas, é definido o caso de estudo que descreve um sistema especificado para este trabalho. O sistema possui uma especificação contendo a listagem de funcionalidades documentadas através de documentação de caso de uso. Esta especificação, por sua vez, é detalhada através da elaboração de um protótipo executável utilizando Redes de Petri Coloridas. As informações necessárias para simular o protótipo auxiliam, em seguida, a atividade especificação dos casos de teste. Os casos de teste são apresentados em tabela que serve como auxílio para a execução dos testes do sistema.

Ao final, são discutidos os benefícios das práticas colocadas, e como é possível desenvolver um software de melhor qualidade. São também apresentadas as limitações deste trabalho assim como trabalhos futuros que podem colaborar para melhores resultados.

1.2 Trabalhos relacionados

Com o objetivo de colaborar com o aspecto da qualidade de software, alguns trabalhos têm se concentrado na atividade de modelagem e na sua utilização para a definir os casos de testes de um sistema, semelhante ao objetivo deste trabalho apresentado na seção 1.3.

Souza H. e Luiz L. [10] propõem uma metodologia para elaborar casos de testes. A ideia central consiste na extração de casos de teste a partir de modelos de processos de negócio. A partir do modelo de processos, os fluxos do sistema são investigados, e, em seguida, a documentação de testes é elaborada. Os casos de testes são identificados com base nas heurísticas usadas na descrição de requisitos, e novas heurísticas inspiradas em métodos de geração de casos de testes a partir de casos de uso e diagrama de estados. Um dos problemas dessa abordagem é a ausência de um maior grau de detalhamento sobre o funcionamento do sistema em desenvolvimento. O modelo de processo, além de não permitir que sejam colocados os tipos de dados usados, é um

modelo estático, sem a possibilidade de validar os requisitos em seu estágio inicial, isto é, antes do sistema ficar pronto.

Liliane Vale [12] propõe a especificação de testes funcionais usando mapeamento de redes de Petri para objetos para softwares orientados a objetos. Para a obtenção do modelo de teste funcional, o trabalho utilizou os conceitos de processos de Workflow e redes de Petri, originando, posteriormente, as Workflow-Nets com redes de Petri ordinárias para, por último, obter as Workflow-Nets a objetos que realizam o mapeamento de redes de Petri a Objetos. Este trabalho exige um conhecimento de um número considerável de técnicas, ferramentas e linguagens, o que torna a atividade menos prática, podendo aumentar significativamente o volume de trabalho na atividade de testes. Além disso, ao final, é obtido um modelo de teste para testes funcionais. O modelo, apesar de fornecer uma visualização dos fluxos de teste, não é tão prático para guiar quanto uma documentação com descrição dos casos de teste, que funcionam diretamente como um roteiro para a equipe que irá testar.

No trabalho de Neto Baumgartner [1] são apresentados dois métodos para geração de casos de teste. Um dos métodos conduz os testes a partir de modelo desenvolvido em rede de Petri hierárquica. O segundo modelo ajuda a validar os resultados após os testes serem concluídos utilizando um modelo em OWL-S. Neste trabalho são apresentados resultados com alto nível de cobertura de cenários do sistema, possibilitando identificar testes incompleto de uma funcionalidade. O estudo proporciona também a definição de casos de teste como resultado do trabalho, que é um dos objetivos aqui colocados. Porém, a realização das atividades a que se propõe, utiliza um método que exige uma elevada curva de aprendizado, e também um conhecimento aprofundado do sistema em desenvolvimento para construção de modelos em Rede de Petri e Ontologia.

Tabela 1 - Comparação de trabalhos relacionados

	Souza H. e Luiz L.	Neto Baumgartner	Liliane Vale
Uso de formalismo		X	X
Uso prático	X	X	
Geração de casos de teste	X	X	
Necessidade de baixo nível de aprendizado	X		X

Fonte: Elaborada pela autora.

1.3 Objetivos

O presente trabalho busca por facilitar a execução da atividade de teste de forma acessível, apresentando dados que podem ser inseridos em modelo executável. Além disso, é proposto que os testes possam ser elaborados inicialmente, sem profundo conhecimento do sistema, pois este ainda pode se encontrar em suas fases iniciais, onde a formalização e documentação de seu funcionamento encontram-se em construção. Muitas vezes um sistema em desenvolvimento pode ter alterações, e é preciso de práticas flexíveis que não demandem alto nível de trabalho para que novos testes sejam elaborados.

1.4 Organização do documento

A apresentação a seguir encontra-se dividida em 6 tópicos que fundamentam o trabalho e, em seguida, apresentam o estudo realizado. No capítulo 2 é apresentada a fundamentação teórica sobre o processo de desenvolvimento de software, com foco nas atividades de validação de requisitos e teste de software, e também sobre a prática de modelagem utilizando redes de Petri Coloridas. No capítulo 3 é apresentado o desenvolvimento do trabalho, ilustrando a validação utilizando redes de Petri Coloridas para modelagem e elaboração dos testes do sistema. Por fim, são apresentadas as conclusões sobre o resultado no capítulo 4.

2. Revisão de literatura

Nesta seção são apresentados os fundamentos e os conceitos usados no desenvolvimento deste trabalho. Os tópicos abordados aqui estão divididos em duas seções principais: o Processo de desenvolvimento de Software e Redes de Petri Coloridas. Dentro da primeira parte são abordados os estudos sobre Engenharia de Requisitos, Validação e Verificação, Teste de Software, Modelagem e Simulação, e na segunda parte é abordado o conceito de Redes de Petri Coloridas e processo de Validação usando esta técnica.

2.1 O processo de desenvolvimento de software

O processo de criação de um sistema envolve uma série de tarefas relacionadas, que visam garantir que os objetivos de seus futuros usuários sejam alcançados. Não existem processos ideais para produzir software, e as organizações podem seguir modelos de processo já definidos na engenharia, ou desenvolver seu próprio processo da maneira que achar mais adequada.

Tanto nos processos personalizados ou nos modelos mais genéricos, existem quatro atividades fundamentais para a engenharia de software que, de alguma forma, fazem parte de todos os processos: Engenharia de Requisitos, Projeto e Implementação, Validação e Evolução de software [9]. Essas etapas de desenvolvimento possuem diferentes objetivos, e têm como aspecto central garantir que as expectativas dos usuários sejam alcançadas. Satisfazer o cliente, e garantir que suas necessidades com o produto elaborado sejam alcançadas, significa ter um software com qualidade.

Modelos de processo de software são introduzidos na engenharia como forma de simplificar as atividades do desenvolvimento. Os padrões propostos possuem formato genérico e podem ser adaptados para diferentes projetos, sem possuir uma reprodução definitiva das atividades de desenvolvimento [9]. Como exemplo é possível citar o modelo em cascata, cujo uma de suas variações, o Modelo V, é utilizada neste trabalho. O Modelo V faz uma relação direta entre as atividades de análise e projeto com as atividades de qualidade e teste. Em suas etapas, os testes são planejados de forma paralela as atividades de desenvolvimento correspondentes, e servem como maneira de verificar o projeto de software.

Tendo como objetivo principal realizar o debate sobre qualidade de software e apresentar possíveis maneiras de alcançar este critério, o trabalho aqui apresentado irá centrar em duas dessas atividades: Engenharia de Requisitos e Validação. Esses estágios, aprofundados nos

próximos tópicos, são considerados de grande importância para garantir que os critérios definidos pelo usuário sejam alcançados. A etapa de Engenharia de Requisitos, inclusive, pode ser considerada crítica, pois, além de se encontrar na fase inicial do processo, é utilizada nas demais etapas.

2.1.1 A Engenharia de Requisitos

A Engenharia de Requisitos é utilizada para descrever o propósito do sistema em desenvolvimento e suas restrições. Nessa etapa, se especifica o software de acordo com a finalidade que os clientes têm com o produto a ser desenvolvido. "O processo de descobrir, analisar, documentar e verificar esses serviços e restrições é chamado engenharia de requisitos" [9].

As principais atividades da etapa podem ser divididas em Estudo de Viabilidade (verifica se o sistema é útil), Elicitação e Análise (descobrindo requisitos), Especificação (convertendo-os em alguma forma padrão) e Validação de Requisitos (verificar se eles realmente definem o sistema). Tais práticas devem produzir documentos e um conjunto de modelos gráficos que serão apresentados tanto para os clientes quanto para os desenvolvedores do sistema. Ao final desta fase, todos os levantamentos, depois de validados, devem ser organizados e documentados, servindo como uma declaração oficial do o que os desenvolvedores do sistema devem implementar.

A atividade de descobrir requisitos, durante a Elicitação e Análise, depende da comunicação entre os clientes e os engenheiros de software para obter informações sobre o sistema, e envolve diferentes tipos de interessados. A interação entre as partes envolvidas, para especificar as necessidades do cliente, pode ser facilitada por meio de práticas que aproximam o diálogo, principalmente quando existem distintos níveis familiaridade com os termos técnicos e aspectos do domínio do software.

Existem diferentes formas de descrever a especificação de um sistema, tais como documentos de caso de uso, documentos de descrição de requisitos e diagramas de modelagem. O material produzido neste levantamento serve como base as demais atividades do processo. Os documentos de Caso de uso são o tipo de narrativa mais usado atualmente. Neles encontram-se definidas as interações do sistema com seu usuário ou com outro sistema para atingir um determinado objetivo. Cada documento, geralmente, é descrito através de cenários que expressam as diferentes situações possíveis de se alcançar durante uma interação. Os cenários detalham as ações e interações, esperadas ou não, envolvidas neste processo. "Cenários e casos de uso são técnicas eficazes para elicitar requisitos dos stakeholders que vão interagir diretamente com o sistema" [9].

Outra maneira de especificar um sistema é através a técnica de modelagem. Com modelos do sistema, também é possível apresentar suas características de funcionamento e obter uma visão geral dos seus fluxos de execução. Esta representação pode possuir diferentes perspectivas, e ser elaborada com ferramentas específicas. As Redes de Petri Coloridas são oferecidas como técnica de modelagem formal que permite, além da abstração do modelo geral do sistema, a execução dos fluxos de funcionamento do software antes da sua construção. A possibilidade de executar o modelo funciona também como forma de validar a sua especificação.

Com os requisitos descobertos e documentados, é de grande importância verificar se eles realmente definem o que o usuário quer ou se possuem algum erro. Um analista ou desenvolvedor pode ter interpretações equivocadas durante a especificação, assim como o cliente interessado pode não expressar da melhor maneira seus objetivos com o software em criação. Erros são comuns, e a validação e verificação dos requisitos, quando realizadas, podem evitar o aparecimento de problemas no sistema posteriormente, gerando grandes transtornos para os clientes e desenvolvedores.

É possível verificar o sistema de diferentes maneiras até chegar na sua validação como um todo. Além da revisão do que foi documentado na Engenharia de Requisitos e da prototipação de modelos do sistema, a geração dos casos de testes são técnicas que também são úteis para validar os requisitos levantados [9].

2.1.1.1 Validação de requisitos

A validação de requisitos faz parte do processo de Engenharia de Requisitos do sistema, e tem a finalidade de verificar se os requisitos levantados definem o sistema que o cliente realmente quer. Esta prática possui a finalidade de garantir que o produto satisfaça suas especificações e ofereça a funcionalidade esperada pelos usuários. A verificação, permite conferir se o atendimento aos requisitos funcionais e não funcionais do sistema. A validação possui uma maior abrangência, e ajuda a garantir que o software faça aquilo que o cliente tem como expectativa [9].

A Validação e Verificação são fundamentais para melhorar o nível de confiança do software, e, desta forma, a sua qualidade. Tais atividades podem ser feitas de forma estática, através de inspeções e revisões de documentos e modelos durante o processo de software, ou de forma dinâmica com a realização de testes. Diferentes atividades de verificação podem ser realizadas durante o processo de validação dos requisitos.

Dentre essas atividades de verificação a prototipação e geração de casos de teste são elencadas como abordagens efetivas como auxílio para esta finalidade. Através da prototipação define-se um modelo executável do sistema, permitindo uma demonstração mais qualitativa do que será projetado para os futuros usuários. Além disso, é importante verificar se os requisitos definidos são testáveis. Se existe dificuldade em projetar os testes a partir do que foi levantado, isso é um indicativo de que o requisito pode está errado.

2.1.1.2 Testes de Software

O Teste de Software é uma das atividades do processo de Verificação e Validação. Testar o software é executar o sistema para encontrar problemas ou determinar se ele funciona de acordo com suas especificações, e é uma forma efetiva de preparar o sistema para produção. Com esta atividade é possível investigar e prevenir o acontecimento de falhas. Os testes têm como principais objetivos provar que o produto de software atende a seus requisitos, e descobrir erros no seu comportamento [3].

O processo de teste básico é constituído das atividades: planejamento e controle, análise e modelagem, implementação e execução, avaliação dos critérios de saída e relatórios, e atividades de encerramento de teste [11]. A atividade de planejamento e controle consiste em definir os objetivos dos testes, como pretende-se alcança-los, e comparar o que foi progredido com o que foi planejado. Em análise e modelagem, os objetivos definidos são analisados e transformados em modelos e condições concretas para o teste. Na implementação e execução são especificados os testes no ambiente escolhido e, em seguida, estes são executados. Na avaliação do critério de saída, o resultado da execução é avaliado e comparado com os objetivos do processo. E, por fim, na atividade de encerramento, os resultados são coletados e podem ser utilizados como referência para outras experiências.

Durante o processo de teste, é possível aplicá-lo em diferentes níveis durante o desenvolvimento do sistema. De acordo com programa de estudos produzido pela ISTQB, organização de certificação em teste de software, os possíveis níveis em que os testes podem ser aplicados são: teste de componente, teste de integração, teste de sistema, e teste de aceitação [11]. Em resumo, os testes podem ser aplicados desde a estrutura mais interna, no desenvolvimento, até a interface e seus componentes. A decisão do que testar depende do nível de criticidade do software em produção, da fase de desenvolvimento em que o produto se encontra e das exigências feitas pelos clientes.

Os objetivos de teste definidos podem possuir diferentes alvos, e, desta forma, o tipo usado vai depender das metas colocadas. Os testes funcionais e os não-funcionais são duas importantes divisões possíveis de se realizar num produto de software. Os não-funcionais, também chamados

de testes caixa-branca, se aplicam nos aspectos mais internos do sistema, envolvem código e possibilitam que partes específicas da estrutura sejam analisadas. Já os testes funcionais, chamados de caixa-preta, são aplicados independente da estrutura externa do sistema, e são baseados nos requisitos funcionais da aplicação, como forma de verificar se os requisitos foram satisfeitos.

Na atividade de análise e modelagem, citada anteriormente, são definidas as condições de teste (o que testar), e esta condição será verificada através de casos de teste que serão executados posteriormente. Os casos de teste são centrais para a execução dos testes, e são usados para orientar este procedimento. Os casos de teste declaram o que está sendo testado, e apresentam a definição dos valores entrada, as pré-condições necessárias, as saídas que se espera do sistema e as pós-condições de execução [3].

Um importante aspecto da atividade de teste é medir a sua abrangência, chamada de cobertura. Esta medida está baseada na cobertura dos requisitos levantados e também pela cobertura do código implementado. Ao medir a abrangência dos testes, é possível obter uma melhor visão de quais cenários possíveis da funcionalidade em questão estão sendo verificados. Esta medida é importante e pode ser utilizada na definição dos casos de teste e servir inclusive para critérios de riscos e impactos.

A elaboração dos casos de teste é feita a partir de uma documentação base da especificação do sistema em questão. Materiais como os casos de uso e modelos do sistema oferecem informações que servem como auxílio na construção dos casos de teste. Dentre os artefatos utilizados para derivar os casos de teste, é comum o uso dos modelos UML e diagramas de objetos. Assim como é necessário especificar o sistema em desenvolvimento, é preciso especificar os testes que serão nele aplicados.

Especificar os casos de teste nem sempre é uma tarefa simples, principalmente se os requisitos não estiverem bem definidos e possuírem ambiguidades. É possível ter um grande número de cenários com desvios de execução, e assim se tornar necessário aplicar uma quantidade considerável de entradas garantindo uma melhor cobertura. Desta forma, é importante ter informações que possibilitem uma atividade efetiva, e ajudem na tomada de decisões.

A utilização de modelos executáveis do sistema podem facilitar a definição dos casos de testes. O protótipo executável do sistema, além de ser uma forma de validar os requisitos elaborados na Engenharia de Requisitos, tornando-a mais segura, possibilita uma visualização efetiva do funcionamento do sistema. Nos tópicos seguintes apresento a técnica de modelagem com Redes de Petri Coloridas, propondo-a como opção efetiva para esta atividade de testes.

2.2 Modelagem com redes de Petri Coloridas

Nesta seção apresento a técnica de modelagem redes de Petri e sua extensão redes de Petri Coloridas. bem como suas diferenças e indicações de uso. As redes de Petri Coloridas são centrais neste trabalho, e, por isso, abordadas com maior nível de detalhamento [6]. Ao final é colocado como é possível obter benefício utilizando esta técnica como forma de verificação do sistema, e na elaboração dos testes de um sistema.

2.2.1 Redes de Petri

O uso de métodos formais no desenvolvimento de software contribui para a confiabilidade e robustez do projeto. Sua implementação baseia o processo de desenvolvimento em princípios matemáticos, e pode ser utilizada para auxiliar todas as etapas como, por exemplo, na especificação formal para definir os requisitos. Redes de Petri é uma técnica formal de especificação com forte base matemática indicada para a modelagem de sistemas. Esta técnica permite representar graficamente a estrutura de sistemas com características como paralelismo, concorrência, que são assíncronos e não determinísticos [6]. Alguns exemplos adequados para sua utilização são protocolos de comunicação, processos de negócios e sistemas embarcados/embutidos.

Com esta técnica de modelagem pode-se representar o funcionamento de um sistema distribuído como um grafo direcionado. Sua estrutura consiste basicamente em nós de lugares, nós de transição, e também arcos direcionados que conectam lugares com transições. O inventor desta teoria, Carl Adam Petri [6], definiu uma rede de Petri como uma quíntupla RP = (P, T, A, W, Mo). Na quíntupla, os elementos de (P,T,A) formam um grafo bipartido dirigido que representa um sistema. Os vértices podem ser transições (T), que representam as ações possíveis do sistema realizar, ou podem ser lugares (P), que são as variações de estados. Os arcos do grafo (A) podem partir de um lugar para uma transição ou de uma transição para um lugar. Na primeira ocasião o lugar constitui uma pré-condição para t, e é chamado lugar de entrada para a referida transição, já na segunda ocasião, tem-se um lugar de saída para a transição que representa um estado atingido após a t ser realizada. O W da quíntupla é chamado peso, e representa uma função que associa a cada arco de A um número natural.

Os lugares modelados são marcados com uma ou mais fichas (tokens), e a quantidade e posição inicial das fichas nos lugares indicam a marcação inicial (Mo) do sistema. A partir da marcação inicial da rede, é possível fazer o sistema mudar de estado de acordo com as fichas (tokens) que se encontram nos lugares e também pelos pesos indicados nos arcos. O peso de uma função W é

que determina quantas fichas serão retiradas do lugar de entrada vinculado ao arco de W, ou quantas fichas serão adicionadas no lugar de saída do arco. Para habilitar uma transição é necessário que todos os lugares de entrada tenham uma marcação com peso igual ou maior do que o peso indicado na transição. Quando esta condição é atendida, a transição pode ser acionada e alterar o estado da rede, retirando fichas dos lugares de entrada e adicionados nos lugares de saída.

As transições nas redes de Petri permitem o consumo de tokens de múltiplos lugares, diferente do que acontece nos sistemas mais tradicionais, que processam somente um único fluxo de token numa entrada. Esta ação de consumir os tokens de entrada é denominada disparo, e pode representar alguma tarefa de processamento no sistema. O disparo de uma transição consiste basicamente em retirar tokens da entrada, realizar um processamento, e adicionar os tokens no lugar de saída. A representação das mudanças de estado permitem simular o funcionamento do sistema e investigar o seu comportamento em diferentes momentos.

A partir das redes de Petri, diferentes extensões deste formalismo foram introduzidos com o objetivo de suprir algumas restrições encontradas na versão clássica, e também possibilitar outras formas de simular sistemas. Dentre as extensões pode-se citar as redes de Petri Coloridas, que apresento na seção abaixo.

2.2.2 Redes de Petri Coloridas

Redes de Petri Coloridas é uma extensão do formalismo redes de Petri, que combina as habilidades deste com uma linguagem de alto nível [6]. As redes de petri possibilitam descrever sincronização de processos correntes, através de suas primitivas, enquanto linguagens de programação oferecem os meios para descrever tipos de dados e manipulação de dados.

Com redes de Petri coloridas tem-se uma melhor visualização de controle de dados em processos de negócios, devido à possibilidade de representação de dados que antes não era possível com as redes de Petri clássicas. É uma linguagem para simulação de sistemas de eventos discretos e orientada a estados e ações. Assim como nas redes de Petri, sua modelagem é feita descrevendo os diferentes estados alcançados pelo sistema e os eventos que fazem o sistema mudar de estado.

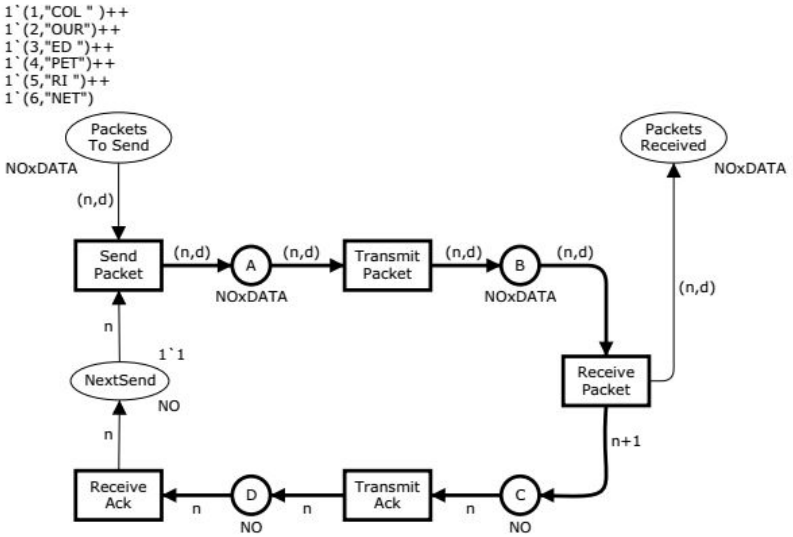
Usando a linguagem CPN-ML, é possível definir tipos de dados, chamados de token colours, e realizar a manipulação desses dados durante a execução do protótipo do sistema. Os modelos criados podem ser executados de forma interativa, passo a passo ou automaticamente. Diferente das redes de Petri clássicas, nas RCPs as expressões são inscrições textuais na linguagem de programação CPN-ML, e são formadas por variáveis, constantes, operadores e funções. Se as variáveis das expressões de arco de uma transição correspondem com valores válidos nos

multiconjuntos de token colours presentes no correspondente lugar de entrada, tornando possível uma atribuição, significa que a transição está habilitada. Quando a transição está habilitada, com as variáveis nas expressões ligadas a tipos correspondentes, isso significa que as expressões nos arcos podem ser calculadas.

A estrutura, assim como nas redes de Petri, é um grafo dirigido com dois tipos de vértices (lugares e transições). Os lugares são representados graficamente por círculos (ou por elipses) e as transições por retângulos. Com este formalismo pode-se armazenar em cada lugar diferentes de tipos/cores, e também representar valores associados a tipos de dados mais complexos. Declarações compreendem a especificação dos conjuntos de cores e declarações de variáveis. As inscrições variam de acordo com o componente da rede. Os lugares possuem três tipos de inscrições: nomes, conjunto de cores e expressão de inicialização (marcação inicial). As transições têm dois tipos de inscrições: nomes e expressões guarda, e os arcos apenas um tipo de inscrição dado pela expressão. Como formas para distinguir as inscrições, nomes são escritos com letras normais, cores em itálico, expressões de inicialização sublinhadas e as expressões guarda são colocadas entre colchetes [6].

A ilustração 1, retirada de [6], ajuda a exemplificar este estudo, e possibilitar uma melhor visualização da estrutura de uma rede de Petri Colorida. O modelo utilizado é formado por sete lugares, cinco transições, doze arcos dirigidos entre lugares e transições, e inscrições textuais feitas com a linguagem de programação CPN ML em alguns destes componentes. Juntas estas partes formam a estrutura da rede de Petri Colorida.

Figura 1 - Exemplo de rede de Petri Colorida



Fonte: [6]

Outra vantagem obtida a modelagem com redes de Petri Coloridas é poder construir um modelo geral a partir de modelos menores, submodelos, que são construídos separados, mas se comunicam através de interfaces sem perder a consistência. Esta forma diferenciada é feita por causa do conceito de hierarquia introduzido nas redes de Petri Coloridas. Esta hierarquia, além de reduzir o tamanho do modelo, permite representar diferentes processos ou recursos em uma mesma sub-rede.

Usar redes de Petri Coloridas no projeto de um sistema ajuda a simular os diferentes cenários possíveis, e a conhecer melhor suas ações e comportamentos. Uma simulação mais detalhada, com nível de abstração apropriado, é fundamental para investigar situações com maior criticidade na execução do software. Além disso, ajuda também a evitar resultados inesperados, que muitas vezes não são identificados na fase de Engenharia de Requisitos. Esta modelagem formal pode complementar o processo de software, e auxiliar a tomada de decisões.

Diante disto, a elaboração dos casos de testes do sistema pode ser realizada com o auxílio deste formalismo, que ajuda a obter maiores informações sobre o sistema e agrega segurança na atividade de teste. Mais adiante, apresento como é possível melhorar a qualidade e segurança de um sistema em desenvolvimento a partir da combinação deste método de modelagem com os testes de software.

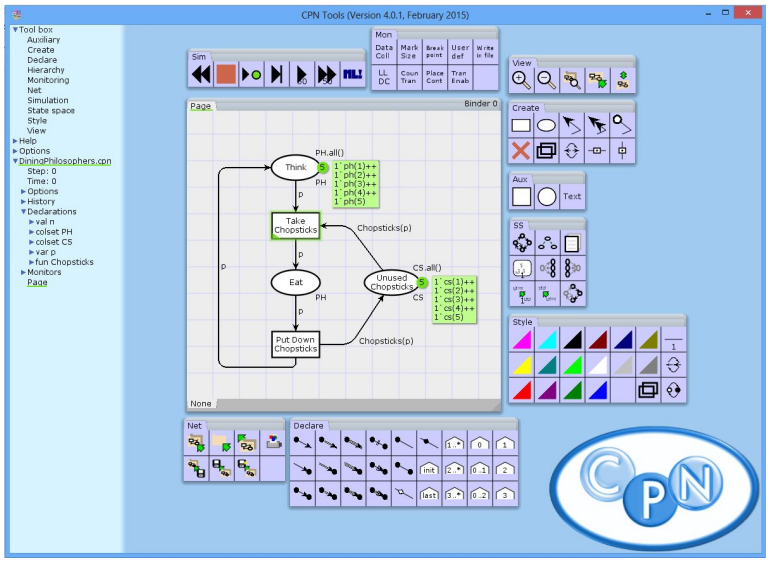
2.2.3 CPN Tools

A ferramenta de suporte para redes de Petri de alto nível utilizada neste trabalho é a CPN Tools. "CPN Tools é uma ferramenta para edição, simulação, espaço estado análise, e análise de performance de modelos CPN" [6]. Com esta ferramenta o usuário trabalha com a representação gráfica do modelo do sistema, e com ela pode realizar tanto a verificação de sintaxe como a de tipo usados na construção. Ao mesmo tempo que se faz a edição é possível gerar código e verificar a sintaxe, visto que são atividades realizadas em paralelo e de forma incremental. O código de simulação, resultado da geração de código, contém as funções que inferem os eventos habilitados em um dado estado, e que computam o estado resultante após a ocorrência de um evento.

É possível realizar a simulação com CPN Tools de forma interativa ou automática. A simulação interativa depende do controle do usuário que determina cada passo realizado através da seleção de eventos habilitados em um dado estado. Após o usuário selecionar um evento, o sistema apresenta o estado resultante como efeito do passo executado. Já na simulação automática o modelo é executado sem interações com o usuário, ele apenas define quantos passos serão executados, e também critérios de parada ou breakpoints. A partir destas informações, o sistema executa automaticamente as ações habilitadas no estado apresentado. Na captura de tela 2,

apresento a estrutura da CPN Tools, carregando modelo de exemplo *DiningPhilosophers.cpn* disponibilizado pela ferramenta para os seus usuários. Além de carregar o modelo de exemplo fornecido, exibo na tela as paletas que podem ser usadas em todos os projetos criados a depender do nível de complexidade e necessidades do sistema em construção.

Figura 2 - Ferramenta CPN Tools



Fonte: Elaborada pela autora.

CPN Tools possui um pacote de visualização que suporta diversos tipos de diagramas e gráficos. Ao utilizar esta funcionalidade, pode-se observar as características dinâmicas do sistema modelado, e também tornar o modelo mais compreensível para os interessados. Com uma melhor compreensão sobre o funcionamento do sistema, é possível verificar os requisitos documentados e também ter um mapeamento mais claro na elaboração dos testes que serão realizados.

Neste trabalho, a CPN Tools será usada para construir um modelo de sistema estudado e simular seu funcionamento. Em seguida, será demonstrado como utilizar esta técnica pode ser útil nas atividades de verificação e validação, contribuindo para o aspecto de qualidade no processo de software como um todo.

3. Desenvolvimento

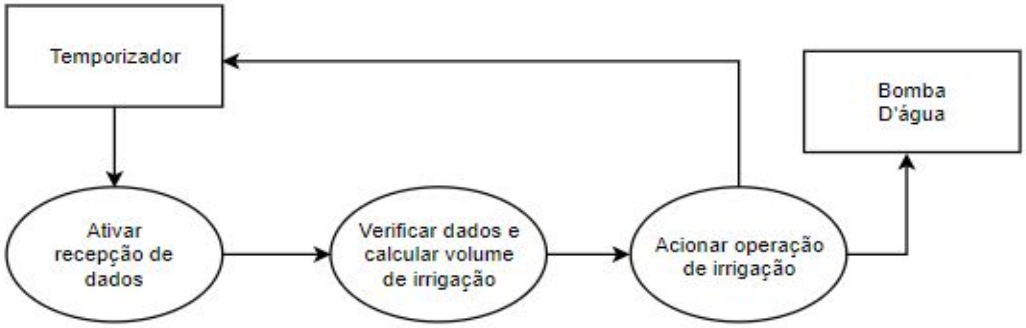
Apresentadas as atividades de testes da verificação e validação de software, e as bases teóricas das redes de Petri Coloridas, é descrito nesta seção como melhorar o aspecto de qualidade de um sistema utilizando estas técnicas durante o desenvolvimento.

Para averiguar as propostas colocadas, foi elaborada a especificação de um sistema como caso de estudo que possibilitasse aprofundar este trabalho. A ideia consiste em verificar e validar a especificação deste sistema com as técnicas propostas, a partir dos modelos formais elaborados com redes de Petri Coloridas. Para isso, o trabalho considera a geração dos casos testes a partir do protótipo construído. Por fim, os resultados são utilizados para ressaltar as vantagens obtidas com estas práticas para a qualidade do software como um todo.

Como caso de estudo deste trabalho, foi utilizado o sistema Água Viva em sua fase de Engenharia de Requisitos. Água Viva possui características convenientes para o estudo aqui levantado, e foi definido para fins exclusivamente acadêmicos como parte deste trabalho de conclusão de curso. Água Viva é um sistema de controle de irrigação construído com o propósito de reduzir a quantidade de água desperdiçada no procedimento de irrigação na agricultura. Seu comportamento depende dos valores de ph da terra recolhidos através de sensores inseridos na plantação. Os sensores monitoram as condições atuais e influenciam a forma de irrigação do sistema. Estes indicativos são utilizados para calcular a quantidade de água utilizada na irrigação, e o intervalo de tempo necessário para uma próxima execução. A coleta de dados dos sensores é ativada através de um temporizador, que captura informações em diferentes momentos.

Uma representação inicial do caso de estudo pode ser observada na Figura 3, que apresenta o comportamento geral do sistema. O funcionamento concentra-se em três etapas principais: recebimento dos dados dos sensores, cálculo dos valores a serem utilizados na irrigação e a irrigação, com possível ajuste posterior do temporizador. O sistema tem sua descrição apresentada através de documento de caso de uso, e seus detalhes são colocados no modelo feito utilizando redes de Petri Coloridas.

Figura 3 - Fluxo de funcionamento do sistema Água Viva



Fonte: Elaborada pela autora.

Água Viva foi escolhido por possuir requisitos não-funcionais críticos, em especial, confiabilidade, e por apresentar características que o enquadram enquanto concorrente e assíncrono, o que o torna adequado para a modelagem com redes de Petri Coloridas. Além disso, é importante que possua um procedimento de teste bem definido, ajudando a prevenir problemas com o sistema já em produção, desperdiçando água ou deixando de executar a irrigação. Abaixo é apresentado documento de caso de uso do sistema Água Viva.

3.1 Apresentação do Caso de Uso

É apresentado a seguir a especificação do sistema Água Viva através de sua documentação de caso de uso. O caso de uso, como colocado anteriormente, consiste numa narrativa do sistema, e é uma das principais bases para outras atividades do processo de desenvolvimento de software. No documento, além do fluxo de funcionamento principal do caso de estudo, foram adicionados os eventos que podem vir a alterar o fluxo de execução e as saídas esperadas do sistema nestas situações.

A execução do sistema é iniciada por um ator do tipo temporizador, que é responsável por acionar o sistema em intervalos de tempo pré-definidos, e finaliza com o acionamento da bomba de água. O fluxo principal consiste basicamente de três etapas: receber os dados coletados pelos sensores, realizar os cálculos do volume de água a partir dos dados recebidos, e, por fim, acionar a irrigação com a bomba de água. O intervalo de tempo para o início de cada ciclo é pré-definido, porém ele pode ser ajustado caso, após os cálculos realizados, verifique-se que a irrigação deve ser adiada. Além disso, outras alterações de fluxo podem ocorrer quando o sistema encontrar inconsistência nos dados recebidos pelos sensores ou nos resultados do cálculo do volume de irrigação.

No caso de uso abaixo os fluxos são apresentados com passos que representam as interações realizadas, e os subfluxos, fluxos alternativos e exceções, quando possíveis de acontecer, são indicadas no final da descrição dos passos. Além disso, as regras de execução, quando necessárias, são apresentadas no formato de requisitos especiais no decorrer dos passos do caso de uso.

Especificação de Caso de Uso:

Nome: Controle de irrigação Água Viva

Introdução: Este documento tem como objetivo apresentar o fluxo de eventos que compõem as interações entre atores e o sistema. Além disso, contém outras informações consideradas essenciais como: pré e pós - condições, e cenários principais.

Descrição: O fluxo apresentado descreve a coleta dos dados de sensores que são usados para calcular a quantidade de irrigação necessária, dadas as circunstâncias externas durante sua execução. Em seguida, o resultado é aplicado para definir o volume de água usado na irrigação, ou para indicar a necessidade de uma nova coleta posteriormente através de ajuste do temporizador.

Atores: Temporizador e Bomba d'água.

Precondições: Os sensores devem estar ativados. Os dados transmitidos entre sensores e o sistema serão enviados através de mensagens.

Pós-condições: A Bomba d'água deve acionar a irrigação ou o intervalo de tempo do temporizador deve ser ajustado.

Fluxo principal:

Passo 1: O Temporizador ativa a captação de dados dos sensores [Requisito especial RE1];
Passo 2: O sistema recebe os dados coletados e verifica-os; [Fluxo de exceção 1] [Fluxo de exceção 2]
Passo 3: O sistema executa os cálculos de volume de água da irrigação; [Subfluxo 1]
Passo 4: O sistema aciona a Bomba d'água para iniciar a irrigação de acordo com resultado calculado; [Fluxo alternativo 1]
Passo 5: O caso de uso se encerra.

Fluxos alternativos:

Fluxo alternativo 1: Ajustar temporizador para novos cálculos
Passo 1: O sistema verifica que o cálculo resultante indica que ainda não é necessário iniciar a irrigação;
Passo 2: O sistema ajusta o intervalo de tempo do temporizador para uma nova coleta posteriormente;
Passo 3: O fluxo retorna para o Passo 1 do fluxo básico.

Subfluxos:

Subfluxo 1: Calcular com base nos dados dos sensores
Passo 1: O sistema realiza cálculo com os dados do PH recebidos pelos sensores;
Passo 2: O sistema verificar o resultado do cálculo e ajusta valor da quantidade de água da irrigação; [Requisito especial RE2] [Fluxo de exceção 3]
Passo 3: O fluxo retorna para Passo 3 do fluxo básico.

Exceções:

Exceção 1: Dados não recebidos
Passo 1: O sistema verifica que os dados necessários não foram recebidos dos sensores;
Passo 2: O fluxo retorna para o Passo 1 do fluxo básico.

Exceção 2: Dados de pH fora do limite
Passo 1: O sistema verifica que os dados recebidos dos sensores possuem valores de pH fora dos limites possíveis estabelecidos;
Passo 2: O fluxo retorna para o Passo 1 do fluxo básico.

Exceção 3: Dados inconsistentes
Passo 1: O sistema verifica que o resultado do cálculo apresenta valor inconsistente para realizar a irrigação;
Passo 2: O fluxo retorna para o Passo 1 do fluxo básico.

Requisitos especiais:

RE1: Os valores recebidos pelos sensores não podem ser nulos ou possuir nível acima de 7.0;
RE2: O resultado do cálculo deve ser positivo e não nulo. Além disso, se o valor final for menor do que constante mínima para iniciar a irrigação, esta ação deve ser adiada para um próximo ciclo de execução.

3.2 Apresentação do modelo

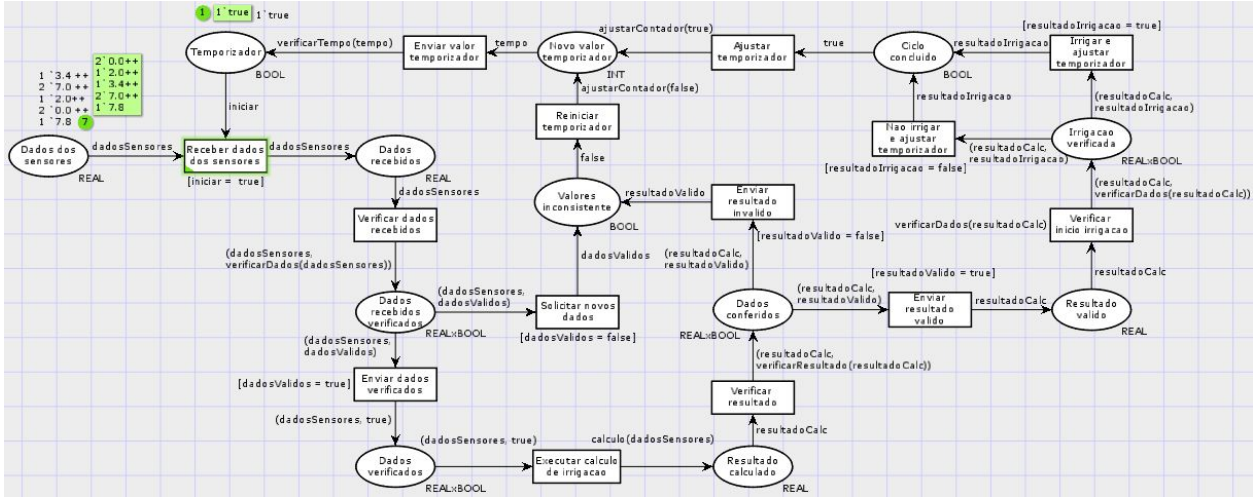
A partir do caso de uso do sistema de irrigação Água Viva, é possível compreender o seu funcionamento e identificar seus fluxos de execução. Como forma de verificar a especificação realizada, apresento sua modelagem utilizando redes de Petri Coloridas. Em seguida o modelo executável serve de auxílio para a especificação dos casos de teste do sistema, que também são uma forma de verificação.

A prototipação auxilia a comunicação entre as partes interessadas, ajudando no entendimentos sobre como o software. Desta forma é possível evitar próximas etapas do processo acabem comprometidas, inclusive a atividade de testes. Definir casos de testes baseados em informações incorretas sobre o funcionamento do sistema, acaba por gerar um retrabalho para os testadores do sistema, e também mais custos para o desenvolvimento.

O modelo definido, apresentado na Figura 4, teve os fluxos de execução do sistema extraídos a partir da documentação de caso de uso. A marcação inicial possui como tokens os dados de ph recolhidos pelos sensores, e a entrada como verdadeira para o temporizador acionar o sistema. A depender do token enviado pelos sensores para o recebimento de dados pelo sistema, diferentes fluxos podem ser executados.

São identificadas quatro situações possíveis de se alcançar com o sistema em funcionamento. Duas execuções representam os cenários de falha, em que os dados não são recebidos ou o resultado do cálculo está inconsistente. E duas execuções que representam cenários de sucesso, porém apenas um realiza a irrigação de imediato.

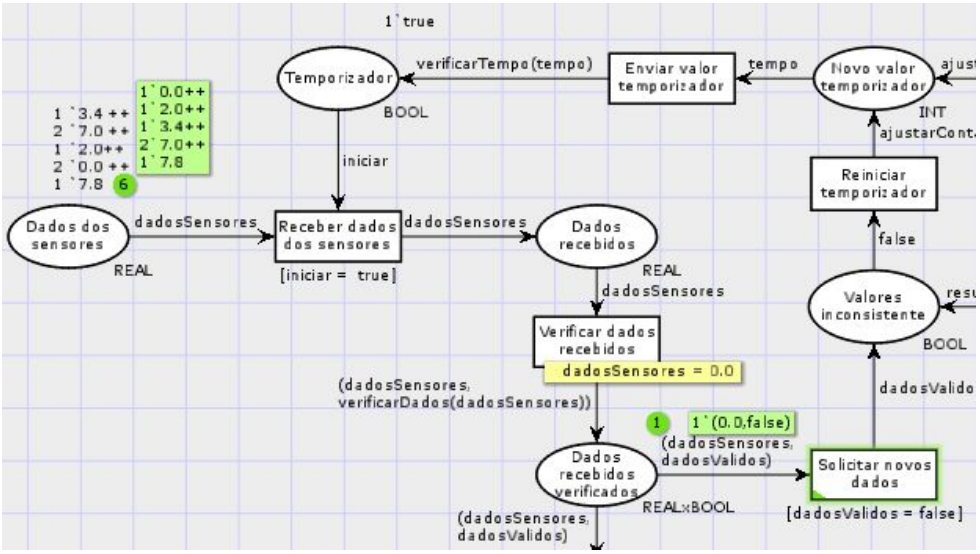
Figura 4 - Modelagem com redes de Petri Coloridas do sistema Água Viva



Fonte: Elaborada pela autora.

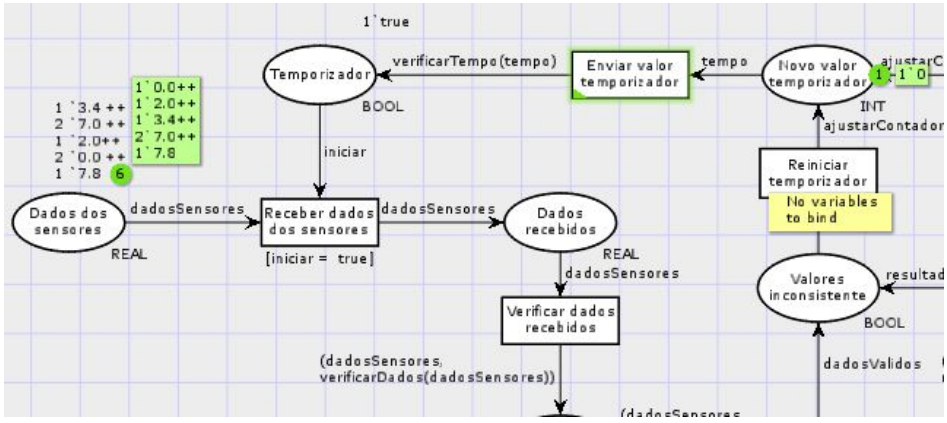
A execução de cenário com falha deve finalizar o ciclo com o temporizador possuindo valor de tempo igual a zero horas, e ficha no temporizador com valor *true*, para que novos dados sejam recolhidos de imediato. Se o cenário for de sucesso o tempo é ajustado para cinco horas, intervalo de tempo determinado entre as coletas de dados dos sensores. Para demonstrar a simulação do modelo, apresento as Figuras 5, 6 e 7 com os resultado e diferentes marcações encontrados na execução do protótipo diante de um cenário inválido.

Figura 5 - Execução de cenário inválido parte 1



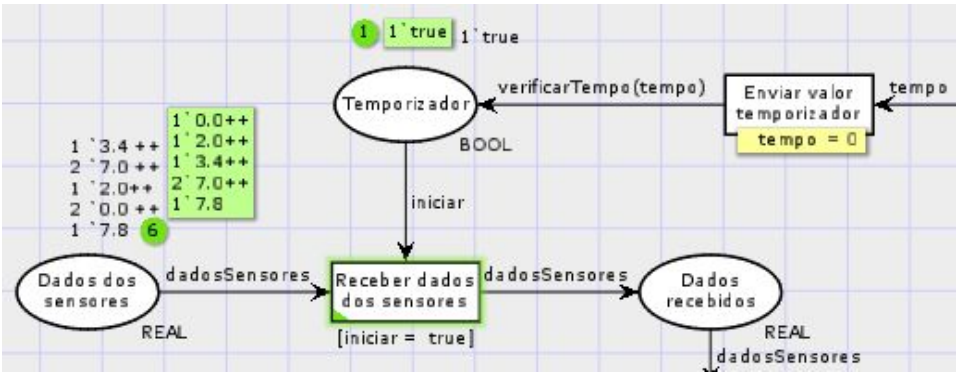
Fonte: Elaborada pela autora.

Figura 6 - Execução de cenário inválido parte 2



Fonte: Elaborada pela autora.

Figura 7 - Execução de cenário inválido parte 3



Fonte: Elaborada pela autora.

3.3 Apresentação dos Casos de Teste

No geral, os dados necessários para construir os casos de teste podem variar de um projeto para o outro, e dependem de aspectos como criticidade do sistema, exigência dos clientes, e experiência da equipe de testadores. Não existem regras para esta atividade, porém, alguns elementos são indicados constantemente em materiais que guiam esta atividade. O documento com casos de teste apresentado nesta seção foi elaborado para servir como guia na realização de testes funcionais do sistema. As informações elencadas são baseadas no "Standard for Software Test Documentation" [4], material desenvolvido com o objetivo de prover praticidade a atividade de testes.

Os elementos que utilizei na definição dos casos de teste foram: Cenário, indica a ação que inicia o caso de teste em questão; Item de teste, consiste na identificação do caso de teste; Entrada, apresenta os dados utilizados como entrada para realizar o cenário em questão; Requisito, coloca quais são os requisitos do caso de uso relacionados ao cenário de teste em questão; Tipo de fluxo, informa se é um caso de teste de fluxo válido ou inválido; Ações, indica as atividades envolvidas na execução do cenário; Resultado esperado, diz qual é o estado ou condição necessária para encerramento do teste.

Adicionar as informações elencadas nos casos de testes é uma tarefa dos testadores, que dedicam seu tempo para compreender a documentação de especificação, e, em seguida, extrair os dados necessários para um procedimento bem sucedido. Esta tarefa demanda um certo tempo de trabalho, e, se a especificação não estiver compreensível, pode ser ainda mais laboriosa. Utilizando o modelo formal do sistema, pude facilitar a definição dos casos de teste do sistema Água Viva. Os dados extraídos são colocados a seguir e indicados na Tabela 1.

Um primeiro passo realizado foi extrair do modelo os itens de teste a serem executados. Com a intenção de cobrir todos os cenários possíveis, pude observar que, no modelo especificado, os

itens possíveis de se executar estão evidenciados, bastou seguir os fluxos orientados pelos arcos. São no total quatro caminhos possíveis de se obter na execução do sistema, e indico-os na coluna "Item de teste".

Pude também extrair do modelo os dados de entrada necessários para a execução dos testes. No sistema Água Viva, os dados que são utilizados na sua execução são provenientes dos sensores e do temporizador. É possível identificar estas entradas observando os tokens existentes na marcação inicial do sistema. No caso de sistemas com muitas entradas possíveis isto pode se tornar ainda mais confuso para o testados, e com o modelo de redes de Petri Colorida basta observar os tokens adicionados para a simulação. As entradas dos cenários são indicados na coluna "Entrada".

Outros dados retirados a partir do modelo foram as ações necessárias para os casos de teste. Os itens que forneceram estas informações foram as transições indicadas no protótipo. Cada transição, como colocado anteriormente, apresenta as ações ou eventos possíveis da execução que fazem o sistema mudar de estado. Por exemplo, as transições que compõem o caminho do fluxo de execução principal são: Receber os dados dos sensores, Verificar dados recebidos; Transmitir dados verificados; Executar cálculo de irrigação; Verificar resultado do cálculo; Transmitir resultado verificado; Avaliar inicio de irrigação; Irrigar; Ajustar temporizador; Enviar valor temporizador. Estas transições, após identificadas, foram resumidas nas ações: 1. Temporizador inicia o recebimento dos dados; 2. O sistema valida os dados recebidos 3. O sistema calcula volume de irrigação e verifica resultado; 4. O sistema avalia início da irrigação e executa irrigação; 5. O sistema ajusta dados do temporizador. Estas informações encontram-se na coluna "Ações" da tabela com casos de teste.

Tabela 2 - Casos de teste do sistema Água Viva

Cenário	Item de teste	Entrada	Requisito	Tipo de fluxo	Ações	Resultado esperado
Executar irrigação	Executar irrigação com sucesso.	Dados coletados por sensores e informação do temporizador.	Fluxo básico.	Válido	1. Temporizador inicia o recebimento dos dados; 2. O sistema valida os dados recebidos 3. O sistema calcula volume de irrigação e verifica resultado; 4. O sistema avalia início da irrigação e executa irrigação; 5. O sistema ajusta dados do temporizador.	Execução da irrigação e confirmação de sucesso sem necessidade de ajuste de temporizador.

	Tentar executar irrigação com dados de entrada inválidos.	Informação do temporizador com dados de sensores inválidos.	Fluxo básico e fluxo de exceção 1.	Inválido	1. Temporizador inicia o recebimento dos dados; 2. O sistema valida os dados recebidos e verifica que estão inválidos; 3. O sistema inicia solicitação de novos valores; 4. O sistema reinicia temporizador para coleta de dados imediata.	Acionar novamente inicio da coleta de dados dos sensores.
	Tentar executar irrigação com resultado do cálculo de irrigação inconsistente.	Dados coletados por sensores e informação do temporizador.	Fluxo básico e fluxo de exceção 2.	Inválido	1. Temporizador inicia o recebimento dos dados; 2. O sistema valida os dados recebidos 3. O sistema calcula volume de irrigação e verifica que o resultado é inválido; 4. O sistema inicia solicitação de novos valores; 5. O sistema reinicia temporizador para coleta de dados imediata.	Acionar novamente inicio da coleta de dados dos sensores.
Ajustar temporizador	Não executar irrigação e ajustar temporizador para nova coleta de dados.	Dados coletados por sensores e informação do temporizador.	Fluxo alternativo 1.	Válido	1. Temporizador inicia o recebimento dos dados; 2. O sistema valida os dados recebidos 3. O sistema calcula volume de irrigação e verifica resultado; 4. O sistema avalia início da irrigação e não executa irrigação; 5. O sistema ajusta dados do temporizador.	Ajustar intervalo de tempo do temporizador para nova coleta de dados.

Fonte: Elaborada pela autora.

3.1.1 Resultados

Ao final do estudo foi obtido o modelo executável com redes de Petri Coloridas, e também a definição dos casos de testes funcionais que serão executados. Estes materiais são resultado do uso das técnicas aqui apresentadas.

O modelo do sistema com redes de Petri Colorida oferece uma abstração do sistema que, além de facilitar a comunicação entre as partes envolvidas, auxilia outras atividades do processo, como a de testes, central neste trabalho.

Foi escolhido um protótipo de estrutura simples e direta para desenvolver o estudo, pois a finalidade central é apresentar a utilidade do formalismo para a atividade de testes. Porém, apesar do foco não ser a construção do protótipo, é possível elaborar modelos mais complexos utilizando de forma mais profunda as propriedades das redes de Petri Coloridas.

As informações dos casos de testes após elaboradas servem para guiar o procedimento de testes de forma segura. Os cenários possíveis e suas propriedades foram mapeados, e exigiram uma menor quantidade de trabalho para o procedimento de testes. Com os fluxos possíveis de se executar no sistema evidenciados no modelo, pode-se extrair diretamente os casos de teste a partir deles, sem precisar recorrer a outros documentos de especificação do sistema. A atividade de modelar e de testar em conjunto serviram para verificar e validar o sistema, e contribuíram para o aspecto da qualidade.

4. Considerações finais

4.1 Conclusão

Neste trabalho foi desenvolvido um caso de estudo utilizando técnica formal de modelagem para a definição de casos de testes, de forma a contribuir com o aspecto de qualidade do software. O desenvolvimento do trabalho foi realizado com o intuito de avaliar se os benefícios obtidos são satisfatórios e possibilitam um melhor processo de software.

Agregar qualidade a produtos de software pode vir a ser, dependendo do sistema em construção, uma tarefa laboriosa, porém profundamente necessária. Utilizar as atividades de modelagem e testes no processo de desenvolvimento de software, o mais cedo possível, contribui para otimizar o sistema e evitar o descobrimento de erros em fases mais avançadas. Verificar e validar a elaboração de um sistema são aspectos centrais para a qualidade. Tanto a atividade de modelar quanto a de testar são úteis na verificação e validação, e podem, inclusive, ser aplicadas em conjunto.

A aplicação da técnica de modelagem formal com redes de Petri Coloridas, para representar o sistema do caso de estudo Água Viva, evidenciou um conjunto de informações que puderam ser extraídas e disponibilizadas para a elaboração dos casos de testes. O mapeamento dos fluxos do modelo formal para definir os testes trouxe praticidade e segurança, visto que o modelo executável ajudou a abstrair o funcionamento do sistema e a verificar sua especificação. Após a utilização das técnicas mencionadas, foi possível constatar que, além de auxiliarem os procedimentos de testes, são eficientes para a validação e verificação dos requisitos do sistema, e influenciam positivamente no aspecto de qualidade.

4.2 Trabalhos futuros

O trabalho desenvolvido ainda precisa ser aprofundado em alguns aspectos para superar suas limitações. Um dos problemas que não foi abordado, é o de como incidir no aspecto do tempo de trabalho do processo de software como um todo. O estudo tratou da praticidade gerada com a modelagem para a atividade de definição de casos de teste, porém seria uma prática a mais a ser realizada no desenvolvimento do sistema. Além disso, o uso desta técnica exige um certo estudo prévio sobre a modelagem e linguagem utilizadas nas redes de Petri Coloridas.

Uma melhoria também importante seria a introdução de modelos mais complexos, que utilizassem as potencialidades oferecidas pelo formalismo redes de Petri Coloridas, e pela ferramenta CPN Tools. O modelo desenvolvido no caso de estudo é limitado diante possibilidades das suas possibilidades de uso. Incorporar ao trabalho modelos hierárquicos ou o

conceito de tempo, pode tornar a prototipação e simulação mais sofisticadas, e ajudaria na validação o estudo.

Por fim, uma outra meta é a de adicionar o conceito de espaço estado, que é possível de se desenvolver com as redes de Petri Coloridas. Esta poderosa forma de análise fornece uma melhor exploração do modelo construído e ajuda na construção de relatórios sobre os comportamentos padrões da rede de Petri elaborada. Muitas vezes se o sistema, mesmo após a validação, apresenta erros, provavelmente este será refletido no relatório de espaço estado.

Como forma de avaliar os resultados, seria importante realizar comparativos entre os processos de desenvolvimento com e sem o formalismo de redes de Petri Coloridas para definir os casos de testes. Após definir critérios de avaliação, o resultado serviria para legitimar o trabalho e ter uma melhor visão sobre suas vantagens e desvantagens.

Superar as limitações citadas e trazer os conceitos mais avançados para o trabalho pode contribuir para maiores progressos e fortalecer a importância do estudo. A área de qualidade vem ganhando espaço na engenharia de software e sendo cada vez mais incorporada por organizações que compreendem os ganhos obtidos com suas práticas. É fundamental e promissor o desenvolvimento de soluções que se enquadram neste assunto.

Referências

1. BAUMGARTNER NETO, August. **Extração de casos de teste utilizando Redes de Petri hierárquicas e validação de resultados utilizando OWL**. 2015. Dissertação (Mestrado em Engenharia de Controle e Automação Mecânica) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2015. doi:10.11606/D.3.2016.tde-22062016-075239. Acesso em: 2017-12-13.

2. Delamaro, M. E., Maldonado, J. C., Jino, M. **Introdução ao Teste de Software**. Editora Campus, 2007.

3. Heumann, J. **Generating Test Cases From Use Cases**, The Rational Edge: e-zine for the rational community, 2001.

4. IEEE Computer Society. **IEEE Std 829: Standard for Software Test Documentation**. September, 1998.

5. ISO 25010 **Systems and software Engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models.** ISO/IEC - International Organization for Standardization and International Electrotechnical Comission. Switzerland, 1 edition, 2011.

6. JENSEN, Kurt; KRISTENSEN, Lars M. **Coloured Petri Nets: Modelling and Validation of Concurrent Systems**. New York: Springer, 2009.

7. Pressman, R. (2001). **Engenharia de Software**. McGrawHill, 4th edition.

8. Robert V. Binder. 1999. **Testing Object-Oriented Systems: Models, Patterns, and Tools.** Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

9. SOMMERVILLE, Ian. **Engenharia de Software.** 9. ed. São Paulo: Pearson, 2011.

10. Sousa, Henrique & Luiz de Castro Leal, André & Staa, Arndt & Leite, Julio, **Extração de casos de teste a partir de modelos de processos de negócio**, XVII Workshop em Engenharia de Requisitos, (2014).

11. THOMAS MÜLLER, Debra Friedenberg, Istqb Wg Foundation Level. European Organisation for Quality – Software Group (Org.). **International Software Testing Qualifications Board:** Foundation Level Syllabus. 2011. Disponível em: <https://www.istqb.org/downloads>. Acesso em: 05 dez. 2017.

12. VALE, Liliane do Nascimento. **Especificação de testes funcionais usando Redes de Petri a objetos para softwares orientados a objetos.** 2009. 139 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Universidade Federal de Uberlândia, Uberlândia, 2009.