Pós-Graduação em Ciência da Computação

THIAGO MONTEIRO PROTA

ANÁLISES ESTRUTURAL E COMPORTAMENTAL ORIENTADAS A CONFORMIDADE PARA O DESENVOLVIMENTO DE APLICAÇÕES MULTIMÍDIA

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE
2016

Thiago Monteiro Prota

Análises Estrutural e Comportamental Orientadas a Conformidade para o Desenvolvimento de Aplicações Multimídia

Este trabalho foi apresentado à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito para obtenção do grau de doutor em ciência da computação.

ORIENTADOR: **Prof.. Carlos A. G. Ferraz**

RECIFE
2016

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

P967a Prota, Thiago Monteiro
Análises estrutural e comportamental orientadas a conformidade para o desenvolvimento de aplicações multimídia / Thiago Monteiro Prota. – 2016.
170 f.: il., fig., tab.

Orientador: Carlos André Guimarães Ferraz.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2016.
Inclui referências e apêndice.

1. Sistemas distribuídos. 2. Representação comportamental. 3. Linguagens declarativas. I. Ferraz, Carlos André Guimarães (orientador). II. Título.

004.36 CDD (23. ed.) UFPE- MEI 2017-16

Thiago Monteiro Prota

Análises Estrutural e Comportamental Orientadas a Conformidade para o Desenvolvimento de Aplicações Multimídia

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de **Doutor** em Ciência da Computação.

Aprovado em: 07/10/2016.

_____

Prof. Carlos André Guimarães Ferraz
Orientador do Trabalho de Tese

BANCA EXAMINADORA

_____

Prof. Dr. Fernando da Fonseca de Souza
Centro de Informática / UFPE

_____

Prof. Dr. Vinicius Cardoso Garcia
Centro de Informática / UFPE

_____

Prof. Dr. Alexandre Marcos Lins de Vasconcelos
Centro de Informática / UFPE

_____

Prof. Dr. Carlos de Salles Soares Neto
Departamento de Informática / UFMA

_____

Prof. Dr. Gibeon Soares de Aquino Junior
Departamento de Informática e Matemática Aplicada/UFRN

Eu dedico esta tese a minha esposa Roberta Nazário.

Agradecimentos

Primeiramente, a minha esposa Roberta Nazário e a toda minha família por toda paciência, compreensão e incentivo que foram fundamentais para o meu sucesso.

Ao meu orientador Carlos Ferraz pela confiança depositada e por todos os ensinamentos que foram de grande importância para a realização deste trabalho.

A todos os meus companheiros da UFPE, do projeto Samsung-TVD e da Dataprev (UDPB) pelo suporte oferecido que foram essenciais durante toda a pesquisa.

Aos meus amigos de uma vida toda, que, mesmo que não tenham feito contribuições diretas para este trabalho, em alguma etapa da minha vida me incentivaram e me apoiaram para eu chegar onde estou.

E a todos aqueles, que de alguma forma, contribuíram para o desenvolvimento deste trabalho.

A todos vocês, muito obrigado!

Resumo

As linguagens declarativas normalmente atuam no desenvolvimento de aplicações multimídia, pois suas características permitem suportar adequadamente a natureza assíncrona e descritiva dessas aplicações. Neste cenário, a robustez surge como um fator determinante para a qualidade dessas aplicações, dada a vasta quantidade de plataformas e dispositivos usuais, que, ocasionalmente, apresentam problemas de execução. Neste contexto, as especificações dessas linguagens são de grande importância para o processo de desenvolvimento, pois além de direcionar a codificação, definindo as restrições léxicas, sintáticas e semânticas, também formalizam como os conteúdos devem ser executados pelos interpretadores. Este trabalho tem por objetivo investigar a viabilidade de se aplicar análises estruturais e comportamentais orientadas a conformidade para o desenvolvimento de aplicações multimídia, a fim de eliminar a dependência de interpretadores para atestar sua corretude. Para tal, a linguagem NCL foi utilizada como alvo do estudo, devido à sua representatividade para o problema.

Palavras-chave: Máquina de Estados. Assertiva. Representação Comportamental. Linguagens Declarativas. NCL.

Abstract

Declarative languages are typically used in the development of multimedia applications, as its features allow to properly support the asynchronous and descriptive nature of these applications. In such a scenario, robustness appears as a determining factor in the quality of these applications, given the vast amount of platforms and devices that occasionally have implementation problems. In this context, specifications of these languages are of great importance to the development process, as they impose lexical, syntactic and semantic restrictions, and also formalize how content must be interpreted. This work aims at investigating the feasibility of applying structural and behavioral analysis oriented to compliance to the development of multimedia applications in order to eliminate dependence on interpreters to prove its correctness. To this end, NCL was used as a target of study due to its problem representativeness.

Keywords: State Machine. Assertion. Behavioral Representation. Declarative Languages. NCL.

Lista de Figuras

Lista de Quadros

Sumário

Introdução

Aplicações multimídia consistem essencialmente de uma combinação de diferentes formas de conteúdos (mídias) como, por exemplo, textos, áudios, vídeos, animações, entre outros, permitindo a interação de algumas delas. Para criar relações consistentes também descrevem como os objetos de mídia se relacionam entre si, com o espaço e o com o tempo, permitindo uma interação livre limitada pelas restrições espaciais e temporais (LI *et. al.*, 2014). A sincronização representa um dos principais recursos na definição dessas aplicações, pois permite associar um conjunto de mídias com o tempo (contínua) e com as ações do usuário (discreta), construindo interfaces gráficas dinâmicas e consistentes (HUANG *et. al.*, 2013). As aplicações multimídia podem ser classificadas como sistemas dirigidos a eventos, isto é, a aplicação não faz nada enquanto espera por um evento, e quando este ocorre, ela reage e volta para o estado de espera (WAGNER *et. al.*, 2006; SANTOS; BRAGA; MUCHALUAT-SAADE, 2013). Porém, como a ocorrência desses eventos é imprevisível, o paradigma orientado a eventos normalmente é indicado para definir todas as etapas do ciclo de vida da aplicação (SANTOS; BRAGA; MUCHALUAT-SAADE, 2013). Outra característica relevante deste tipo de aplicação é o fato de ser independente de dispositivos e plataformas, o que possibilita a sua execução em diversos contextos de uso (CHANG, 2005). Neste cenário, a robustez e a interoperabilidade surgem como fatores determinantes para a qualidade dessas aplicações, dado a vasta quantidade de plataformas e dispositivos usuais, que, ocasionalmente, apresentam problemas durante a execução. As aplicações Web são um dos exemplos mais conhecidos de aplicações multimídia.

A programação declarativa geralmente se mostra adequada para o desenvolvimento de aplicações multimídia, pois suas características permitem suportar adequadamente a natureza dinâmica e descritiva das aplicações, permitindo que os desenvolvedores apenas descrevam "o que" deve ser computado, sem definir "como" isto vai ocorrer (LLOYD, 1994; BOSTOCK, 2010; SANTOS; BRAGA; MUCHALUAT-SAADE, 2013). No paradigma imperativo cabe ao

desenvolvedor definir todo o fluxo de controle da aplicação, necessitando ter um elevado conhecimento em programação (LLOYD, 1994; BOSTOCK, 2010; SANTOS; BRAGA; MUCHALUAT-SAADE, 2013). Enquanto isso, no declarativo, geralmente, utiliza-se uma programação de alto nível, pois a linguagem utilizada possui aspectos de controle próprios, pré-definidos por suas especificações, bastando ao desenvolvedor apenas definir a lógica da aplicação sem especificar o fluxo de controle (LLOYD, 1994; BOSTOCK, 2010; SANTOS; BRAGA; MUCHALUAT-SAADE, 2013). É importante ressaltar que a responsabilidade de renderizar a aplicação conforme a sua definição declarativa fica a cargo da plataforma sobre a qual ela foi executada (KING; SCHMITZ, 2000). Neste contexto, algumas linguagens se destacam quanto a sua aplicabilidade para as plataformas Web, Mobile e de TV Digital (TVD), tais como: HTML5 (W3C, 2014); XHTML (W3C, 2010); SVG (W3C, 2011); NCL (ITU, 2009); e SMIL (W3C, 2008).

Muitas organizações buscam especificar linguagens declarativas para os diversos contextos de uso (Web, Mobile e TV) e para as variadas finalidades (Aplicações Multimídia, Aplicações Interativas, Modelagem e Animações). Essas organizações atuam estabelecendo padrões públicos consistentes para criação e interpretação dessas linguagens. Como exemplo dessas organizações pode-se destacar a World Wide Web Consortium (W3C[1]), a Associação Brasileira de Normas Técnicas (ABNT[2]) e a *International Telecommunication Union* (ITU[3]). Esses padrões definem todas as manipulações possíveis a serem utilizadas pelas linguagens que eles especificam, desta forma especificam: a gramática da linguagem; as relações estruturais; os aspectos visuais, estilo e apresentação; e as interfaces para manipulação de estruturas multimídia simples (textos, imagens e vídeos) e complexas (imagens 3D e animações) (KING; SCHMITZ, 2000). Os padrões públicos são de extrema importância para o processo de desenvolvimento de seus conteúdos, pois além de direcionarem a codificação por parte dos desenvolvedores, definindo as restrições léxicas, sintáticas e semânticas, também formalizam como os conteúdos devem ser executados pelos seus interpretadores (geralmente desenvolvidos por terceiros) e, ocasionalmente, também especificam suítes de teste (Ex.: Suíte de Testes para SMIL[4] e para SVG[5]) (PINHEIRO *et. al.*, 2012; LAMADON *et. al.*, 2003; EIDENBERGER, 2003).

---

1 *World Wide Web Consortium* (W3C). Disponível em: <www.w3.org>. Acesso em: 20 de Setembro de 2016.

2 Associação Brasileira de Normas Técnicas (ABNT). Disponível em: <www.abnt.org.br>. Acesso em: 20 de Setembro de 2016.

3 *International Telecommunication Union* (ITU). Disponível em: <www.itu.int>. Acesso em: 20 de Setembro de 2016.

4 W3C. SMIL 3.0 Test Suite. Disponível em: <www.w3.org/2007/SMIL30/testsuite>. Acesso em: 20 de Setembro de 2016.

As aplicações multimídia demandam configurações de hardware cada vez mais complexas que devem dar suporte a uma diversidade de novas funcionalidades e prover recursos computacionais avançados para permitir uma rica experiência ao usuário (HUANG *et. al.*, 2013). Desta forma, apesar da vasta quantidade de interpretadores existentes, fica praticamente inviável confiar na robustez deles, pois está cada vez mais difícil preservar as especificações (HUANG *et. al.*, 2013). Para que um usuário possa executar corretamente um aplicativo em um determinado interpretador, é necessário que ambos, o aplicativo e o interpretador, estejam conforme os padrões estabelecidos. Porém, várias linguagens não definem mecanismos para atestarem a conformidade de seus conteúdos e dos seus interpretadores, obrigando os desenvolvedores a elegerem, empiricamente, um interpretador para testar exaustivamente suas aplicações, condicionando a qualidade dos seus produtos ao uso de uma determinada versão de um interpretador (EIDENBERGER, 2003; CHOUDHARY *et. al.*, 2014). A Figura 0.1 apresenta o contexto sob o qual esta tese está inserida

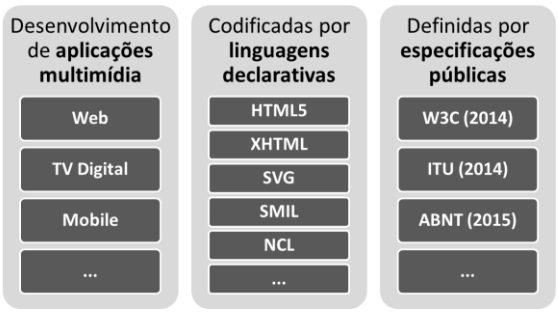

Figura 0.1. Resumo do contexto da tese.

1.1 Motivação

O desenvolvimento de aplicações multimídia é bastante complexo, devido à vasta quantidade de plataformas e dispositivos de uso, que, ocasionalmente, apresentam problemas de execução. Por isso, a robustez surge como um fator determinante para a qualidade desses aplicativos. Nesse cenário, é comum utilizar os interpretadores disponíveis para garantir essa robustez. Este processo de desenvolvimento baseado na execução em interpretadores questionáveis traz alguns problemas. Primeiramente, tende a diminuir a importância das especificações das linguagens, muitas vezes desprezadas durante a codificação, pois o desenvolvedor erroneamente assume que a corretude da aplicação é definida pela execução em um interpretador específico (SANTOS; LENZ, 2014;

5 W3C. SVG 1.1 Second Edition Test Suite. Disponível em: <www.w3.org/Graphics/SVG/Test/20110816/>. Acesso em: 20 de Setembro de 2016.

AMMONS *et. al.*, 2002). Consequentemente, não garante a qualidade dos artefatos produzidos sob sua execução, por não existir uma certificação acerca da sua conformidade. Além disto, caracteriza-se por ser um processo ineficiente, exigindo um elevado esforço manual para identificar os problemas existentes, pois demanda inúmeras execuções manuais para cobrir todos os comportamentos possíveis (AMMONS *et. al.*, 2002; SANTOS, 2012; EIDENBERGER, 2003).

Para evitar a dependência de interpretadores de terceiros, é necessário que os desenvolvedores amadureçam seu processo de desenvolvimento. Uma alternativa é utilizar análises orientadas à conformidade, que, diferentemente do processo praticado, busca realizar a rastreabilidade para os trechos normativos da especificação a fim de fundamentar suas análises. Além disto, é mais eficiente na identificação de problemas, dado que as análises podem ser realizadas de forma completa e automática. Desta forma, podem identificar todos os erros existentes em uma determinada versão da aplicação com um esforço reduzido. Também sugere uma maior eficiência na resolução dos problemas, pois fornece insumos para a correção por meio da rastreabilidade para os trechos normativos que fornecem todo o conhecimento relacionado com o problema identificado. Neste modelo é imprescindível que essas análises supram a necessidade de execução nos interpretadores, e desta forma, deve-se prover facilidades de validação (léxica e sintática) e verificação (semântica).

Apesar deste tipo de abordagem se mostrar mais apropriado para o processo de desenvolvimento de aplicações multimídia, é importante avaliar a viabilidade de definir soluções que promovam análises orientadas à conformidade. Uma vez que, as especificações, geralmente, não apresentam as regras em um formato adequado, podendo definir uma regra em várias seções ou até em normas diferentes (PROTA *et. al.*, 2014), elevando, consequentemente, o esforço para garantir a rastreabilidade das análises para essas regras. E também, devido à complexidade em definir abordagens de verificação (comportamental) de software, a qual apresente evidências de comportamentos inesperados próprios da aplicação, não se limitando apenas a identificar padrões de inconsistência conhecidos. Existem algumas questões conhecidas, inerentes a este tipo de análise, que elevam a complexidade das soluções, tais como: a definição de abstrações concisas e consistentes para representar o comportamento do software sob a perspectiva demandada (ROBILLARD *et. al.*, 2012; OLIVEIRA *et. al.*, 2001; CHENG; KRISHNAKUMAR, 1993; WALKINSHAW *et. al.*, 2013; PICININ *et. al.*, 2012; LORENZOLI *et. al.*, 2008); a elevada quantidade de informações a ser processada ("explosão de estados" (WAGNER *et. al.*, 2006)) e apresentada para prover informações relevantes para o contexto de análise.

1.2 Objetivos

Esta seção visa descrever os objetivos deste trabalho, primeiramente, de forma mais geral para mostrar a dimensão da tese, e posteriormente, de forma mais detalhada, especifica a fim de descrever cada contribuição proposta.

1.2.1 Objetivo geral

Como objetivo geral, este trabalho se propõe a investigar a viabilidade de se aplicar análises orientadas a conformidade a fim de prover uma alternativa para atestar a qualidade das aplicações multimídia construídas a partir de linguagens declarativas padronizadas por especificações públicas. Por meio da identificação de problemas estruturais (léxicos e sintáticos) e comportamentais (semânticos) de forma precisa e confiável, fundamentada pela rastreabilidade aos trechos normativos que especificam a linguagem de desenvolvimento, busca-se eliminar a dependência de utilização de interpretadores no processo de desenvolvimento dessas aplicações.

1.2.2 Objetivos específicos

Especificamente, pretende-se:

I. Prover facilidades para a validação de aplicações multimídia declarativas, a fim de identificar automaticamente os problemas léxicos, sintáticos e semânticos estáticos, por meio de uma análise estrutural orientada à conformidade;
II. Prover facilidades para a verificação de aplicações multimídia declarativas, a fim de permitir a identificação dos problemas comportamentais, por meio de uma análise comportamental orientada à conformidade.
III. Prover a rastreabilidade para os trechos normativos das especificações que visam fundamentar os resultados das análises estruturais e comportamentais de aplicações multimídia declarativas; e
IV. Definir interfaces gráficas para a manipulação das soluções de análises estrutural e comportamental a partir de um Ambiente de Desenvolvimento Integrado (IDE – *Integrated Development Environment*) próprio para o desenvolvimento de aplicações multimídia declarativas.

1.3 Metodologia

Para a realização da tese as principais fases da metodologia utilizadas são:

1. **Levantamento do estado da arte** – etapa responsável pela revisão do estado da arte, a fim de identificar os principais trabalhos relacionados e compreender o contexto teórico da área de atuação. Esta fase é de extrema importância para o sucesso do trabalho, pois busca prover insumos para todas as outras fases da pesquisa;
2. **Definição do escopo da pesquisa** – etapa responsável por consolidar o entendimento do problema, a fim de identificar os principais desafios para propor as metas do trabalho com o objetivo de delimitar o escopo da pesquisa;
3. **Definição da proposta de solução** – identificação dos requisitos associados e projeto de uma solução de qualidade aderente. Desta forma, buscou-se avaliar os trabalhos relacionados, identificando as principais contribuições e limitações, e compreender as abordagens alternativas a fim de propor uma solução adequada;
4. **Planejamento de avaliações** – esta etapa é responsável por definir e planejar cenários de teste e experimentos controlados relevantes para o contexto da solução. Desta forma, buscou-se identificar as premissas, o método adequado e o esforço necessário para os processos de execução e análise das avaliações da solução, aplicados em dois momentos (descritos pelas etapas 6 e 8);
5. **Desenvolvimento da prova de conceito** – nesta etapa buscou-se definir, implementar e testar o escopo da solução a ser desenvolvido como prova de conceito. Para o contexto deste trabalho a prova de conceito é definida como uma implementação incompleta da solução, porém, que seja suficiente para verificar as contribuições e limitações da proposta apresentada;
6. **Avaliação da prova de conceito** – esta etapa teve por objetivo atestar a qualidade da prova de conceito e evidenciar as contribuições e limitações da solução a fim de encontrar oportunidades de melhoria. Os cenários de teste executados e analisados nesta etapa foram pré-definidos na etapa 4;
7. **Realizar melhorias e refinamentos** – nesta etapa buscou-se evoluir a prova de conceito implementada, realizando as correções, melhorias e refinamentos identificados na etapa 6, a fim de melhorar os resultados até o momento obtidos; e
8. **Realizar experimentos controlados** – esta etapa teve por objetivo atestar a viabilidade da solução definida por este trabalho. Desta forma, buscou, através de experimentos formais,

avaliar a confiabilidade das análises estruturais e comportamentais. Os experimentos executados e analisados nesta etapa foram pré-definidos na etapa 4.

1.4 Estrutura do Documento

Este trabalho está organizado da seguinte forma:

- **Capítulo 2 – Fundamentação.** Neste capítulo são apresentados os principais conceitos que fundamentaram a proposta da solução utilizando as subseções: **Processo de Desenvolvimento Orientado a Análise** – que identifica as principais contribuições e limitações do processo usual de desenvolvimento de aplicações multimídia e apresenta uma abordagem alternativa para este fim, a ser aplicada na proposta da solução deste trabalho; **Análises Orientadas a Conformidade** – que apresenta alguns esforços que visam manter a rastreabilidade para as especificações públicas de algumas linguagens de desenvolvimento para as mais diversas finalidades e define uma alternativa de manutenção da rastreabilidade a ser aplicada neste trabalho; e **Representação Comportamental das Aplicações** – que analisa alguns trabalhos que visam fornecer representações comportamentais das aplicações com o objetivo de auxiliar algumas atividades do processo de desenvolvimento, tais como: documentação, manutenção, validação e verificação. Além disto, propõe uma alternativa de representação a ser utilizada pela proposta da solução;

- **Capítulo 3 – Proposta de Análises Estrutural e Comportamental Orientadas a Conformidade para o Desenvolvimento de Aplicações Multimídia.** Este capítulo tem por objetivo apresentar a proposta da solução, que visa aplicar uma abordagem orientada a conformidade para prover análises, estrutural e comportamental, no processo de desenvolvimento e manutenção das aplicações multimídia construídas a partir de linguagens declarativas padronizadas por especificações públicas.

- **Capítulo 4 – Implementação da Proposta.** Este capítulo tem por objetivo apresentar os detalhes acerca da implementação da proposta que foi utilizada para investigar a viabilidade técnica e funcional da abordagem. A proposta foi concretizada por meio da implementação de uma prova de conceito centrada na linguagem NCL. Desta forma, a solução apresentada buscou propor uma alternativa para atestar a qualidade das aplicações NCL permitindo a identificação de problemas estruturais (léxicos e sintáticos) e comportamentais (semânticos) de forma precisa e confiável, fundamentada pela rastreabilidade aos trechos normativos que especificam a linguagem de desenvolvimento. Além disto, evidencia a flexibilidade da

abordagem ao apresentar os estudos realizados na definição de duas novas soluções de análise, para as linguagens HTML e SVG, a partir da solução desenvolvida.

- **Capítulo 5 – Avaliação Experimental.** Este capítulo visa fornecer informações acerca das avaliações realizadas até o presente momento, que buscaram evidenciar principalmente a confiabilidade da representação comportamental provida pela solução. Sendo assim, apresenta detalhes do plano, da execução da análise estatística e dos resultados alcançados do experimento proposto;
- **Capítulo 6 – Trabalhos Relacionados.** Neste capítulo são apresentados os principais esforços que consolidaram contribuições relevantes para o contexto deste trabalho. Tais esforços buscam analisar o código fonte das aplicações multimídia com o objetivo de identificar problemas estruturais e comportamentais nessas aplicações. A análise desses trabalhos permitiu fundamentar a definição da solução proposta, evidenciando os requisitos aderentes, além de permitir a prevenção dos problemas já conhecidos. Desta forma, este capítulo busca apresentar as principais contribuições e limitações de cada trabalho relacionado; e
- **Capítulo 7 – Conclusão.** Este capítulo tem por objetivo apresentar as contribuições, limitações e conclusões deste trabalho, bem como as perspectivas de trabalhos futuros.

Fundamentação

Este capítulo tem por objetivo apresentar os principais conceitos que fundamentaram a proposta da solução. Sendo assim, busca descrever brevemente cada conceito, além de citar as motivações acerca das definições realizadas.

1.1 Linguagens Declarativas utilizadas no Desenvolvimento de Aplicações Multimídia

Linguagens declarativas normalmente são aplicadas no desenvolvimento de aplicações multimídia, pois suas características permitem suportar adequadamente a natureza dinâmica e descritiva dessas aplicações. Desta forma, buscam descrever como os objetos de mídia se relacionam entre si, com o espaço e o com o tempo a partir de uma programação de alto nível, permitindo que os desenvolvedores apenas descrevam "o que" deve ser computado, sem definir "como" isto vai ocorrer (LLOYD, 1994; BOSTOCK, 2010; SANTOS; BRAGA; MUCHALUAT-SAADE, 2013). As linguagens declarativas utilizadas para este fim possuem aspectos de controle próprios, pré-definidos por suas especificações. Como exemplo dessas linguagens pode-se destacar: NCL (ITU, 2009); SMIL (W3C, 2008); SVG (W3C, 2011); XHTML (W3C, 2010); e HTML5 (W3C, 2014). O Quadro 0.1 apresenta algumas características comuns a essas linguagens.

Quadro 0.1. Características das linguagens declarativas utilizadas no desenvolvimento de aplicações multimídia

Característica	NCL	SMIL	SVG 1.1	XHTML	HTML
Versão avaliada	3.0	3.0	1.1	2.0	5.0
Aplicabilidade	- TV Digital - IPTV	- Mobile - TV Digital - Web - Sistemas	- Mobile - Web - Sistemas de impressão	- Mobile - Web - Smart TV	- Mobile - Web - Smart TV

		embarcados	- Sistemas de design gráfico - Sistemas embarcados		
Especificação	Pública	Pública	Pública	Pública	Pública
Data de publicação	29/11/2014	01/12/2008	16/08/2011	16/12/2010	28/10/2014
Formato	*XML Schema*	*XML DTD*	*XML DTD*	*XML DTD*	*SGML*
Órgão Responsável	ABNT ITU	W3C	W3C	W3C	W3C
Define mecanismo oficial para avaliar a conformidade dos interpretadores?	NÃO	NÃO	NÃO	NÃO	NÃO
Define mecanismo oficial para avaliar a conformidade das aplicações?	NÃO	NÃO	NÃO	NÃO	NÃO
Requer robustez das aplicações?	SIM	SIM	SIM	SIM	SIM
Requer interoperabilidade das aplicações?	SIM	SIM	SIM	SIM	SIM
Avaliar as aplicações em um ambiente real é custoso para os desenvolvedores?	SIM	SIM	NÃO	NÃO	NÃO

Este trabalho utiliza a linguagem NCL para atestar a viabilidade da abordagem proposta, pois suas características a torna bastante representativa para o contexto desse trabalho:

- Especificada pelos padrões públicos: ABNT (2011) para aplicações do SBTVD; e ITU (2009) para aplicações de IPTV, que não são formatados adequadamente para o requisito de rastreabilidade. Esses padrões definem as regras da linguagem por meio de um texto corrido organizado apenas em seções e em documentos distintos, sem qualquer preocupação em estruturar e concentrar as informações relacionadas. Este formato tende a dificultar a compreensão das regras da linguagem, pois uma regra pode ser definida em várias sessões ou até em documentos diferentes (PROTA *et. al.*, 2014);
- Executada em diversas plataformas e dispositivos, que não apresentam garantias de conformidade com as especificações definidas (SANTOS; LENZ, 2014; PINHEIRO *et. al.*, 2012). Existem diversas implementações de interpretadores que buscam, sem sucesso, seguir as especificações definidas, dado que, frequentemente, os usuários identificam problemas na execução de seus aplicativos e comunicam ao SAC – Serviço

de Atendimento ao Consumidor – dos grandes fabricantes de TV e também das emissoras (SANTOS; FERRAZ; LENZ, 2013; SANTOS; LENZ, 2014);

- O processo de desenvolvimento é propenso a erros, a replicação de cópias de estruturas e o reuso de documentos preexistentes podem causar a criação de estruturas incompletas, incorretas ou inacessíveis (SANTOS, 2012). Isto, alinhado à ausência de mecanismos para atestarem a conformidade de seus conteúdos (SANTOS; LENZ, 2014), impactam a qualidade dos conteúdos produzidos; e
- Requer robustez e interoperabilidade dos aplicativos desenvolvidos, devido à sua aplicação crítica no processo de transição da TV analógica para a TV Digital de diversos países (SANTOS; LENZ, 2014). As aplicações desenvolvidas são componentes fundamentais para o sistema de TV Digital, pois sua execução pode apresentar problemas para o usuário, mesmo que toda a infraestrutura de transmissão e recepção esteja conforme as normas (SANTOS; LENZ, 2014).

1.2 Processo de Desenvolvimento Orientado a Análise

Esta seção apresenta as limitações do processo de desenvolvimento orientado à execução, normalmente utilizado no contexto das aplicações multimídia, a fim de confrontá-las com uma abordagem orientada a análise, que fundamentou a proposta deste trabalho.

O processo de desenvolvimento das aplicações multimídia não é trivial, primeiramente por conta da natureza dinâmica dessas aplicações, nas quais o programador não determina o fluxo de controle da aplicação, pois se limita a definir o sincronismo entre seus conteúdos. A sincronização representa um dos principais recursos na definição dessas aplicações, pois permite associar um conjunto de mídias com o tempo (contínua) e com as ações do usuário (discreta), construindo interfaces gráficas dinâmicas e consistentes (LI *et. al.*, 2014; HUANG *et. al.*, 2013). Essa característica potencializa a quantidade de interações possíveis, por permitir uma livre interação ao usuário, que, consequentemente, afeta o processo de teste na prevenção de comportamentos inesperados. Além disto, essas aplicações, normalmente, são independentes de plataformas e dispositivos, isto é, elas são construídas para serem interoperáveis, cabendo ao usuário definir onde elas devem executar. Isso permite que a aplicação seja executada em diversos contextos de uso, por uma vasta quantidade de plataformas e dispositivos, o que, consequentemente, também afeta o processo de testes na preservação dos requisitos. Neste contexto, a robustez e a interoperabilidade surgem como fatores

determinantes para a qualidade das aplicações, e, portanto, devem ser tratadas adequadamente pelo processo de desenvolvimento (SANTOS; LENZ, 2014; AMMONS *et. al.*, 2002).

Nesse cenário, é natural utilizar um processo de desenvolvimento orientado a execução, no qual os desenvolvedores simulam as interações manualmente através da execução em um interpretador específico, eleito para atestar a corretude das aplicações desenvolvidas (Figura 0.1). Por conseguinte, neste processo, os testes são limitados e, geralmente, insuficientes para garantir a qualidade da aplicação.

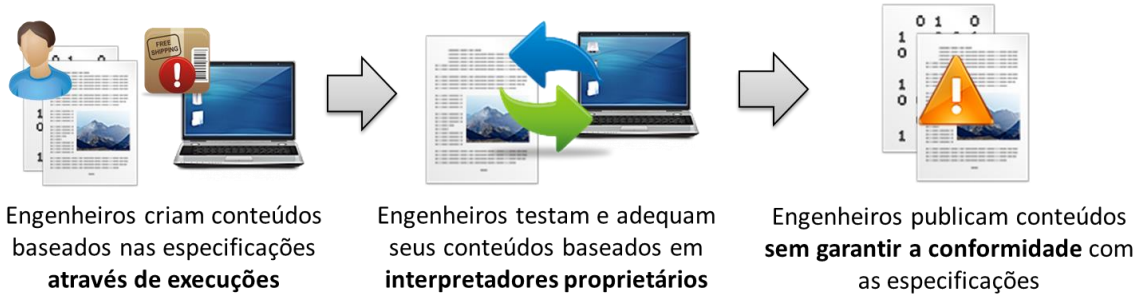

Figura 0.1. Processo de desenvolvimento orientado a execução.

Algumas características ajudam a justificar a imaturidade deste tipo de abordagem no desenvolvimento de aplicações multimídia, por exemplo:

- Os interpretadores usuais, normalmente, apresentam qualidade questionável, dado que não há garantias da sua conformidade perante as especificações (SANTOS; LENZ, 2014; PINHEIRO *et. al.*, 2012);
- Tende a diminuir a importância das especificações das linguagens, muitas vezes desprezadas durante a codificação, pois a corretude da aplicação é determinada pela execução em um interpretador específico (SANTOS; FERRAZ; LENZ, 2013; SANTOS; LENZ, 2014; AMMONS *et. al.*, 2002);
- Induz o desenvolvedor a inserir trechos desnecessários, geralmente conflitantes, para tratar as limitações dos interpretadores elegíveis (SANTOS; LENZ, 2014);
- Limita a simulação das interações do usuário no processo de teste, dado que a percepção das interações possíveis (que afetam o comportamento da aplicação) depende da usabilidade;

- Demanda inúmeras execuções para cobrir as interações possíveis, geralmente por meio de atividades manuais com elevado grau de repetição e esforço (AMMONS *et. al.*, 2002; SANTOS, 2012; EIDENBERGER, 2003);
- Força o desenvolvedor a realizar vários ciclos de correção (iterativos), dado que alguns erros são inalcançáveis por conta da existência de outros (predecessores); e
- Estimula o uso de várias versões de diversos interpretadores, por parte dos usuários finais, que geralmente são indispensáveis para executar alguns aplicativos (SANTOS; LENZ, 2014).

Apesar desta abordagem não ser indicada para o desenvolvimento das aplicações multimídia, ela possui algumas características interessantes que podem ser destacadas e devem ser consideradas:

- Estimula o processo de teste, dado que é mais natural e confortável testar a aplicação atuando como um usuário, simulando as interações diretamente na camada de apresentação (ZELLER, 2009);
- Facilita a verificação da ocorrência de um determinado comportamento, por meio do retorno visual (ZELLER, 2009);
- Permite identificar se um determinado problema está visível para o usuário, isto é, se ele afeta a camada de apresentação de alguma forma (ZELLER, 2009);
- Facilita a percepção dos requisitos funcionais, que geralmente são visivelmente rastreados; e
- Permite avaliar a usabilidade da aplicação durante a execução de testes funcionais.

Uma alternativa, a fim de amadurecer o processo de desenvolvimento, é eliminar a dependência de interpretadores questionáveis para atestar a corretude das aplicações desenvolvidas. As abordagens que utilizam ferramentas ou modelos que intervenham na execução, fornecendo análises que evidenciem os problemas, são fortemente indicadas (SANTOS, 2012; AMMONS *et. al.*, 2002). Essas abordagens orientadas a análises buscam suprir as limitações da execução, geralmente fornecem funcionalidades que permitem identificar tanto os problemas estruturais (léxica e sintática) quanto os problemas comportamentais. Além disto, também fornecem recursos para avaliar o software sob diversas perspectivas, os quais permitem identificar, por exemplo: a dependência entre estruturas, as cópias de código, a máquina de estados da aplicação, a utilização da memória, organização do layout visual, entre outras informações (GLEIRSCHER *et. al.*, 2013). Essa abordagem pode ser aplicada com baixo custo e permite potencializar detecção de defeitos críticos

para a qualidade das aplicações (GLEIRSCHER *et. al.*, 2013). As soluções buscam prover suas análises por meio de métodos distintos, entre eles pode-se destacar: a análise estática automática; a análise dinâmica automática; e a análise a partir de logs (SCHUR *et. al.*, 2013; BELLUCCI *et. al.*, 2012; KRKA *et. al.*, 2010; MARIANI *et. al.*, 2011).

Algumas características potencializam o uso da abordagem orientada a análise no processo de desenvolvimento de aplicações multimídia, pois busca prover um mecanismo, independente de interpretadores, que permite automatizar a identificação de problemas, por meio da busca de padrões de inconsistências, da simulação automática das interações possíveis e da extração de modelos que facilitam a verificação (GLEIRSCHER *et. al.*, 2013). Desta forma, otimiza a identificação de problemas, pois todos os trechos podem ser avaliados independentemente de estarem graficamente visíveis ou funcionalmente acessíveis. Essa característica também contribui para a cobertura dos testes, por meio da automação das validações e verificações (AMMONS *et. al.*, 2002; SANTOS, 2012; EIDENBERGER, 2003). Além disto, também é possível prover acesso às informações das especificações, para justificar as análises e complementar o entendimento do problema. Porém, é importante destacar algumas preocupações que precisam ser tratadas na definição de uma solução de análise para o contexto das aplicações multimídia (PROTA *et. al.*, 2014):

- **Completude**: diz respeito à necessidade de suportar todas as regras normativas existentes, pois é imprescindível identificar todos os problemas existentes em uma análise estrutural, além de ser essencial para a realização de uma análise comportamental, a fim de evitar a identificação de falsos comportamentos motivada pelo desconhecimento de alguma regra;
- **Confiabilidade**: para que a utilização da ferramenta seja produtiva, suas análises precisam ser confiáveis, evitando que o desenvolvedor precise recorrer à especificação para confirmar a existência de algum problema ou de algum comportamento evidenciado;
- **Desempenho**: nesse contexto de desenvolvimento o desempenho é impactado por dois fatores: a velocidade na realização das análises e o tempo de resposta no seu acesso durante o processo de desenvolvimento. Em um cenário de desenvolvimento ágil, a adoção de novas ferramentas é bastante crítica, pois o custo da sua utilização pode inviabilizar seu uso; e
- **Usabilidade**: é essencial a solução atingir um grau de usabilidade aceitável, pois ela pode gerar uma elevada quantidade de informações, oriundas das análises, inviabilizando seu consumo. O perfil do usuário (desenvolvedor, testador, entre outros.) é muito importante para este contexto, pois ele pode determinar a relevância e o nível de detalhamento das

informações, contribuindo para a diminuição da curva de aprendizado, além de evidenciar a agregação de valor com o menor esforço de interação.

Este trabalho utilizou a abordagem orientada a análises para definir sua solução (Figura 0.2). Desta forma, buscou prover uma alternativa para atestar a qualidade das aplicações multimídia sem depender da execução em interpretadores de terceiros. Para isto, utilizou os métodos baseados na análise estática e dinâmica, a fim de, respectivamente, identificar os problemas estruturais (léxico, sintático e semântico estático) automaticamente e prover facilidades para a identificação de problemas comportamentais.

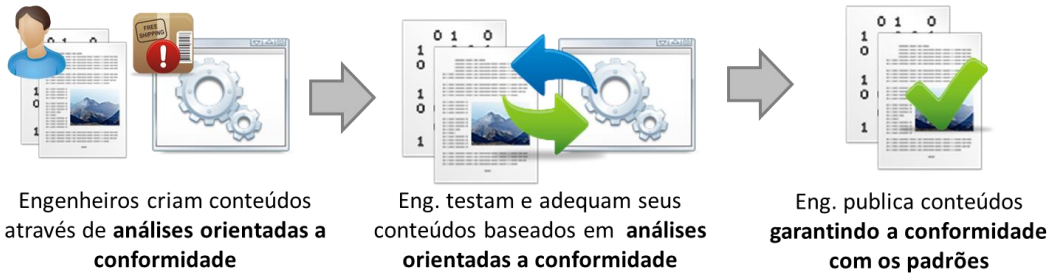

Figura 0.2. Processo de desenvolvimento orientado a análises.

1.3 Análises Orientadas a Conformidade

Esta seção tem por objetivo apresentar alguns esforços que visam manter a rastreabilidade para as especificações públicas de algumas linguagens de desenvolvimento para as mais diversas finalidades e define uma alternativa de manutenção da rastreabilidade que fundamentou a proposta deste trabalho.

Em um processo de identificação de problemas, que busca evidenciar infrações e interpretações incorretas das especificações, faz-se necessário apresentar seus resultados de forma precisa e confiável, para que o usuário evite recorrer às especificações para questionar os resultados ou para complementar o entendimento. Uma alternativa para atender a esses requisitos é utilizar uma abordagem orientada a conformidade, que busca referenciar cada trecho normativo das especificações para justificar seus resultados. Desta forma, além de assegurar os resultados e complementar o entendimento, esta abordagem tem como efeitos adicionais: a diminuição da curva de aprendizado, por facilitar o acesso às informações que especificam a linguagem; e o aumento da produtividade durante o processo de identificação e correção de problemas, por evidenciar os

problemas automaticamente e por fornecer insumos para a correção por meio do acesso às informações relevantes.

O contexto deste trabalho, em que as aplicações multimídia são construídas a partir de linguagens declarativas padronizadas por especificações públicas, traz algumas limitações para a aplicação dessa abordagem, uma vez que, normalmente, as especificações públicas são escritas em linguagem natural e não apresentam as regras em um formato adequado, podendo definir uma regra em várias sessões ou até em documentos diferentes (PROTA *et. al.*, 2014). Desta forma, as especificações em seu formato original podem fornecer informações inconsistentes (Ambíguas, Conflitantes ou Incompletas) ou desnecessárias (Contextuais ou Repetidas) ao longo de seu conteúdo. Porém, para aplicar a abordagem orientada a conformidade é indicado que as regras que especificam a linguagem estejam estruturadas de forma concisa e consistente, a fim de facilitar: a percepção da completude da regra; o escopo da regra; a identificação de regras dependentes; a criticidade da regra (obrigação ou recomendação); entre outras informações relevantes. Sendo assim, é importante definir uma proposta para estruturar os padrões públicos. Na literatura foram encontrados alguns esforços que buscam estruturar especificações descritivas para objetivos diversos, tais como ITU-T HSTP.CONF-H761 (2012) Pinheiro *et. al.* (2012), W3C (2002) e Bossi e Gaggi (2007).

A ITU-T HSTP.CONF-H761 (2012) define uma especificação padrão, orientada a assertivas, para realização de testes de conformidade no contexto do Ginga-NCL. Esta especificação é organizada em três categorias: assertivas, instruções de testes e casos de testes. No contexto deste trabalho as assertivas são estruturas que consolidam algumas informações relevantes da especificação. Elas são compostas por um identificador, por uma descrição, pelas referências e pelo alvo (componente NCL). Pinheiro *et al* (2012) especificam um conjunto contendo 253 assertivas referentes a ITU-T (ITU-T HSTP.CONF-H761, 2012), utilizando as assertivas da especificação padrão para criação de seus casos de testes. Ainda com o objetivo de especificar testes, O W3C (2002) define um processo para escrita de suítes de testes para o HTML, para isto também utiliza o paradigma orientado a assertivas. Bossi e Gaggi (2007) propõem a utilização de assertivas no processo de validação semântica de aplicações SMIL, definindo as entradas e saídas das regras de validação devido à possibilidade de se expressar as propriedades temporais. Todos os trabalhos avaliados fazem uso de assertivas para formatar regras descritivas, seja para definir a cobertura de suítes de testes (ITU-T HSTP.CONF-H761, 2012; PINHEIRO *et. al.*, 2012; W3C, 2002) ou justificar uma transformação em um processo de validação semântica (BOSSI; GAGGI, 2007).

No processo de análises orientadas a conformidade, a utilização de assertivas é indicada para a estruturação de especificações descritivas, pois representam unidades de funcionalidade ou comportamento extraídas das proposições normativas e constituem uma especificação modular, estruturada e completa. Neste paradigma, a primeira etapa é caracterizada pela identificação de assertivas testáveis, agrupadas por categorias que auxiliem a sua organização, tais como: seção, subseção, funcionalidades e escopo. Dado que as assertivas representam unidades de funcionalidade, a sua utilização pelas ferramentas de análise também permite avaliar a representatividade da solução, pois ao relacionar as regras implementadas com as assertivas existentes pode-se determinar o grau de completude da solução, dado que para a solução ser considerada completa todas as assertivas devem estar relacionadas com alguma regra implementada pela ferramenta. A abordagem orientada a assertiva também agrega valor à confiabilidade da solução, pois assim como W3C (2002), ITU-T HSTP.CONF-H761 (2012) e Pinheiro *et. al.* (2012) utilizaram esta abordagem para testar a conformidade das implementações, pode-se utilizar as assertivas para testar a conformidade das análises da solução. Apesar de não ser essencial, é importante apresentar os trechos descritivos da especificação que estão referenciados pelas assertivas, para permitir que os usuários possam confrontar os resultados da análise com as proposições normativas originais.

Este trabalho utilizou o conceito de **assertiva** para prover análise orientadas a conformidade. Desta forma, elas foram utilizadas para prover a rastreabilidade aos trechos normativos, onde cada regra é associada unicamente a uma assertiva, tendo como responsabilidade identificar a conformidade da aplicação (estrutural) e inferir comportamentos (comportamental) de acordo com a sua assertiva. Este trabalho utilizou as 253 assertivas identificadas pelo ITU-T (2015) como ponto de partida na implementação da prova de conceito. Elas originalmente são compostas por um identificador, por uma descrição textual, pelas referências à norma e pelo alvo (componente NCL). Para o modelo proposto por este trabalho ainda foi sugerido incluir a área funcional (existente na especificação), uma classificação da finalidade (Estrutural, Relacional, Interativa e Descritiva) e uma descrição assertiva, a fim de facilitar a organização das 253 assertivas e prover informações concisas e relevantes aos desenvolvedores. Foi proposta a classificação das assertivas de acordo com a sua finalidade, de forma a agrupá-las em quatro categorias:

- **Estruturais** – definem as regras léxicas e sintáticas da linguagem, tratando basicamente da gramática. Ex.: O elemento <media> deve possuir um atributo 'id';
- **Relacionais** – buscam descrever as regras que relacionam as estruturas entre si, que existem apenas para prover facilidades para a codificação, tais como: importação,

referência, contexto, hierarquia e reuso. Esses tipos de estruturas são conhecidos como o "açúcar sintático" da linguagem. Ex.: O atributo 'region' do elemento <descriptor> deve referenciar o atributo 'id' de um elemento <region>;

- **Interativas** – buscam definir as regras que determinam o funcionamento da máquina de estados da linguagem. Ex.: Quando a aplicação inicia, os nós de mídia referenciados pelo atributo 'component' dos elementos <port> devem ser iniciados; e
- **Descritivas** – definem o que deve ser apresentado pelos interpretadores, dados determinados estados da aplicação. Ex.: Regiões com o atributo 'zIndex' maior devem ser sobrepostas às regiões de 'zIndex' menor.

Para garantir a consistência quantitativa e qualitativa das regras implementadas com as 253 assertivas identificadas pelo ITU-T foi necessário aumentar a granularidade das assertivas, para que cada assertiva pudesse representar uma única regra da especificação. A Figura 0.3 exemplifica a relação entre a especificação, as referências, as regras e as assertivas. Essa reestruturação foi necessária, pois alguns casos possuíam vários alvos para uma mesma assertiva, por exemplo: a assertiva "region_10" define o comportamento de todos os atributos do elemento <region>. Outras consolidavam asserções distintas independentes em uma mesma assertiva, por exemplo: a assertiva "bind_03" define as regras relacionais e de comportamento para o atributo 'descriptor'. Com isso, foi necessário redefinir as assertivas, em duas ou mais, de acordo com a quantidade de alvos e de asserções independentes.

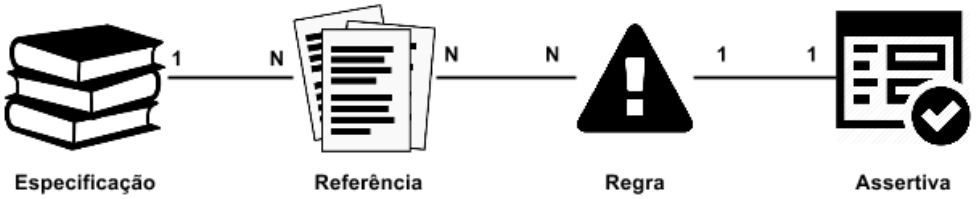

Figura 0.3. Relação entre Especificação, Referência, Regra e Assertiva.

1.4 Representação Comportamental das Aplicações

Esta seção tem por objetivo analisar alguns trabalhos que visam fornecer representações comportamentais das aplicações com o objetivo de auxiliar algumas atividades do processo de desenvolvimento, tais como: documentação, manutenção, validação e verificação. Além disto, propõe uma alternativa de representação que fundamentou a proposta deste trabalho.

As aplicações multimídia podem ser classificadas como **sistemas dirigidos a eventos**, nas quais a aplicação fica inoperante enquanto espera por um evento, e quando um evento ocorre, a aplicação reage e volta para o estado de espera (WAGNER *et. al.*, 2006). Comumente, algumas soluções buscam prover abstrações para representar o comportamento desse tipo de aplicação com o objetivo de auxiliar as atividades de documentação, manutenção, validação e verificação do software. Por exemplo, essas abstrações podem ser aplicadas em testes de regressão, na identificação de anomalias comportamentais, na geração de casos de testes automáticos, na representação da compreensão do software, na compactação das possibilidades de interação, entre outros (WALKINSHAW *et. al.*, 2013).

Existem diversos esforços que buscam definir abstrações adequadas para representar o comportamento de acordo com sua finalidade, cada qual com sua limitação e complexidade (ROBILLARD *et. al.*, 2012; OLIVEIRA *et. al.*, 2001; CHENG; KRISHNAKUMAR, 1993; WALKINSHAW *et. al.*, 2013; PICININ *et. al.*, 2012; LORENZOLI *et. al.*, 2008). Por exemplo, existem modelos que buscam evidenciar o fluxo de controle, enquanto outros buscam evidenciar o comportamento dos dados (entrada e saída) (WALKINSHAW *et. al.*, 2013). Entretanto, existem modelos que visam fornecer uma visão mais completa do comportamento do software, evidenciando tanto o fluxo de controle quanto os dados de entrada e saída, e que, dependendo da complexidade do software, podem apresentar modelos ilegíveis ou até computacionalmente inviáveis ("explosão de estados") (WAGNER *et. al.*, 2006).

Um dos objetivos deste trabalho é prover facilidades no processo de identificação de problemas comportamentais. Desta forma foi necessário categorizar os tipos de comportamentos que podem ocorrer nessas aplicações a fim de delimitar o escopo da solução. Assim como as assertivas os descrevem, os comportamentos também podem ser classificados em:

- **Interativo** – Relacionado com o modelo de interação da aplicação, no qual o comportamento é originado a partir de uma interação. O **Comportamento Interativo** é originado a partir da interação do usuário com a máquina de estados da aplicação. Desta forma, para identificar a existência de inconsistências desse tipo de comportamento é necessário avaliar a máquina de estado associada. Por exemplo: a interação em uma determinada aplicação pode conduzir o usuário a um estado intermediário que não possui transição de saída (Figura 0.4); e

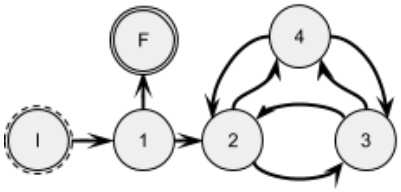

Figura 0.4. Exemplo de Inconsistência Interativa.

- **Descritivo** – Relacionado com a camada de apresentação, no qual o comportamento é originado a partir do layout visual da aplicação e dos dados correntes. O **Comportamento Descritivo** é originado a partir da interpretação das variáveis de domínio em um determinado momento (estado). Desta forma, para identificar existência de inconsistências desse tipo de comportamento é necessário avaliar os estados de forma isolada. Por exemplo: em um determinado momento, a aplicação apresenta uma imagem sobrepondo totalmente outra imagem ou fora dos limites da interface gráfica (Figura 0.5).

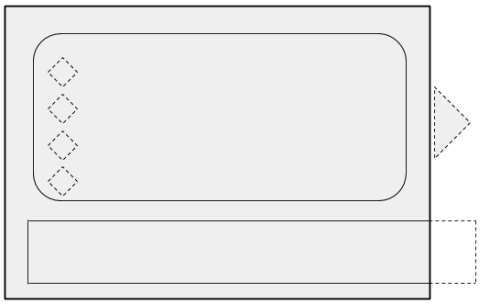

Figura 0.5. Exemplo de Inconsistência Descritiva.

Além disto, também é possível classificar os comportamentos de acordo com sua origem:

- **Padrão** – Relacionado com as recomendações da linguagem ou com as limitações de hardware (própria da linguagem). A classificação do comportamento entre 'esperado' e 'inesperado' depende dessas recomendações e limitações. Sendo assim, o **Comportamento Padrão** pode ser avaliado em todas as aplicações, indiscriminadamente. Desta forma, é possível identificar automaticamente as inconsistências comportamentais existentes (**identificação passiva**). Por exemplo: é esperado que as aplicações possuam um estado final, no qual todas as mídias devem estar paradas (Figura 0.6); e

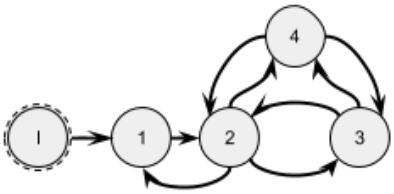

Figura 0.6. Exemplo de Inconsistência Padrão.

- **Próprio** – Relacionado com os requisitos funcionais das aplicações. A classificação do comportamento entre 'esperado' e 'inesperado' depende totalmente dos requisitos da aplicação. Sendo assim, para avaliar o **Comportamento Próprio** é necessário ter conhecimento dos requisitos da aplicação. Desta forma, é importante fornecer representações que permitam ao usuário identificar as inconsistências existentes (**identificação ativa**). Por exemplo: É esperado em um aplicativo específico, que o pressionamento de uma determinada tecla direcione o usuário para uma nova interface gráfica (Figura 0.7).

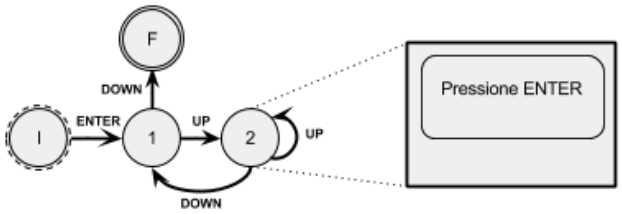

Figura 0.7. Exemplo de Inconsistência Própria.

Desta forma, definiu-se que a abstração proposta para representar os comportamentos das aplicações para o contexto deste trabalho deve prover recursos que permitam identificar tanto as inconsistências comportamentais de acordo com a natureza (**Interativa** e **Descritiva)**, quando as de acordo com a origem (**Padrão** e **Própria**). Com isso, o modelo comportamental abstrato precisa evidenciar o fluxo de controle e também os dados de entrada e saída. Além disto, é essencial que a solução atinja um grau de usabilidade aceitável, pois ela pode gerar uma elevada quantidade de informações, oriundas da complexidade das aplicações, inviabilizando seu consumo. Com isso, a abstração deve permitir a execução de estratégias de redução configuráveis, a fim de omitir ou detalhar comportamentos de acordo com o propósito da análise. Desta forma, o usuário pode determinar a relevância e o nível de detalhamento das informações, contribuindo para a diminuição da curva de aprendizado, além de evidenciar a agregação de valor com o menor esforço de interação.

Uma alternativa para a representação comportamental das aplicações neste contexto é a utilização do modelo definido como **Máquina de Estados Finita Estendida** (EFSM - *Extended Finite State Machine*) (CHENG; KRISHNAKUMAR, 1993), pois, além de serem indicados para este fim (WALKINSHAW *et. al.*, 2013; OLIVEIRA *et. al.*, 2001; PICININ *et. al.*, 2012), permitem evidenciar o fluxo de controle e os dados de entrada e saída e também possibilitam a utilização de técnicas de redução para ajustar o detalhamento das transições comportamentais (PICININ *et. al.*, 2012). Segundo Wagner (2006) o comportamento do software é definido pela história das entradas e não apenas pelas entradas possíveis, e por isso, diferentemente das Máquinas de Estado Finitas convencionais, as EFSM mantêm a história dos dados em memória (LORENZOLI *et. al.*, 2008; WALKINSHAW *et. al.*, 2013; CHENG; KRISHNAKUMAR, 1993). Segundo Cheng e Krishnakumar (1993), formalmente uma Máquina de Estados Finita Estendida E é definida por $E = (S, I, O, D, F, U, T)$, onde :

- S representa o conjunto dos estados;
- I representa o conjunto das entradas;
- O representa o conjunto das saídas;
- D representa o conjunto do domínio das variáveis;
- F representa o conjunto das funções de permissão, tal que, $p: D \rightarrow \{0,1\}$;
- U representa o conjunto das funções de transformação, tal que, $t: D \rightarrow D'$; e
- T representa a relação de transição $T: SxFxI \rightarrow SxUxO$.

Algumas características e limitações da análise comportamental proposta por este trabalho motivaram algumas alterações na EFSM. Por exemplo, diferentemente da EFSM, as operações de entrada e saída não são independentes, isto quer dizer que em um dado estado uma determinada entrada sempre irá produzir uma mesma saída. Desta forma, o modelo proposto buscou encapsular essas operações em uma única estrutura denominada Evento. Outra convergência decorrente da definição do Evento foi a associação das funções de permissão e transformação com as entradas (evento passivo) e saídas (evento ativo), respectivamente. Determinando quando uma transição deve ocorrer (função de permissão) e realizando as transformações necessárias no domínio das variáveis (função de transformação). Além disto, as aplicações desse contexto apresentam apenas um estado inicial e um estado final. No estado inicial todas as mídias estão paradas existindo apenas uma transição de inicialização, enquanto que no estado final todas as mídias estão paradas e não há nenhuma transição partindo dele. Desta forma, foi definido um

modelo de representação denominado **Modelo Comportamental**, onde um Modelo Comportamental B (*Behavior*) é definido por $B = (S, E, D, T)$, tal que :

- S representa o conjunto dos estados. Um estado $s \in S$ é representado por $s = (d', T')$, onde: $d' \subset D$ representa os modelos descritivos resultantes das transformações que tem como destino s; e $T' \subset T$ representa o conjunto de todas as transições ocorridas em s. $s_0 \in S$, é o estado inicial e $s_f \in S$, é o estado final;
- E é definido como o conjunto de eventos possíveis (de qualquer natureza). Um $e \in E$ é representado por $e = (Ec, Ea)$, onde:
    - Ep representa o conjunto de eventos condicionais, isto é, um $ec \in Ec$ é responsável por determinar a ocorrência do evento no domínio de variáveis $d \in D$, por meio de uma função de permissão $p: D \rightarrow \{0,1\}$. Cada evento condicional possui um classificador que define o seu tipo (Ex.: Temporal); e
    - Ea representa o conjunto de eventos acionáveis, isto é, um $ea \in Ea$ é responsável por realizar transformações no domínio de variáveis $d \in D$, por meio de uma função de transformação $t: D \rightarrow D'$. Cada evento acionável possui um classificador que define o seu tipo (Ex.: Atribuição);
- D representa o conjunto do Domínio das Variáveis, responsável por manipular os dados de entrada e saída. O Domínio das Variáveis $d \in D$, denominado neste trabalho como **Modelo Descritivo**, é representado por $d = (N, n_0)$, onde: D representa o conjunto de todas as variações possíveis do Modelo Descritivo Inicial $d_0 \in D$; N representa o conjunto de nós existentes representados por (N', A, P). $N' \subset N$ representa os nós filhos de um determinado nó. A representa o conjunto de âncoras existentes, onde cada $a \in A$ representa uma interface para um $n \in N'$ ou para um $p \in P$; P representa o conjunto de propriedades descritivas existentes, onde cada $p \in P$ possui um valor associado determinante para a interpretação descritiva; e $n_0 \in N$ representa o nó inicial;
- T é o conjunto de transições existentes. Uma Transição $t \in T$ é representada por t (s_o, E', s_d), onde: $s_o \in S$ representa o estado origem de uma determinada transição; $E' \subset E$ representa o conjunto dos eventos ocorridos; e $s_d \in S$ representa o estado destino de uma determinada transição.

Com o objetivo de endereçar o problema da "explosão de estados", este trabalho propôs a identificação de estados equivalentes e a utilização de alguns tipos de reduções a serem aplicadas no modelo, a fim de omitir ou detalhar comportamentos de acordo com o propósito da análise:

- **Equivalência de Estados –** Para o modelo comportamental proposto buscou-se definir como estados equivalentes os estados que permitem exatamente as mesmas interações (eventos), independentemente do seu modelo descritivo. Desta forma, um dado estado $s_1 = (d_1, T_1)$ é considerado equivalente a um determinado estado $s_2 = (d_2, T_2)$ se, e somente se, o conjunto de todos os eventos condicionais referentes a T_1 for igual ao conjunto de todos os eventos condicionais referentes a T_2. Na Figura 0.8 **S3** e **S4** são equivalentes, pois $T_{s3} = T_{s4} = \{E4, E5\}$;

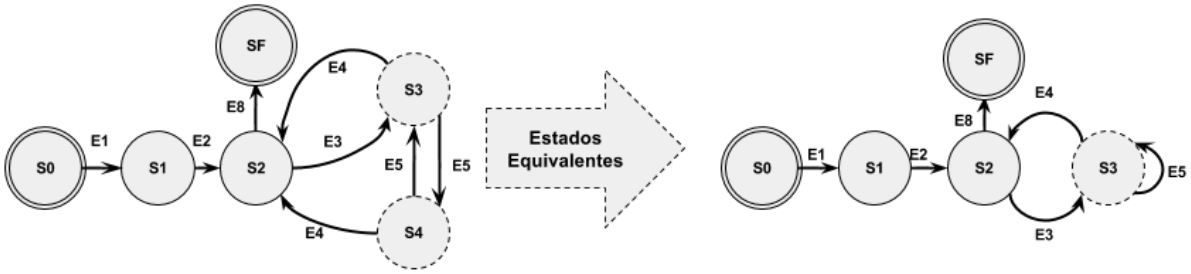

Figura 0.8.Exemplo de Equivalência de Estados.

- **Redução de Transição Direta** – Busca eliminar transições diretas existentes no modelo. Uma transição direta é definida por duas transições subsequentes intermediadas por um estado que não possui outras transições. Desta forma, dadas duas transições $t_1 = (s_{o1}, E_1, s_a)$ e $t_2 = (s_a, E_2, s_{d2})$ elas são consideradas diretas no estado $s_a = (d_a, T_a)$ se, e somente se, $T_a = \{t_2\}$ e $\exists!\ T_n \mid t_1 \in T_n$. Com isso, a fim de reduzir o modelo comportamental, é possível aplicar uma função de redução definida por $\delta(t_a, t_b) \rightarrow t' \Rightarrow \{t_a, t_b \in T \mid t_a = (s_{oa}, E_a, s_x),\ t_b = (s_x, E_b, s_{db}) \rightarrow t' \in T \mid t' = (s_{oa}, E', s_{db}),\ E' = E_a \cup E_b\}$. A Figura 0.9 exemplifica a redução direta, onde: $t_1, t_2 \in T \mid t_1 = (S0, E1, S1),\ t_2 = (S1, E2, S2) \rightarrow t' \in T \mid t' = (S0, E1E2, S2)$; e

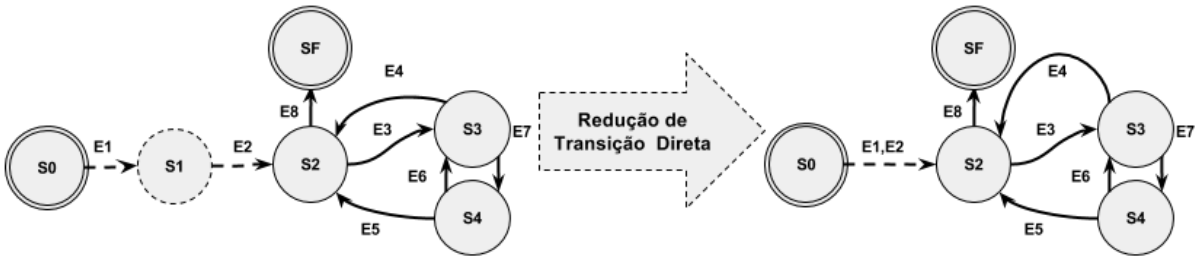

Figura 0.9.Exemplo de Redução de Transição Direta

- **Redução Parametrizada** – Busca eliminar eventos irrelevantes para o usuário. Este tipo de redução permite que o usuário possa selecionar quais tipos de eventos condicionais ele deseja visualizar, para que o algoritmo de redução elimine as transições constituídas integralmente por eventos condicionais cujos tipos não foram selecionados pelo usuário. Desta forma, dado o conjunto dos eventos visíveis $E_v \subset E$ é aplicada uma função de redução a fim de eliminar todas as transições indesejadas definidas por $T_i = \{t \in T, t = (s_o, E', s_d) \mid e \in E' \wedge e \notin E_v\}$. A Figura 0.10 exemplifica a redução parametrizada a fim de eliminar o evento $E3$.

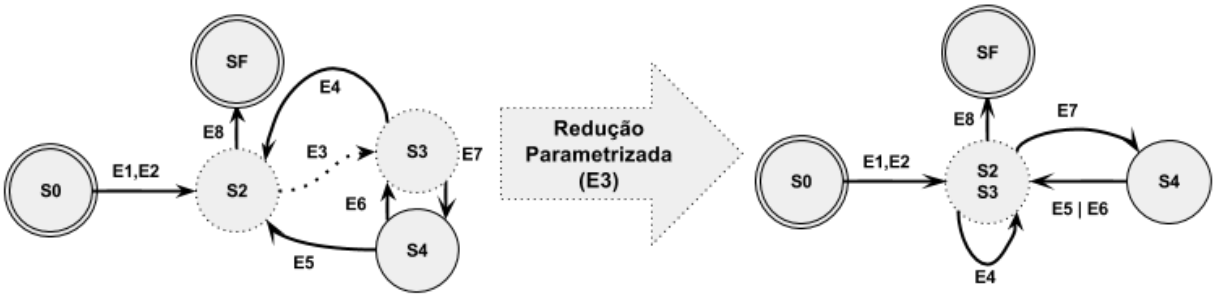

Figura 0.10. Exemplo de Redução Parametrizada

Vale salientar que outras estratégias de redução, que busquem controlar a apresentação dos comportamentos, também podem ser incorporadas à abordagem proposta, desde que elas não impactem na consistência dos modelos.

1.5 Considerações Finais

Este capítulo apresentou os principais conceitos que fundamentaram a proposta de solução deste trabalho. Inicialmente foram apresentadas as características comuns às **Linguagens Declarativas usadas no Desenvolvimento de Aplicações Multimídia**. Em seguida, buscou-se

descrever detalhes da definição da solução relacionados aos conceitos apresentados. Com relação ao **Processo de Desenvolvimento Orientado a Análises**, foi proposta a utilização dos métodos baseados na análise estática e dinâmica, a fim de, respectivamente, identificar os problemas estruturais (léxico, sintático e semântico estático) automaticamente e prover facilidades para a identificação de problemas comportamentais. As **Análises Orientadas a Conformidade** sugeriram utilizar o conceito de assertiva para evidenciar a conformidade das aplicações e prover a rastreabilidade das análises aos trechos normativos das especificações. Enquanto que, as **Representações Comportamentais das Aplicações** evidenciaram a utilização da Máquina de Estados Finitos Estendida (EFSM - *Extended Finite State Machine*) na definição do Modelo Comportamental.

O próximo capítulo tem como objetivo apresentar a proposta da solução, que visa aplicar uma abordagem orientada a conformidade para prover análises, estrutural e comportamental, no processo de desenvolvimento e manutenção das aplicações multimídia construídas a partir de linguagens declarativas padronizadas por especificações públicas.

Proposta de Análises Estrutural e Comportamental Orientadas a Conformidade para o Desenvolvimento de Aplicações Multimídia

A abordagem proposta por este trabalho foi fundamentada na utilização de dois componentes responsáveis por fornecer funcionalidades de análise estrutural e comportamental, para permitir a identificação de problemas sob essas duas perspectivas. Porém, não é possível generalizar todo o processo de análise, é necessário considerar as especificidades da linguagem para prover uma solução de análise aderente ao contexto do desenvolvimento. A Figura 0.1 apresenta uma visão geral da arquitetura abstrata da solução (A) e a visão geral da arquitetura concreta de duas ferramentas com exemplos de instanciação: uma destinada à análise de aplicações multimídia NCL (B) e outra de aplicações multimídia HTML (C). Sendo assim, cabe a cada componente fornecer todo o conhecimento necessário para a realização da análise estrutural e comportamental de acordo com sua linguagem de domínio, enquanto que o componente Core tem por objetivo definir as interfaces de comunicação e padronizar os resultados das análises.

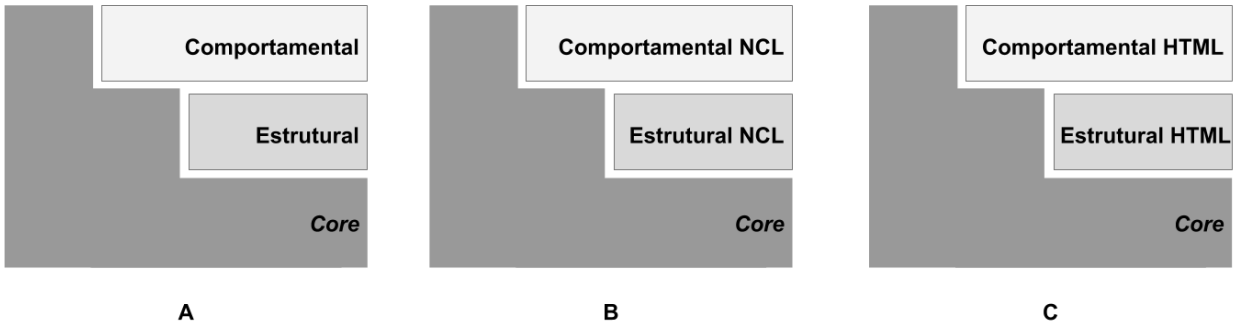

Figura 0.1. (A) Arquitetura abstrata da solução; (B) arquitetura concreta da ferramenta de análise para aplicações NCL; e (C) arquitetura concreta da ferramenta de análise para aplicações HTML.

De forma geral, independente da perspectiva de análise, definiu-se que a orientação a conformidade fosse baseada no conceito de **assertivas**, pois elas fornecem uma alternativa para estruturar as regras das especificações mantendo a rastreabilidade para trechos normativos. Desta forma, podem ser usadas para fundamentar a construção dos modelos propostos, que fornecem recursos para a identificação dos problemas estruturais e comportamentais. Na prática, o processo de análise busca realizar transformações estáticas e dinâmicas a partir do código fonte da aplicação para construir modelos que permitam a identificação dos problemas. Cada modelo representa a interpretação e reestruturação da aplicação sob alguma perspectiva. Para a análise estrutural, o **Modelo Estrutural** permite identificar os problemas de natureza estática (sintático e estrutural). Para a análise comportamental, o **Modelo de Relacionamentos** promove a simplificação da estrutura da aplicação ao eliminar o "açúcar sintático", a fim de simplificar a aplicação analisada para facilitar as futuras traduções, o **Modelo Comportamental** permite identificar problemas comportamentais de natureza interativa e o **Modelo Descritivo** permite identificar problemas comportamentais de natureza descritiva. Para cada assertiva foram definidos identificadores únicos (denominados ***Tokens*)** que são utilizados no processo de tradução dos modelos, onde cada etapa de tradução precisa referenciar explicitamente os *Tokens* utilizados no processo, provendo, consequentemente, a rastreabilidade para as assertivas. Para fins de organização, tal como as assertivas (conforme apresentado no Capítulo 2), os *Tokens* foram classificados em Estruturais, Relacionais, Comportamentais e Descritivos. Os tipos das assertivas identificadas pelos *Tokens* precisam estar consistentes com a classificação dos *Tokens*. A Figura 0.2 apresenta a relação entre a especificação da linguagem (ex.: Especificação da linguagem NCL (ITU, 2009)), as assertivas, os *Tokens* e os modelos. Mais detalhes acerca da construção e utilização desses modelos serão apresentados nas próximas seções.

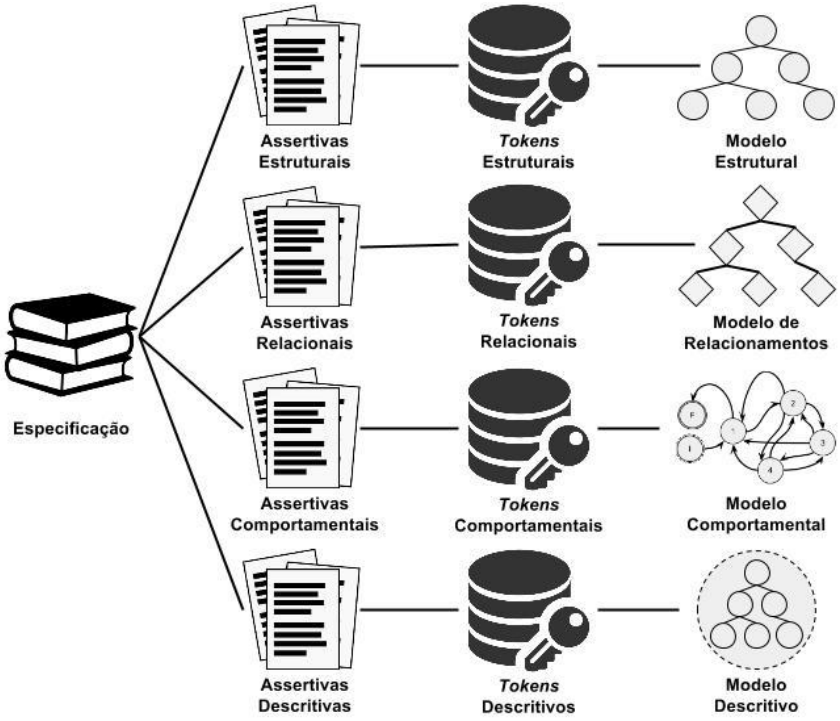

Figura 0.2. Relacionamento entre especificação, assertivas, *Tokens* e modelos.

As seções a seguir detalham como a proposta de análise orientada a conformidade foi aplicada nas análises estruturais, que buscam identificar os problemas sintáticos e estruturais, e nas análises comportamentais, que provêm facilidades para identificar a existência de problemas comportamentais. Desta forma, buscam apresentar os problemas e os desafios existentes e o modelo conceitual proposto, inerentes a cada tipo de análise.

1.1 Análise Estrutural

Esta seção apresenta a proposta da solução referente à análise estrutural, descrevendo a abordagem sob a perspectiva conceitual e técnica.

Problemas estruturais são mais fáceis de serem identificados, geralmente as ferramentas de autoria fornecem esta funcionalidade de alguma forma, limitando-se a evidenciar erros, léxicos e sintáticos (SANTOS; LENZ, 2014). O **Modelo Estrutural** proposto para este tipo de análise buscou atuar na identificação de problemas de origem:

- **Léxica** – Referente à estrutura léxica, geralmente definida pelo XML Schema ou DTD da linguagem;
- **Sintática** – Referente à hierarquia, obrigatoriedade e cardinalidade dos elementos e atributos;

- **Referencial** – Referente às relações de referência entre componentes da linguagem; e
- **Contextual** – Referente às relações de escopo entre os elementos.

Apesar das ferramentas de autoria normalmente proverem funcionalidades para a identificação de problemas estruturais (AZEVEDO *et. al.*, 2009; HONORATO; BARBOSA, 2010), geralmente não garantem a conformidade com as especificações, nem fornecem facilidades para que o desenvolvedor possa atestar a qualidade desta identificação. Neste cenário, basta que o desenvolvedor encontre um falso positivo ou um falso negativo para questionar todos os resultados. Entretanto, quando a ferramenta de autoria não evidencia erros estruturais, cabe ao interpretador evidenciar estes problemas. Este processo é ainda mais traumático para o desenvolvimento, dado que, além de não estarem certificados perante as especificações, os interpretadores também forçam o desenvolvedor a realizar mais ciclos de correções (iterativos), já que alguns erros são inalcançáveis por conta da existência de outros erros predecessores (ex.: uma mídia sobrepõe a indicação de uma interação possível).

A fim de prover uma alternativa apropriada para este contexto é preciso prover soluções que busquem endereçar adequadamente essas limitações do processo de desenvolvimento usual. Como dito anteriormente no Capítulo 2, a abordagem de desenvolvimento centrada na análise orientada a conformidade é indicada para este fim, principalmente, para auxiliar a identificação de problemas estruturais, pois essa abordagem pode ser aplicada com baixo custo, dada a natureza estática da análise estrutural, e permite potencializar a detecção de defeitos críticos para a qualidade das aplicações (GLEIRSCHER *et. al.*, 2013). O modelo proposto para a análise estrutural de aplicações utilizou assertivas como estrutura intermediária para manter a rastreabilidade entre as regras que avaliam as estruturas das aplicações e os trechos normativos que justificam esta avaliação. Esta abordagem agrega valor à avaliação da completude e da confiabilidade da solução. Desta forma, possibilita ao desenvolvedor identificar quais regras estão sendo consideradas pela análise, mesmo que esta não contemple todas as regras. Além disto, permite realizar a rastreabilidade às proposições normativas, a fim de atestar a conformidade com as especificações. Vale salientar que apenas as assertivas classificadas como estruturais e relacionais fazem sentido para a análise estrutural.

Para a análise estrutural, além de utilizar uma abordagem orientada a assertivas para garantir a conformidade, buscou-se definir as regras de avaliação de forma independente, isto é, uma regra deve ser autossuficiente para determinar se a estrutura apresenta ou não o problema investigado por ela. Desta forma, funcionalmente, a análise apresentará todos os problemas existentes na aplicação analisado de acordo com as regras suportadas, evitando que erros predecessores impactem a

avaliação de trechos subsequentes. Além disto, este formato permite que a análise possa ser paralelizada, a fim de otimizar a análise, e parametrizada, permitindo a análise de regras específicas de acordo com a necessidade do desenvolvedor. A análise estrural de aplicações orientadas a conformidade consiste em (Figura 0.3):

1. Traduzir a aplicação para uma árvore sintática abstrata (*AST - Abstract Syntax Tree*) que representa a linguagem;
2. Coletar as regras existentes e selecionadas (parametrizadas pelo usuário) para a análise de cada nó da AST da aplicação;
3. Avaliar cada regra coletada, a fim de identificar as possíveis inconsistências; e
4. Apresentar os problemas estruturais identificados.

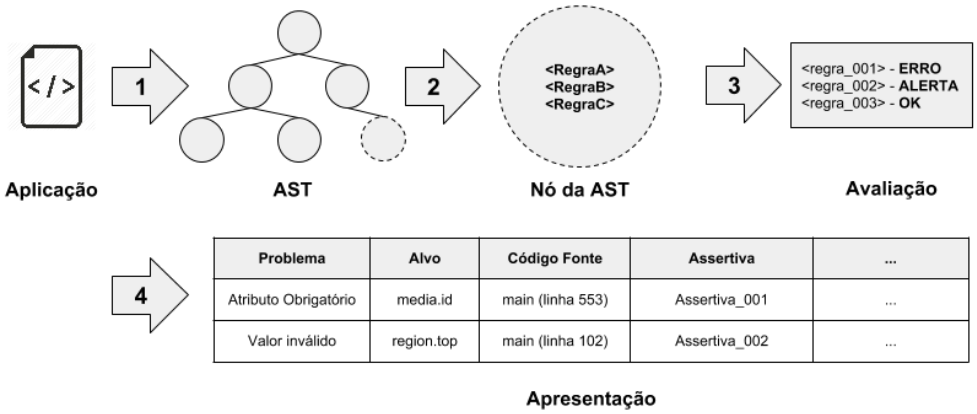

Figura 0.3. Análise Estrutural de Aplicações Orientada a Conformidade.

1.1.1 Tradução da Aplicação

Processo caracterizado pela leitura e estruturação da aplicação. Desta forma, realiza um *parsing* da aplicação de acordo com sua gramática a fim de extrair a árvore sintática abstrata (*AST - Abstract Syntax Tree*). A tradução deve iniciar pelo arquivo principal da aplicação identificando todos os arquivos referenciados recursivamente. Além de realizar a tradução, esta etapa também é responsável por realizar coleta de informações acerca do código traduzido (artefato e linha), a fim de evidenciar a rastreabilidade para o código analisado na apresentação dos resultados.

1.1.2 Coleta das Regras Estruturais

Esta etapa tem por objetivo coletar as regras a serem executadas no processo de identificação automática dos problemas estruturais. Cada nó da *AST* mantém um conjunto de regras estruturais

que o avalia, e essas, por sua vez, referenciam o *Token* que identifica a assertiva correspondente. Sendo assim, para identificar todas as regras que compõem a análise estrutural da aplicação, é necessário caminhar em toda a estrutura da *AST* coletando as regras existentes para cada nó. Nessa etapa, também é possível selecionar um conjunto pré-estabelecido de regras, permitindo a realização de uma análise estrutural parcial, parametrizada de acordo com as necessidades do desenvolvedor, coletando apenas as regras que se deseja investigar, relevantes para alguns contextos de desenvolvimento.

1.1.3 Avaliação das Regras Estruturais

Nesta etapa, todas as regras coletadas são avaliadas e encapsuladas juntamente com seus resultados. A especificação da linguagem pode conter regras rígidas, definindo restrições que não possuem tratamentos excepcionais para a estrutura que não as seguem. Além disso, a especificação da linguagem também pode definir recomendações, sugerindo diretrizes a serem seguidas, mas também definindo tratamentos no caso delas não serem seguidas. Desta forma, o resultado das avaliações pode ser classificado em: OK – atende a regra estrutural; ERRO – não atende a regra estrutural; e ALERTA – não atende a recomendação. Dado que a avaliação das regras é realizada de forma independente, este processo permite que a execução das avaliações possa ser paralelizada, a fim de otimizar a análise.

1.1.4 Apresentação dos Problemas Identificados

Esta etapa é responsável por apresentar as inconsistências identificadas, destacando o problema, o alvo (elemento da linguagem), a referência para o código fonte (artefato e linha) e a assertiva (identificador, descrição assertiva, classificação e trechos normativos referenciados). Apesar de ser dispensável, por não demandar ação do desenvolvedor, é importante considerar as regras atendidas (OK), a fim de evidenciar, em sua totalidade, o conjunto de regras que foram utilizadas no processo de análise estrutural.

1.2 Análise Comportamental

Esta seção apresenta a proposta da solução referente à análise comportamental, descrevendo a abordagem sob a perspectiva conceitual e técnica.

As análises comportamentais geralmente buscam prover abstrações para representar o comportamento com o objetivo de auxiliar as atividades de documentação, manutenção, validação e

verificação do software. Por exemplo, essas abstrações podem ser aplicadas: em testes de regressão; na identificação de anomalias comportamentais; na geração de casos de testes automáticos; na representação da compreensão do software; na compactação das possibilidades de interação; entre outros (WALKINSHAW *et. al.*, 2013). É comum utilizar a abstração do modelo comportamental como etapa intermediária de processos de verificação formais (SANTOS, 2012; DAHMANI *et. al.*, 2014; GAGGI; BOSSI, 2011; BOUYAKOUB; BELKHIR, 2011) sem apresentá-la visualmente. No entanto, esses procedimentos se limitam a identificar inconsistências comportamentais classificadas como Padrão (da linguagem), que buscam verificar a consistência temporal das aplicações. Desta forma, não permitem identificar as inconsistências comportamentais referentes às regras de negócio da aplicação, classificadas como Própria (da aplicação). Nesse sentido, existem algumas soluções (HONORATO; BARBOSA, 2010; PICININ *et. al.*, 2012) que atuam nessa limitação, permitindo que o usuário possa incrementar a base de regras utilizada, possibilitando a inclusão de regras próprias da aplicação. Porém, essas abordagens demandam um esforço do desenvolvedor para retroalimentar a solução com regras específicas da aplicação, que normalmente não são reutilizadas para análises de outras aplicações. Além disto, essas soluções tendem a utilizar uma linguagem de domínio própria para definição de novas regras, o que tende a aumentar a curva de aprendizado do usuário.

No contexto das aplicações multimídia, a identificação dos problemas comportamentais pelos desenvolvedores normalmente segue o mesmo processo, ineficiente e arbitrário, utilizado na identificação de problemas estruturais: por meio da execução em interpretadores. Para a investigação de problemas comportamentais, a utilização deste método é ainda mais agravante, pois falsos negativos ou falsos positivos são mais difíceis de serem identificados. Mesmo tomando como premissa que a aplicação esteja estruturalmente correta, o comportamento apresentado é resultante dos relacionamentos entre diversos componentes, das interações realizadas e do layout utilizado pela aplicação. Além disto, nem todas as variáveis são visíveis ao desenvolvedor em um processo natural de execução.

A fim de prover uma alternativa apropriada para o contexto deste trabalho, é relevante prover soluções que busquem endereçar adequadamente as limitações do processo de desenvolvimento usual. Como dito anteriormente, no Capítulo 2, a abordagem de desenvolvimento centrada na análise orientada a conformidade, juntamente com uma representação comportamental apropriada, que permita a identificação de quaisquer problemas comportamentais, é indicada para este fim.

Sendo assim, o modelo proposto fornece uma abstração comportamental que apresenta todos os estados da aplicação, permitindo ao desenvolvedor, em cada estado, identificar:

- As interações possíveis que resultem em uma mudança de estado;
- A descrição das variáveis; e
- As inconsistências padrão (da linguagem).

Além disto, este modelo fornece interfaces que possibilitam:

- Simular a execução da aplicação manualmente, navegando sobre a abstração comportamental;
- Executar fluxos de navegações relevantes, baseados nos caminhos possíveis originados pela abstração comportamental; e
- Depurar a abstração comportamental, por meio do uso de *breakpoints* associados aos eventos e variáveis da aplicação.

Para atingir estes objetivos foi necessário atuar em alguns desafios intrínsecos da natureza da linguagem de desenvolvimento e do problema de análise comportamental, além dos consequentes da proposta da solução:

- Tratar adequadamente os diferentes tipos de estruturas: os que facilitam o relacionamento entre os componentes ("açúcar sintático"); os que definem o modelo de interação; e os que determinam o layout da aplicação;
- Identificar evidências de comportamentos inesperados próprios da aplicação, não se limitando apenas a identificar padrões de inconsistência conhecidos;
- Atuar no problema da "explosão de estados" (WAGNER *et. al.*, 2006), existente no contexto das análises comportamentais;
- Prover a rastreabilidade às proposições normativas que justificam as interpretações comportamentais; e
- Suprir a necessidade de simulação manual em interpretadores arbitrários durante o processo de desenvolvimento para avaliar a corretude da aplicação.

Sendo assim, este trabalho propõe a utilização de modelos abstratos bem definidos e coesos, cada um com suas responsabilidades, para tratar alguns desses desafios. Algumas estruturas existem apenas para prover um "açúcar sintático" da linguagem, isto é, provêm facilidades para a codificação. Computacionalmente, em um processo de análise comportamental, não se faz

necessário utilizar essas estruturas. Desta forma, com o objetivo de simplificar a tradução dos comportamentos, é importante eliminar toda e qualquer estrutura desnecessária. Nesse sentido, este trabalho propõe, como primeira etapa da análise comportamental, a tradução da árvore estrutural em um modelo simplificado, denominado **Modelo de Relacionamentos**. Ainda no sentido de simplificar a tradução dos comportamentos, é relevante criar modelos que permitam a identificação de cada tipo de problema comportamental (Interativo ou Descritivo), dado que o processo de análise e as estruturas investigadas em cada tipo são divergentes. Nesse contexto, definiu-se a utilização do **Modelo Comportamental** e do **Modelo Descritivo**, responsáveis por permitir a identificação das inconsistências comportamentais classificadas como Interativas e Descritivas, respectivamente. Anteriormente, no Capítulo 2, foi apresentada a definição formal desses modelos. Ainda no sentido de tratar os desafios identificados, este trabalho propõe a utilização de *tokens* no processo de tradução dos modelos, no qual cada *token* identifica unicamente as assertivas que justificam um determinado comportamento. A análise comportamental de aplicações orientadas a conformidade consiste nas seguintes etapas (Figura 0.4):

1. Realizar a tradução da aplicação para uma árvore sintática abstrata (*Abstract Syntax Tree - AST*) que representa a linguagem, a fim de permitir a manipulação de toda aplicação em memória;
2. Construir o Modelo de Relacionamentos por meio da tradução da *AST*, a fim de eliminar as estruturas que, essencialmente, constituem o "açúcar sintático" da linguagem;
3. Identificar todos os eventos E possíveis de ocorrer na aplicação, essenciais para a construção do Modelo Comportamental $B = (S, E, D, T)$;
4. Construir o Modelo Descritivo Inicial $d_0 \in D$, através da tradução do Modelo de Relacionamentos, que representa o ponto de partida para a construção iterativa do Modelo Comportamental, por constituir o estado inicial s_0;
5. Construir o Modelo Comportamental $B = (S, E, D, T)$ iterativamente a partir do estado inicial s_0, buscando as transições que têm origem neste estado por meio da identificação e acionamento dos eventos permitidos (interagindo com o Modelo Descritivo). Para cada novo estado criado a partir das transições, é necessário realizar uma nova iteração a fim de identificar as transições originadas deste novo estado;
6. Identificar as inconsistências classificadas como Padrão, existentes no modelo comportamental (interativas) e nos modelos descritivos de cada estado (descritivas); e

7. Apresentar os resultados a partir dos Modelos gerados que evidenciam os comportamentos da aplicação.

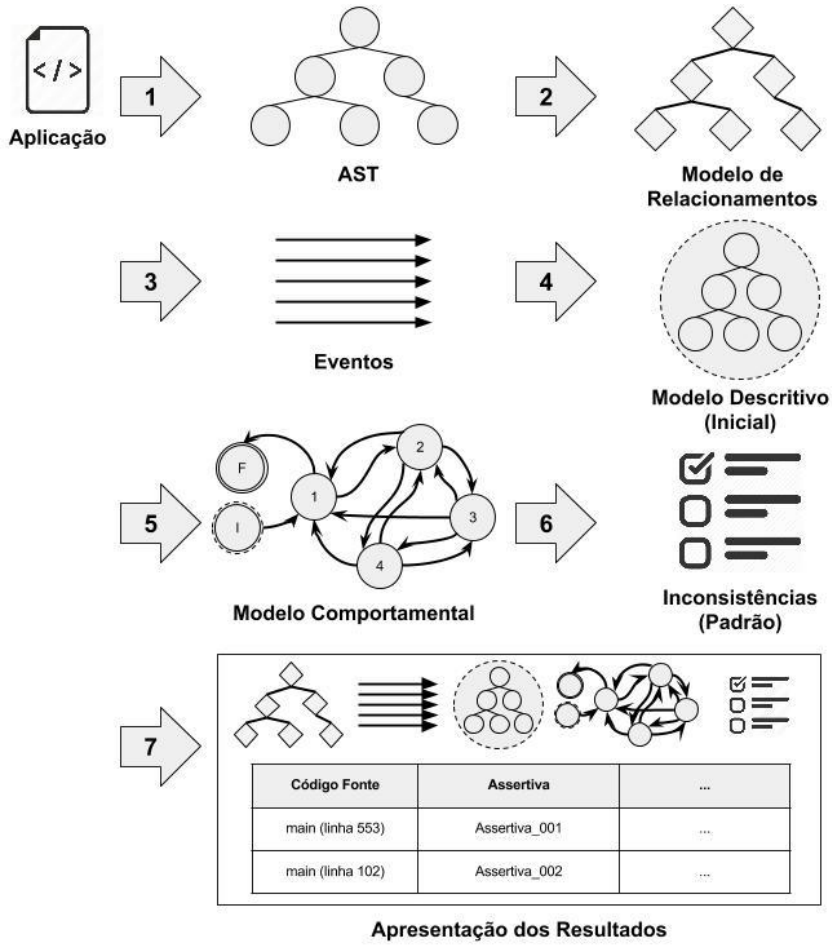

Figura 0.4. Análise Comportamental de Aplicações orientada a conformidade.

1.2.1 Tradução da Aplicação

Processo reutilizado da análise estrutural, responsável pela leitura e estruturação da aplicação em uma árvore sintática abstrata (*AST - Abstract Syntax Tree*). Segue o mesmo procedimento da análise estrutural (apresentado na seção 3.1.1).

1.2.2 Construção do Modelo de Relacionamentos

Com o objetivo de simplificar a tradução dos modelos abstratos, é importante eliminar toda e qualquer estrutura desnecessária, tal como as estruturas conhecidas como o "açúcar sintático" da linguagem que existem apenas para prover facilidades para a codificação. Nesse sentido, foi proposta a definição de um modelo simplificado que represente a aplicação em memória, livre de estruturas

desnecessárias para a análise comportamental, denominado **Modelo de Relacionamentos**. Este modelo visa eliminar o "açúcar sintático" da linguagem traduzindo a aplicação a partir da *AST* para uma estrutura mais concisa, eliminando os relacionamentos de importação, referência, contexto, hierarquia e reuso. Vale salientar que existem trechos na especificação, e consequentemente assertivas (Relacionais), que regem cada um desses relacionamentos entre as estruturas. Desta forma, com o objetivo de manter a rastreabilidade, é necessário coletar os *Tokens* em cada etapa da tradução;

1.2.3 Identificação dos Eventos de Transição

Nesta etapa, todos os eventos possíveis E são identificados e encapsulados. Como definido pela abordagem proposta, cada evento possível $e \in E$ é constituído de uma parte condicional Ec e outra parte acionável Ea. Desta forma, este processo procura observar nós específicos do Modelo de Relacionamentos (ex.: as estruturas relacionais que representam os Links) com o objetivo de coletar as informações necessárias para a criação dos eventos em memória. Cada evento criado deve prover acesso aos *tokens* referentes a suas partes condicional e acionável, para serem coletados sempre que o evento for utilizado pelo Modelo Comportamental da Aplicação. Como apresentado no Capítulo 2, a parte condicional do evento Ec é essencial para a definição do Modelo Comportamental, pois a incidência dos eventos é quem determina a equivalência de estados e as possibilidades de reduções. Desta forma, é relevante classificar os eventos de acordo com sua natureza condicional:

- **Eventos Temporais –** Os eventos temporais são aqueles definidos pela espera de um determinado tempo, por exemplo, uma mídia iniciará em 10s enquanto outra iniciará em 20s. Desta forma, a cada tempo de espera definido, um evento temporal será disparado podendo alterar um Modelo Descritivo $d \in D$, mas é possível que outro evento ocorra antes do evento temporal;
- **Eventos Instantâneos –** Os eventos instantâneos são um subconjunto dos eventos temporais que são executados instantaneamente após a execução de algum outro evento, desta forma não se pode intervir na sua execução;
- **Eventos de Seleção –** Os eventos de seleção são aqueles que dependem da interação do usuário da aplicação. Para este contexto, considera-se que não é possível pressionar concorrentemente duas ou mais teclas, por exemplo; e

- **Eventos de Avaliação –** Os eventos de avaliação são eventos que dependem do Modelo Descritivo $d \in D$, no qual uma condição específica desse modelo determina a incidência do evento.

Os eventos representam as possibilidades de transição da aplicação e são essenciais para a construção iterativa do Modelo Comportamental. Porém, em um contexto de depuração, o usuário também poderá definir seus *breakpoints* por meio dos eventos, determinando que a construção iterativa do Modelo Comportamental deva parar após a incidência de um evento específico, para que alguma verificação seja realizada.

1.2.4 Construção do Modelo Descritivo Inicial

Esta etapa é responsável por realizar a tradução do Modelo de Relacionamentos para o Modelo Descritivo (Inicial) $d_0 = (N, n_0)$. Um Modelo Descritivo $d \in D$ representa o domínio de variáveis da aplicação cuja interpretação resulta na interface gráfica da aplicação (Figura 0.5). Sendo assim, este modelo tem um papel fundamental na geração do Modelo Comportamental $B = (S, E, D, T)$, pois ele é utilizado por cada evento $e \in E$ para determinar sua ocorrência por meio da função de permissão $p: D \rightarrow \{0,1\}$ da parte condiciona Ec, além de ser entrada da função de transformação $t: D \rightarrow D'$ da parte acionável, que altera o valor das variáveis gerando novos Modelos Descritivos. Além disto, é importante destacar que cada nó do Modelo Descritivo deve fornecer os *Tokens* descritivos durante sua interpretação visual ou textual. O Modelo Descritivo Inicial $d_0 \in D$ também é importante em um contexto de depuração, pois a partir dele o usuário pode identificar todas as variáveis da aplicação e definir seus *breakpoints* por meio dessas variáveis, determinando que a construção iterativa do Modelo Comportamental deva parar após o acesso ou alteração de uma variável específica.

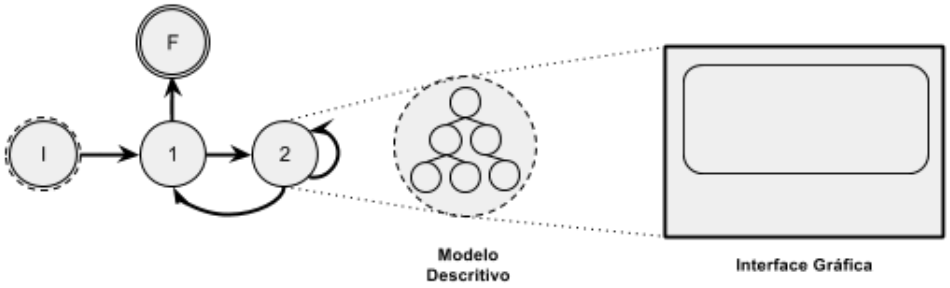

Figura 0.5. Relação entre Estado, Modelo Descritivo e Interface Gráfica

1.2.5 Construção do Modelo Comportamental

Etapa responsável por construir o Modelo Comportamental $B = (S, E, D, T)$ identificando os estados existentes por meio dos eventos possíveis e da interação deles com os modelos descritivos gerados. Este modelo é construído por meio de ciclos iterativos, a fim de identificar todos os estados S e transições T, partindo do estado inicial s_0 que é constituído pelo Modelo Descritivo Inicial d_0. O ciclo iterativo é dividido em três etapas (Figura 0.6): 1) identificar os eventos a ocorrer no estado corrente; 2) acionar eventos para gerar as transições; e 3) executar os algoritmos de redução. A primeira etapa consiste em testar o Modelo Descritivo do estado para cada evento (por meio da parte condicional, como dito anteriormente) e definir quais eventos podem ocorrer em paralelo. A segunda executa a transformação de cada evento permitido (por meio da parte acionável, como dito anteriormente), gerando, portanto, uma nova transição t com origem no estado corrente e destino para um novo estado $s \notin S$ ou para um estado equivalente já criado $s \in S$. Por fim, após a execução de todas as transformações, o estado é considerado fechado e executa os algoritmos de redução. A cada novo estado criado, uma nova iteração é executada a partir deste novo estado.

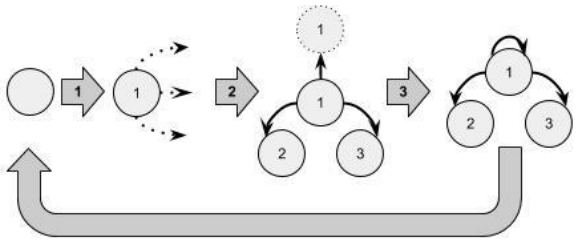

Figura 0.6. Processo de construção do Modelo Comportamental.

É importante salientar que a definição dos eventos possíveis de um estado não é apenas determinada pelas funções de permissão de cada evento. Também é preciso definir quais desses podem ocorrer em paralelo. Existem algumas regras de priorização que precisam ser satisfeitas com o objetivo de manter a consistência temporal:

- **Eventos Temporais –** Caso um conjunto de eventos temporais seja permitido, apenas aqueles de menor tempo de espera serão selecionados como possíveis, por conta da continuidade do tempo. Dado o exemplo da seção anterior, para a mídia de 20s ser iniciada, obrigatoriamente a mídia de 10s deve ter sido iniciada, pois, logicamente, não tem como chegar aos 20s sem passar pelos 10s (Figura 0.7). Como regra geral, dado um conjunto de eventos temporais permitidos, apenas aqueles com o

menor tempo de espera devem ser selecionados para execução. Esta regra também é aplicada para eventos instantâneos, considerando o tempo de espera como 0s (zero segundo);

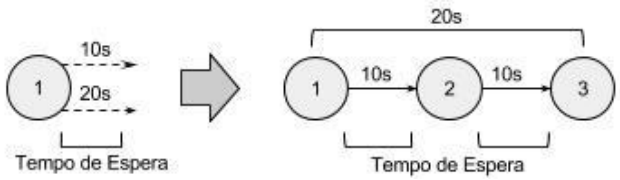

Figura 0.7. Exemplo de evento temporal.

- **Eventos Instantâneos –** Não se pode intervir na execução dos eventos instantâneos. Por exemplo, dado que uma mídia deva iniciar imediatamente após outra, não há a possibilidade de realizar alguma ação entre o fim da primeira mídia e o início da mídia seguinte (Figura 0.8). Sendo assim, como regra geral, dado um conjunto de eventos permitidos, caso existam eventos instantâneos, apenas estes devem ser selecionados para ocorrerem em uma única transição; e

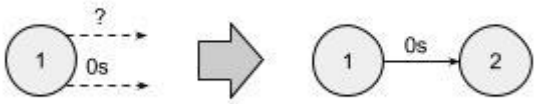

Figura 0.8. Exemplo de evento instantâneo.

- **Eventos de Seleção –** Cada seleção representa um evento discreto e por isso não podem ser sequênciados em uma mesma transição. Por exemplo, dado que uma mídia seja iniciada ao pressionar a tecla verde e outra seja iniciada ao pressionar a tecla vermelha, não é possível iniciar as duas mídias ao mesmo tempo, obrigatoriamente deve-se pressionar apenas uma das duas teclas (Figura 0.9). Sendo assim, como regra geral, dado um conjunto de eventos de seleção permitido, cada um deles deve ser selecionado para ocorrer em transições distintas, representando fluxos comportamentais diferentes dentro do modelo.

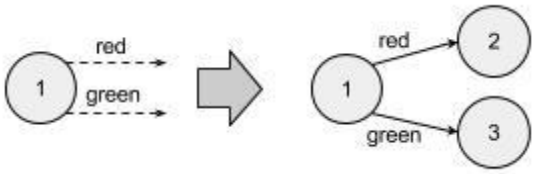

Figura 0.9. Exemplo de evento de seleção.

1.2.6 Identificação das Inconsistências Padrão

Esta etapa tem por objetivo identificar as inconsistências **Padrão Interativa**, existente no Modelo Comportamental, e **Descritiva,** existente no Modelo Descritivo de cada estado. Desta forma, após a construção completa do Modelo Comportamental, é necessário navegá-lo para investigar a existência de padrões de inconsistências no modelo e nos estados. Sendo assim, deve-se predefinir quais padrões serão considerados nessa investigação de acordo com a linguagem de desenvolvimento. Exemplos: com exceção do estado final, todos os outros estados devem possuir uma transição de saída (Interativa); e nenhuma mídia pode extrapolar os limites absolutos da interface do dispositivo (Descritiva).

1.2.7 Apresentação dos Resultados

Como resultado da análise comportamental, deve-se apresentar o Modelo Comportamental gerado, detalhando sua estrutura (estados e transições) e os Modelos Descritivos associados aos estados, pois ambos fornecem uma visão do comportamento da aplicação e permitem ao usuário identificar **Inconsistências Comportamentais Próprias** (interativas e descritivas) presentes na aplicação. Além disto, é necessário apresentar também as **Inconsistências Comportamentais Padrão** (interativas e descritivas), pois sinalizam a existência de problemas desta natureza (ex.: a inexistência de um estado final) presentes na aplicação. Apesar de serem estruturas intermediárias, também é importante apresentar o Modelo de Relacionamentos e os eventos possíveis identificados, a fim de fornecer acesso ao desenvolvedor, para que ele possa verificar a consistência dos modelos. Todos os artefatos apresentados devem referenciar o código fonte (artefato e linha) e apresentar os *Tokens* e as assertivas relacionadas.

1.3 Considerações Finais

Este capítulo buscou apresentar as abordagens propostas por este trabalho: Análise Estrutural Orientada a Conformidade e Análise Comportamental Orientada a Conformidade. Com isso,

apresentou todas as preocupações, decisões, limitações e detalhes conceituais das abordagens. Com relação à Análise Estrutural, a proposta buscou:

- Agregar valor ao processo de correção de *bugs*, permitindo a identificação automática de todos os erros estruturais de acordo com as regras com suporte na solução;
- Fornecer insumos para a correção por meio da rastreabilidade para as especificações, que referencia todo o conhecimento relacionado com o problema identificado; e
- Permitir a aplicabilidade da solução no processo de desenvolvimento de aplicações declarativas multimídia.

Porém, é preciso destacar algumas limitações consideradas:

- Para atingir a completude, dando suporte a todas as regras estruturais, é necessário garantir a consistência quantitativa e qualitativa das regras estruturais com as assertivas, e destas com as especificações, a fim de fornecer uma solução completa e confiável; e
- Dado que algumas aplicações utilizam *scripts* em sua essência, é importante destacar que o modelo proposto não provê mecanismos para avaliação estrutural desses *scripts*.

Com relação à Análise Comportamental, a proposta buscou:

- Permitir a identificação, tanto das inconsistências comportamentais Padrão, quanto das Próprias;
- Permitir a identificação, tanto das inconsistências comportamentais Descritivas, quanto das Interativas;
- Fornecer insumos para a correção por meio da rastreabilidade para as especificações, que referencia todo o conhecimento relacionado com o problema identificado;
- Prover uma alternativa para atuar no problema da "explosão de estados", por meio dos algoritmos de redução e das configurações de análise; e
- Permitir a aplicabilidade da solução no processo de desenvolvimento de aplicações declarativas multimídia.

Assim como na Análise Estrutural, também é preciso destacar algumas limitações:

- Não provê a identificação de inconsistências comportamentais de forma automática, isto é, a solução busca prover insumos por meio dos modelos extraídos que facilitem a identificação dessas inconsistências;

- Não define estratégias para reutilizar o resultado de análises prévias, isto é, cada análise realizada deve executar todo o processo, independente da modificação realizada;
- Para atingir a completude, dando suporte a todas as regras comportamentais, é necessário garantir a consistência quantitativa e qualitativa das regras comportamentais com as assertivas, e destas com as especificações, a fim de fornecer uma solução completa e confiável; e
- Não foi considerada a avaliação dos *scripts* que podem constituir as aplicações. Eles são tratados como uma mídia qualquer, que pode ser executada durante a aplicação.

O próximo capítulo tem como objetivo detalhar a implementação da proposta definida neste capítulo.

Implementação da Proposta

Como visto no capítulo anterior, a abordagem proposta nesta tese foi definida de forma flexível para o contexto das aplicações multimídia podendo ser aplicada a quaisquer linguagens declarativas padronizadas por especificações públicas desse contexto. Porém, para avaliar a abordagem proposta é necessário realizar uma prova de conceito, delimitando o escopo para uma linguagem específica, a fim de permitir a investigação da viabilidade técnica e funcional (valor agregado no uso) a partir de uma solução que a instancie. Além disso, dada a característica flexível da abordagem, também foi necessário investigar a extensibilidade da solução desenvolvida, a fim de identificar os pontos de reuso e de extensão na definição de novas soluções para outras linguagens de desenvolvimento.

Neste processo, a linguagem NCL foi eleita como alvo da prova de conceito, dado o conhecimento prévio nos problemas desse contexto de desenvolvimento e as características da linguagem que a torna bastante representativa para este estudo. Dado que, como dito anteriormente, é uma linguagem declarativa, especificada pela ABNT (ABNT, 2015) para o desenvolvimento de aplicativos para a TV Digital e, recentemente, foi certificada pelo ITU como a primeira recomendação internacional para suporte à interação e à multimídia para dispositivos de IPTV (*Internet Protocol Television*) (ITU, 2009). Além disto, é executada em diversas plataformas e dispositivos, que não apresentam garantias de conformidade com as especificações definidas (SANTOS; LENZ, 2014; PINHEIRO *et. al.*, 2012) e requer robustez e interoperabilidade dos aplicativos desenvolvidos, devido à sua aplicação crítica no processo de transição da TV analógica para a TV Digital de diversos países (SANTOS; LENZ, 2014). As aplicações desenvolvidas em NCL são componentes fundamentais para o sistema de TV Digital, pois sua execução pode apresentar problemas para o usuário, mesmo que toda a infraestrutura de transmissão e recepção esteja conforme as normas. Santos, Ferraz e Lenz (2013) sugerem que é importante analisar a conformidade das aplicações, pois elas devem seguir especificações precisas para o seu correto funcionamento. Segundo Santos, Braga e Muchaluat-Saade (2013), as aplicações desenvolvidas em

NCL podem apresentar três tipos de problemas: **Problemas sintáticos**, quando não seguem a gramática da linguagem; **Problemas estruturais**, quando, apesar de se limitar à gramática da linguagem, não respeitam a semântica estática da linguagem (ex.: referência e escopo); e **Problemas comportamentais**, quando, apesar de seguir fielmente as normas, a computação induzida pela aplicação resulta em comportamentos inesperados (interações ou exibições indesejadas).

Este capítulo tem por objetivo mostrar a aplicabilidade da proposta a partir da implementação de uma prova de conceito, apresentando detalhes acerca da implementação dos módulos de análise estrutural e comportamental. Além de descrever brevemente a utilização da solução implementada em alguns contextos de uso, a fim de evidenciar o valor agregado da solução na identificação de problemas. E, por fim, apresenta uma discussão acerca da flexibilidade da abordagem a partir de extensões da solução para as linguagens HTML e SVG.

1.1 Prova de Conceito

Como prova de conceito, para instanciar a abordagem proposta, uma solução foi implementada utilizando a linguagem Java. Essa solução provê recursos para análise estrutural e comportamental de aplicações NCL. Na prática essas duas perspectivas de análise determinaram a separação da solução em dois módulos. Com o objetivo de facilitar o acesso aos módulos durante o processo de desenvolvimento e manutenção das aplicações, foi provida uma integração com o ambiente de desenvolvimento NCL Eclipse (AZEVEDO *et. al.*, 2009), consolidada por meio de *plug-ins* para a plataforma Eclipse. Para o desenvolvimento da primeira versão da solução foram consideradas apenas algumas assertivas estruturais e comportamentais representativas, pelo fato dessa versão servir para investigar a viabilidade técnica da abordagem proposta.

Apesar da solução não ser caracterizada com uma ferramenta para autoria de aplicações, ela busca prover recursos complementares para um ambiente de desenvolvimento a partir da extensão do NCL Eclipse. Desta forma, além dos requisitos funcionais que foram definidos de acordo com o propósito de avaliar a viabilidade técnica da abordagem, também é importante definir requisitos que tornem as funcionalidades mais produtivas e atrativas para os usuários em um processo de desenvolvimento. Akiyama, Oore e Watters (2014) realizaram um estudo com usuários iniciantes e avançados no contexto das ferramentas de autoria de aplicações multimídia, a fim de identificar os principais problemas de usabilidade dessas soluções e propor diretrizes para a melhoria desses problemas. Para o contexto desse trabalho, algumas orientações se mostraram relevantes para a

implementação das funcionalidades de análise, principalmente no contexto da análise comportamental, como por exemplo (AKIYAMA; OORE; WATTERS, 2014):

- Prover aos usuários recursos para gerar e manter os fluxos de interação;
- Fornecer passos claros para alcançar cada estado da aplicação;
- Permitir o usuário realizar um conjunto de ações desde uma visão mais contextual até ações de baixo nível;
- Exibir caminhos alternativos para atingir um mesmo resultado; e
- Prover informações acerca do estado corrente, bem como as ações que podem ser realizadas e os resultados consequentes dessas ações.

As seções seguintes apresentam os detalhes de implementação dos módulos de análise estrutural e comportamental, destacando os requisitos identificados, a arquitetura proposta, o escopo da implementação e as funcionalidades implementadas para cada perspectiva de análise. É importante destacar que, apesar de ter sido fundamentada pelos desafios da proposta e pelos trabalhos relacionados, a identificação dos requisitos funcionais e sua classificação de acordo com a relevância foram realizadas de forma empírica sem seguir nenhum processo formal, representando uma ameaça desta prova de conceito.

1.1.1 Análise Estrutural

Os requisitos da análise estrutural (Quadro 0.1) foram definidos pelas funcionalidades propostas pela a abordagem, a fim de evidenciar a viabilidade técnica na implementação dessas. Além desses, também foram definidos requisitos baseados nas funcionalidades e limitações encontradas nos trabalhos relacionados.

Quadro 0.1. Requisitos funcionais do módulo de análise estrutural

Id	Requisito Funcional	Relevância
RF001	Identificar problemas estruturais léxicos	Essencial
RF002	Identificar problemas estruturais sintáticos	Essencial
RF003	Identificar problemas estruturais semânticos estáticos	Essencial
RF004	Prover a rastreabilidade para os trechos de código com problemas identificados	Essencial
RF005	Prover a rastreabilidade para as assertivas que justificam os problemas identificados	Essencial
RF006	Fornecer insumos para a correção dos problemas identificados	Essencial

RF007	Apresentar as assertivas que fundamentaram a análise estrutural	Importante
RF008	Prover a rastreabilidade para as assertivas que estão associadas com a aplicação	Importante
RF009	Permitir a configuração do escopo da análise estrutural referente ao código da aplicação	Importante
RF010	Permitir a configuração do escopo da análise estrutural referente às assertivas estruturais	Importante
RF011	Informar a presença de heurísticas estáticas nas aplicações que determinam as boas práticas	Opcional
RF012	Permitir estender a base de heurísticas estáticas	Opcional

Para o desenvolvimento desta primeira versão da solução, foram considerados apenas os requisitos funcionais categorizados como **Essencial**, pelo fato desta solução servir para provar a viabilidade técnica da abordagem proposta. Porém, para facilitar a extensão da solução, todos os requisitos foram considerados para o projeto da arquitetura. Para o módulo estrutural da solução foram idealizados dois projetos de arquitetura. Primeiramente foi projetada uma solução de análise estrutural completa, sem reutilizar bibliotecas existentes, a fim de fornecer a idealização de um projeto alternativo para as linguagens que não possuam bibliotecas confiáveis para serem reusadas. Desta forma, a arquitetura do módulo completo foi projetada conforme a abordagem proposta, buscando definir componentes responsáveis por cada etapa (Figura 0.1):

- **StructuralAnalyzerFacade** – Componente responsável por centralizar o acesso aos serviços do módulo estrutural, implementa o padrão de projeto *Facade* (GAMMA *et. al.*, 1994);
- **ApplicationToASTParser** – Realiza a tradução da aplicação para uma árvore sintática abstrata (*Abstract Syntax Tree - AST*) **ASTNCL** que representa o NCL por meio da utilização da API SAX[6] que é responsável por ler e permitir a manipulação da aplicação. A *AST* foi definida de acordo com a gramática da linguagem (ABNT, 2015);
- **NormativeRuleFinder** – Componente responsável por coletar as regras estruturais a serem executadas para identificar os problemas. Este Componente utiliza o padrão *Visitor* para caminhar na AST NCL coletando as instâncias das regras (***INormativeRule***). Cada nó da AST está associado a um conjunto de regras e essas,

[6] *SAX API Documentation*. Disponível em: <docs.oracle.com/javase/8/docs/api/org/xml/sax/package-summary.html>. Acesso em: 20 de Setembro de 2016.

por sua vez, referenciam explicitamente o *token* (***EnumStructuralToken***) que identifica a assertiva correspondente; e

- **StructuralAnalyzer –** Componente responsável por validar as regras coletadas. Cada instância de regra retorna um objeto que encapsula o resultado da validação (***ValidationData***), mantendo a rastreabilidade para o código (artefato e linha) e para a especificação (*token* e assertiva).

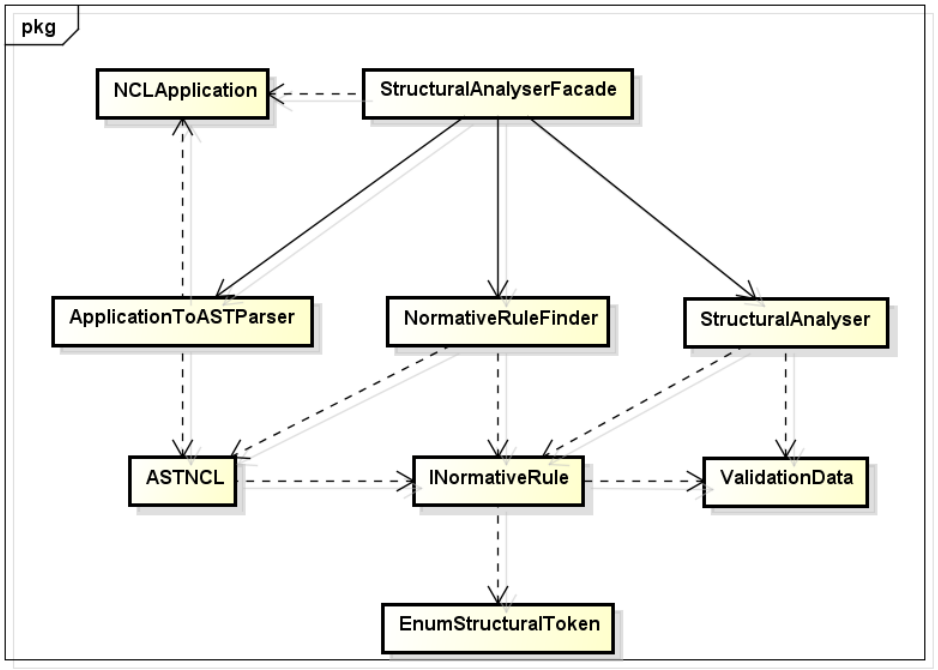

Figura 0.1. Arquitetura da Análise Estrutural (Completa)

Por fim, projetou-se a solução que realmente foi utilizada na prova de conceito, considerando a utilização do *NCL-Validator* (ARAÚJO *et. al.*, 2008) para o processo de identificação de problemas estruturais, provendo apenas o tratamento das limitações desta API. Desta forma, a arquitetura do módulo foi projetada com dependência para o *NCL-Validator* (Figura 0.2), reutilizando a funcionalidade responsável pela validação do NCL, realizada por meio do componente denominado *NCLValidator*. O *NCLValidator* é responsável por realizar a tradução da aplicação, representada por *Document*, para uma árvore sintática abstrata (*NclValidatorDocument*) que representa o NCL, por meio

da utilização da *API DOM*[7]. Além disto, o *NCLValidator* também é responsável por identificar os problemas estruturais existentes na aplicação, criando uma mensagem (*Message*) para cada problema, contendo as informações referentes ao problema identificado, inclusive o *Token* (*EnumStructuralToken*) que permite realizar a rastreabilidade para as assertivas estruturais existentes. É importante destacar, que além do motor de validação do *NCL-Validator*, as funcionalidades integradas ao ambiente de desenvolvimento também foram reusadas, como por exemplo, o recurso que destaca os problemas estruturais no próprio editor.

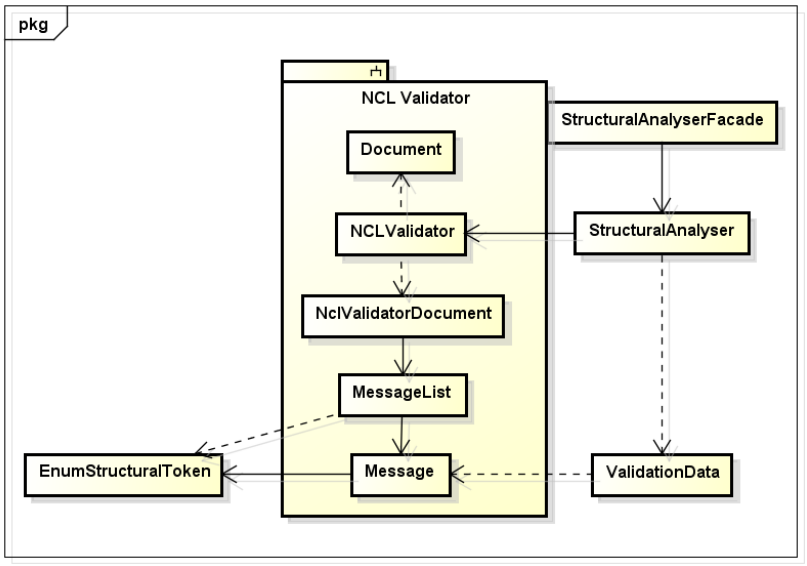

Figura 0.2. Arquitetura da Análise Estrutural considerando a *API NCL-Validator*

Para a implementação da versão completa da solução foram consideradas apenas algumas assertivas assertivas estruturais com o objetivo avaliar a viabilidade técnica do projeto da solução. Essas assertivas foram selecionadas empiricamente levando em consideração o tipo, a importância e a complexidade dessas, para definir um subconjunto representativo da linguagem. Desta forma, foram consideras apenas as regras apresentadas no

Quadro 0.2. Sendo assim, a implementação da solução buscou identificar os problemas estruturais validados por estas regras, além de prover a rastreabilidade para as assertivas e para os trechos de código relacionados.

---

7 *DOM API Documentation*. Disponível em: <docs.oracle.com/javase/8/docs/api/org/w3c/dom/package-summary.html>. Acesso em: 20 de Setembro de 2016.

Quadro 0.2. Regras estruturais selecionadas para implementação

Regra	Descrição	Exemplo de Entidades Relacionadas
Opcional	Responsável por validar os elementos e atributos opcionais	head:descriptorBase region.top
Obrigatório	Responsável por validar os elementos e atributos obrigatórios	importedDocumentBase:ImportNCL region.id
Coleção	Responsável por validar os elementos que possuam coleção de outros elementos	regionBase:region
Xor	Responsável por validar elementos que elementos com disjunção exclusiva	casualConnector:simpleAction casualConnector:compoundAction
Or	Responsável por validar elementos que elementos com disjunção inclusiva	descriptorBase:importBase descriptorBase:descriptor descriptorBase:descriptorSwitch
Unicidade	Responsável por validar atributos que devem possuir valores únicos no documento NCL	media.id

A implementação da versão que utiliza o *NCL-Validator* buscou atuar nas limitações da biblioteca, suas funcionalidades contemplam: a identificação de problemas léxicos, sintáticos e semânticos estáticos e a rastreabilidade para os trechos de código. Desta forma, foi necessário atuar para prover a rastreabilidade para as assertivas estruturais que justificam a utilização da regra de validação implementada. Esta atividade foi facilitada, pois a biblioteca também utilizava uma árvore sintática abstrata (*AST - Abstract Syntax Tree*) para estruturar a aplicação em memória, onde cada nó desta árvore implementava suas regras de validação. Além disto, cada regra de validação referenciava explicitamente o identificador para a sua mensagem de erro, como, por exemplo, o identificador "3301" que diz respeito a regra de validação que verifica se os conectores (<*causalConnector*>) possuem identificadores únicos (Figura 0.3). Desta forma, foi necessário criar constantes que consolidassem o identificador da mensagem de erro com o identificador da assertiva (*token*) para que esta pudesse ser referenciada explicitamente pela assertiva de validação (Figura 0.4), para que os problemas identificados referenciassem essas constantes (Figura 0.5). Vale salientar que, apesar do *NCL-Validator* dar suporte a inúmeras assertivas de validação estrutural, a rastreabilidade foi provida apenas para as assertivas selecionadas, apresentadas no

Quadro 0.2.

```
if (this.roles.containsKey(strRole)) {
    Vector<String> args = new Vector<String>();
    args.add(strRole);
    args.add(eCausalConnector.getAttribute("id"));

    MessageList.addError(doc.getId(), 3301, childElement,args);
    ret = false;
}
```

Figura 0.3. Exemplo de identificação de um problema estrutural pelo *NCL-Validator*

```
public enum EnumStructuralToken {

    //CausalConnector
    UNIQUE_RULE_CAUSAL_CONECTOR_ID_TOKEN(3301); // O valor do atributo role '%s' deve ser único no <causalConnector> com id '%s'.

    int idMsg;

    private EnumStructuralToken(    int idMsg) {
        this.idMsg = idMsg;
    }

    public int getIdMsg() {
        return idMsg;
    }

    public void setIdMsg(int idMsg) {
        this.idMsg = idMsg;
    }
}
```

Figura 0.4. Constante que representa o identificador da (*token*) da assertiva possui o identificador da mensagem de erro.

```
if (this.roles.containsKey(strRole)) {
    Vector<String> args = new Vector<String>();
    args.add(strRole);
    args.add(eCausalConnector.getAttribute("id"));

    MessageList.addError(doc.getId(), EnumStructuralToken.UNIQUE_RULE_CAUSAL_CONECTOR_ID_TOKEN, childElement,args);
    ret = false;
}
```

Figura 0.5. Trecho do código após a implementação da rastreabilidade de assertivas no *NCL-Validator*

Com o objetivo de facilitar o acesso do módulo de análise estrutural, durante o processo de desenvolvimento e manutenção das aplicações, foi provida uma integração com um ambiente de desenvolvimento de aplicações desse contexto. Desta forma, a fim de usufruir das funcionalidades providas pelo ambiente NCL Eclipse (AZEVEDO *et. al.*, 2009) e dos benefícios de integração da plataforma Eclipse, a solução foi encapsulada na forma de plug-ins dessa plataforma. Com isso, buscou prover uma extensão do editor próprio da linguagem, acessível por meio de uma aba que apresente os resultados da análise estrutural (*Structural Model*), fornecendo a identificação dos problemas estruturais relacionados com a referência para os trechos de código impactado,

permitindo a identificação das assertivas que justificaram a análise, bem como os trechos da especificação referenciados por ela (Figura 0.6). Vale salientar que os problemas estruturais também são destacados no próprio editor textual, recurso comumente utilizado na plataforma Eclipse.

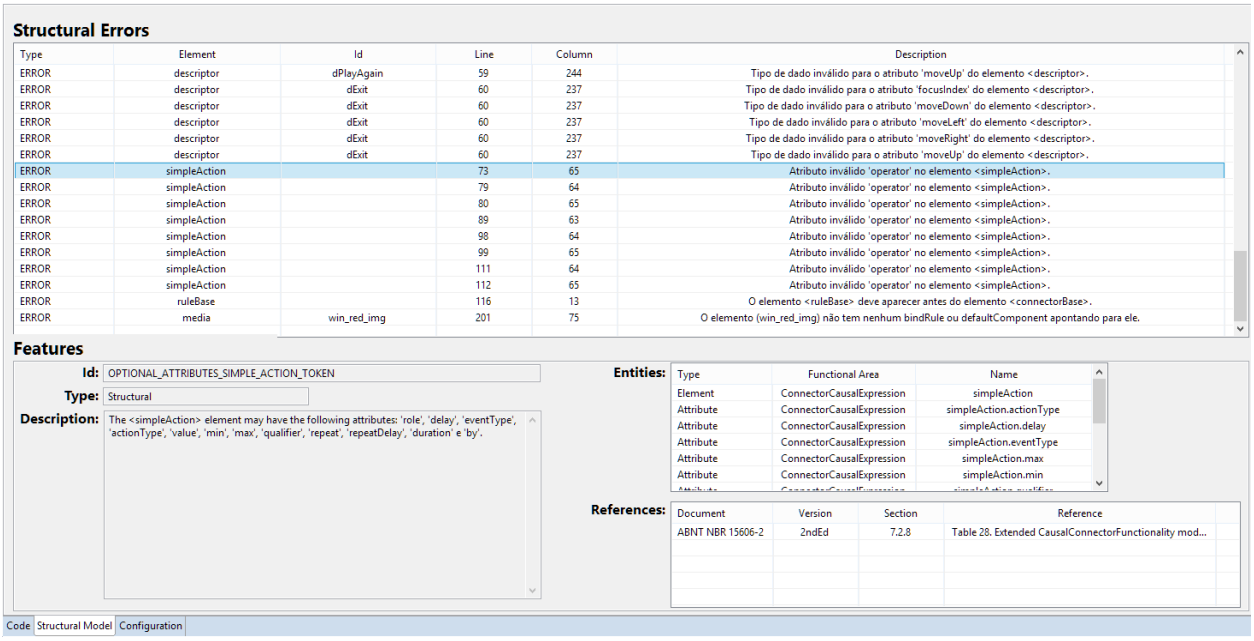

Figura 0.6. Interface do módulo de análise estrutural

1.1.2 Análise Comportamental

Os requisitos da análise comportamental (Quadro 0.3) foram definidos pelas funcionalidades propostas pela abordagem, pelas funcionalidades e limitações encontradas nos trabalhos relacionados e pelas orientações de Akiyama, Oore e Watters (2014). Desta forma, buscou evidenciar a viabilidade técnica na implementação das funcionalidades da abordagem proposta, além de prover recursos produtivos e atrativos integrados a um ambiente de desenvolvimento de aplicações multimídia.

Quadro 0.3. Requisitos funcionais do módulo de análise comportamental

Id	Requisito Funcional	Relevância
RF001	Permitir a identificação de inconsistências referentes à linguagem (Padrão)	Essencial
RF002	Permitir a identificação de inconsistências referentes à aplicação (Própria)	Essencial
RF003	Permitir a identificação de inconsistências comportamentais (Interativas)	Essencial
RF004	Permitir a identificação de inconsistências descritivas	Essencial

	(Descritivas)	
RF005	Prover a rastreabilidade para os trechos de código	Essencial
RF006	Prover a rastreabilidade para as assertivas	Essencial
RF007	Fornecer insumos para a correção dos problemas identificados	Essencial
RF008	Apresentar as assertivas que estão sendo utilizadas na análise comportamental	Importante
RF009	Prover a rastreabilidade para as assertivas que estão associadas com a aplicação	Importante
RF010	Permitir a configuração do escopo da análise estrutural referente ao código da aplicação	Importante
RF011	Permitir a configuração do escopo da análise estrutural referente às assertivas comportamentais	Importante
RF012	Informar a presença de heurísticas comportamentais nas aplicações que determinam as boas práticas	Opcional
RF013	Informar a presença de heurísticas descritivas nas aplicações que determinam as boas práticas	Opcional
RF014	Permitir estender a base de heurísticas comportamentais	Opcional
RF015	Permitir estender a base de heurísticas descritivas	Opcional

Assim como no módulo de análise estrutural, para o desenvolvimento desta primeira versão da solução, foram considerados apenas os requisitos funcionais categorizados como essenciais, pelo fato desta solução servir para provar a viabilidade técnica da abordagem proposta. Porém, para facilitar a extensão da solução, todos os requisitos foram considerados para o projeto da arquitetura. É importante destacar outra limitação dessa implementação: para evitar tratar as inconsistências estruturais durante o processo de análise comportamental, definiu-se que a análise comportamental só seria realizada se a aplicação não possuísse problemas estruturais.

A arquitetura da solução foi projetada conforme a abordagem proposta, desta forma buscou definir componentes responsáveis por cada etapa (Figura 0.7):

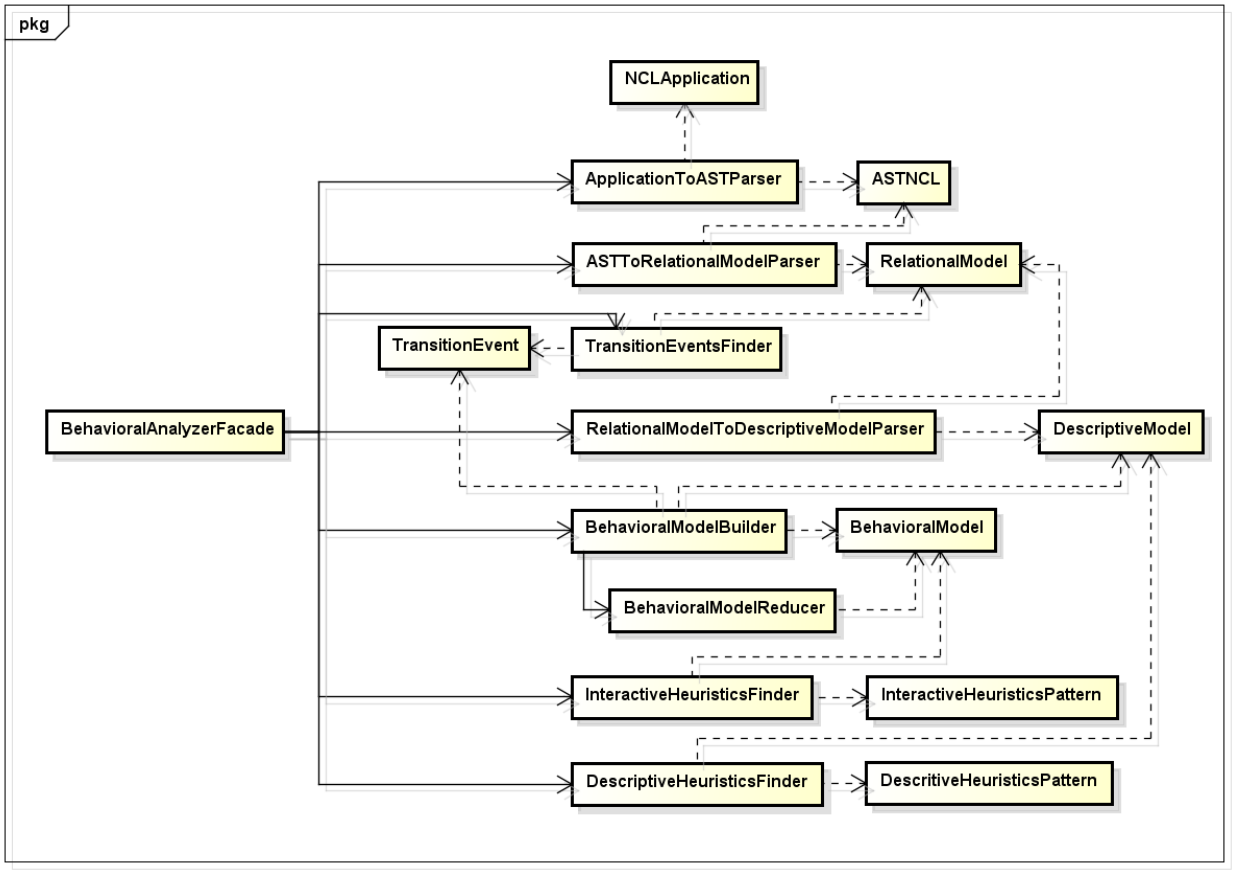

Figura 0.7. Principais componentes da arquitetura da Análise Comportamental

- **ApplicationToASTParser** – Mesmo componente utilizado pela módulo estrutural, realiza a tradução da aplicação para uma árvore sintática abstrata (*Abstract Syntax Tree - AST*) que representa o NCL;
- **ASTToRelationalModelParser** – Realiza a tradução da AST NCL para o Modelo de Relacionamentos por meio da implementação do padrão de projeto *Visitor* (GAMMA *et. al.*, 1994) que permite caminhar em todos os nós da AST realizando as traduções necessárias. Como definido pela abordagem proposta, esta etapa visa eliminar o "açúcar sintático" da linguagem traduzindo a aplicação para uma estrutura mais concisa. Vale salientar que cada entidade do Modelo de Relacionamentos (*RelationalModel*) fornece acesso aos *tokens* das assertivas que justificaram a sua construção;
- **TransitionEventsFinder** – Responsável por identificar todos os eventos possíveis (*TransitionEvent*) de ocorrer na aplicação. Este componente também foi implementado por meio do padrão de projeto *Visitor*, realizando investigações em cada nó com o

objetivo de identificar as possíveis transições. Como definido pela proposta, o *TransitionEventsFinder* procura observar nós específicos do Modelo de Relacionamentos (ex.: as estruturas relacionais que representam os Links) com o objetivo de criar os *TransitionEvent*, mesmo que eles não estejam sendo utilizados pela aplicação. Vale salientar que, cada tipo de evento (ex.: Iniciação, Temporal e Seleção) possui uma classe concreta parametrizável que estende a classe abstrata *TransitionEvent*. Esta classe também é responsável por coletar e fornecer os *tokens* das assertivas (Comportamentais) utilizados pelas partes condicionais ou acionáveis;

- **RelationalModelToDescriptiveModelParser** – Realiza a tradução do Modelo de Relacionamentos para o Modelo Descritivo Inicial. Este componente também foi implementado por meio do padrão de projeto *Visitor*, realizando as traduções necessárias. Como definido pela abordagem proposta, o Modelo Descritivo (*DescritptiveModel*) representa o domínio de variáveis da aplicação e tem um papel fundamental para a geração do Modelo Comportamental (*BehavioralModel*). Além disto, é importante destacar que cada nó do Modelo Descritivo fornece acesso aos *tokens* descritivos relacionados com sua interpretação.
- **BehavioralModelBuilder** – Responsável por construir o Modelo Comportamental (*BehavioralModel)* identificando os estados existentes por meio dos eventos possíveis e da interação deles com os Modelos Descritivos gerados. Assim como definido pela abordagem proposta, este componente busca realizar ciclos iterativos a fim de identificar todos os estados S (*State*) e transições T (*Transition*). O estado inicial s_0 é constituído pelo modelo descritivo inicial, d_0, fornecido pelo componente *RelationalModelToDescriptiveModelParser*, enquanto os eventos possíveis, E, são fornecidos pelo componente *TransitionEventsFinder*. Além disto, executa os algoritmos de redução por meio do componente *BehavioralModelReducer*.;
- **InteractiveHeuristicsFinder** – Componente não implementado para esta primeira versão da solução. É responsável por identificar as inconsistências existentes no Modelo Comportamental. Pretende-se utilizar alguma API que facilite a pesquisa em grafos, como, por exemplo, a JGraph[8] . Desta forma, após a construção completa do

[8] JGraph Documentation. Disponível em: <jgraph.github.io/mxgraph/>. Acesso em: 20 de Setembro de 2016.

Modelo Comportamental busca investigar a existência de cada padrão interativo implementado (*InteractiveHeuristicsPattern*); e

- **DescriptiveHeuristicsFinder** – Componente não implementado para esta primeira versão da solução. É responsável por identificar as inconsistências existentes no modelo descritivo de cada estado. Desta forma, após a geração de cada estado busca investigar a existência de cada padrão de heurística descritivo implementado (*DescritiveHeuristicsPattern*).

A implementação considerou algumas assertivas relacionais, comportamentais e descritivas com o objetivo de avaliar a viabilidade técnica do projeto da solução. Desta forma, foram consideras apenas as regras apresentadas no Quadro 0.4. Sendo assim, a implementação da solução permitiu a identificação de problemas comportamentais dependentes dessas regras, além de prover a rastreabilidade para as assertivas e para os trechos de código relacionados. Vale salientar que, apesar dos modelos abstratos implementados fornecerem todo o conhecimento necessário para a identificação de inconsistências (padrões, próprias, interativas e descritivas), é necessário prover interfaces usáveis para facilitar a identificação dessas inconsistências.

Quadro 0.4. Regras comportamentais selecionadas para implementação

Regra	Descrição	Exemplo de Entidades Relacionadas
Importação	Responsável por definir as regras de relacionamento referentes à importação de base	connectorBase:importBase regionBase:importBase
Valor Padrão	Responsável por definir as regras de relacionamento referentes valor *default* dos atributos	region.top region.height
Parametrização	Responsável por definir as regras de relacionamento referentes à parametrização de atributos	bind:bindParam link:linkParam
Referência	Responsável por definir as regras de relacionamento referentes à referência de atributos	descriptor.region media.descriptor
Eventos de Apresentação	Responsável por definir as regras comportamentais referentes aos comportamentos de apresentação	simpleCondition.role = “onBegin” simpleAction.role = “stop”
Eventos de Atribuição	Responsável por definir as regras comportamentais referentes aos comportamentos de atribuição	simpleCondition.role = “onEndAttribution” simpleAction.role = “set”
Eventos de Seleção	Responsável por definir as regras comportamentais referentes aos	simpleCondition.role = “onSelection” simpleCondition.key = “GREEN”

	comportamentos de seleção	
Eventos Temporais	Responsável por definir as regras comportamentais referentes aos comportamentos temporais	area.begin area.end
Eventos de Foco	Responsável por definir as regras comportamentais referentes aos comportamentos de foco	descriptor.focusIndex
Exibição de mídias	Responsável por definir as regras descritivas referentes à exibição das mídias	media.type = "image/png" media.src = "sbtvd-ts://video"
Exibição de propriedades	Responsável por definir as regras descritivas referentes à exibição das propriedades	property.name property.value
Exibição de propriedades abstratas	Responsável por definir as regras descritivas referentes à exibição das propriedades abstratas	media.state property.name="service.currentFocus"

Com o objetivo de facilitar o acesso do módulo de análise comportamental, durante o processo de desenvolvimento e manutenção das aplicações, assim como o módulo estrutural, também foi provida uma integração com o ambiente NCL Eclipse (AZEVEDO *et. al.*, 2009). Com isso, buscou prover uma extensão do editor próprio da linguagem, acessível por meio de abas que apresentem os recursos da análise comportamental, permitindo a identificação dos problemas comportamentais, fornecendo as referências para os trechos de código impactado, além de fornecer acesso às assertivas que justificaram a análise, bem como os trechos da especificação referenciados por ela.

A interação com o módulo comportamental da solução é definida por três abas: interface de configuração, visualização do modelo de relacionamentos e visualização do modelo comportamental. A primeira aba, denominada *"Configuration"*, visa prover interfaces para a escolha no nível de análise, atualmente categorizado por LOW, MEDIUM e HIGH. Conforme a abordagem proposta, o nível de análise determina quais regras devem estar visíveis na apresentação dos modelos (Comportamental e Descritivo). É importante destacar que, para a prova de conceito, os níveis de análise foram utilizados apenas para avaliar a viabilidade de se aplicar as regras de redução de acordo com o nível selecionado. Para ter um melhor aproveitamento desses níveis de análise é necessário investigar os perfis do público alvo da ferramenta, para prover níveis aderentes com as expectativas de análise de cada perfil. Além de permitir a seleção do nível, a aba detalha quais regras, e consequentemente assertivas, constituem cada nível de análise (Figura 0.8).

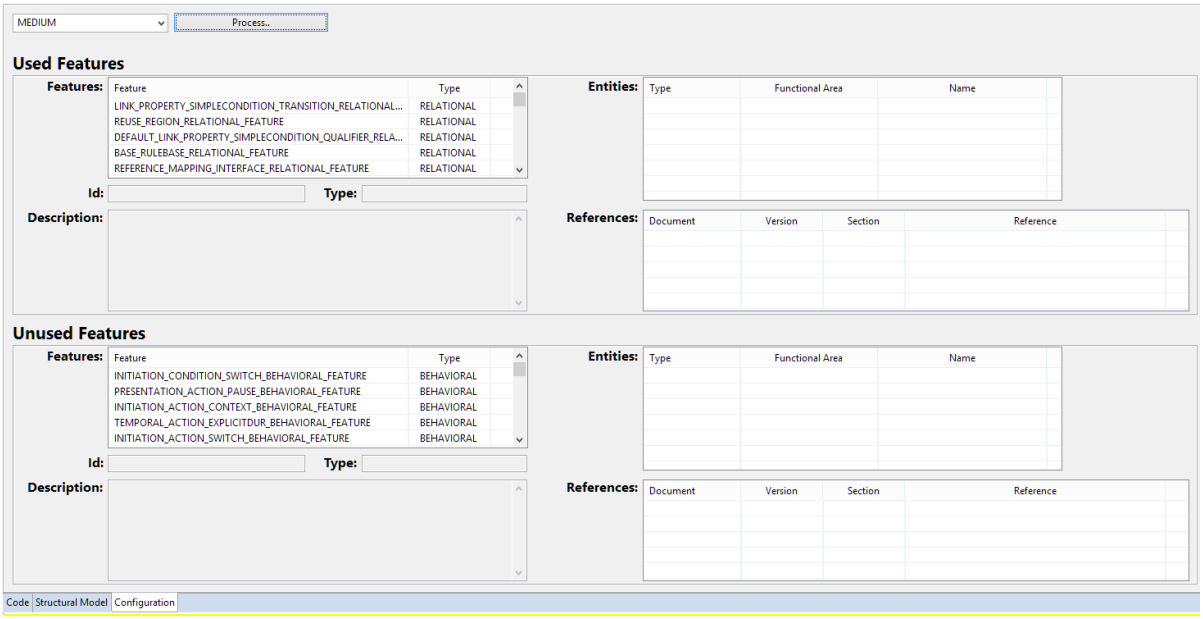

Figura 0.8. Exemplo da configuração da análise comportamental

A segunda aba, denominada *"Relational Model"*, fornece a visualização interativa para o modelo de relacionamentos, a interação neste modelo permite identificar os nós filhos e as propriedades de cada entidade do modelo, além de possibilitar a identificação e detalhamento de cada assertiva relacionada com a transformação de cada entidade (Figura 0.9).

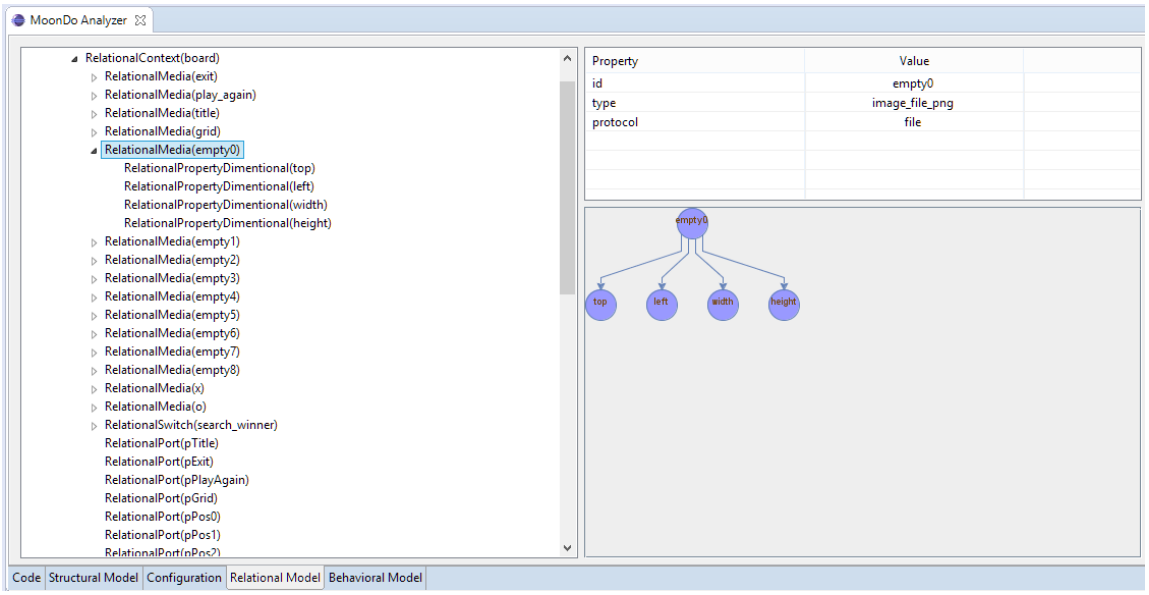

Figura 0.9. Exemplo do Modelo de Relacionamentos

A terceira aba, denominada *"Behavioral Model"*, representa a principal interface de interação do usuário, pois fornece uma série de funcionalidades que visam contribuir para a identificação das inconsistências comportamentais (Figura 0.10):

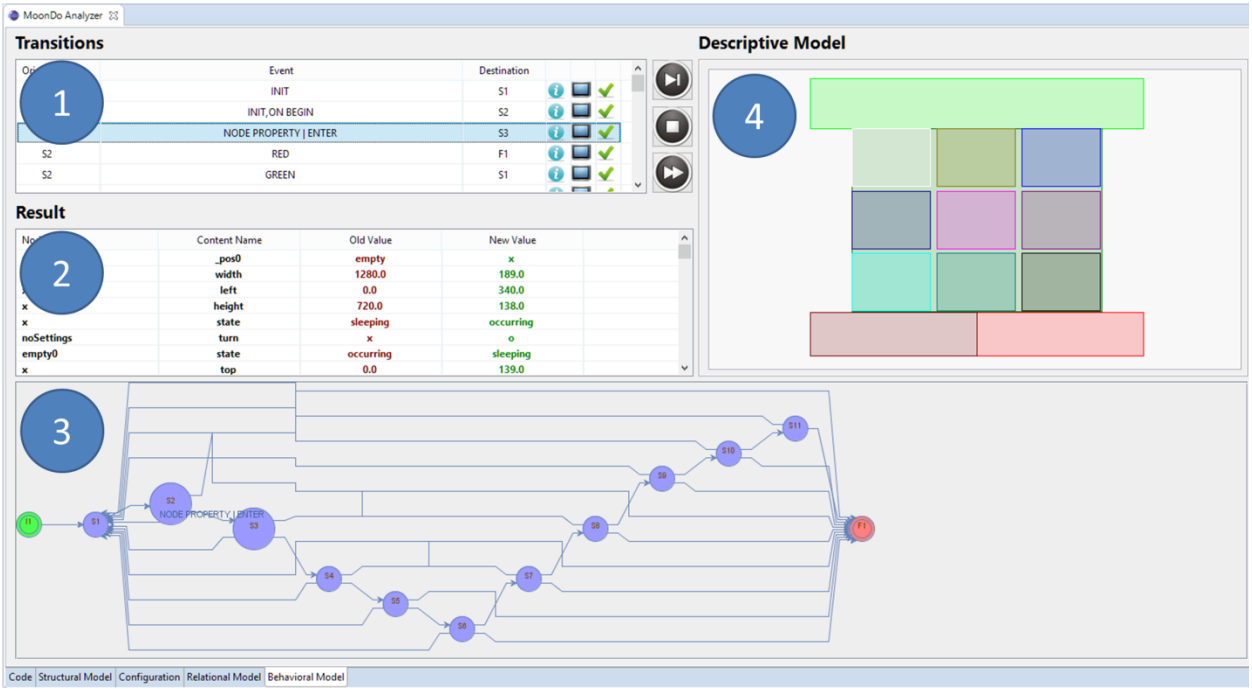

Figura 0.10. Funcionalidades da análise comportamental: (1) Visualização das Transições; (2) Visualização das Transformações Ocorridas; (3) Visualização do Modelo Comportamental; e (4) Visualização do Esboço do Modelo Descritivo.

- **Visualização das Transições** (Figura 0.10.1) – Apresenta as transições de cada estado textualmente, podendo estas ser virtuais ou concretas. Apresenta o nome do estado de origem, o *label* dos eventos de transição (condicionais) e o nome do estado de destino. A nomenclatura dos estados segue a seguinte orientação: O estado Inicial inicia com a letra "I", os estados finais iniciam com a letra "F", os estados intermediários iniciam com a letra "S" e os estados virtuais iniciam com a letra "V", seguidos de um número incremental. Esta visualização é atualizada a cada requisição da funcionalidade de construção do modelo;
- **Visualização das Transformações Ocorridas** (Figura 0.10.2) – Permite visualizar as transformações ocorridas em alguma transição. Na tabela é possível identificar as propriedades que foram alteradas, os valores antigos e os valores novos. Esta visualização é carregada sempre que uma transição é selecionada, seja por meio da Visualização de Transições ou da Visualização do Modelo Comportamental;
- **Visualização do Modelo Comportamental** (Figura 0.10.3) – Apresenta a visualização gráfica interativa do Modelo Comportamental, permitindo que o usuário interaja com o modelo detalhando as transições selecionadas com o mouse. Apresenta

as transições de cada estado visualmente, podendo estas ser virtuais (estado de destino com bordas pontilhadas), que ainda não foram acionadas, e concretas (estado de destino com bordas contínuas). Esta visualização é atualizada a cada Manipulação da Construção do Modelo Comportamental ou a cada Manipulação da Visibilidade das Transições;

- **Visualização do Esboço do Modelo Descritivo** (Figura 0.10.4) **–** Permite visualizar o esboço do Modelo Descritivo, que contém a disposição das mídias ativas. Esta visualização é carregada sempre que uma transição é selecionada, seja por meio da Visualização de Transições ou da Visualização do Modelo Comportamental;
- **Manipulação da Construção do Modelo Comportamental –** Permite manipular a construção do Modelo Comportamental. Desta forma, o usuário pode: **Avançar** – aciona a transição virtual corrente; **Avançar Rápido** – acionam as transições geradas automaticamente até a geração completa do modelo; e **Parar** – interrompe o acionamento automático das transições. Para versões futuras da ferramenta, pretende-se estender esta funcionalidade para incluir mais duas ações: **Avançar Seleção** – aciona a transição virtual selecionada; e **Voltar** – desfaz a última transição acionada. Após qualquer uma dessas ações, as Visualizações do Modelo Comportamental e das Transições são atualizadas;
- **Detalhamento da Transformação Ocorrida na Transição –** Permite detalhar cada uma das transições concretas, fornecendo a visualização completa, visual e textual, dos Modelos Descritivos da origem e do destino. Este detalhamento é exibido em uma janela *popup* após a seleção do ícone da TV ( ) na transição que deseja detalhar;
- **Detalhamento das Assertivas Relacionadas com a Transição –** Permite detalhar as assertivas relacionadas com cada uma das transições concretas, apresentando as informações que constituem a assertiva. Este detalhamento é exibido em uma janela *popup* após a seleção do ícone de informação ( ) na transição que deseja detalhar; e
- **Manipulação da Visibilidade das Transições –** Permite alterar a visibilidade de qualquer transição, possibilitando esconder as transições irrelevantes ou destacar as relevantes durante a investigação. Ao selecionar o ícone de visibilidade ( ) ou de invisibilidade ( ), a Visualização do Modelo Comportamental é atualizada.

A implementação da prova de conceito permitiu investigar a viabilidade técnica da abordagem proposta, além de permitir a avaliação da viabilidade funcional para alguns contextos de uso (apresentada na próxima subseção). Com o objetivo de apresentar todas a utilização da ferramenta na prática, foi disponibilizado um vídeo exemplo[9]..

1.2 Contextos de Uso

Esta seção busca idealizar um cenário de uso a fim de evidenciar as contribuições e limitações deste módulo da solução. Durante o processo de desenvolvimento deste módulo, alguns testes foram realizados a fim de confrontar os resultados obtidos pela solução com os resultados esperados. Esses testes tiveram como objetivo principal avaliar a confiabilidade na identificação dos problemas, bem como identificar *bugs* e oportunidades de melhoria na solução. Além disto, também permitiu investigar a viabilidade da solução sob alguns aspectos: Técnico – se a implementação da abordagem proposta é viável para a construção de soluções concretas; e Funcional – se resultados da análise agregam valor na identificação de problemas estruturais e comportamentais. Para este processo foi necessário, inicialmente, definir o conjunto de aplicações a ser utilizado, contendo: aplicações reais com problemas existentes conhecidos; aplicações manipuladas contendo os mais diversos tipos de problemas (estruturais e comportamentais); e aplicações corretas (sem problemas). Para estes dois últimos grupos de aplicações foi necessário desenvolver aplicações testes, a fim de manipular a existência dos diversos tipos de problema e de criar interações que pudessem ser visualmente ilegíveis (explosão de estados). As seções a seguir apresentam os resultados obtidos nas análises de algumas aplicações, a fim de evidenciar os principais recursos dos módulos de análise estrutural e comportamental. As aplicações apresentadas foram selecionadas empiricamente a fim de evidenciar as principais funcionalidades da ferramenta, bem como destacar suas as potencialidades e limitações.

1.2.1 Análise Estrutural

Para a análise estrutural, optou-se por apresentar neste trabalho os resultados obtidos através do uso da solução no processo de manutenção de uma aplicação real denominada HardRace[10],

---

9 Vídeo Exemplo da Ferramenta de Análise Estrutural e Comportamental de Aplicações NCL. Disponível em: <https://dl.dropboxusercontent.com/u/5228583/Doutorado/Video-Exemplo.mp4>. Acesso em: 20 de Setembro de 2016.

10 Aplicação *HardRace*. Disponível em: <club.ncl.org.br/node/2>. Acesso em: 20 de Setembro de 2016.

disponibilizada pelo repositório público Clube NCL [11]. Com isso, buscou-se evidenciar os benefícios no processo de identificação de erros estruturais.

A aplicação **HardRace** foi desenvolvida pelo Laboratório Telemídia[12] e disponibilizada pelo Clube NCL. Ela consiste em uma corrida de cavalos onde o usuário deve apostar em um cavalo favorito e torcer para que ele cruze a linha de chegada em primeiro. A interação nesta aplicação utiliza basicamente as teclas coloridas, tanto para selecionar os cavalos ("RED", "GREEN", "YELLOW" e "BLUE") quanto para iniciar a corrida ("GREEN") e terminar a aplicação ("RED"). A análise estrutural desta aplicação permitiu identificar 58 erros estruturais ao longo do código da aplicação (Figura 0.11), divididos em três categorias. O Quadro 0.5 apresenta a distribuição dos erros pela categoria.

Quadro 0.5. Quantitativo de erros estruturais de acordo com sua categoria, existentes na aplicação HardRace

Regra	**Quantidade de Erros**
Opcional	9
Unicidade	19
Valor Possível	30

A rastreabilidade da assertiva é fornecida para cada erro identificado, permitindo que o usuário possa complementar o entendimento do problema identificado, além de justificar a acusação do erro através da evidência da conformidade com a especificação (Quadro 0.5). Vale salientar, que, para esta versão da implementação, as assertivas estruturais da categoria "Valor Possível" não foram mapeadas devido à limitação do escopo da solução, impossibilitando a sua rastreabilidade.

11 Repositório público Clube NCL. Disponível em: <clube.ncl.org.br>. Acesso em: 20 de Setembro de 2016.
12 Laboratório Telemídia. Disponível em: <www.telemidia.puc-rio.br>. Acesso em: 20 de Setembro de 2016.

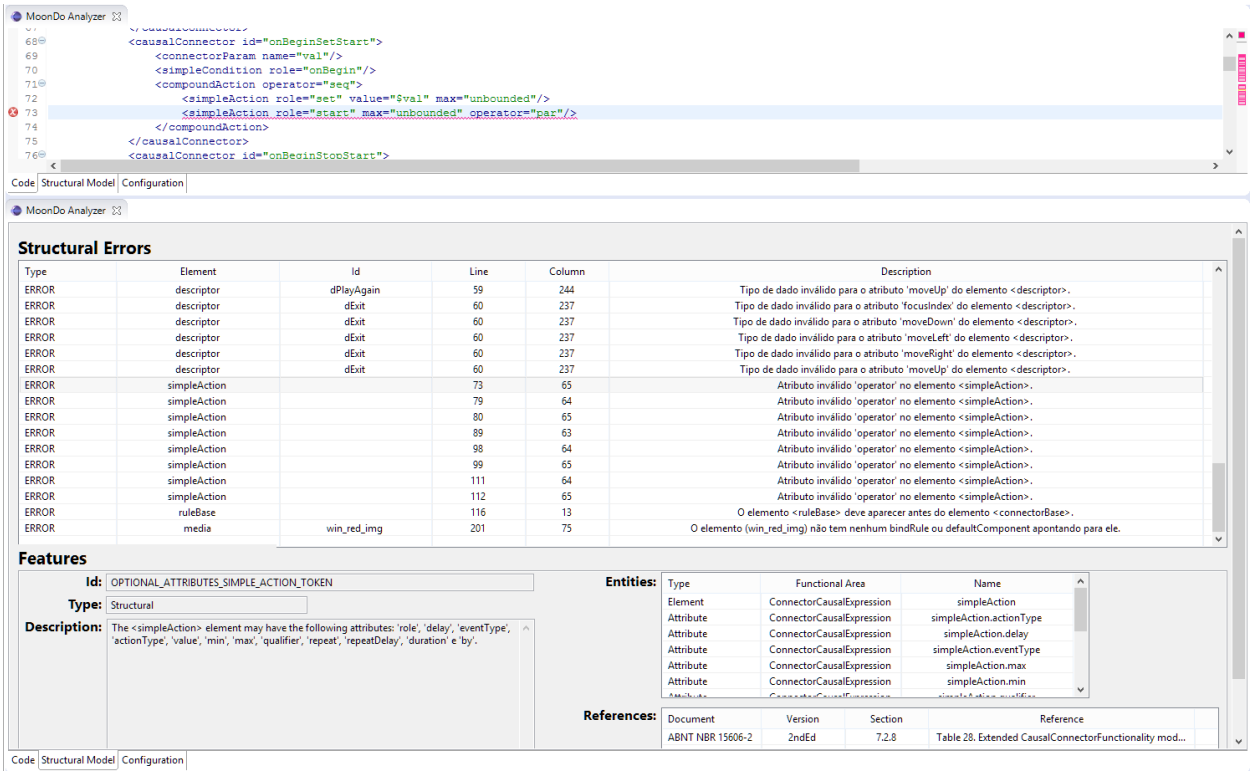

Figura 0.11. Exemplo de problemas estruturais encontrados na aplicação HardRace

1.2.2 Análise Comportamental

Para a análise comportamental, optou-se por apresentar neste trabalho os resultados obtidos da análise de duas aplicações reais: Supermercado[13] e TicTacToe[14]. Com isso, buscou-se evidenciar os benefícios no processo de identificação de erros comportamentais, a partir da análise da aplicação Supermercado, e apresentar a potencialidade da representação do modelo comportamental, a partir da análise da aplicação TicTacToe.

A aplicação **Supermercado** foi desenvolvida por terceiros, seu objetivo é divulgar as ofertas da semana de um determinado supermercado, permitindo a interação do usuário entre os diversos produtos a fim de fornecer algumas informações, tais como: quantidade, preço, marca e foto ilustrativa. Apesar dos requisitos da aplicação não terem sido fornecidos pelo desenvolvedor, a análise comportamental permitiu identificar alguns problemas (Descritivos e Interativos) existentes nesta aplicação.

---

13 Aplicação Supermercado. Disponível em: <gingarn.wikidot.com/gingancl>. Acesso em: 20 de Setembro de 2016.

14 Aplicação TicTacToe. Disponível em: <club.ncl.org.br/node/6>. Acesso em: 20 de Setembro de 2016.

Primeiramente, foi possível identificar alguns problemas comportamentais descritivos, dado que a aplicação está fazendo uso de valores absolutos limitados pela área de 800x600 *pixels* para definir a disposição de seus componentes visuais, criando desta forma interfaces reduzidas que não aproveitam toda área do dispositivo (Figura 0.12). Para corrigir este problema o desenvolvedor teria que utilizar valores percentuais, ou, se preferir trabalhar de forma absoluta, poderia utilizar a resolução padrão HD, 1280x720 *pixels*.

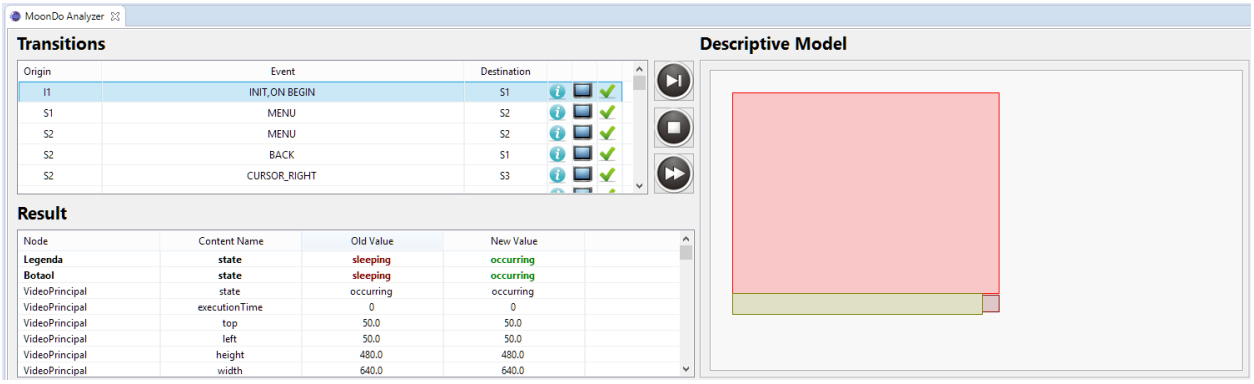

Figura 0.12. Problema descritivo existente na aplicação Supermercado

Também foi possível identificar, por meio do modelo comportamental, a existência de um problema comportamental interativo. Ao observar o modelo comportamental extraído, é possível inferir que a interação na aplicação entre os produtos é determinada pelos botões direcionais "LEFT" e "RIGTH". Porém, é possível perceber que ao alcançar o estado S9, onde é exibido o produto "promo5.png", apenas a interação da tecla "RIGTH" é permitida, impedindo que o usuário retorne para o produto "promo4.png" por meio do pressionamento da tecla "LEFT" (Figura 0.13). Ao acessar o código (Figura 0.14), identificou-se que aparentemente ocorreu um erro de cópia de código. Para corrigir este problema o desenvolvedor teria que substituir o valor da mídia que determina a condição dessa interação, que foi colocada equivocadamente, de "promo3" para "promo5" (Figura 0.15). Como isso, o modelo comportamental extraído apresenta o comportamento consistente com a aplicação (Figura 0.16).

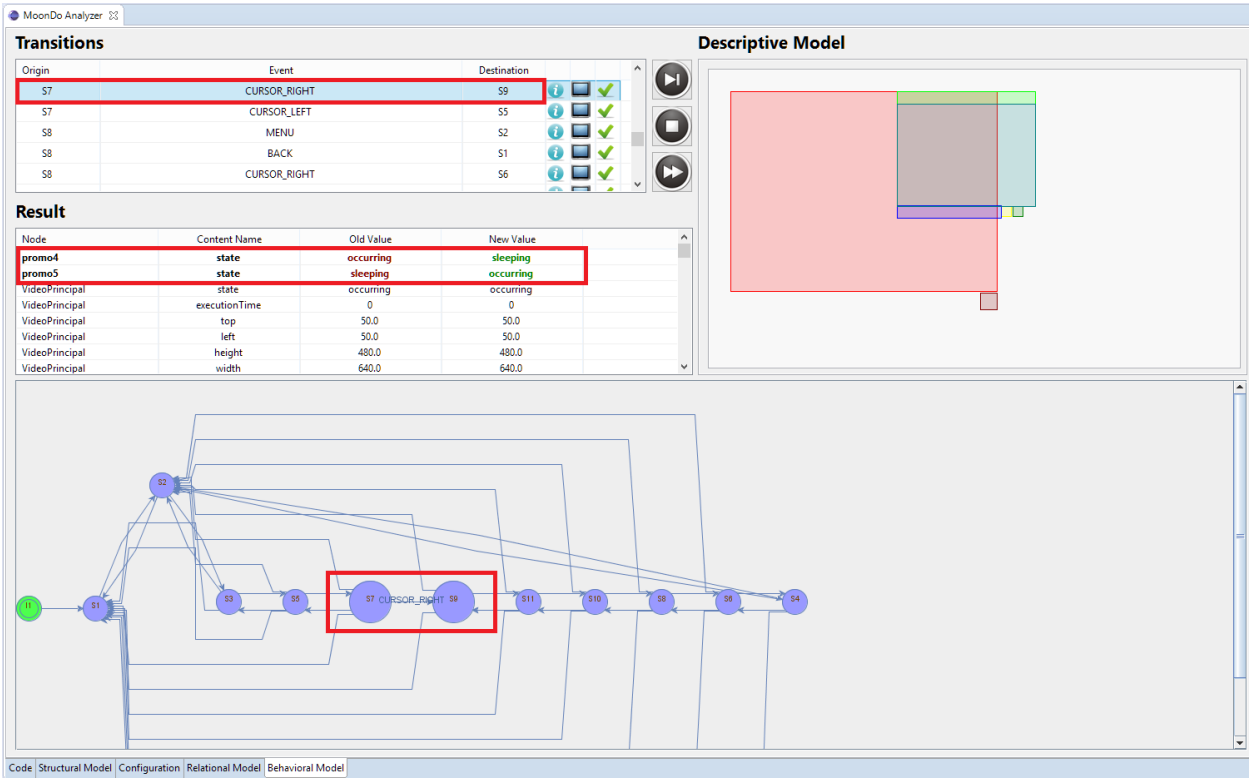

Figura 0.13. Problema interativo existente na aplicação Supermercado

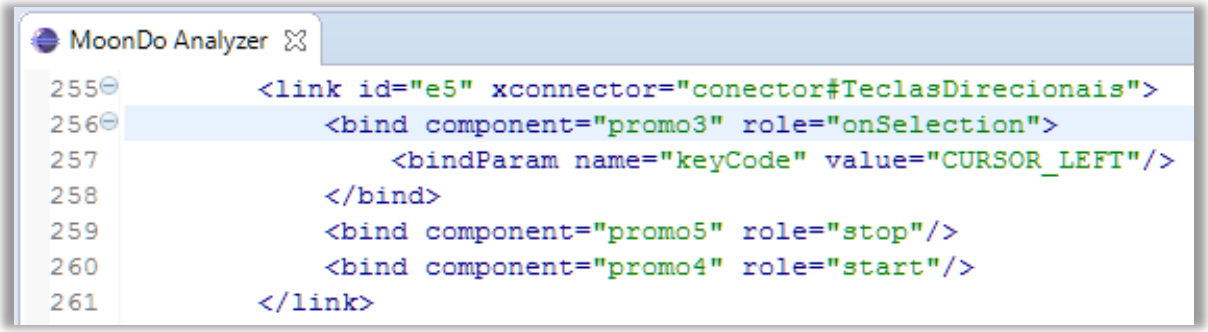

Figura 0.14. Trecho do código referente ao problema interativo existente na aplicação Supermercado

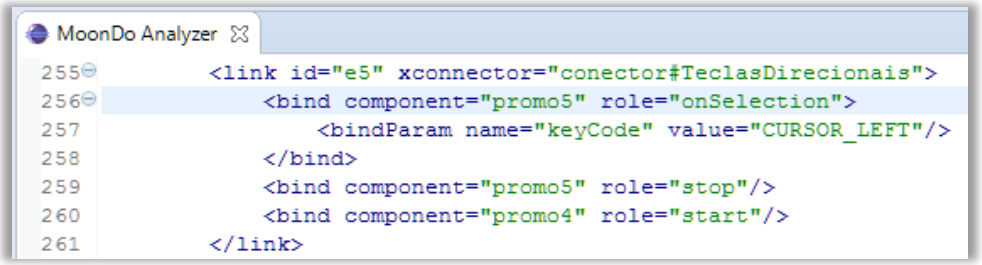

Figura 0.15. Trecho do código corrigido referente ao problema interativo existente na aplicação Supermercado

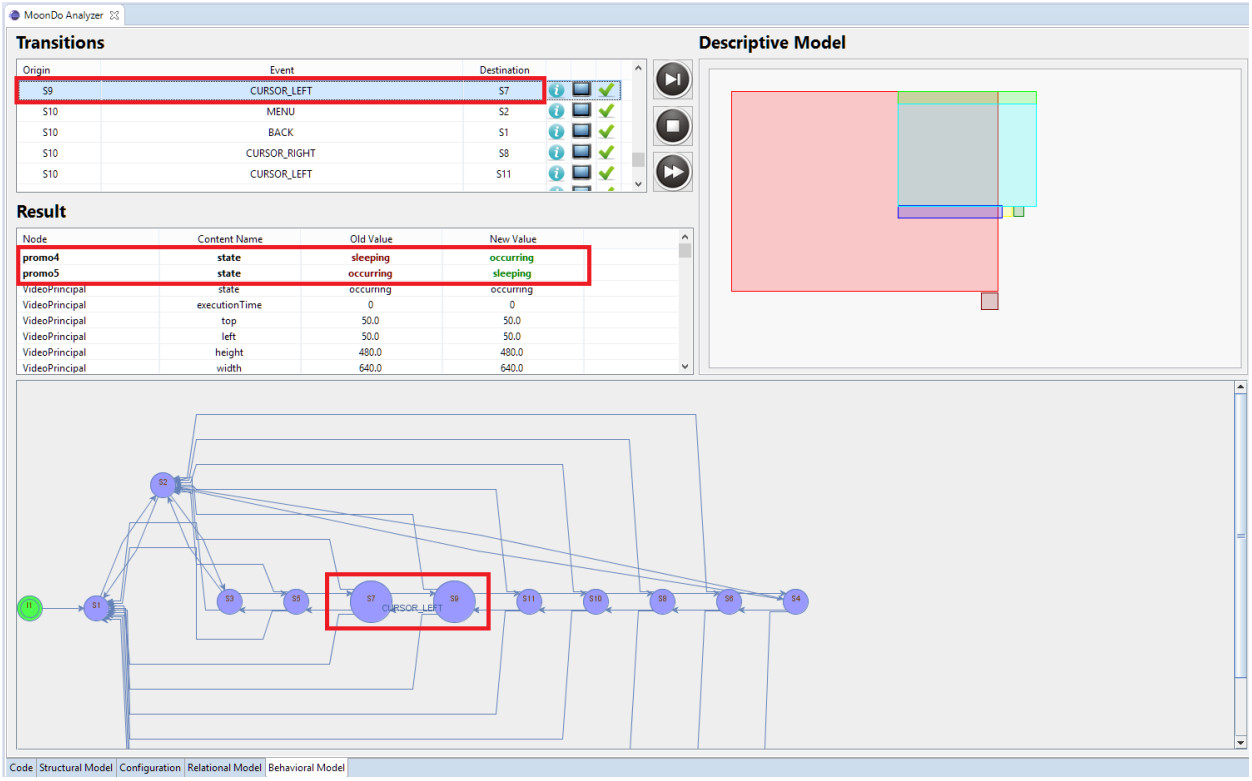

Figura 0.16. Modelo Comportamental após a correção do problema interativo existente na aplicação Supermercado

A aplicação **TicTacToe,** foi desenvolvida pelo Telemídia e disponibilizada pelo Clube NCL, representa um "Jogo da Velha" desenvolvido em NCL. Este jogo, originalmente, é realizado com lápis e papel por dois jogadores: “o” e “x”, que se revezam ao ocupar espaços em uma grade de tamanho 3x3, a fim de conseguir ocupar três espaços seguidos em uma linha vertical, horizontal ou diagonal, de modo que quem conseguir este feito ganha o jogo. A aplicação permite a interação a partir das teclas direcionais para navegação no tabuleiro e da tecla “OK” para ocupar o espaço em foco. Além disso, utilizar as teclas coloridas “GREEN” e “RED” para reiniciar e finalizar o jogo, respectivamente. Por ser um jogo popular, seus requisitos funcionais são conhecidos, e, portanto, é possível determinar a quantidade de combinações de valores que o tabuleiro pode assumir (9! = 362.880). Consequentemente, também é possível determinar a quantidade de estados e transições que um processo de tradução comportamental sem reduções geraria (como para cada estado existe 9 interações possíveis, tem-se 9^9 = 387.420.489). Esse cenário poderia inviabilizar a identificação de um problema comportamental por meio de uma representação visual, dada a magnitude do modelo gerado, problema conhecido como “explosão de estados” (WAGNER *et. al.*, 2006). A análise comportamental dessa aplicação permitiu investigar como a solução iria se comportar nesse cenário. Como era esperado, caso o desenvolvedor parametrize a configuração da análise para “HIGH”, que

considera todo e qualquer evento de interação, a solução irá apresentar problema de memória ao tentar simular todas as 387.420.489 interações. Porém, a solução permite que a configuração seja parametrizada para "MEDIUM", que desconsidera os eventos de mudança de foco, reduzindo o modelo comportamental da aplicação para 31 interações (Figura 0.17). No geral, sugere-se a eliminação deste tipo de evento do modelo comportamental, dado que, geralmente, a manipulação do foco não determina novos comportamentos funcionais.

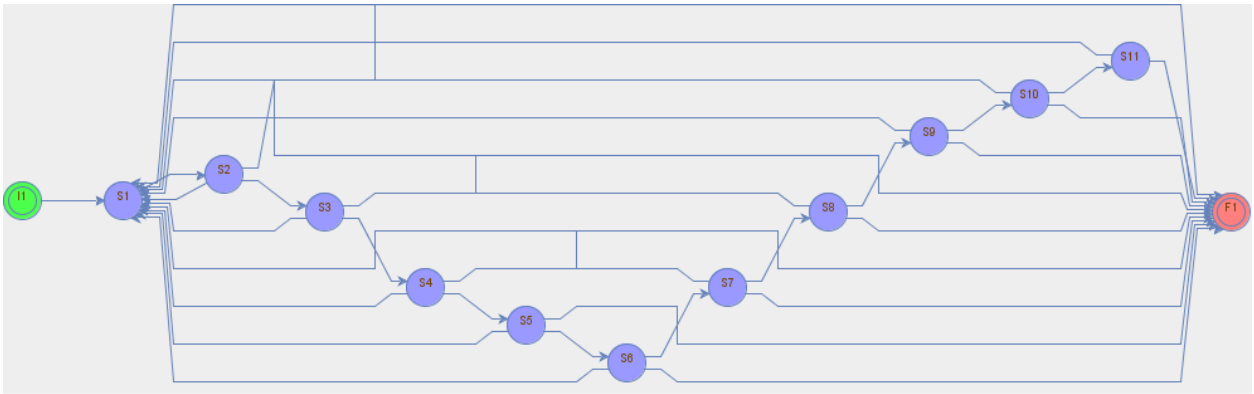

Figura 0.17. Modelo Comportamental da aplicação TicTacToe

A Figura 0.17 apresenta o modelo comportamental da aplicação TicTacToe extraído a partir da solução configurada com o grau de análise "MEDIUM". Esse modelo consiste em 13 estados, sendo 1 estado inicial, 1 estado final e 11 estados intermediários, com 32 transições. A aplicação se comporta da seguinte forma: ao iniciar vai para um estado de inicialização (S1), e posteriormente para um estado em que aguarda a primeira jogada (S2), que é realizada ao pressionar a tecla "OK"; os estados subsequentes realizam as outras jogadas possíveis que determinam a ocupação de cada um dos nove espaços existentes (S3, S5, S6, S7, S8, S9, S10, S11 e S12); pressionando, em qualquer estado intermediário, o botão verde (GREEN), a aplicação reinicia, e o botão vermelho (RED), a aplicação termina. O modelo descritivo complementa a interpretação do comportamento, descrevendo, por exemplo, qual a jogada corrente ('x' e 'o') em cada estado que aguarda a jogada (Figura 0.18).

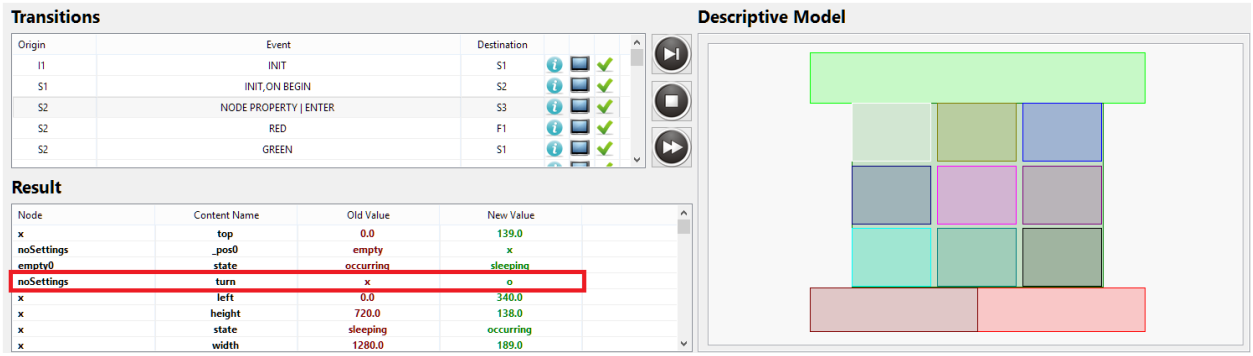

Figura 0.18. Modelo Descritivo da aplicação TicTacToe

Esta subseção apresentou alguns indícios que evidenciam a viabilidade funcional da solução, apesar de várias aplicações terem sido analisadas pela solução durante a implementação da prova de conceito, apenas algumas foram selecionadas para demostrarem os principais recursos funcionais implementados, a partir de aplicações com erros estruturais, com erros comportamentais e sem erros (com problema de explosão de estados).

1.3 Flexibilidade da Abordagem Proposta

Apesar da implementação da solução de análise de aplicações NCL sugerir a viabilidade técnica da abordagem proposta, é relevante avaliar a flexibilidade da abordagem, que foi definida para ser aplicada independentemente da linguagem de desenvolvimento. Desta forma, foi definido que a avaliação da flexibilidade seria realizada a partir da implementação de duas novas soluções de análise para outras linguagens (HTML e SVG) aderentes à abordagem proposta. Assim como na implementação da solução de análise de aplicações NCL, também foi necessário delimitar um escopo reduzido das linguagens a fim de investigar a flexibilidade com o menor esforço possível. Outra definição acerca dessa avaliação foi delimitar o escopo apenas para o módulo comportamental, dado que as características funcionais intrínsecas das linguagens podem inserir complexidades inesperadas para a implementação da abordagem. O módulo estrutural possui regras comuns a todas as linguagens (estruturais, relacionais, comportamentais e descritivas), o que torna a abordagem de análise estrutural naturalmente mais simples.

Primeiramente foi necessário classificar os componentes implementados identificando os que fazem parte do *core* (que independem da linguagem) e os que são específicos da solução implementada (dependente das especificações). Desta forma, os componentes do *core* da solução NCL foram reutilizados no desenvolvimento das novas soluções (HTML e SVG) e novos componentes foram desenvolvidos para implementar especificamente as regras definidas pelas

especificações. Vale salientar que todos os componentes da camada de apresentação foram reusados, mantendo a consistência da interface gráfica da solução independente da linguagem.

Para ambas linguagens, foi necessário estender alguns componentes do *core* responsáveis por: representar o Modelo de Relacionamentos (*RelationalModel*) e Descritivo (*DescriptiveModel*); por realizar as lógicas de tradução da *AST* para o Modelo de Relacionamentos (*ASTToRelationalModelParser*) e do Modelo de Relacionamentos para o Modelo Descritivo (*RelationalModelToDescriptiveModelParser*); e por representar os eventos de transição (passivo e ativo). Todas essas alterações foram realizadas para atender o escopo de análise definido para a solução. É importante destacar, que a implementação das soluções fez uso da API Jsoup[15] para representar as *AST* das linguagens (HTML e SVG) e para realizar a tradução das aplicações para as *AST* (*ApplicationToASTParser*) implicitamente.

Para avaliar a consistência da análise comportamental das soluções, foi definido um escopo mínimo de análise a ser implementado para possibilitar a avaliação de três versões de uma mesma aplicação, codificadas pelas três linguagens avaliadas (NCL, HTML e SVG). Desta forma, definiu-se que as soluções de análise HTML e SVG deveriam identificar apenas as transições definidas pelos hyperlinks e interpretar apenas as propriedades dimensionais dos textos e imagens, escopo já com suporte a solução de análise NCL. A Figura 0.19, Figura 0.20 e Figura 0.21 apresentam a execução das aplicações codificadas em NCL, HTML e SVG, respectivamente. Enquanto que, as figuras Figura 0.22, Figura 0.23 e Figura 0.24 apresentam seus modelos comportamentais.

Figura 0.19. Execução da aplicação codificada em NCL

15 *JSOUP API Documentation*. Disponível em: <jsoup.org>. Acesso em: 20 de Setembro de 2016.

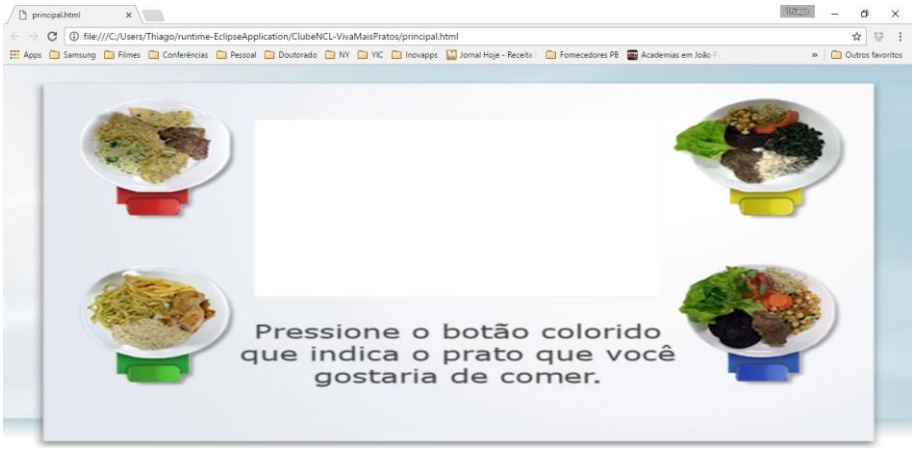

Figura 0.20. Execução da aplicação codificada em HTML

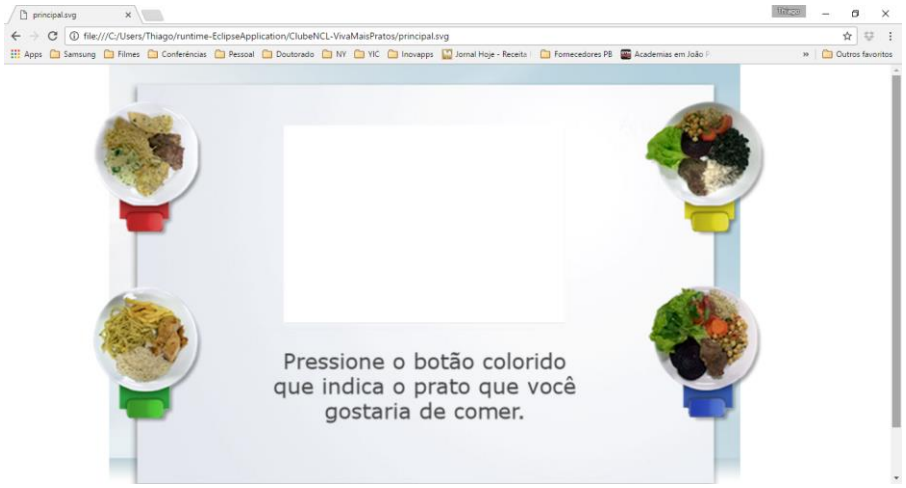

Figura 0.21. Execução da aplicação codificada em SVG

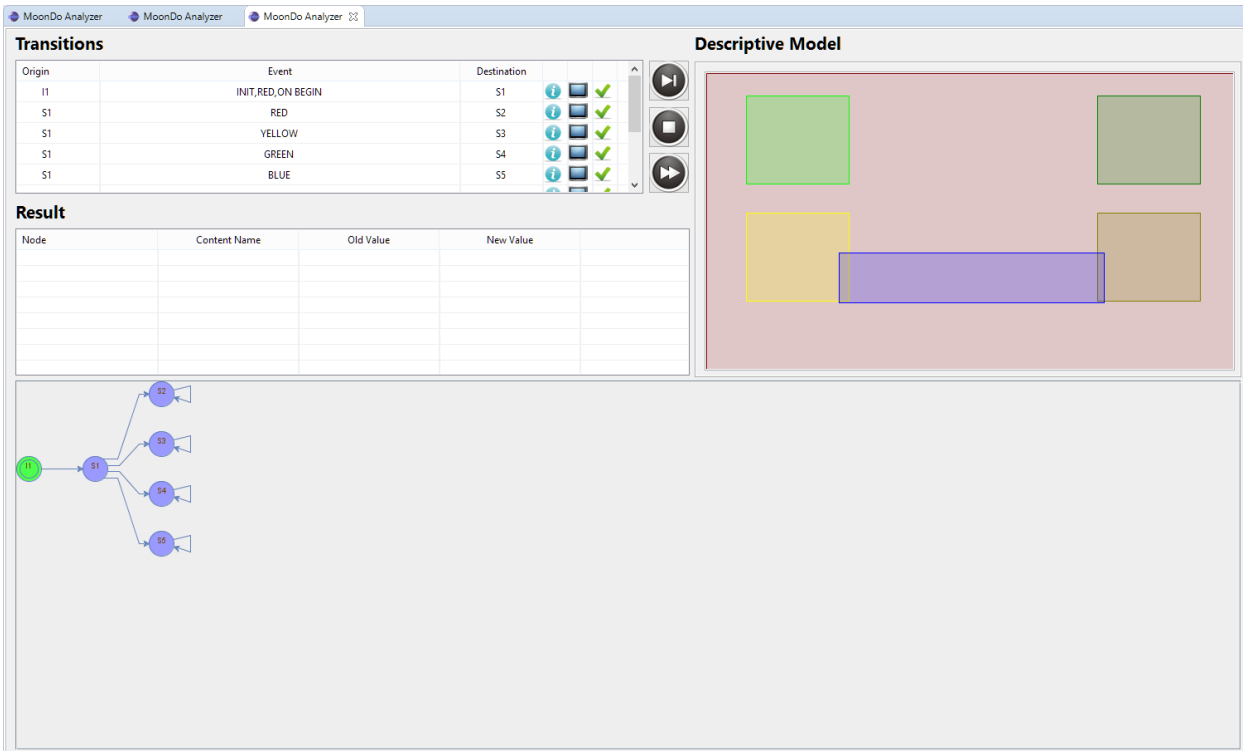

Figura 0.22. Exemplo de análise comportamental NCL

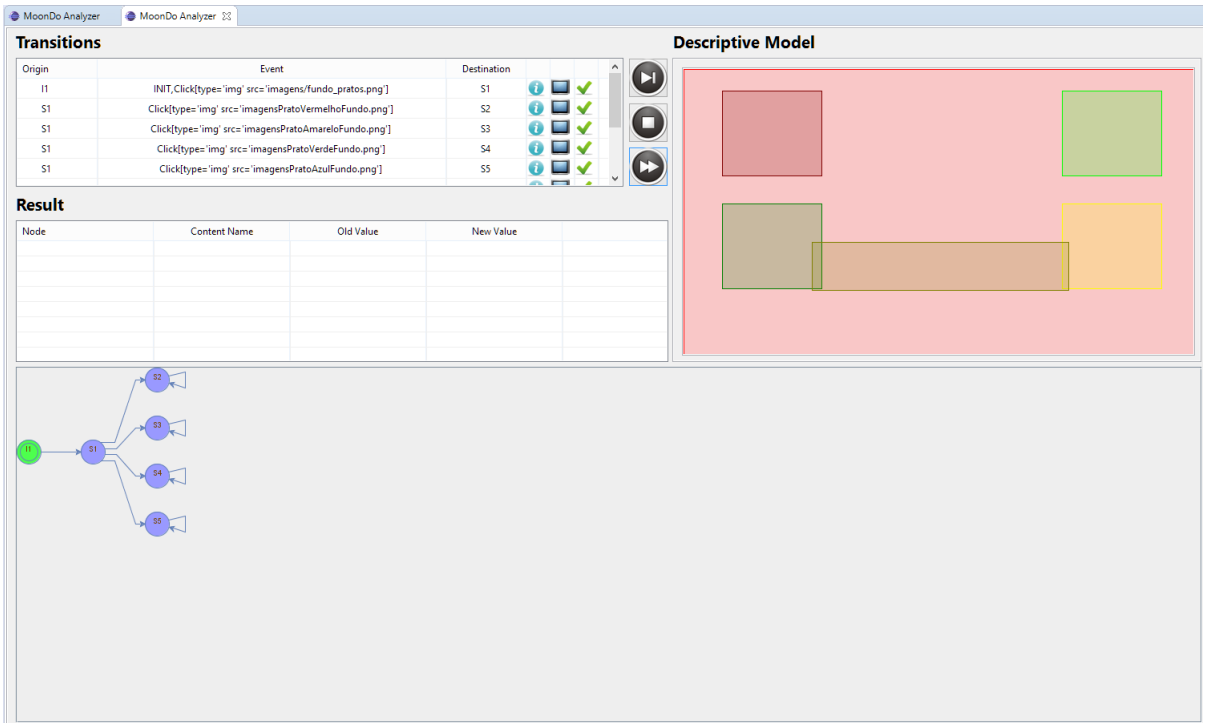

Figura 0.23. Exemplo de análise comportamental HTML

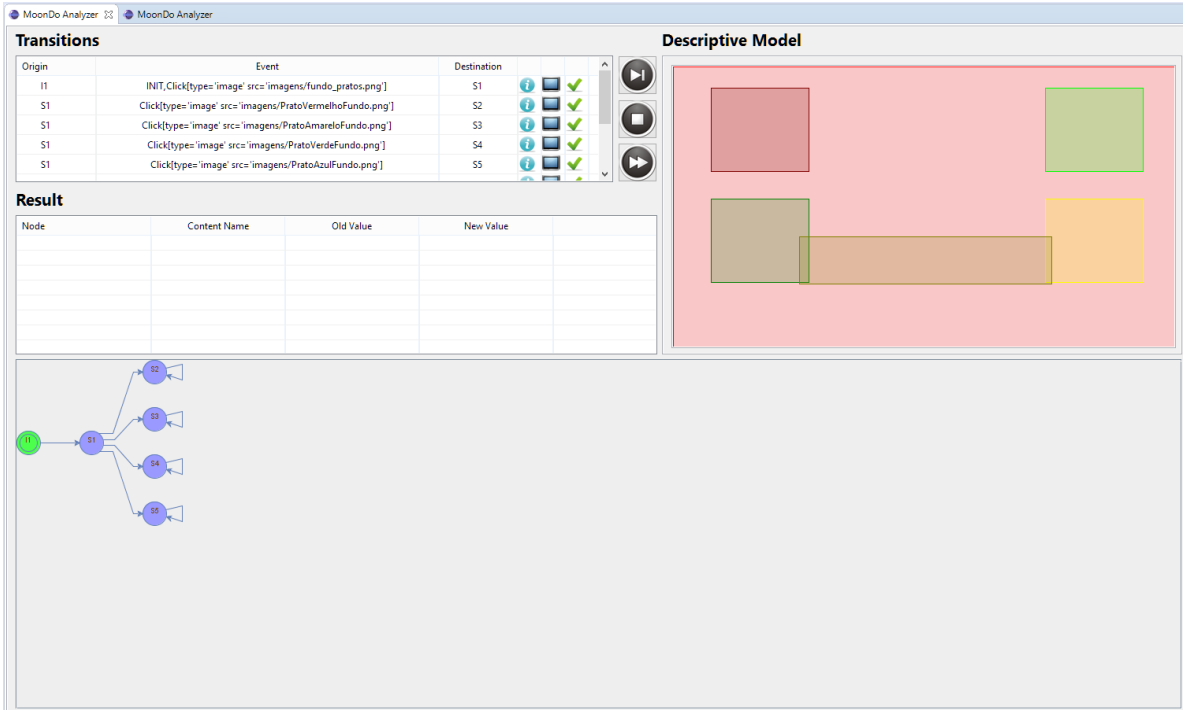

Figura 0.24. Exemplo de análise comportamental SVG

Apesar do escopo limitado, a consistência das análises pôde ser observada através dos modelos gerados. Apesar de possuir interações distintas, evidenciadas pelas transições (*Transitions*), os modelos comportamentais e descritivos gerados são equivalentes. É importante destacar que, a coloração diferente das mídias apresentadas nos esboços do Modelo Descritivo não determina que as aplicações tenham uma interface diferente. Funcionalmente o esboço deve ser utilizado apenas para apresentar uma visão espacial das interfaces da aplicação.

1.4 Considerações Finais

Este capítulo buscou apresentar a implementação da proposta: Análise Estrutural Orientada a Conformidade e Análise Comportamental Orientada a Conformidade. Com isso, apresentou problemas, decisões e detalhes de implementação, descreveu alguns contextos de uso e evidenciou a flexibilidade da solução a partir da implementação de duas novas soluções de análise. Com relação à Análise Estrutural, foram identificados alguns resultados para o contexto de desenvolvimento de aplicações NCL:

- Aplicou a proposta definida com sucesso, para um contexto de desenvolvimento real, demostrando a viabilidade técnica para este contexto;

- Propôs uma extensão para uma API de análise estrutural já consolidada, a fim de promover a rastreabilidade para as especificações;
- Sugere a melhoria na produtividade das correções, quando comparado à utilização de interpretadores para o processo de desenvolvimento e manutenção. Promove a diminuição do tempo de identificação de *bugs*, pelo fato da solução apontar os problemas automaticamente, e, também, contribui para a diminuição do tempo de correção, dado que a solução fornece automaticamente detalhes acerca do problema identificado; e
- Sugere a melhoria na qualidade das correções, quando comparado à utilização de interpretadores para o processo de desenvolvimento e manutenção, dado que valoriza o uso das especificações da linguagem durante a codificação, facilitando o acesso às informações por meio da implementação da rastreabilidade.

Porém, é preciso destacar algumas limitações de implementação do módulo de Análise Estrutural da solução, além daquelas já destacadas anteriormente pela proposta:

- Apesar de a abordagem definida poder ser utilizada por outras linguagens declarativas que atuam no desenvolvimento de aplicações multimídia, as soluções implementadas são suporte apenas às linguagens NCL, HTML e SVG;
- A solução implementada buscou estender uma solução já existente (*NCL-Validator*) para a identificação de problemas estruturais e com isso delimitou seu escopo de análise ao escopo da solução estendida; e
- A solução implementada promoveu a rastreabilidade apenas para um subconjunto de assertivas estruturais, devido à limitação pré-definida do escopo da prova de conceito.

Com relação às análises comportamentais, foram identificadas algumas contribuições para o contexto de desenvolvimento de aplicações NCL:

- Aplicou a proposta definida com sucesso, para um contexto de desenvolvimento real, demostrando a viabilidade técnica para este contexto;
- Possibilita a visualização dos valores das variáveis, inclusive as abstratas, entre cada interação;
- Permite avaliar, minimamente, a usabilidade da aplicação, por meio do esboço da tela da aplicação;

- Sugere a melhoria na produtividade das correções, quando comparado à utilização de interpretadores para o processo de desenvolvimento e manutenção. Promove a diminuição do tempo de identificação de *bugs*, pelo fato da solução delimitar cada estado possível fornecendo detalhes acerca das variáveis, da disposição visual e das interações possíveis. Também, contribui para a diminuição do tempo de correção, dado que a solução fornece os passos e as transformações que condicionaram a existência do problema identificado; e
- Sugere a melhoria na qualidade das correções, quando comparado à utilização de interpretadores para o processo de desenvolvimento e manutenção, dado que valoriza o uso das especificações da linguagem durante a codificação, facilitando o acesso às informações por meio da implementação da rastreabilidade.

Assim como no módulo de Análise Estrutural, também é preciso destacar algumas limitações existentes na implementação do módulo de Análise Comportamental da solução, além daquelas já destacadas anteriormente pela proposta:

- Não reutiliza o resultado de análises prévias, nem provê uma atualização sincronizada dos modelos em relação ao código. Para esta versão da solução é necessário que o desenvolvedor determine o momento da análise, por meio da execução na aba de configurações. Com isso, a solução sempre reanalisa todo o código fonte, independente das alterações realizadas;
- Como no módulo de Análise Estrutural, a abordagem definida pode ser utilizada por outras linguagens declarativas que atuam no desenvolvimento de aplicações multimídia, porém as soluções implementadas dão suporte apenas às linguagens NCL, HTML e SVG;
- A solução implementada utiliza alguns algoritmos de redução a fim de prover modelos comportamentais concisos e consistentes, porém identificou-se a necessidade de se utilizar outros algoritmos de redução, principalmente na análise de aplicações mais complexas;
- Como no módulo estrutural, a solução implementada promoveu a rastreabilidade apenas para um subconjunto de assertivas comportamentais, devido à limitação pré-definida do escopo da prova de conceito; e

- A implementação de alguns recursos visuais foi despriorizada, como, por exemplo, a ordenação e filtros em tabelas e a manipulação do zoom nos modelos visuais. Esses e outros recursos são relevantes para prover interfaces visuais amigáveis, dada a elevada quantidade de informações extraídas. Porém, não foram implementados nessa versão da solução por não impactarem os objetivos das avaliações executadas.

O próximo capítulo tem como objetivo detalhar as avaliações realizadas até o presente momento, que buscaram evidenciar a confiabilidade da representação comportamental provida pela solução.

Avaliação Experimental

Este capítulo visa fornecer informações acerca das avaliações realizadas que buscaram evidenciar a confiabilidade da representação comportamental provida pela solução (Modelo Comportamental). A avaliação da confiabilidade do Modelo Comportamental é essencial para a evolução deste trabalho, dado que essa variável pode impactar qualquer outro experimento que tenha como premissa o uso deste modelo, como por exemplo: um experimento que avalie a produtividade da abordagem proposta. Nesse sentido, foi necessário definir um plano experimental aderente aos objetivos da avaliação que trate adequadamente as limitações desse contexto. Desta forma, definiu-se uma abordagem que buscou confrontar a confiabilidade da solução com interpretadores reais da linguagem a partir da comparação das máquinas de estados inferidas (WALKINSHAW; BOGDANOV, 2013). A partir do plano definido, foram realizadas duas rodadas de avaliação considerando aplicações distintas. As próximas subseções apresentam o plano experimental proposto e detalham a execução e análise das avaliações realizadas.

1.1 Plano Experimental

No contexto das aplicações multimídia, as máquinas de estados podem determinar as sequências de interações possíveis de uma dada aplicação multimídia (WALKINSHAW; BOGDANOV, 2013). Os interpretadores, responsáveis por executar as aplicações, funcionam como tradutores dessas máquinas de estados (implícitas), determinando as transições comportamentais das aplicações. Desta forma, a partir de um **interpretador confiável**, que segue corretamente as especificações da linguagem, é viável determinar as **máquinas de estados reais** das aplicações (explícitas). Consequentemente, é possível avaliar a confiabilidade de quaisquer soluções cujas máquinas de estados possam ser inferidas, utilizando abordagens que verificam a equivalência dessas com a máquina real (WALKINSHAW; BOGDANOV, 2013).

Entretanto, algumas limitações foram identificadas na aplicação dessa abordagem: primeiramente, é impossível comparar explicitamente a linguagem de duas máquinas de estado se elas são infinitas (WALKINSHAW; BOGDANOV, 2013). Além disso, nesse contexto, normalmente, não há mecanismos para atestarem a conformidade dos interpretadores. Desta forma, não é indicado utilizar interpretadores para determinar as máquinas de estados reais das aplicações, pois eles podem omitir transições existentes (Figura 0.1.A), omitir estados existentes (Figura 0.1.B) ou gerar comportamentos equivocados (Figura 0.1.C).

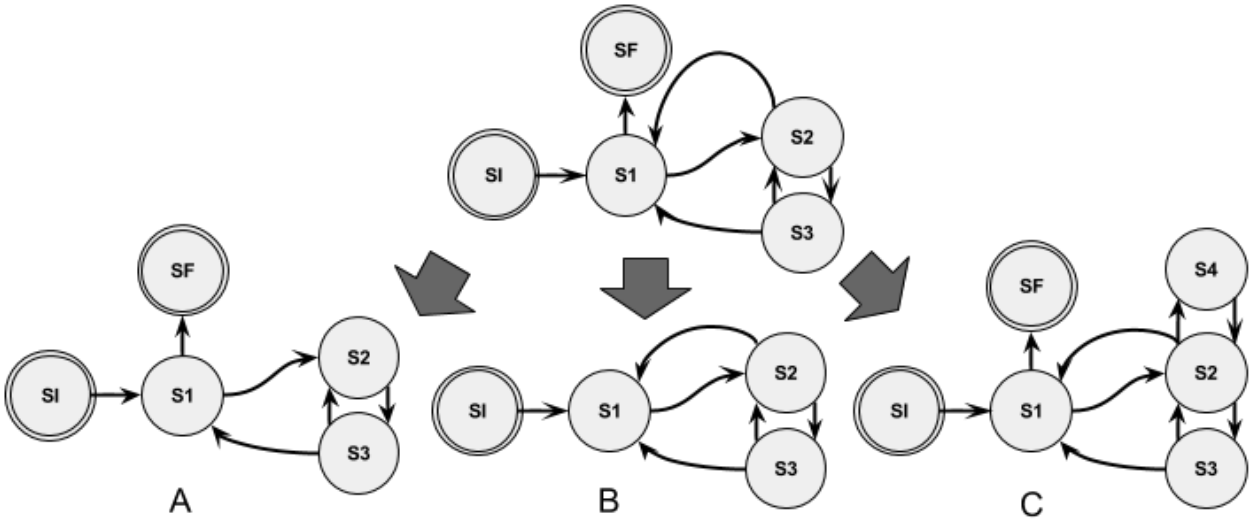

Figura 0.1. Exemplo de inconsistências entre a máquina de estados real e a máquina de estados inferida.

O plano experimental proposto buscou endereçar as limitações identificadas, aplicando um processo semiautomático para determinar a máquina de referência e determinando a utilização de um conjunto finito de sequências representativo, gerado a partir de técnicas de geração de testes, contendo sequências existentes e não existentes na máquina de estados, a fim de medir a quantidade de:

- **Verdadeiros-positivos (VP)** – sequências aceitas tanto pela máquina de estado avaliada quanto pela de referência;
- **Falsos-positivos (FP)** – sequências aceitas pela máquina de estado avaliada, porém não é aceita pala máquina de estados de referência;
- **Verdadeiros-negativos (VN)** – sequências não aceitas pela máquina de estado avaliada nem pela de referência; e
- **Falsos-negativos (FN)** – sequências não aceitas pela máquina de estado avaliada, porém é aceita pala máquina de estados de referência.

Essas medidas definem a Matriz de Confusão (SOKOLOVA; LAPALME 2009) (Figura 0.2) e permitem calcular indicadores relevantes (WALKINSHAW; BOGDANOV, 2013) para evidenciar a precisão de uma determinada máquina de estados.

		Interpretador Alvo	
		Aceita	**Rejeita**
Modelo Comportamental	**Aceita**	VP	FP
	Rejeita	FN	VN

Figura 0.2. Exemplo da Matriz de Confusão

Desta forma, permitiu-se avaliar a confiabilidade da representação comportamental gerada pela solução comparando-a com dois interpretadores reais da linguagem NCL: o **Astrobox**[16], emulador desenvolvido pela TQTVD[17] que fornece suporte a execução de aplicações NCL; e o **EiTV smartBox**[18], set-top-box desenvolvido e comercializado pela EiTV[19]. Com isso, resumidamente, o objetivo do estudo é:

Analisar a utilização de três abordagens na identificação de comportamentos válidos,

Com o propósito de identificar qual delas é mais confiável,

Com respeito à identificação dos comportamentos das aplicações NCL,

Do ponto de vista do pesquisador,

No contexto de um laboratório de testes.

1.1.1 Questões

As questões que devem ser respondidas no experimento são:

1. Qual abordagem é a mais confiável na identificação dos comportamentos reais das aplicações NCL: a que utiliza a solução provida por este trabalho (Modelo Comportamental) ou a que utiliza o Astrobox?

---

16 TQTVD. *Astrobox Documentation*. Disponível em: <astrodevnet.com.br/AstroDevNet/restrict/astrobox_sobre.html>. Acesso em: 20 de Setembro de 2016.

17 TQTVD. Disponível em: <www.tqtvd.com>. Acesso em: 20 de Setembro de 2016.

18 EiTV smartBox. Disponível em: <eitv.com.br/produtos/eitv-smartbox>. Acesso em: 20 de Setembro de 2016.

19 EiTV. Disponível em: <www.eitv.com.br>. Acesso em: 20 de Setembro de 2016.

2. Qual abordagem é a mais confiável na identificação dos comportamentos reais das aplicações NCL: a que utiliza a solução provida por este trabalho (Modelo Comportamental) ou a que utiliza o EiTV smartBox?

1.1.2 Métricas

Segundo Walkinshaw e Bogdanov (2013) a matriz de confusão pode ser utilizada para calcular alguns indicadores relevantes para comparar máquina de estados:

- ***Precision*** **–** Proporção das sequências aceitas pela máquina avaliada em comparação com as aceitas pela máquina de referência. Definida por: $\frac{VP}{VP \cup FP}$;
- ***Recall / Sensitivity*** **–** Proporção das sequências aceitas pela máquina de referência em comparação com as aceitas pela máquina avaliada. Definida por: $\frac{VP}{VP \cup FN}$;
- ***F-Measure*** **–** Média harmônica entre ***Precision*** e ***Recall***. Definida por: $\frac{2*Precision*Recall}{Precision+Recall}$;
- ***Specificity*** **–** Proporção das sequências rejeitadas pela máquina de referência em comparação com as rejeitadas pela máquina avaliada. Definida por: $\frac{VN}{VN \cup FP}$; e
- ***Balanced Classification Rate (BCR)*** **–** Acurácia da máquina avaliada, balanceada com relação a classificação positiva e negativa. Definida por: $\frac{Sensitivity+Specificity}{2}$.

Uma orientação acerca da utilização desses indicadores é a seguinte (WALKINSHAW; BOGDANOV, 2013): se o objetivo é priorizar o grau de aceitação da linguagem pela máquina avaliada (se realmente aceita as sequências positivas), sem dar a mesma ênfase ao complemento da linguagem (se realmente rejeitam as sequências negativas), é indicado utilizar apenas os indicadores ***Precision*** e ***Recall*** (usando ***F-Measure*** para agregar os dois). Porém, se tanto a linguagem quanto seu complemento tem a mesma importância para o estudo, é mais indicado utilizar a ***Sensitivity*** e ***Specificity*** (usando ***BCR*** para agregar os dois). Para o contexto deste experimento, dado que o objetivo é avaliar a confiabilidade do Modelo Comportamental, é importante evidenciar todos os indicadores.

1.1.3 Definição das hipóteses

Antes de definir as hipóteses, é necessário apresentar alguns símbolos e siglas que serão utilizados durante a seção.

- μ_{FM-SOL} — Média da medida ***F-Measure*** referente à abordagem proposta por este trabalho.
- $\mu_{FM-ASTB}$ — Média da medida ***F-Measure*** referente à abordagem que utiliza o **Astrobox**.
- $\mu_{FM-EITV}$ — Média da medida ***F-Measure*** referente à abordagem que utiliza o **EiTV smartBox**.
- $\mu_{BCR-SOL}$ — Média da medida ***BCR*** referente à abordagem proposta por este trabalho.
- $\mu_{BCR-ASTB}$ — Média da medida ***BCR*** referente à abordagem que utiliza o **Astrobox**.
- $\mu_{BCR-EITV}$ — Média da medida ***BCR*** referente à abordagem que utiliza o **EiTV smartBox**.

Hipóteses Nulas (H01):

- (H_{01}): O***F-Measure*** da abordagem proposta por este trabalho é menor ou igual a da abordagem que utiliza o Astrobox;
    - $\mu_{FM-SOL} \leq \mu_{FM-ASTB}$
- (H_{02}): O ***F-Measure*** da abordagem proposta por este trabalho é menor ou igual a da abordagem que utiliza o EiTV;
    - $\mu_{FM-SOL} \leq \mu_{FM-EITV}$
- (H_{03}): O ***BCR*** da abordagem proposta por este trabalho é menor ou igual a da abordagem que utiliza o Astrobox; e
    - $\mu_{BCR-SOL} \leq \mu_{BCR-ASTB}$
- (H_{04}): O ***BCR*** da abordagem proposta por este trabalho é menor ou igual a da abordagem que utiliza o EiTV.
    - $\mu_{BCR-SOL} \leq \mu_{BCR-EITV}$

Hipótese Alternativa

- (H_{11}): O ***F-Measure*** da abordagem proposta por este trabalho é melhor do que a que utiliza o Astrobox;
    - $\mu_{FM-SOL} > \mu_{FM-ASTB}$
- (H_{12}): O ***F-Measure*** da abordagem proposta por este trabalho é melhor do que a que utiliza o EiTV;

- $\mu_{FM-SOL} > \mu_{FM-EITV}$

- (H_{13}): O ***BCR*** da abordagem proposta por este trabalho é melhor do que a que utiliza o Astrobox; e
  - $\mu_{BCR-SOL} > \mu_{BCR-ASTB}$
- (H_{14}): O ***BCR*** da abordagem proposta por este trabalho é melhor do que a que utiliza o EiTV.
  - $\mu_{BCR-SOL} > \mu_{BCR-EITV}$.

1.1.4 Variáveis

Objetos Experimentais

Por definição, o objeto ou unidade experimental é o elemento sobre o qual será aplicado o tratamento (JURISTO, 2001) pelos sujeitos experimentais (participantes do experimento). Dentro do contexto deste estudo, o objeto experimental é a **Representação Comportamental das Aplicações Multimídia Codificadas em NCL**.

Sujeitos Experimentais

Os sujeitos experimentais desse estudo serão as **Aplicações Multimídia Codificadas em NCL**. Para definir a amostra das aplicações a serem utilizadas deve-se considerar diversas fontes, tais como: os repositórios públicos; desenvolvedores independentes; produtores de conteúdo profissionais; e desenvolvedores acadêmicos.

Variáveis de Resposta

As variáveis de resposta (*response variables*) representam as variáveis de saída do experimento, que serão utilizadas para avaliar os tratamentos empregados. No caso deste estudo, as variáveis de resposta será a **Matriz de Confusão** (SOKOLOVA; LAPALME 2009) contendo: a quantidade de: **Verdadeiros-positivos (VP)**; **Falso-positivos (FP)**; **Verdadeiros-negativos (VN)**; e **Falso-Negativos (FN)**.

Parâmetros

Os parâmetros representam as variáveis cujos valores serão mantidos constantes ao longo do experimento. Para o contexto deste estudo, tem-se como parâmetro o **Algoritmo de Geração de Exemplos de Sequências.**

Dado que os exemplos de sequência devem ser representativos para a máquina de estados de referência, recomenda-se utilizar técnicas de geração de testes baseadas em máquinas de estados, por ser uma abordagem rigorosa que busca explorar os comportamentos da máquina (com um conjunto finito de sequências) (WALKINSHAW; BOGDANOV, 2013). Para este estudo utilizou-se o *W-Method* (CHOW, 1978).

Fatores

Os fatores são as variáveis que representam os tratamentos a serem recebidos pelos objetos experimentais. Desta forma, são as características que serão modificadas para caracterizar uma eventual influência do tratamento nas variáveis de resposta. Sendo assim, este estudo é um experimento de fator único: **identificação dos comportamentos reais das aplicações**, com três tratamentos (como dito anteriormente):

- Por meio do **Modelo Comportamental**;
- Por meio do emulador **Astrobox**; e
- Por meio do interpretador **EiTV smartBox**.

1.1.5 Instrumentação

Além das soluções que representam os tratamentos (**API da Análise Comportamental**; **Emulador Astrobox**; e **EiTV smarBox)** também foram definidos alguns instrumentos necessários para apoiar a realização do experimento:

- **TraceGenerator –** implementação Java do *W-Method*, utilizada para gerar o conjunto de sequências comportamentais de cada aplicação. Desta forma, permite: acessar as aplicações avaliadas; gerar o conjunto de sequências comportamentais; e exportar o conjunto gerado para um arquivo *CSV*, que servirá de entrada para o **AstroboxSimulator** e o **EiTVSimulator**. Dado que, o **Astrobox** e o **EiTV smartBox** não evidenciam suas máquinas de estados, a geração dos traces foi orientada pelo **Modelo Comportamental**. Segundo Walkinshaw e Bogdanov (2013), nesses casos, para permitir a identificação de estados extras existentes nas implementações, é necessário parametrizar o *W-Method* com a estimativa da quantidade desses estados (k), para serem considerados no conjunto de sequências. Normalmente, utiliza-se k=1, pois já permite identificar a incidência de qualquer desvio comportamental com um aumento mínimo na quantidade de sequências (pois

esse parâmetro aumenta a quantidade das sequências exponencialmente) (WALKINSHAW; BOGDANOV, 2013). Outra observação acerca dessa ferramenta é que ela já acessa a **API da Análise Comportamental** para determinar o aceite de cada sequência gerada no Modelo Comportamental;

- **AstroboxSimulator –** implementação Java para automatizar o processo de execução das sequências comportamentais no emulador Astrobox, utilizada para apoiar a verificação do aceite dessas. Desta forma, permite: inicializar a aplicação no emulador **Astrobox**; simular o pressionamento de teclas no emulador, por meio da *API Robot*[20]; capturar a tela da aplicação em execução no emulador (também por meio da *API Robot*); e gerar um relatório HTML com o resultado da execução do conjunto de sequências comportamentais;
- **EiTVSimulator –** implementação Java para automatizar o processo de execução das sequências comportamentais no emulador EiTV smartBox, utilizada para apoiar a verificação do aceite dessas sequências. Desta forma, permite: inicializar a aplicação no **EiTV smartBox** a ser exibida no **Monitor TV LCD Samsung**; simular o pressionamento de teclas no emulador, por meio do envio de requisições HTTP ao servidor do EiTV smartBox; capturar a tela da aplicação em execução no interpretador, por meio da *API JavaCV*[21] que permite manipular a **WebCam**; e gerar um relatório HTML com o resultado da execução do conjunto de sequências comportamentais;
- **Oracle VM VirtualBox** – máquina virtual responsável por executar o emulador **Astrobox**;
- **Monitor TV LCD Samsung (24')** – utilizada pelo **EiTV smartBox** para exibir as aplicações em execução;
- **WebCam** – utilizada pelo **EiTVSimulator** para capturar a tela da aplicação em execução;
- **Relatório de execução HTML** – relatório de execução do conjunto de sequências comportamentais gerado pelo **AstroboxSimulator** e **EiTVSimulator**, utilizado para

---

20 *Robot API Documentation*. Disponível em: <docs.oracle.com/javase/8/docs/api/java/awt/Robot.html>. Acesso em: 20 de Setembro de 2016.

21 *JavaCV API Documentation*. Disponível em: <https://github.com/bytedeco/javacv>. Acesso em: 20 de Setembro de 2016.

determinar o aceite das sequências comportamentais. Desta forma, fornece a descrição visual e textual da execução de cada sequência para possibilitar a verificação por um especialista da linguagem; e

- **R Studio** – Ferramenta utilizada para a análise estatística.

1.1.6 Design do Experimento

Esta seção contém as etapas necessárias para a realização do experimento que tem como objetivo avaliar quais dos tratamentos são mais precisos na identificação dos comportamentos das aplicações NCL. Os experimentos devem ser conduzidos pelas seguintes etapas (Figura 0.3):

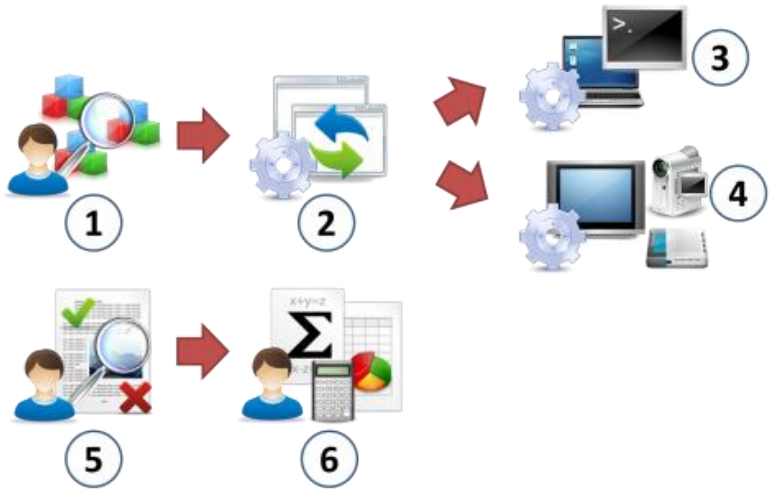

Figura 0.3. Etapas da avaliação experimental

1. Definir a amostra das aplicações a serem utilizadas pelo experimento. Para isto deve-se considerar: aplicações reais desenvolvidas por terceiros e aplicações criadas para este estudo a fim de explorar algumas funcionalidades da linguagem;
2. Definir o conjunto de sequências comportamentais de cada aplicação por meio do **TraceGenerator**;
3. Executar o conjunto de sequências comportamentais de cada aplicação no **Astrobox** por meio do **AstroboxSimulator;**
4. Executar o conjunto de sequências comportamentais de cada aplicação no **EiTV smartBox** por meio do **EiTVSimulator;**
5. Analisar os relatórios de execução gerados pelas etapas 3 e 4 para determinar o aceite de cada sequência comportamental executada. Nesta etapa, para cada conjunto de sequências executado em cada um dos interpretadores alvos, é necessário:

a. Identificar os conflitos gerados entre a saída esperada (definida pelo Modelo Comportamental) e a saída real (após a sua execução). Nesta etapa, deve-se considerar como conflito qualquer saída diferente da esperada (WALKINSHAW; BOGDANOV, 2013);
b. Determinar qual é o real resultado de cada conflito, por meio da análise de um especialista da linguagem, a fim de identificar qual abordagem está inconsistente com a máquina de estados real da aplicação; e
c. Montar duas Matrizes de Confusão que evidenciem a comparação entre: o Modelo Comportamental e a Máquina de Estados Real da Aplicação Inferida; e o interpretador alvo (Astrobox ou EiTV smartBox) e a Máquina de Estados Real da Aplicação Inferida (Figura 0.4).

6. Calcular as métricas para realizar análises exploratórias e formais a fim de responder as questões do experimento. Sendo assim, além de apresentar a análise exploratória dos dados por meio de gráficos, que evidencie a tendência central (média, mediana, moda) e dispersão dos dados (desvio padrão) (BUSSAB; MORETIN, 2010), é necessário realizar um teste de hipótese de acordo com as características da amostra que permite avaliar as hipóteses definidas. Para esta etapa, propõe-se utilizar o software R[22], pois ele fornece suporte a bibliotecas destinadas a análises estatísticas e permite a geração de diversos tipos de gráficos apropriados para a análise exploratória dos dados (JEDLITSCHKA; PFAHL, 2005) tais como: o *violin plot*, que permite visualizar a densidade, a dispersão e a mediana dos dados; e o *dot-plot,* que é utilizado para visualizar as medidas (*F-Measure* e *BCR*) de cada aplicação. Para a análise formal, é indicada a utilização de um teste de hipótese apropriado de acordo com a amostragem utilizada.

---

22 R. Disponível em: <www.r-project.org>. Acesso em: 20 de Setembro de 2016.

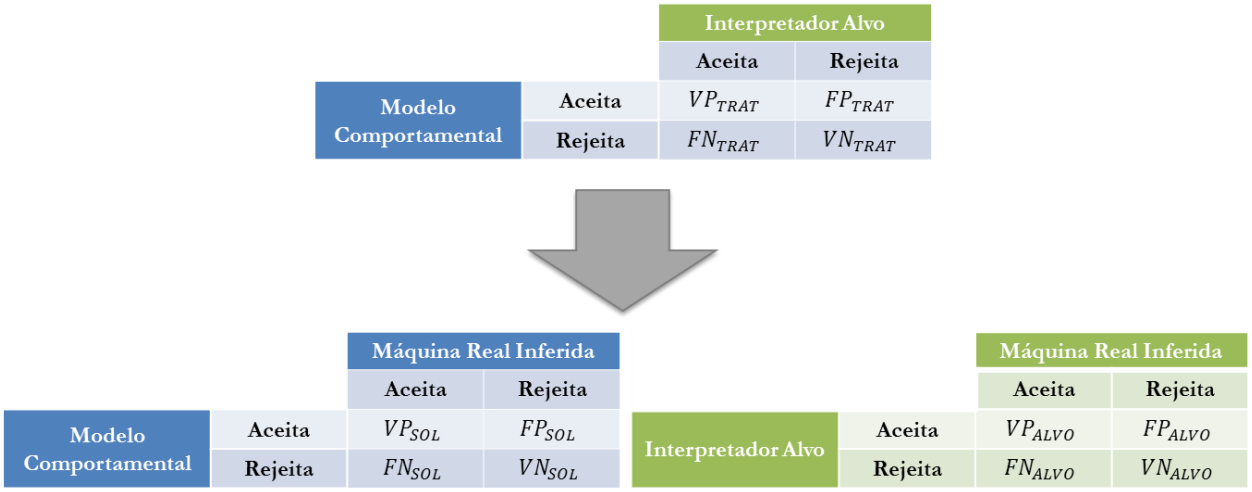

Figura 0.4. Inferência das Matrizes de Confusão dos tratamentos

A Figura 0.4 apresenta como inferir as Matrizes de Confusão de cada tratamento com a Máquina de Estados Real da Aplicação a partir da Matriz de Confusão dos tratamentos. Desta forma, dado:

- $\boldsymbol{VP_{TRAT}}$ – Quantidade de **verdadeiros positivos** para a Matriz de Confusão dos tratamentos (Astrobox ou EiTV smartBox);
- $\boldsymbol{FP_{TRAT}}$ – Quantidade de **falsos positivos** para a Matriz de Confusão dos tratamentos (Astrobox ou EiTV smartBox);
- $\boldsymbol{VN_{TRAT}}$ – Quantidade de **verdadeiros negativos** para a Matriz de Confusão dos tratamentos (Astrobox ou EiTV smartBox); e
- $\boldsymbol{FN_{TRAT}}$ – Quantidade de **falsos negativos** para a Matriz de Confusão dos tratamentos (Astrobox ou EiTV smartBox).

É necessário analisar cada um das sequências existentes em $\boldsymbol{FP_{TRAT}}$ e $\boldsymbol{FN_{TRAT}}$ (conflitantes) a fim de determinar qual seria o real comportamento em uma máquina de estados real. Desta forma, infere-se:

- $\boldsymbol{FP_{SOL}}$ – Quantidade real de falsos positivos para o Modelo Comportamental (Solução) após a análise do especialista;
- $\boldsymbol{FP_{ALVO}}$ – Quantidade real de falsos positivos para o interpretador alvo (Astrobox ou EiTV smartBox) após a análise do especialista;
- $\boldsymbol{FN_{SOL}}$ – Quantidade real de falsos negativos para o Modelo Comportamental (Solução) após a análise do especialista;

- $\boldsymbol{FN_{ALVO}}$ – Quantidade real de falsos negativos para o interpretador alvo (Astrobox ou EiTV smartBox) após a análise do especialista;
- $\boldsymbol{VP_{SOL}}$ – Quantidade real de verdadeiros positivos para o Modelo Comportamental, determinada por $\boldsymbol{VP_{TRAT} + FP_{ALVO}}$;
- $\boldsymbol{VP_{ALVO}}$ – Quantidade real de verdadeiros positivos para o interpretador alvo (Astrobox ou EiTV smartBox), determinada por $\boldsymbol{VP_{TRAT} + FP_{SOL}}$;
- $\boldsymbol{VN_{SOL}}$ – Quantidade real de verdadeiros negativos para o Modelo Comportamental, determinada por $\boldsymbol{VN_{TRAT} + FN_{ALVO}}$; e
- $\boldsymbol{VN_{ALVO}}$ – Quantidade real de verdadeiros negativos para o interpretador alvo (Astrobox ou EiTV smartBox), determinada por $\boldsymbol{VN_{TRAT} + FN_{SOL}}$.

1.2 Execuções

Com base no Plano Experimental definido, foram realizadas duas rodadas de avaliação, a primeira avaliou uma versão inicial da ferramenta e teve por objetivo destacar a viabilidade da solução por meio da investigação da sua confiabilidade em um escopo limitado da linguagem. Desta forma, além das aplicações codificadas por terceiros, também foram consideradas aplicações criadas para explorar algumas estruturas consideradas fundamentais para a linguagem, nessa primeira avaliação foram utilizadas 12 aplicações. Com base nos resultados desse experimento, a solução foi evoluída a fim de corrigir os problemas identificados e incrementar o escopo anterior. A partir dessa nova versão um segundo experimento foi realizado, considerando apenas aplicações reais produzidas por terceiros. Vale salientar que apesar do incremento do escopo da solução, a segunda versão ainda não contempla todas as estruturas e regras da linguagem. Os experimentos utilizaram um conjunto de 24 aplicações, ambas as rodadas utilizaram um conjunto de 12 aplicações distintas (totalizando 24 aplicações) selecionadas a partir das seguintes atividades:

1. **Coletar aplicações –** as aplicações foram coletadas a partir: dos repositórios públicos oficiais (Clube NCL[23] e o Repositório de Conteúdos e Aplicações Interativas de TV Digital[24] mantido pelo Ministério das Comunicações) e não oficiais

23 Clube NCL. Disponível em: <clube.ncl.org.br>. Acesso em: 20 de Setembro de 2016.

24 Ministério das Comunicações. Repositório de Conteúdos e Aplicações Interativas de TV Digital. Disponível em: <gingabr.comunicacoes.gov.br/>. Acesso em: 20 de Setembro de 2016.

(Github[25]); do contato com desenvolvedores por meio de fóruns de desenvolvimento; do contato com os principais produtores de conteúdo (Lifia[26], HxD[27], Lavid[28], TQTVD[29], SBT[30], Globo[31], Laboratório Telemidia[32], C.E.S.A.R [33]); e da consulta livre na Web. Além destas, para o primeiro experimento, também foram consideradas aplicações criadas para explorar alguns tipos de interações da linguagem; e

2. **Filtrar aplicações** – todas as aplicações coletadas foram analisadas a fim de descartar aquelas que: apresentavam muitos erros estruturais; e não eram essencialmente escritas em NCL. Já que, aplicações com erros estruturais podem apresentar comportamentos inesperados (não definidos pelas especificações), uma prática comum, nesses casos, é a interrupção da execução da aplicação.

Apesar de ter propósitos diferentes e de usarem subconjuntos distintos, ambas as execuções seguiram o protocolo definido pelo Design do Experimento. Para cada uma das aplicações foi necessário definir o conjunto de sequências comportamentais a ser utilizado, por meio do **TraceGenerator**. Após a definição dos conjuntos de sequências comportamentais, o **AstroboxSimulator** e o **EiTVSimulator** foram utilizados para executar automaticamente cada uma dessas sequências no **Astrobox** e no **EiTV smartBox**, respectivamente. A Figura 0.5 apresenta o ambiente de execução do EiTVSimulator. Algumas limitações foram identificadas durante este processo. O EiTV smartBox apresentou uma certa instabilidade na execução das sequências, por diversas vezes, sem motivo aparente, o servidor web da ferramenta ficou inoperante, impossibilitando a simulação do pressionamento das teclas. Além disto, frequentemente, a aplicação ficava inacessível por meio do Menu, não permitindo a sua execução. Para retomar a execução era necessário realizar o *reset* do equipamento e refazer todas as configurações de rede. Desta forma, a fim de diminuir o esforço, algumas sequências tiveram que ser reexecutas manualmente. Na sequência, os relatórios de execução foram analisados, a fim de identificar e tratar os conflitos, as

25 Github. Disponível em: <github.com>. Acesso em: 20 de Setembro de 2016.
26 Lifia. Disponível em: <www.lifia.info.unlp.edu.ar/lifia/es>. Acesso em: 20 de Setembro de 2016.
27 HxD. Disponível em: <www.hxd.com.br>. Acesso em: 20 de Setembro de 2016.
28 Lavid. Disponível em: <www.lavid.ufpb.br>. Acesso em: 20 de Setembro de 2016.
29 TQTVD. Disponível em: <www.tqtvd.com>. Acesso em: 20 de Setembro de 2016.
30 SBT. Disponível em: <www.sbt.com.br>. Acesso em: 20 de Setembro de 2016.
31 Rede Globo. Disponível em: <redeglobo.globo.com>. Acesso em: 20 de Setembro de 2016.
32 Telemídia. Disponível em: <www.telemidia.puc-rio.br>. Acesso em: 20 de Setembro de 2016.
33 C.E.S.A.R. Disponível em: <www.cesar.org.br>. Acesso em: 20 de Setembro de 2016.

Matrizes de Confusão resultantes dessa etapa para as duas execuções são apresentadas no Apêndice B. Por fim, foi necessário calcular os indicadores relevantes para este estudo: *Precision*; *Recall / Sensitivity*; *F-Measure*; *Specificity*; *Balanced Classification Rate (BCR)* para realizar a análise exploratória e formal. Esses indicadores também são apresentados no Apêndice B, bem como os scripts e os resultados das análises formais (realizada com o software R).

Figura 0.5. Execução das aplicações no EiTV smartBox através do EiTVSimulator

A seguir serão detalhadas algumas informações acerca das aplicações utilizadas em cada experimento, bem como a análise exploratória e formal dos resultados.

1.2.1 Primeira Rodada

Para a primeira rodada foram selecionadas 12 aplicações, codificadas por terceiros e criadas para explorar algumas estruturas consideradas fundamentais para a linguagem (Figura 0.6).

Figura 0.6. Exemplo das execuções das aplicações durante a Primeira Rodada

O

Quadro 0.1 apresenta algumas informações acerca do tamanho e da representatividade das aplicações utilizadas na primeira rodada do experimento. A representatividade foi calculada de acordo com a quantidade de entidades (elementos ou atributos) existentes: na linguagem NCL (Ling.); no escopo implementado pela versão inicial da solução (Esc.); e na aplicação utilizada (Apl.). Vale salientar que, para o escopo da solução, foram consideradas tanto as entidades cujas regras foram implementadas integralmente, como as que foram implementadas parcialmente, desconsiderando apenas aquelas que não tiveram nenhuma regra implementada pela versão inicial da solução. A análise do

Quadro 0.1 permite perceber que boa parte das aplicações utilizadas apresenta uma representatividade de 100,00% em relação ao escopo implementado, o que quer dizer que todas as entidades existentes nessas aplicações foram implementadas pela versão inicial da solução. Como a linguagem NCL classifica suas entidades de acordo com a Área Funcional (subconjunto semântico da linguagem), é relevante apresentar a distribuição das entidades de acordo com essa classificação,

considerando a linguagem, o escopo e as aplicações utilizadas (Quadro 0.2). Desta forma, é possível verificar que as Áreas Funcionais fundamentais da linguagem foram priorizadas, tais como: *Components*, *Presentation Specification e Layout*, que definem a maioria das regras descritivas; e *Connectors* e *Linking*, que definem grande parte das regras comportamentais.

Quadro 0.1. Aplicações selecionadas para realização do experimento para a primeira rodada de avaliação

	Aplicação											
	1	2	3	4	5	6	7	8	9	10	11	12
Tamanho (LOC)	346	158	198	187	203	209	143	204	209	755	385	662
Representatividade da Apl./Ling. (%)	40,00	29,74	28,21	28,21	28,21	28,21	28,21	28,21	28,72	35,90	28,72	47,69
Representatividade da Apl./Esc. (%)	67,74	62,37	59,14	59,14	59,14	59,14	59,14	59,14	60,22	67,74	56,99	72,04
Representatividade do Esc./Apl. (%)	80,77	100,00	100,00	100,00	100,00	100,00	100,00	100,00	100,00	90,00	94,64	72,04

Quadro 0.2. Distribuição das entidades de acordo com a Área Funcional para a primeira rodada de avaliação

Área Funcional	Linguagem	Escopo	Aplicação											
			1	2	3	4	5	6	7	8	9	10	11	12
Interfaces	26	11	6	9	6	6	6	6	6	6	10	6	7	6
Connectors	46	35	24	17	16	16	16	16	16	16	16	25	15	24
Linking	14	13	10	9	12	12	12	12	12	12	9	13	10	13
Presentation Control	22	2	0	0	0	0	0	0	0	0	0	0	0	14
Structure	7	6	6	5	5	5	5	5	5	5	5	5	5	5
Components	7	7	7	6	5	5	5	5	5	5	5	5	5	7
Reuse	14	0	3	0	0	0	0	0	0	0	0	0	3	3
Navigational Key	11	0	3	0	0	0	0	0	0	0	0	4	0	9
Presentation Specification	9	8	4	4	4	4	4	4	4	4	4	4	4	4
Timing	2	0	0	0	0	0	0	0	0	0	0	0	0	0
Transition Effects	17	0	7	0	0	0	0	0	0	0	0	0	0	0
Meta-Information	4	0	0	0	0	0	0	0	0	0	0	0	0	0
Layout	14	11	8	8	7	7	7	7	7	7	7	8	7	8

Animation	2	0	0	0	0	0	0	0	0	0	0	0	0	0
Total	**195**	**93**	**78**	**58**	**55**	**55**	**55**	**55**	**55**	**55**	**56**	**70**	**56**	**93**

Como dito anteriormente, para cada aplicação foram geradas sequências positivas e negativas (esperadas) que foram executadas e analisadas por um especialista da linguagem a fim de determinar qual dessas são realmente positivas e negativas de acordo com a especificação da linguagem. O Quadro 0.3 apresenta a distribuição das sequências (positivas e negativas) esperadas e reais de acordo com o Modelo Comportamental.

Quadro 0.3. Distribuição das sequências de avaliação esperadas (geradas com base no Modelo Comportamental) e reais (após o tratamento dos conflitos) para a primeira rodada de avaliação

Aplicação	Sequências Positivas (esperada)	Sequências Negativas (esperada)	Sequências Positivas (real)	Sequências Negativas (real)
Aplicação 01	27	30	29	28
Aplicação 02	16	91	16	91
Aplicação 03	78	108	78	108
Aplicação 04	58	134	58	134
Aplicação 05	82	135	82	135
Aplicação 06	56	77	56	77
Aplicação 07	9	27	9	27
Aplicação 08	55	77	55	77
Aplicação 09	45	120	29	136
Aplicação 10	390	148	390	148
Aplicação 11	275	180	1	454
Aplicação 12	65	46	65	46

Com relação à análise exploratória dos dados, a Figura 0.7 apresenta o gráfico *violin plot* para a medida *F-Measure* , enquanto que a Figura 0.8 o apresenta para a medida BCR. Nesses gráficos é possível observar que a mediana das medidas para o Modelo Comportamental tende a 1,00 em ambos os gráficos, enquanto que para o Astrobox tende a 0,00 (*F-Measure*) e 0,33 (*BCR*) e para o EiTV smartBox tende a 0,70 (*F-Measure*) e 0,88 (*BCR*). Porém, apesar das medianas indicarem um bom grau de confiabilidade para o Modelo Comportamental, em uma das aplicações essa abordagem não obteve um bom resultado.

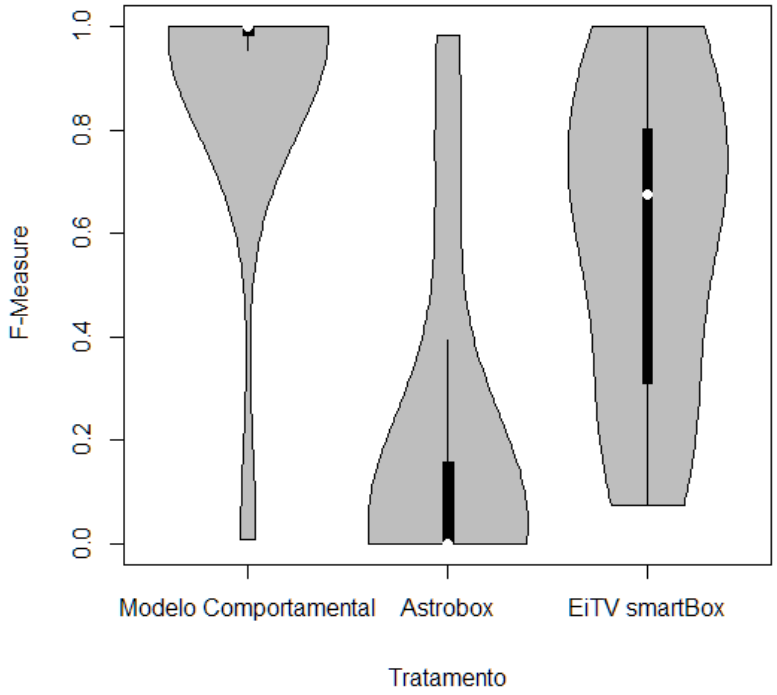

Figura 0.7. Gráfico *violin plot* para a medida *F-Measure* para a primeira rodada de avaliação

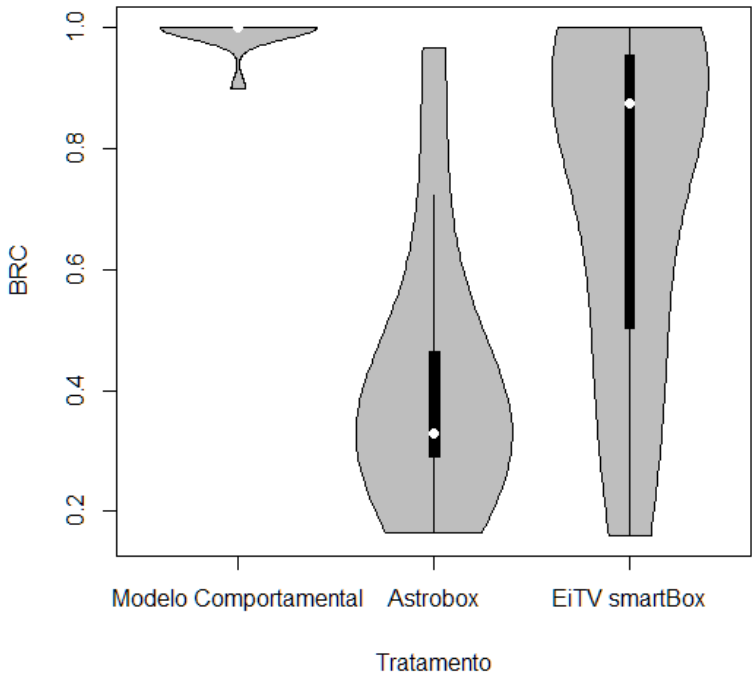

Figura 0.8. Gráfico *violin plot* para a medida *BCR* para a primeira rodada de avaliação

Com o objetivo de visualizar as métricas individuais, foram construídos os Diagramas Pontuais (JEDLITSCHKA; PFAHL, 2005) para cada medida avaliada, no qual é possível comparar o resultado de cada unidade experimental. Como pode ser visualizado, a maioria dos indicadores do

Modelo Comportamental teve uma diferença considerável, tanto em comparação com o *Astrobox* quanto com o *EiTV smartBox*, para ambas medidas *F-Measure* (Figura 0.9) e *BRC* (Figura 0.10). Este cenário sugere que o Modelo Comportamental é mais confiável na identificação dos comportamentos das aplicações multimídia codificadas em NCL quando comparado às abordagens que utilizam esses interpretadores. Apesar da consolidação dos resultados ter indicado um bom desempenho do Modelo Comportamental, uma das amostras teve a confiabilidade próxima a zero com relação a medida *F-Measure* (A11). Após realizar uma análise mais detalhada da aplicação A11, constatou-se que o resultado foi impactado pela limitação do escopo da prova de conceito, por não dar suporte a uma regra essencial para a análise dessa aplicação que determina as teclas válidas para a interação com a aplicação. Esse caso é um indicio da importância da completude da implementação, principalmente no da solução em um processo produtivo real.

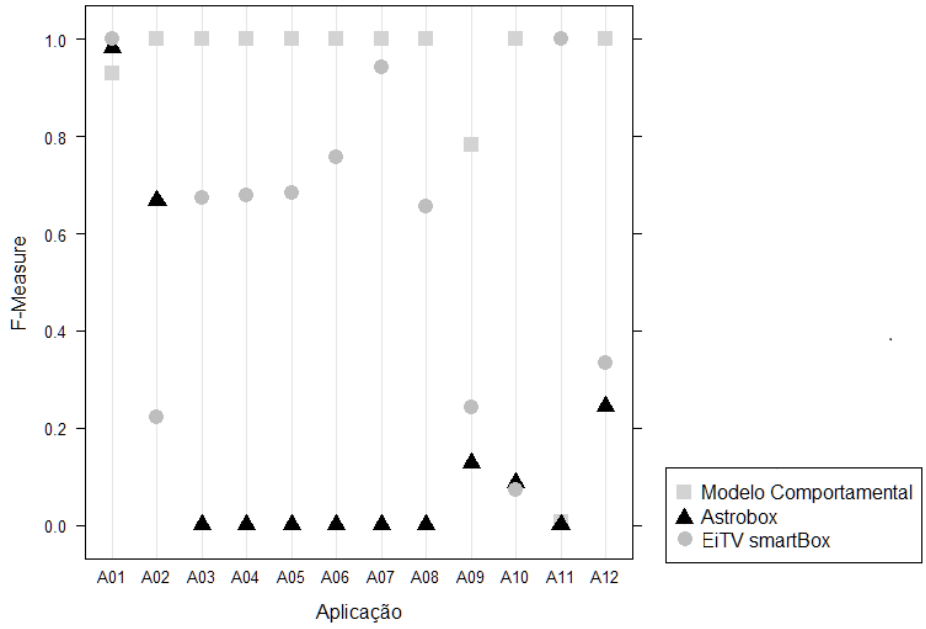

Figura 0.9. Diagrama pontual para a medida *F-Measure* para a primeira rodada de avaliação

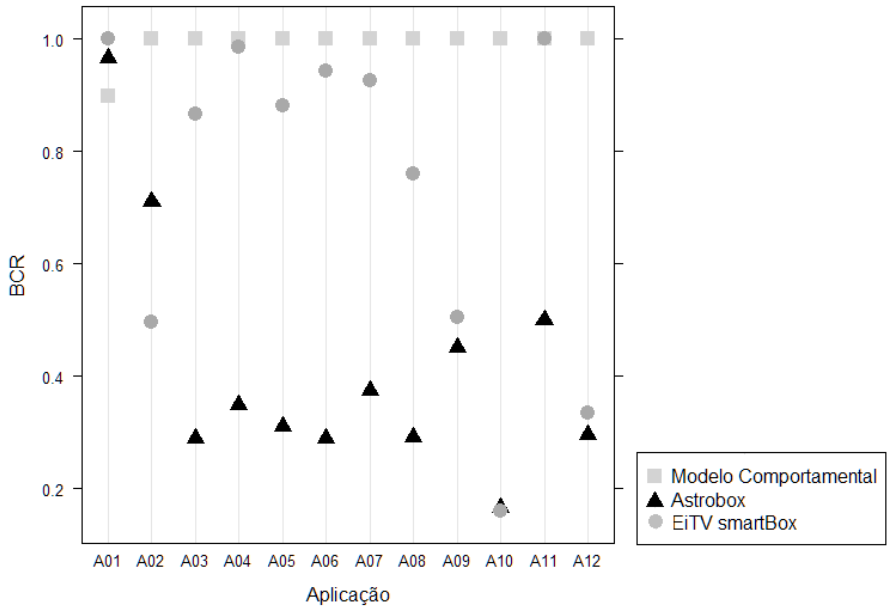

Figura 0.10. Diagrama pontual para a medida *BCR* para a primeira rodada de avaliação

Uma vez realizada a análise exploratória dos dados, é importante utilizar um método mais formal e sistemático que permita verificar a generalização dos resultados obtidos. Nesse sentido, considerando a distribuição da amostra como normal, foi aplicado o Teste T emparelhado (JURISTO, 2001), dado que a mesma população foi utilizada (nas três observações) e que a variável de resposta não possui natureza nominal ou ordinal (JURISTO, 2001). Além disto, é necessário determinar o valor-p (*p-value*) ou nível descritivo do teste que é o indicador utilizado para decidir sobre a aceitação ou rejeição da hipótese nula (JURISTO, 2001). Como este experimento utilizou uma amostra de apenas 12 elementos utilizou-se a variável t_{n-1} da distribuição *t* de *Student* (AHSANULLAH *et. al.*, 2014). Sendo assim, considerando o nível de significância igual a 0,05, identificou-se o $t_{critico}$ igual a -1,796, que determina que a hipótese nula deva ser aceita no caso do *t* calculado (t_{calc}) ser maior, e que deva ser rejeitada em caso contrário. O Quadro 0.4 apresenta o resultado da análise formal realizada sobre os dados do primeiro experimento, onde: o t_{calc1} representa o resultado do Astrobox para a medida *F-Measure*; o t_{calc2} representa o resultado do EiTV smartBox para a medida *F-Measure*; o t_{calc1} representa o resultado do Astrobox para a medida *BCR*; e o t_{calc1} representa o resultado do EiTV smartBox para a medida *BCR*.

Quadro 0.4. Resultado da avaliação experimental para a primeira rodada de avaliação

t_{calc}	**Conclusão**
-6.1975	$t_{calc1} < t_{critico}$, então H_{01} deve ser rejeitada
-2.0204	$t_{calc1} < t_{critico}$, então H_{02} deve ser rejeitada

-8.1419	$t_{calc1} < t_{critico}$, então H_{03} deve ser rejeitada
-2.9323	$t_{calc1} < t_{critico}$, então H_{04} deve ser rejeitada

Com isso, pode-se concluir que o Modelo Comportamental apresenta uma confiabilidade mais alta que as abordagens que utilizam o Astrobox ou EiTV smartBox, independente do propósito: se o grau de aceitação da linguagem é priorizado (*F-Measure*) ou se o grau de aceitação e o grau de rejeição têm a mesma relevância (*BCR*).

1.2.2 Segunda Rodada

Como mencionado anteriormente, a segunda rodada buscou avaliar uma versão evoluída da solução, que contemplou novas estruturas e regras inicialmente desprezadas na primeira versão. Desta forma, foram selecionadas apenas aplicações (12) codificadas por terceiros (Figura 0.11).

Figura 0.11. Exemplo das execuções das aplicações durante a Segunda Rodada

O Quadro 0.5 apresenta algumas informações acerca do tamanho e da representatividade das aplicações utilizadas na segunda rodada do experimento. Assim como no primeiro experimento, a representatividade foi calculada de acordo com a quantidade de entidades (elementos ou atributos) existentes: na linguagem NCL (Ling.); no escopo implementado pela versão inicial da solução (Esc.); e na aplicação utilizada (Apl.). Após a análise do Quadro 0.5 pode-se perceber que no geral as aplicações são maiores e estruturalmente mais complexas do que as utilizadas pelo primeiro experimento. O Quadro 0.6 apresenta a distribuição das entidades (elementos ou atributos) das aplicações de acordo com a classificação de Área Funcional. Nele é possível observar que as Áreas Funcionais incrementadas pela segunda versão da solução foram as seguintes: *Presentation Control*,

Navigational Key, Structure e *Reuse*. As duas primeiras complementam as regras comportamentais que definem o controle da apresentação a partir de condicionais e a transição do foco a partir das teclas de navegação, respectivamente. Por outro lado *Structure* e *Reuse* atuam na limitação das regras relacionais, permitindo o uso da importação de bases e documentos, normalmente utilizados na codificação de aplicações reais. Além disto, é importante destacar que, apesar de boa parte das aplicações utilizarem recursos da Área Funcional *Transition Effects*, a ausência desta área no escopo da solução não impacta a análise das aplicações, pois seus recursos não determinam novos comportamentos, eles apenas definem os efeitos de transição das mídias.

Quadro 0.5. Aplicações selecionadas para realização do experimento para a segunda rodada de avaliação

	Aplicação											
	1	2	3	4	5	6	7	8	9	10	11	12
Tamanho (LOC)	313	2100	2426	1996	436	832	1723	1723	1723	431	2079	2002
Representatividade da Apl./Ling. (%)	28,21	38,97	34,87	49,74	40,51	35,90	49,74	49,74	49,74	39,49	30,26	37,95
Representatividade da Apl./Esc. (%)	44,63	52,89	52,07	65,29	61,16	50,41	65,29	65,29	65,29	57,02	46,28	52,07
Representatividade do Esc./Apl. (%)	98,18	84,21	92,65	81,44	93,67	87,14	81,44	81,44	81,44	89,61	94,92	85,14

Quadro 0.6. Distribuição das entidades de acordo com a Área Funcional para a segunda rodada de avaliação

Área Funcional	Linguagem	Escopo	Aplicação											
			1	2	3	4	5	6	7	8	9	10	11	12
Interfaces	26	11	6	6	6	13	5	6	13	13	13	5	3	6
Connectors	46	35	15	19	19	19	29	22	19	19	19	24	19	19
Linking	14	13	9	10	9	10	9	10	10	10	10	10	9	9
Presentation Control	22	15	0	0	0	14	0	0	14	14	14	0	0	0
Structure	7	7	6	6	6	6	6	5	6	6	6	6	6	6
Components	7	7	4	5	5	5	5	7	5	5	5	7	4	5
Reuse	14	9	3	3	3	3	3	2	3	3	3	3	3	3
Navigational Key	11	5	0	4	5	3	7	0	3	3	3	3	0	3
Presentation Specification	9	8	4	5	5	5	5	4	5	5	5	4	5	5
Timing	2	0	0	0	0	1	0	0	1	1	1	0	0	0
Transition Effects	17	0	0	8	0	8	0	5	8	8	8	7	0	8
Meta-Information	4	0	0	0	0	0	0	0	0	0	0	0	0	0
Layout	14	11	8	9	9	9	9	9	9	9	9	8	9	9

Animation	2	0	0	1	1	1	1	0	1	1	1	0	1	1
Total	**195**	**121**	**55**	**76**	**68**	**97**	**79**	**70**	**97**	**97**	**97**	**77**	**59**	**74**

O Quadro 0.7 apresenta a distribuição das sequências (positivas e negativas) esperadas e reais de acordo com o Modelo Comportamental. Pode-se observar que na maioria das aplicações o Modelo Comportamental teve um bom resultado, ao confirmar a expectativa das sequências positivas e negativas.

Quadro 0.7. Distribuição das sequências de avaliação esperadas (geradas com base no Modelo Comportamental) e reais (após o tratamento dos conflitos) para a segunda rodada de avaliação

Aplicação	Sequências Positivas (esperada)	Sequências Negativas (esperada)	Sequências Positivas (real)	Sequências Negativas (real)
Aplicação 01	96	92	96	92
Aplicação 02	60	10	60	10
Aplicação 03	146	307	123	330
Aplicação 04	11	9	11	9
Aplicação 05	106	110	106	110
Aplicação 06	111	434	110	435
Aplicação 07	11	4	11	4
Aplicação 08	11	9	11	9
Aplicação 09	11	9	11	9
Aplicação 10	22	88	22	88
Aplicação 11	58	0	40	18
Aplicação 12	33	10	33	10

Com relação à análise exploratória dos dados, a Figura 0.12 apresenta o gráfico *violin plot* para a medida *F-Measure* , enquanto que a Figura 0.13 o apresenta para a medida BCR. Nesses gráficos é possível observar que a mediana das medidas para o Modelo Comportamental tende a 1,00, enquanto que para o Astrobox tende a 0,50 e para o EiTV smartBox tende a 0,50, em ambos os gráficos e para ambas medidas *F-Measure* (Figura 0.12) e *BRC* (Figura 0.13).

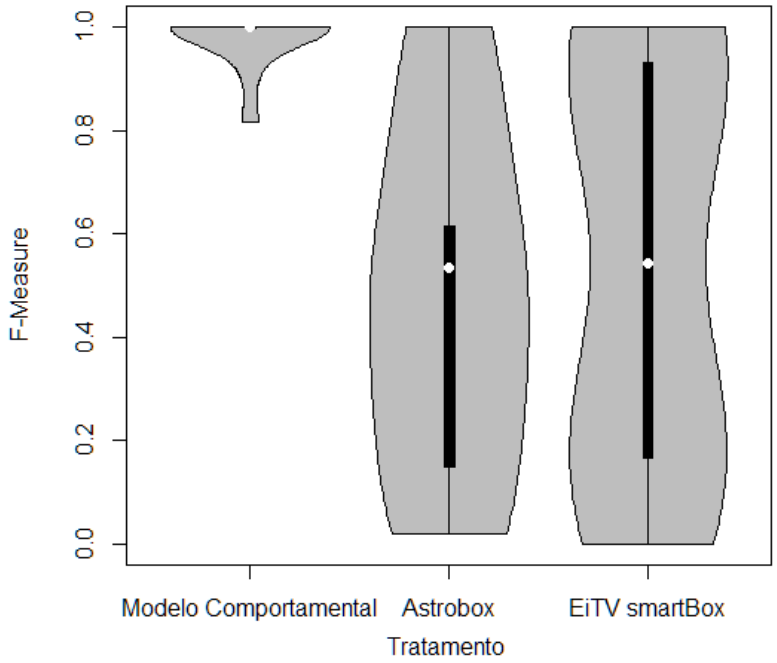

Figura 0.12. Gráfico *violin plot* para a medida *F-Measure* para a segunda rodada de avaliação

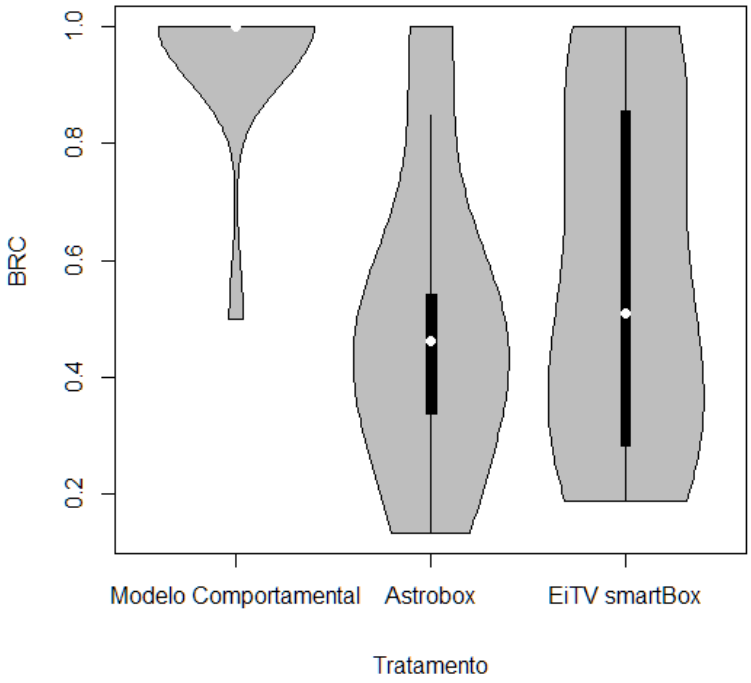

Figura 0.13. Gráfico *violin plot* para a medida *BCR* para a segunda rodada de avaliação

Assim como na análise do primeiro experimento, os Diagramas Pontuais (JEDLITSCHKA; PFAHL, 2005) evidenciaram que o Modelo Comportamental é mais confiável na identificação dos comportamentos das aplicações multimídia codificadas em NCL quando comparado às abordagens que utilizam os interpretadores avaliados, pois a maioria dos indicadores do Modelo

Comportamental teve uma diferença considerável, tanto em comparação com o *Astrobox* quanto com o *EiTV smartBox*, para ambas medidas *F-Measure* (Figura 0.14) e *BRC* (Figura 0.15). Apesar da consolidação dos resultados terem indicado um bom desempenho do Modelo Comportamental, uma das amostras não obteve um bom resultado com relação à medida *BCR* (A11).

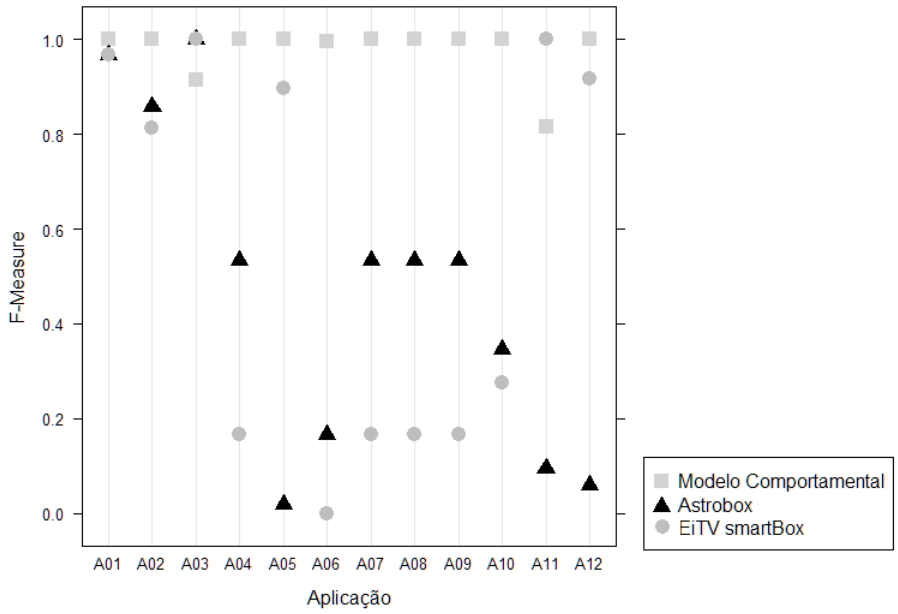

Figura 0.14. Diagrama pontual para a medida *F-Measure* para a segunda rodada de avaliação

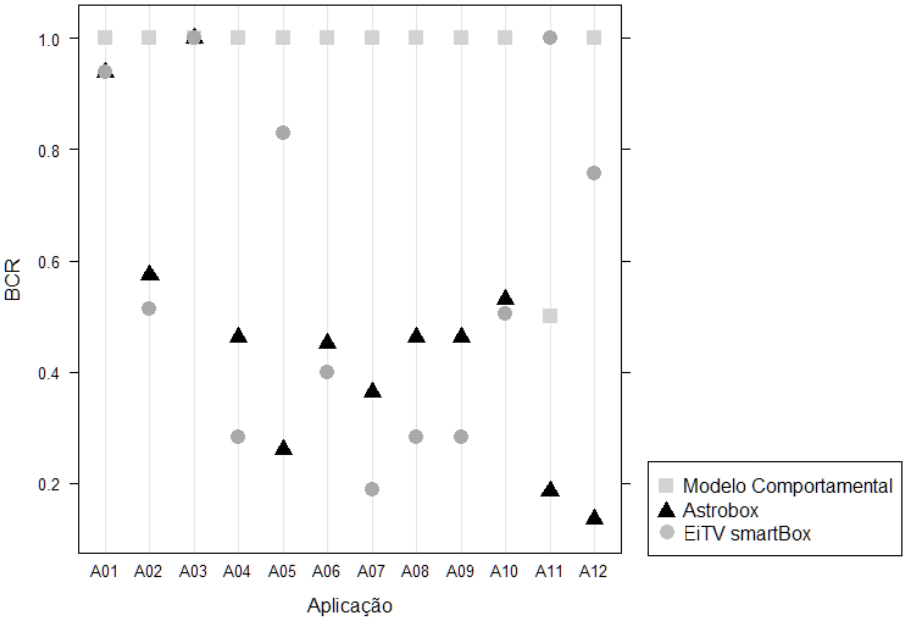

Figura 0.15. Diagrama pontual para a medida *BCR* para a segunda rodada de avaliação

Assim como na análise dos dados do primeiro experimento, também foi aplicado o Teste T emparelhado (JURISTO, 2001). Como este experimento também utilizou uma amostra de apenas 12

elementos utilizou-se a variável t_{n-1} da distribuição *t* de *Student* (AHSANULLAH *et. al.*, 2014). Considerando o nível de significância igual a 0,05, identificou-se o $t_{critico}$ igual a -1,796, que determina que a hipótese nula deva ser aceita no caso do *t* calculado (t_{calc}) ser maior, e que deva ser rejeitada em caso contrário. O Quadro 0.8 apresenta o resultado da análise formal realizada sobre os dados do primeiro experimento, onde: o t_{calc1} representa o resultado do Astrobox para a medida *F-Measure*; o t_{calc2} representa o resultado do EiTV smartBox para a medida *F-Measure*; o t_{calc1} representa o resultado do Astrobox para a medida *BCR*; e o t_{calc1} representa o resultado do EiTV smartBox para a medida *BCR*.

Quadro 0.8. Resultado da avaliação experimental para a segunda rodada de avaliação

t_{calc}	**Conclusão**
-5.1083	$t_{calc1} < t_{critico}$, então H_{01} deve ser rejeitado.
-3.3944	$t_{calc1} < t_{critico}$, então H_{02} deve ser rejeitado.
-6.5353	$t_{calc1} < t_{critico}$, então H_{03} deve ser rejeitado.
-3.3371	$t_{calc1} < t_{critico}$, então H_{04} deve ser rejeitado.

Com isso, pode-se concluir que, assim como no primeiro experimento, o Modelo Comportamental apresenta uma confiabilidade mais alta que as abordagens que utilizam o Astrobox ou EiTV smartBox, independente do propósito: se o grau de aceitação da linguagem é priorizado (*F-Measure*) ou se o grau de aceitação e o grau de rejeição têm a mesma relevância (*BCR*).

1.3 Ameaças à Validade do Experimento

Nesta seção são descritos os elementos que poderiam ameaçar a validade do experimento, categorizados de acordo com (WOHLIN *et. al.*, 2012) em ameaças: de conclusão, interna, de construção e externa.

1.3.1 Validade de Conclusão

A validade de conclusão diz respeito ao relacionamento entre o tratamento e as saídas do experimento controlado (variáveis de resposta/dependentes), e à capacidade de realizar inferências corretas a partir deste relacionamento. A ameaça à validade de conclusão se refere ao baixo poder estatístico e a violação de premissas assumidas para o teste estatístico. Para eliminar esta ameaça foi empregado o teste de hipótese, método amplamente sólido e difundido. Visando garantir o respeito às premissas do teste estatístico. Complementarmente, a ferramenta R Studio apoiou a análise dos dados, minimizando a possibilidade de erros associados ao tratamento manual.

1.3.2 Validade Interna

A validade interna diz respeito a garantir que o relacionamento entre os tratamentos e as saídas do experimento (variáveis de resposta/dependentes) seja determinado pela causalidade. Isto é, os tratamentos causam os diferentes resultados. Por isso, o projeto de experimento buscou eliminar a influência de qualquer variável que possa causar variações indesejadas nos resultados. Um dos elementos que poderiam introduzir uma ameaça à validade interna é a "instrumentação inadequada" para a realização do experimento. Neste caso, houve uma preocupação relevante com a instrumentação do experimento, conforme apresentado anteriormente.

1.3.3 Validade de Construção

A validade de construção visa garantir que existe um relacionamento adequado entre o arcabouço teórico que dê suporte ao experimento e às observações realizadas. Logo, deve ser avaliado se todas as variáveis (dependentes e independentes), métricas e hipótese foram definidas em conformidade com a teoria existente (realidade). A má definição do plano experimental é um exemplo de ameaça de validade de construção. Para minimizar riscos associados a esta ameaça, a execução do experimento foi monitorada, a fim de evitar qualquer inconsistência com relação ao plano. Além disso, a apresentação inadequada das etapas do experimento e o receio e expectativa dos participantes em serem avaliados, representam outras ameaças que não fazem sentido para o contexto deste experimento.

1.3.4 Validade Externa

A validade externa diz respeito à capacidade de generalizar os resultados obtidos em determinado experimento controlado para situações além do escopo do mesmo. Dentro deste contexto, o estudo apresenta algumas limitações quanto à validade externa, uma vez que a quantidade limitada de replicações do estudo pode restringir a generalização. Outra limitação que dificulta a generalização dos resultados deste estudo é o critério não-aleatório de seleção dos sujeitos experimentais, dado que um mesmo sujeito foi aplicado em pelo menos dois tratamentos devido à inexistência da máquina de estados de referência.

1.4 Considerações Finais

Este capítulo apresentou detalhes acerca das avaliações realizadas, que tinham como foco a investigação da confiabilidade do Modelo Comportamental na identificação dos comportamentos

reais da aplicação, comparando esta abordagem com outras que utilizam implementações dos interpretadores da linguagem: Astrobox ou EiTV smartBox. Esta investigação foi conduzida por meio de dois experimentos controlados: ambos determinaram que o Modelo Comportamental apresenta uma confiabilidade mais alta que as abordagens que utilizam os interpretadores avaliados. Além disto, a análise mais detalhada dos resultados permitiu identificar algumas limitações nos ambientes analisados:

- **No Astrobox –** algumas estruturas essenciais para o controle do foco das aplicações não funcionam adequadamente. Essa limitação fez com que algumas aplicações não reagissem aos eventos enviados. Também, não permite a utilização da mídia "sbtvd-ts://vídeo" (que representa o vídeo corrente enviado pela emissora) e do vídeo no formato mp4; e
- **No EiTV smartBox –** ocorreram instabilidades na interação com as aplicações por meio do servidor web. Além disto, frequentemente, a aplicação ficava inacessível por meio do Menu, não permitindo a sua execução. Além desses problemas de execução, também foram identificados alguns problemas comportamentais: assim como no Astrobox, o EiTV smartBox também não implementa o controle do foco adequadamente. Aparentemente, não restringe o controle do foco para o componente predefinido, permitindo que todos os componentes em exibição controlem o foco da aplicação.

O próximo capítulo, busca analisar os principais esforços que consolidaram contribuições relevantes para o contexto deste trabalho.

Trabalhos Relacionados

Neste capítulo os principais trabalhos relacionados serão descritos e analisados, a fim de destacar as contribuições mais relevantes e limitações existentes. Tais informações fundamentaram a definição da solução proposta por este trabalho, evidenciando os requisitos aderentes à proposta, além de permitir a prevenção dos problemas já conhecidos. Com o objetivo de avaliar e classificar os trabalhos relacionados definiu-se critérios de avaliação relevantes para o contexto, no âmbito estrutural e comportamental. O Quadro 0.1, apresentado no final do capítulo, consolida os resultados das avalições de cada critério para cada trabalho relacionado investigado. A seguir segue uma breve descrição de cada critério definido, classificado em Geral, Estrutural e Comportamental.

Geral:

- **Utiliza um modelo simplificado para representar a aplicação** – Indica se o processo de análise faz uso de um modelo simplificado para estruturar a aplicação em memória;
- **Possibilita a integração com um ambiente de desenvolvimento** – Indica se provê facilidades de integração com um ambiente de desenvolvimento;
- **Reusa o resultado das avaliações prévias** – Indica se o processo de análise reutiliza o resultado de análises prévias para compor as novas análises, a fim de otimizar o processo;
- **Permite determinar os trechos de códigos alvos para a análise** – Indica se existe a possibilidade de direcionar os artefatos ou trechos de código que devem ser analisados, limitando o escopo da análise;
- **Permite determinar o conjunto de regras a serem utilizadas pela análise** – Indica se é possível determinar quais regras serão utilizadas no processo de análise.

- **Classifica os problemas identificados** – Indica se os problemas identificados são agrupados em alguma categoria que facilite o entendimento;
- **Fornece insumos para correção dos problemas identificados** – Indica se a solução fornece informações que auxiliam o processo de correção dos problemas, além de apenas indicar a existência do problema;
- **Fornece detalhes da confiabilidade da análise** – Indica se a solução provê informações acerca da confiabilidade de seus resultados;
- **Fornece detalhes da completude da análise** – Indica se a solução provê informações acerca da completude de sua análise, determinando quais regras foram consideradas no seu processo;
- **Utiliza um modelo estruturado para representação das regras** – Indica se a solução utiliza um modelo estruturado para representar suas regras; e
- **Permite a criação/atualização das regras** – Indica se as regras da solução podem ser estendidas ou atualizadas pelo usuário.

Estrutural:

- **Permite realizar a análise estrutural (validação)** – Indica se a solução provê análise estrutural das aplicações de seu contexto;
- **Permite a identificação de problemas estruturais sintáticos** – Indica se a solução provê a identificação de problemas estruturais sintáticos; e
- **Permite a identificação de problemas estruturais semânticos estáticos** – Indica se a solução provê a identificação de problemas estruturais semânticos estáticos.

Comportamental:

- **Permite realizar a análise comportamental (verificação)** – Indica se a solução provê análise comportamental das aplicações de seu contexto;
- **Permite identificar inconsistências descritivas** – Indica se a solução provê mecanismos para a identificação das inconsistências comportamentais descritivas;
- **Permite identificar inconsistências comportamentais** – Indica se a solução provê mecanismos para a identificação das inconsistências comportamentais interativas;

- **Permite identificar inconsistências referentes à linguagem** – Indica se a solução provê mecanismos para a identificação das inconsistências comportamentais padronizadas pela linguagem; e
- **Permite identificar inconsistências referentes à aplicação** – Indica se a solução provê mecanismos para a identificação das inconsistências comportamentais próprias da aplicação.

Honorato e Barbosa (2010) apresentam um sistema de críticas para o código NCL denominado NCL-Inspector, que apoia a autoria de aplicações NCL, por meio da inspeção do código, identificando erros e sugerindo melhorias. O principal objetivo do trabalho é fornecer uma interface para que o usuário possa definir e agrupar regras, constituídas de padrões XML, para serem verificadas de acordo com a necessidade do usuário. Cada regra pode ser classificada de acordo com sua finalidade: que visam identificar erros de codificação baseados na sintaxe e semântica estática da linguagem; que buscam encontrar trechos de códigos ambíguos ou imprevisíveis, que apesar de seguir a especificação apresentam um comportamento inesperado; que permitem identificar *bad smells*, isto é, práticas equivocadas que reduzem a legibilidade, manutenabilidade e o reuso do código NCL; ou que identificam problemas de interação do usuário. Outro recurso importante do trabalho possibilita ao usuário definir e selecionar em tempo de execução grupos de regras de acordo com a sua necessidade. Esse trabalho concretiza sua solução em uma ferramenta que pode ser integrada a um ambiente de desenvolvimento.

A ferramenta é composta pelos seguintes componentes: ***parser***, que é responsável por traduzir o código fonte em uma árvore sintática abstrata (*AST - Abstract Syntax Tree*) que representa o NCL; ***rules engine***, componente principal responsável por caminhar na árvore sintática avaliando as regras associadas a cada elemento a fim de identificar todas as violações detectadas durante o processo de inspeção, além de provê uma interface para manipulação da biblioteca de regras; ***rules library***, que representa a biblioteca de regras e tem por objetivo fornecer as regras para o motor de avaliação; e ***visitors***, que são responsáveis por verificar os padrões de cada regra.

Apesar de fornecer alternativas para que o desenvolvedor possa identificar tanto os problemas estruturais quanto os comportamentais, o NCL-Inspector limita-se a realizar uma análise estritamente estática do código da aplicação, não permitindo identificar as inconsistências temporais ou interativas. Além disto, essa abordagem definida pelo NCL-Inspector força que o desenvolvedor tenha um posicionamento bem mais ativo no processo de identificação de erros, definindo e pré-selecionando o escopo da análise de acordo com as suas necessidades. Uma limitação é o fato de não

prevê estratégias de reuso de suas análises, importantes no processo de autoria incremental, sempre demandando a análise de todo documento.

Santos (2012) propõe uma abordagem orientada a modelos para analisar documentos NCL. Com isto, busca predefinir propriedades estruturais e comportamentais para serem verificadas por meio do sistema Maude. Além disto, fornece a implementação de uma biblioteca que concretiza a abordagem proposta, denominada aNaa (*API for NCL Authoring and Analysis*), e, também, disponibiliza uma interface de acesso na web (aNaa4Web[34]).

Nesse trabalho, as propriedades estruturais representam as regras de sintaxe definidas pela gramática do NCL e as regras resultantes das restrições semânticas estáticas. Estas propriedades foram classificadas em: **propriedades sintáticas**, que representam as restrições léxicas e sintáticas da linguagem; **propriedades hierárquicas**, que representam as restrições relacionadas com a existência e a cardinalidade dos elementos filhos; **propriedades de atributo**, que representam as restrições relacionadas com a existência, a obrigatoriedade e a unicidade dos atributos; **propriedades de referência**, que representam as restrições que definem as referências entre elementos; **propriedade de composição**, que representam as restrições de escopo para a composição dos elementos; **propriedades de composição aninhada**, que representam as restrições de aninhamento para a composição dos elementos; e **propriedades de reuso**, que representam as restrições de reuso para a composição dos elementos. As propriedades estruturais são verificadas por meio da redução equacional. Enquanto que, as propriedades comportamentais, constituem fórmulas lógicas temporais originadas das regras semânticas dinâmicas. Estas propriedades foram classificadas em: **propriedades de acessibilidade**, que permitem verificar a acessibilidade dos elementos; **propriedades de terminação de âncora**; que permitem verificar a consistência do término das apresentações das âncoras; **propriedades de terminação de documentos**, que permitem verificar o a consistência do término da apresentação dos documentos; e **propriedades de recurso**, que permitem verificar a disponibilidade e a simultaneidade do acesso aos recursos. As propriedades comportamentais são avaliadas por meio da verificação de modelos.

Apesar de propor a utilização de um modelo formal simplificado para representar a aplicação, a qual permite a avaliação automática das propriedades, esta abordagem não prevê estratégias de reuso e nem permite limitar o escopo de suas análises, características importantes no processo de

34 SANTOS. aNaa4Web. Disponível em: < www.midiacom.uff.br/~joel/anaa4web>. Acesso em: 20 de Setembro de 2016.

autoria incremental. Uma facilidade provida por esse trabalho é a possibilidade de integrar a sua biblioteca disponibilizada a um ambiente de desenvolvimento. No escopo da análise estrutural, permite a identificação tanto dos problemas sintáticos quanto dos problemas semânticos estáticos. No escopo da análise comportamental, limita-se a identificar a existência de heurísticas temporais preestabelecidas, que representam as inconsistências comportamentais, no fluxo de execução da aplicação. Desta forma, não identifica as inconsistências espaciais e também não permite a identificação de comportamentos indesejáveis relacionados com as regras de negócio da aplicação. Apesar de possuir uma série de limitações, referentes à cobertura de análise tanto no escopo estrutural quanto no escopo comportamental, permite a extensão da cobertura de acordo com a necessidade do usuário por meio da definição de novas regras a serem codificadas por uma metalinguagem pré-definida.

Picinin *et al.* (2012) tem como foco a definição de um modelo formal abstrato que permite identificar a existência de comportamentos indesejáveis durante o processo de desenvolvimento, por meio da execução de uma verificação formal de algumas propriedades expressas por fórmulas lógicas temporais em LTL e CTL. O desempenho dessa abordagem também é uma preocupação explícita deste trabalho, dado que o processo de verificação formal demanda uma elevada quantidade de recursos computacionais, devido, principalmente, ao problema da "explosão de estados" (WAGNER *et. al.*, 2006). Desta forma, busca avaliar apenas as partes do código que foram modificadas ou impactadas desde a última verificação. As propriedades definidas foram classificadas de acordo com sua origem em **propriedades gerais**, aplicadas a todos os objetos da aplicação, e **propriedades específicas**, aplicadas a um conjunto de objetos específicos, determinando o relacionamento entre estes objetos. A ferramenta também provê uma interface para que o usuário possa incluir novas propriedades específicas. As propriedades também podem ser classificadas de acordo com sua finalidade em **propriedades espaciais** e **propriedades temporais**, que, respectivamente, permitem identificar as inconsistências espaciais e temporais.

A abordagem foi concretizada por meio da implementação de uma ferramenta integrada ao ambiente de desenvolvimento Composer. Nesta ferramenta, o processo de verificação é conduzido inicialmente com toda aplicação sendo traduzida para um modelo abstrato no formato de uma grafo, o qual permite facilmente identificar o relacionamento entre os objetos e facilita a tradução para o modelo formal. Após esta etapa, algumas operações de redução são aplicadas no grafo, principalmente com o objetivo de manter apenas as estruturas relacionadas com as recentes modificações, delimitando o escopo da verificação. Após o processo de redução, o grafo resultante é

traduzido para um modelo formal parcial, contendo apenas os estados referentes às recentes modificações do desenvolvedor. Este modelo formal é exercitado exaustivamente, combinando todas as possíveis execuções, avaliando todas as propriedades. E, por fim, gera um contraexemplo para cada propriedade não satisfeita, que tem por objetivo fornecer ao desenvolvedor o caminho onde a inconsistência (temporal ou espacial) ocorreu.

A abordagem definida agrega dois requisitos relevantes para o processo de análise de aplicações multimídia, a verificação de propriedades comportamentais atrelada a um processo de verificação incremental. Apesar da verificação parcial se mostrar eficiente, a solução não permite limitar a verificação de acordo com a necessidade do usuário, relevante em um processo de autoria colaborativo. O processo de verificação é bastante completo, pois permite a identificação tanto das inconsistências temporais quanto das espaciais. Além disto, permite que o desenvolvedor possa incluir novas propriedades a serem avaliadas. Porém, se limita a identificar a existência de heurísticas preestabelecidas, os quais representam as inconsistências comportamentais, no fluxo de execução da aplicação, negligenciando os comportamentos inesperados dinâmicos. Um recurso relevante desse trabalho, é que ele fornece detalhes acerca do problema identificado, além de apontar o problema apresenta a sequência de passos que o ocasionou, porém não define explicitamente as ações corretivas.

Em Santos *et. al.* (2013) é definido e implementado um mecanismo de validação estática incremental de aplicações NCL, o qual busca avaliar apenas as partes do código que foram modificadas ou impactadas desde a última validação. A abordagem incremental busca otimizar o processo de validação natural que, desnecessariamente, reavalia todo o documento NCL. O foco principal é diminuir o esforço de validação, identificando o menor conjunto de segmentos a ser validado, composto pelos trechos que sofreram modificações diretas, ou pelos que, possivelmente, foram afetados. Além disto, permite a configuração automática do tipo de validação a ser feita, sintática ou estrutural, por exemplo, baseado no conjunto de elementos a serem validados. A eficiência desta abordagem é evidenciada, principalmente, quando incorporada a um ambiente de desenvolvimento, independente da interface de codificação (textual ou visual). Dado que, no contexto textual, pode operar de forma "online", atuando imediatamente após o usuário parar de digitar, e, no contexto visual, pode processar a avaliação após a adição, edição ou remoção de um componente visual. A implementação da validação foi incorporada ao Composer, um ambiente de desenvolvimento focado na codificação textual e/ou visual de aplicações NCL.

Fica evidente que a eficiência de validação é o principal requisito desse trabalho, buscando realizar uma validação mais inteligente, limitando o escopo da validação e reutilizando validações prévias para compor o resultado final. A integração com um ambiente de desenvolvimento também agregou eficiência ao processo de validação, pois as verificações podem ser realizadas "online", durante o processo de desenvolvimento. Uma melhoria identificada é a possibilidade de limitar a validação de acordo com a necessidade do usuário, relevante em um processo de autoria colaborativo. Uma vez que, as alterações em um determinado documento não estão necessariamente relacionadas com o escopo de atuação de um desenvolvedor específico, elas podem ter sido originadas pela codificação de outros desenvolvedores. O processo de validação permite a identificação tanto dos problemas sintáticos quanto dos problemas semânticos estáticos. Assim como em alguns trabalhos avaliados, permite inserir e editar as regras de análise a partir da manipulação de uma especificação simplificada constituída por todas as regras suportada pela solução, escrita por meio de uma metalinguagem predefinida.

Santos, Moreno e Soares (2015) apresentam uma ferramenta de monitoramento integrada a uma implementação do middleware Ginga-NCL. A ferramenta fornece *feedback* sobre os estados das variáveis, as propriedades dos objetos, os tempos de apresentação das mídias, entre outros em tempo de execução. Com isso, permite detectar se os problemas visuais estão sendo causados por erros de programação ou por erros de interpretação. Durante a execução da aplicação, o motor de apresentação e o de monitoramento trocam várias mensagens para monitorar a execução e manter a rastreabilidade dos problemas. A interface gráfica da ferramenta apresenta uma linha do tempo contendo a visão geral das mídias, evidenciando os momentos de execução. Quando a apresentação termina, a ferramenta gera um relatório que descreve as etapas da execução a partir de uma linha do tempo contendo a visão geral das mídias, evidenciando os momentos de execução.

A solução proposta por esse trabalho busca estender um interpretador real da linguagem NCL, a fim de viabilizar o processo de depuração das aplicações, evidenciando as variáveis que controlam a execução. Desta forma, se limita a identificar problemas comportamentais, a serem verificados em tempo de execução. Além disso, herda os mesmos problemas das abordagens que utilizam interpretadores nesse processo de verificação. O principal problema dessas abordagens é que todo o esforço fica a cargo do desenvolvedor, tanto para simular os cenários como para identificar os problemas existentes. Assim como a maioria das soluções analisadas, também não provê recursos para delimitar o escopo da análise. Um recurso interessante provido pela solução é a

disponibilização de um relatório de execução, que pode ser utilizado como entrada de alguma solução que busque automatizar as verificações.

Gaggi e Bossi (2011) fornecem um ambiente de desenvolvimento voltado para a construção de aplicações multimídia codificadas em SMIL. Além de fornecer funcionalidades relacionadas ao processo de autoria dessas aplicações, o ambiente proposto também provê uma interface para um modulo denominado *Semantic Validator Module* que permite a análise do comportamento temporal por meio da definição de uma semântica formal para a linguagem SMIL. Esta semântica define os aspectos temporais dos objetos da aplicação por meio de um conjunto de regras de inferência fundamentadas na lógica de Hoare. Esta lógica é fundamentada na seguinte premissa: dado um conjunto de asserções existentes (pré-condição) a execução de um determinado comando resulta em um conjunto de asserções conhecidas (pós-condição). Desta forma, permite enriquecer a estrutura da linguagem com as assertivas que expressam as propriedades temporais, com o objetivo de diminuir a complexidade da análise. Estas assertivas determinam o estado da aplicação anterior e posterior à execução de uma *tag* ou um conjunto de *tags* SMIL. Essa abordagem permite à ferramenta realizar a verificação da corretude do documento multimídia aplicando axiomas pré-definidos, a fim de verificar se a *tag* ou o conjunto de *tags* altera corretamente o estado do sistema.

Bouyakoub e Belkhir (2011) apresentam o SMIL *Builder*, um ambiente de desenvolvimento que fornece uma abordagem gráfica para construção de aplicações multimídia codificadas em SMIL. Formalmente, utiliza uma abordagem baseada em redes de Petri, denominada H-SMIL-Net, a fim de permitir uma verificação comportamental incremental. No processo de desenvolvimento o usuário cria a aplicação visualmente, definindo as relações de sincronia espaço-temporal entre as mídias, e, após cada modificação, a ferramenta verifica a consistência dessas relações referentes ao trecho alterado antes de aceitá-lo. No caso de alguma inconsistência ser identificada, o sistema rejeita a alteração e alerta o usuário para que ele possa atuar no problema. Assim, a alteração em uma parte da aplicação não leva à verificação de todo o modelo, apenas da parte afetada pela modificação. Essa abordagem é adequada para o contexto de desenvolvimento, dado que a complexidade dos modelos temporais da linguagem SMIL pode conduzir os autores, em alguns casos, a especificar relações de sincronização que não poderiam ser satisfeitas durante a apresentação do documento, caracterizando assim a ocorrência de inconsistências temporais. A utilização da H-SMIL-Net possibilita a identificação desses tipos de inconsistências, como por exemplo, a inconsistência intra-elemento, conflito entre os atributos temporais de um mesmo elemento (*dur*, *begin* e *end*), ou a inter-elemento conflito entre os atributos temporais de diferentes elementos relacionados. Vale salientar que a

especificação da linguagem SMIL define algumas convenções que tratam alguns conflitos temporais, ignorando um ou mais atributos conflitantes. Porém, o seguimento dessas convenções fica a cargo da implementação do interpretador. Desta forma, a existência desses conflitos, mesmo que esperados pela especificação pode representar uma espécie de *bad smell*, pois algumas relações podem depender desses atributos que poderão ser ignorados.

Em Dahmani *et al.* (2014) é proposta uma abordagem baseada em redes de Petri, com objetivo de verificar a consistência temporal de aplicações multimídia codificadas em SMIL durante o processo de desenvolvimento. O processo de verificação utiliza o formalismo de um tipo de rede de Petri denominado *Time Extended Recursive Petri Nets Plus* (*Time ERPN+*), o qual permite avaliar a sincronização de objetos de mídia com referências temporais distintas. Os autores buscaram evoluir esta abordagem ao longo dos anos (DAHMANI *et. al.*, 2014; DAHMANI *et. al.*, 2013; DAHMANI *et. al.*, 2008) provendo melhorias ao modelo proposto para dar suporte a verificação temporal de diversos tipos de estruturas com variados níveis de complexidade. Em sua última versão (*Time ERPN+*) o modelo possibilita verificar a consistência temporal entre objetos que possuem referências distintas. As restrições temporais podem ter a mesma referência, com as relações entre os objetos sendo definidas por meio de uma função temporal comum, ou referências distintas, com as relações entre os objetos sendo definidas por meio das possibilidades de intersecção de suas funções temporais específicas. A abordagem consiste em traduzir as aplicações SMIL para o modelo formal *Time ERPN+*, o qual permite verificar a consistência temporal dessas aplicações, desta forma busca avaliar se todos os estados finais podem ser acessados a partir do estado inicial. Com isso, um documento SMIL é considerado consistente apenas se as ações que iniciam a apresentação são seguidas, necessariamente, pelas ações que a terminam. Além disto, esta abordagem também permite identificar os objetos que nunca iniciam ou que iniciam e nunca terminam. Como trabalho futuro, os autores pretendem estender o modelo para permitir a verificação de inconsistências espaciais.

Apesar de apresentar soluções distintas, com modelos formais específicos, que permitem verificar a consistência temporal das aplicações multimídia codificadas em SMIL, os trabalhos de Gaggi e Bossi (2011), Bouyakoub e Belkhir (2011) e Dahmani *et. al.* (2014), possuem contribuições e limitações bem semelhantes. Todos buscam prover mecanismos que permitem identificar inconsistências temporais por meio de uma tradução da aplicação para um modelo formal, porém apenas o SMIL *Builder* (BOUYAKOUB; BELKHIR, 2011) permite reuso de suas avaliações, avaliando, a cada modificação, apenas os trechos que ainda não foram verificados. Todos têm como limitação a impossibilidade de limitar o escopo da avaliação de acordo com a necessidade do usuário.

Além disto, se restringem a identificar as inconsistências temporais determinadas pela especificação da linguagem, negligenciando as inconsistências espaciais e os comportamentos inesperados referentes às regras de negócio da aplicação.

Hallé *et.al.* (2015) introduz o Cornipickle, uma ferramenta de teste automatizado que provê uma linguagem declarativa para expressar propriedades desejáveis de uma aplicação web como um conjunto de assertivas legíveis a serem verificadas nas páginas Web (HTML e CSS). Em Cornipickle, as propriedades são expressas a partir do conteúdo dos atributos da página em um dado momento e utiliza a Lógica Temporal Linear (LTL) para definir assertivas que verificam as propriedades em tempo de execução. O processo de verificação é bastante flexível, permitindo ao usuário definir propriedades a serem avaliadas de acordo com a linguagem e com as regras de negócios.

A ferramenta proposta por esse trabalho atua estritamente no escopo da análise comportamental, partindo de uma interpretação inicial do seu domínio de variáveis. Porém, a flexibilidade provida pela ferramenta traz um alto custo, por demandar um elevado esforço manual tanto para a construção das assertivas como para o processo de verificação, que ocorre em tempo de execução. Além disso, não prevê estratégias de reuso de suas análises, importantes no processo de autoria incremental, sempre demandando a análise de todo documento Um diferencial dessa ferramenta é o fato dela permitir ao usuário definir quais assertivas serão testadas, delimitando o escopo da análise.

Takhma *et.al.* (2015) apresenta uma ferramenta integrada ao Eclipse para análise estática de código para a plataforma MyIC, o seu objetivo é verificar a conformidade dos aplicativos de terceiros (por exemplo, aplicativos Web) com padrões de plataforma MyIC. Sua abordagem utiliza um modelo abstrato como estrutura intermédia para realizar a análise. As regras são construídas a partir de uma metalinguagem (baseada em XML), permitem realizar verificações estruturais e comportamentais (estáticas) e são originadas tanto pelas especificações da plataforma como pelos guias de melhores práticas.

A ferramenta proposta por esse trabalho atua no escopo da análise estrutural (estática) de aplicações Web, realizada a partir do código fonte das aplicações. Desta forma, busca avaliar a conformidade desses códigos com as restrições de uma plataforma de telefonia, a fim de evidenciar quais aplicações podem ser executadas. A essência da solução é atuar nas restrições da linguagem, porém também permite ao usuário criar novas regras, para verificar restrições do negócio, por exemplo. Assim como a maioria das soluções avaliadas, não prevê estratégias de reuso de suas análises. Porém, traz a facilidade de já ser integrada ao seu ambiente de desenvolvimento.

No geral, independente do propósito da solução e do escopo da análise (estrutural ou comportamental), nenhuma das soluções analisadas fornecem detalhes acerca da confiabilidade e da completude das suas regras de análise, não evidenciando, respectivamente, as referências que fundamentaram a criação das regras e o escopo de análise suportado pelas soluções. Outra limitação comum entre esses trabalhos é a ausência de informações que complementam o entendimento do problema, geralmente se restringem a apontar os problemas identificados, não definindo explicitamente as ações corretivas. Os detalhes acerca do problema são importantes para o processo de correção, principalmente para desenvolvedores inexperientes.

1.1 Considerações Finais

Este capítulo trouxe como principal contribuição a análise detalhada dos trabalhos relacionados, evidenciando características relevantes de cada escopo de análise: **Estrutural** e **Comportamental**. Desta forma, foi possível identificar as principais contribuições, as limitações e os problemas reincidentes em cada área. Possibilitando, incorporar os principais requisitos destacados, sugerir melhorias e prevenir os problemas já conhecidos, respectivamente.

A fim de confrontar a solução proposta com os trabalhos relacionados, o Quadro 0.1 a compara a proposta definida por esta tese (**PT16)** com os trabalhos analisado: **ST12** (SANTOS, 2012); **ST13** (SANTOS *et. al.*, 2013); **PN12** (PICININ *et. al.*, 2012); **HT10** (HONORATO; BARBOSA, 2010); **DN14** (DAHMANI *et. al.*, 2014); **GG11** (GAGGI;BOSSI, 2011); **HL15** (HALLÉ *et.al.,* 2015); **TK15** (TAKHMA *et. al.*, 2015); and **ST15** (SANTOS, MORENO e SOARES, 2015) e **BK11** (BOUYAKOUB; BELKHIR, 2011). Cada critério foi avaliado de acordo com a sua completude, onde: **C** (Completo) atende o critério de avaliação completamente; **P** (Parcial) atende o critério de avaliação parcialmente; **N** (Não) não atende o critério de avaliação; **N/I** (Não Informado) não traz informações acerca do critério de avaliação; e **N/A** (Não Aplicado) não é aplicado para o escopo do trabalho avaliado.

Quadro 0.1. Avaliação dos Trabalhos Relacionados

Característica	PT16	HL15	ST15	TM15	ST12	ST13	PN12	HT10	DN14	GG11	BK11
Geral											
Utiliza um modelo simplificado para representar a aplicação	C	N	N	C	C	N	C	C	C	C	C
Possibilita a integração com um ambiente de desenvolvimento	C	N	N	C	C	C	C	C	N/I	N	C

Reusa o resultado das análises prévias	N	N	N	N	N	C	C	N	N	N	N
Permite determinar os trechos de códigos alvos para a análise	N	N	P	N	N	P	P	N	N	N	P
Permite determinar o conjunto de regras a serem utilizadas pela análise	C	C	N	C	N	N	N	C	N	N	N
Classifica os problemas identificados	C	C	N	C	C	N	P	P	N	N	N
Fornece insumos para correção dos problemas identificados	C	P	N	P	N	N	P	N	N	N	N
Fornece detalhes da confiabilidade da análise	C	N	N	N	N	N	N	N	N	N	N
Fornece detalhes da completude da análise	C	N	N	N	N	N	N	N	N	N	N
Utiliza um modelo estruturado para representação das regras	N	C	N	C	C	C	C	C	N	C	N
Permite a criação\atualização das regras	N	C	N	C	C	C	C	C	N	N	N
Estrutural											
Permite realizar a análise estrutural (validação)	P	N	N	C	C	C	N	P	N	N	N
Permite a identificação de problemas estruturais sintáticos	P	N	N	C	C	C	N/A	P	N/A	N/A	N/A
Permite a identificação de problemas estruturais semânticos (estático)	P	N	N	C	C	C	N/A	P	N/A	N/A	N/A
Comportamental											
Permite realizar a análise comportamental (verificação)	P	P	P	N	P	N	C	P	P	P	P
Permite identificar inconsistências descritivas	P	P	P	N/A	P	N/A	C	P	N	N	N
Permite identificar inconsistências comportamentais	P	P	P	N/A	C	N/A	C	P	P	P	P
Permite identificar inconsistências referentes à linguagem	P	C	P	N/A	C	N/A	C	P	C	C	C
Permite identificar inconsistências referentes à aplicação	P	P	P	N/A	N	N/A	P	P	N	N	N

É importante ressaltar que a maioria dos trabalhos avaliados faz uso de um modelo simplificado intermediário para representar a aplicação e permite a integração com ambientes de desenvolvimento, características importantes, independentes do formalismo utilizado pela solução. Tais características possibilitam a identificação dos problemas durante as fases de desenvolvimento, sem demandar esforço de execução. Poucos trabalhos permitem a delimitação do escopo de avaliação, tanto com relação ao código fonte quanto com relação às regras utilizadas, esses, normalmente, promovem essa limitação automaticamente apenas com o objetivo de diminuir o tempo de análise, não se preocupando com as necessidades do usuário. Também é importante destacar que, apesar de boa parte dos trabalhos estruturarem as regras de análise, eles o fazem apenas para viabilizar a identificações de problemas relacionados com as regras de negócio, permitindo visualizar, atualizar ou criar novas regras de acordo com a necessidade do usuário. Com relação aos trabalhos que possibilitam a análise comportamental, é importante destacar que apenas um permite ao usuário realizar uma análise comportamental completa, identificando as inconsistências comportamentais e descritivas referentes tanto às restrições da linguagem quanto às relacionadas com as regras de negócio da aplicação. Todos os trabalhos desse contexto fornecem facilidades para identificar as inconsistências comportamentais referentes às restrições da linguagem, porém poucos identificam inconsistências descritivas e suportam restrições relacionadas com a aplicação. Ainda convém ressaltar que todos os trabalhos avaliados, independente do escopo de análise, não proveem detalhes da confiabilidade e da completude da análise e poucos fornecem diretrizes para a correção dos problemas identificados.

O próximo capítulo busca apresentar as contribuições e conclusões da tese, bem como as perspectivas de trabalhos futuros.

Conclusão

O objetivo geral deste trabalho foi investigar a viabilidade de se aplicar análises orientadas a conformidade a fim de prover uma abordagem alternativa para atestar a qualidade das aplicações multimídia construídas a partir de linguagens declarativas padronizadas por especificações públicas. Para este fim, considerou-se que a qualidade é determinada pela incidência de problemas estruturais (léxicos e sintáticos) e comportamentais (semânticos). Sendo assim, a pesquisa foi conduzida para prover uma solução genérica que auxilie a identificação e correção desses problemas, que possa ser adaptável a qualquer linguagem desse contexto, eliminando a dependência de utilização de interpretadores para avaliar a qualidade dessas aplicações. A solução proposta foi aplicada em um caso de uso, concretizando-a em uma ferramenta de análise estrutural e comportamental de aplicações NCL. Posteriormente, foi avaliada sob a perspectiva da confiabilidade dos seus resultados, por meio de duas rodadas de experimentos aplicados em versões distintas da ferramenta, utilizando conjuntos de aplicações diferentes. Os resultados dos experimentos evidenciaram que a ferramenta é mais confiável na identificação dos comportamentos das aplicações NCL do que alguns interpretadores reais da linguagem (Astrobox e EiTV smartBox). Além disso, para a consolidação da solução proposta, também foi importante aplicar suas definições em outros contextos de desenvolvimento (HTML e SVG). Apesar de utilizar um escopo limitado na implementação das ferramentas de análise de aplicações HTML e SVG, os resultados obtidos mostraram indícios da aplicabilidade flexível da solução proposta.

As seções a seguir apresentam as principais contribuições e limitações desse trabalho, além de destacar algumas diretrizes de trabalhos futuros.

1.2 Contribuições e Limitações

Esta seção tem por objetivo consolidar as principais contribuições e limitações deste trabalho, que foram apresentadas ao longo dos capítulos.

Contribuições:

- Abordagem para promover a rastreabilidade das análises estáticas e dinâmicas no contexto das linguagens declarativas padronizadas, viabilizada por meio do uso de assertivas no processo de estruturação das especificações públicas;
- Representação comportamental aderente com os requisitos das aplicações multimídia, que busca evidenciar tanto o fluxo de controle, quanto o comportamento dos dados de entrada e saída;
- Abordagem sistemática para a realização da análise comportamental no contexto das linguagens declarativas padronizadas, aplicada e avaliada por meio de um caso de uso; e
- Plano experimental para avaliar a confiabilidade de representações comportamentais abstratas perante interpretadores concretos, fundamentada na geração de testes orientada a modelo.

Limitações:

- Apesar de a abordagem proposta poder ser aplicada no contexto de outras linguagens declarativas que atuam no desenvolvimento de aplicações multimídia, a solução implementada na prova de conceito considerou apenas a linguagem NCL para avaliação da proposta. Outras soluções também foram implementadas para as linguagens HTML e SVG, apenas com o objetivo de investigar a viabilidade técnica para implementação da proposta;
- Dado que algumas aplicações multimídia utilizam *scripts* como agente comportamental (que altera o fluxo da aplicação), é importante destacar que a abordagem proposta não considera esses *scripts* em seu processo de análise;
- A solução implementada não buscou fornecer uma implementação completa da proposta, o escopo foi reduzido para permitir a avaliação da proposta com um esforço reduzido por meio de uma prova de conceito consistente;
- Apesar de a abordagem proposta sugerir a melhoria na produtividade e na qualidade do processo de desenvolvimento, ela foi avaliada apenas com relação à confiabilidade da representação comportamental;
- No contexto da análise comportamental, a solução implementada não fornece a identificação das inconsistências comportamentais de forma automática. A solução

busca prover insumos por meio dos modelos extraídos que facilitem a identificação dessas inconsistências;

- Não provê recursos de depuração que permitem depurar a abstração comportamental, por meio do uso de *breakpoints* associados aos eventos e às variáveis da aplicação;
- A abordagem proposta não promove o reuso de análises prévias. Novas análises sempre reavaliam todo o código fonte da aplicação, independente das alterações realizadas; e
- A solução implementada não provê uma atualização sincronizada dos modelos em relação ao código. Desta forma, é necessário que o desenvolvedor determine o momento da análise de acordo com sua necessidade.

1.3 Trabalhos Futuros

Como trabalho futuro, pretende-se atuar nas limitações desse trabalho a fim de:

- Estender a ferramenta de análise de aplicações NCL para suportar todas as regras da sua especificação e incrementar suas funcionalidades, a fim de prover uma solução completa para análise das aplicações desse contexto;
- Estender a ferramenta de análise de aplicações NCL para prover recursos de depuração baseados nas representações gráficas utilizadas pela ferramenta, através do uso de breakpoints associados aos eventos e variáveis da aplicação;
- Estender a ferramenta para realizar o reuso de análises prévias, a fim de viabilizar a automatização do processo de análise;
- Planejar e executar experimentos que busquem evidenciar o impacto da solução proposta em outras atividades importantes do processo de desenvolvimento, além de, investigar a suposta melhoria na produtividade dos desenvolvedores e na qualidade das aplicações desenvolvidas;
- Investigar o impacto dos *scripts* que complementam as linguagens declarativas na solução proposta, para propor extensões que contemplem os *scripts* como agentes modificadores do fluxo da aplicação;
- Investigar a utilizar da representação comportamental no processo de geração de código automático, a fim de aumentar a produtividade na construção das aplicações multimídias; e

- Aplicar e avaliar a solução proposta em outros contextos de desenvolvimento, para comprovar sua flexibilidade. Primeiramente, estendendo as ferramentas já desenvolvidas (HTML e SVG), considerando um escopo mais representativo da linguagem que permita realizar avaliações experimentais.

Referências

ABNT (2015). ABNT NBR 15606-2:2015 - Televisão digital terrestre - Codificação de dados e especificações de transmissão para radiofusão digital Parte 2: Ginga-NCL para receptores fixos e móveis - Linguagem de aplicação XML para codificação de aplicações, 2015. Disponível em: <http://www.abntcolecao.com.br/normavw.aspx?ID=329289>. Acesso em: 20 de Setembro de 2016.

AHSANULLAH, M.; KIBRIA, B. M. G.; SHAKIL, M. (2014). Normal and Student´s t Distributions and Their Applications. Atlantis Press. v.4, 2014. DOI=10.2991/978-94-6239-061-4_3.

AMMONS, G.; RASTISLAV, B.; LARUS, J. R. (2002). Mining specifications. In Proceedings of the 29th ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL '02). ACM, New York, NY, USA, 4-16. DOI=10.1145/503272.503275 http://doi.acm.org/10.1145/503272.503275.

ARAÚJO, E. C.; AZEVEDO, R. G. D. A.; NETO, C. D. S. S. NCL-Validator: um processo para validação sintática e semântica de documentos multimídia NCL. In: II Jornada de Informática do Maranhão, 2008.

AZEVEDO, R. G. A.; SOARES NETO, C. de S.; TEIXEIRA, M. M.; SANTOS, R. C. M.; GOMES, T. A. (2011). Textual authoring of interactive digital TV applications. In Proceddings of the 9th international interactive conference on Interactive television (EuroITV '11). ACM, New York, NY, USA, 235-244. DOI=10.1145/2000119.2000169 http://doi.acm.org/10.1145/2000119.2000169

BELLUCCI, F.; GHIANI, G.; PATERNÒ, F.; Porta, C. (2012). Automatic reverse engineering of interactive dynamic web applications to support adaptation across platforms. Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces - IUI ’12, 217, 2012. http://doi.org/10.1145/2166966.2167004

BOSSI, A.; GAGGI, O. (2007). Enriching SMIL with assertions for temporal validation. Proceedings of the 15th International Conference on Multimedia - MULTIMEDIA'07. 2007. http://doi.org/10.1145/1291233.1291256

BOUYAKOUB, S.; BELKHIR, A. (2008). H-SMIL-Net: A hierarchical Petri Net model for SMIL documents. Proceedings - UKSim 10th International Conference on Computer Modelling and Simulation. EUROSIM/UKSim2008, 2008. http://doi.org/10.1109/UKSIM.2008.54

BOUYAKOUB, S.; BELKHIR, A. (2011). SMIL builder. ACM Transactions on Multimedia Computing, Communications, and Applications. 2011. http://doi.org/10.1145/1870121.1870123

BUSSAB, W.; MORETIN, P. (2010). Estatística Básica. Editora Saraiva. 6º Edição. 2010.

CHANG, A. Y. (2005). An intelligent semantic model for the design of consistent online SMIL presentations. Proceedings - International Conference on Advanced Information Networking and Applications, AINA, 2005. http://doi.org/10.1109/AINA.2005.111.

CHENG, K. T.; KRISHNAKUMAR, A. S. (1993). Automatic functional test generation using the extended finite state machine model. In Proceedings of the 30th international Design Automation Conference (DAC '93). ACM, New York, NY, USA, 86-91, 1993. DOI=10.1145/157485.164585 http://doi.acm.org/10.1145/157485.164585

CHOUDHARY, S. R.; PRASAD, M. R.; ORSO, A. (2014). X-PERT: a web application testing tool for cross-browser inconsistency detection. In Proceedings of the 2014 International Symposium on Software Testing and Analysis (ISSTA 2014). ACM, New York, NY, USA, 417-420, 2014. DOI=10.1145/2610384.2628057 http://doi.acm.org/10.1145/2610384.2628057.

CHOW, T. (1978). Testing software design modelled by finite state machines. IEEE Trans. Softw. Eng. 4,3, 178–187, 1978.

DAHMANI, D.; ILIÉ, J.; BOUKALA, M. (2008). Time Recursive Petri Nets. In Transactions on Petri Nets and Other Models of Concurrency I, Kurt Jensen, Wil M. Aalst, and Jonathan

Billington (Eds.). Lecture Notes In Computer Science, Vol. 5100. Springer-Verlag, Berlin, Heidelberg 104-118, 2008. DOI=10.1007/978-3-540-89287-8_7 http://dx.doi.org/10.1007/978-3-540-89287-8_7

DAHMANI, D.; MAZOUZ, S.; BOUKALA, M. (2013). Modular modeling and analyzing of multimedia documents with repetitive objects. 2013 5th International Conference on Computer Science and Information Technology, CSIT 2013 - Proceedings, 308–316, 2013. http://doi.org/10.1109/CSIT.2013.6588797

DAHMANI, D.; MAZOUZ, S.; BOUKALA, M. (2014). Efficient and modular modeling of hierarchical multimedia objects by using time ERPN+. In Multimedia Computing and Systems (ICMCS), 2014 International Conference on , vol., no., pp.1153-1158, 14-16, April, 2014. DOI=10.1109/ICMCS.2014.6911283

DAMASCENO, A. L.; GALABO, R. J.; SOARES NETO, C. S. (2014). Cacuriá: Authoring Tool for Multimedia Learning Objects. Proceedings of the 20th Brazilian Symposium on Multimedia and the Web, 59–66, 2014. http://doi.org/10.1145/2664551.2664567

EIDENBERGER, H. (2003). SMIL and SVG in Teaching. In SPIE Electronic Imaging Symposium. SPIE, 2003, ISBN: 0819448214.

GAGGI, O.; BOSSI, A. (2011). Analysis and verification of SMIL documents. Multimedia Systems, 17, 487–506, 2011. http://doi.org/10.1007/s00530-011-0233-1

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. M. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, 1994.

GLEIRSCHER, M.; GOLUBITSKIY, D.; IRLBECK, M.; Wagner, S. (2013). Introduction of static quality analysis in small- and medium-sized software enterprises: experiences from technology transfer. Software Quality Journal, 1–44, 2013. http://doi.org/10.1007/s11219-013-9217-z.

HEER, J.; BOSTOCK, Michael (2010). Declarative Language Design for Interactive Visualization. IEEE Transactions on Visualization and Computer Graphics. November, 2010. DOI=10.1109/TVCG.2010.144 http://dx.doi.org/10.1109/TVCG.2010.144

HONORATO, G. D. S. C.; BARBOSA, S. D. J. (2010). NCL-inspector. Proceedings of the 2010 ACM Symposium on Applied Computing - SAC '10, 2010. http://doi.org/10.1145/1774088.1774500

HONORATO, G. S. C.; BARBOSA, S. D. J. (2010). NCL-Inspector. Uma ferramenta para inspeção de aplicações NCL. Rio de Janeiro, 2010. 120p. Dissertação de Mestrado. Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

HUANG, Zixia; NAHRSTEDT, Klara; STEINMETZ, Ralf (2013). Evolution of temporal multimedia synchronization principles: A historical viewpoint. ACM Trans. Multimedia Comput. Commun. Appl. 9, 1s, Article 34 (October 2013), 23 pages. DOI=10.1145/2490821 http://doi.acm.org/10.1145/2490821.

ITU (2012). ITU-T HSTP.CONF-H761 - Conformance testing specification for H.761, 2012. Disponível em: <http://www.itu.int/dms_pub/itu-t/opb/tut/T-TUT-IPTV-2012-H761-PDF-E.pdf>. Acesso em: 20 de Setembro de 2016.

ITU (2014). ITU-T Recommendation H.761 - Nested context language (NCL) and Ginga-NCL. 2014. Disponível em: <https://www.itu.int/rec/T-REC-H.761-201411-I. Acesso em: 20 de Setembro de 2016.

JEDLITSCHKA, A.; PFAHL, D. Reporting guidelines for controlled experiments in software engineering. Empirical Software Engineering. International Symposium, 2005.

JURISTO, N.; MORENO, A. M. (2010). Basics of Software Engineering Experimentation (1st ed.). Springer Publishing Company, Incorporated, 2010.

KING, P.; SCHMITZ, P.; Thompson, S. (2000). Behavioral Reactivity and Real Time Programming in XML. DocEng '04 Proceedings of the 2004 ACM Symposium on Document Engineering, 57–66, 2004. http://doi.org/10.1145/1030397.1030411.

KITCHENHAM, B. (2004). Procedures for Performing Systematic Reviews. Technical Report Software Engineering Group. Keele University, Australia, 2004.

KRKA, I.; BRUN, Y.; POPESCU, D.; GARCIA, J.; MEDVIDOVIC, N. (2010). Using dynamic execution traces and program invariants to enhance behavioral model inference.

Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - ICSE '10, 2, 179, 2010. http://doi.org/10.1145/1810295.1810324.

LAMADON, J. L.; CESAR, P.; HERRERO, C.; VUORIMAA, P. (2003). Usages of a SMIL Player in Digital Television. Proceedings of the IASTED International Conference on Internet and Multimedia Systems and Applications. IMSA2003. Hawaii, USA, 2003.

LI, Z.-N.; DREW, M. S.; LIU, J. (2014). Fundamentals of Multimedia (2nd ed.). Springer Publishing Company, Incorporated, 2014.

LLOYD, J.W. (1994). Practical advantages of declarative programming. Proc. Joint Conf. on Declarative Programming. GULD-PRODE' 94, Peniscola, Spain, 1994.

LORENZOLI, D.; MARIANI, L.; PEZZÈ, M. (2008). Automatic generation of software behavioral models. In Proceedings of the 30th international conference on Software engineering (ICSE '08). ACM, New York, NY, USA, 501-510, 2008. DOI=10.1145/1368088.1368157 http://doi.acm.org/10.1145/1368088.1368157.

MARIANI, L.; PASTORE, F.; PEZZÈ, M. (2011). Dynamic analysis for diagnosing integration faults. IEEE Transactions on Software Engineering, 37(4), 486–508, 2011. http://doi.org/10.1109/TSE.2010.93.

OLIVEIRA, M. C. F.de; TURINE, M. A. S.; Masiero, P. C.(2001). A statechart-based model for hypermedia applications. ACM Trans. Inf. Syst. 19, 1 (January 2001), 28-52, 2001. DOI=10.1145/366836.366869 http://doi.acm.org/10.1145/366836.366869

PETERSEN, K.; Feldt, R.; MUJTABA, S.; MATTSSON, M. (2008). Systematic mapping studies in software engineering. In Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering (EASE'08). Giuseppe Visaggio, Maria Teresa Baldassarre, Steve Linkman, and Mark Turner (Eds.). British Computer Society, Swinton, UK, UK, 68-77, 2008.

PICININ JR., D.; FARINES, J.-M.; KOLIVER, C. (2012). An approach to verify live NCL applications. In Proceedings of the 18th Brazilian symposium on Multimedia and the web

(WebMedia '12). ACM, New York, NY, USA, 223-232, 2012. DOI=10.1145/2382636.2382685 http://doi.acm.org/10.1145/2382636.2382685

PICININ, D. J.; FARINES, J.-M.; SANTOS, C. A. S. (2011). Uma abordagem MDE para modelagem e verificação de documentos multimídia interativos. WebMedia, 2011

PINHEIRO, C. F.; FILHO, E. B. L.; OLIVEIRA, R. R.; CAVALCANTE, A. A. M.; KLEH, V. S.; PEREIRA, D. P.; MELO, H. L. M. (2012). Uma Proposta de Suíte de Testes de Conformidade para o Ginga-NCL. In XXX Simpósio Brasileiro de Telecomunicações (SBRT'12). 2012.

PROTA, T.; VÉRAS, D.; FERRAZ, C. (2014). Análise estrutural e comportamental de aplicações Ginga-NCL orientada a conformidade. In XX Brazilian Symposium on Multimedia and the Web - Webmedia, 2014, João Pessoa, Paraíba, Brasil. IX Workshop on Ongoing Thesis and Dissertations (WTD).

ROBILLARD, M.P.; BODDEN, E.; KAWRYKOW, D.; MEZINI, M.; RATCHFORD, T. (2013). Automated API Property Inference Techniques. In Software Engineering, IEEE Transactions. vol.39, no.5, pp.613-637, May 2013. DOI=10.1109/TSE.2012.63

SANTOS, C. A. S.; FERRAZ, C. A. G.; LENZ, A. R.; Silva, J.S. (2013). Desenvolvimento de Aplicações Sensíveis ao Contexto da Programação diante das Não Conformidades dos Sinais das TV Brasileiras. Revista de Radiodifusão, v. 7, p. 50-59, 2013.

SANTOS, C. A. S.; LENZ, A. R. (2014). Non-conformance and problems related to the digital TV signals transmitted in Brazil and their impact on the Users and Application Developers. In Latin America Transactions, IEEE (Revista IEEE America Latina), vol.12, no.4, pp.772-782, June, 2014. DOI=10.1109/TLA.2014.6868882

SANTOS, J. A. F. dos (2012). Multimedia and hypermedia document validation and verification using a model-driven approach. Dissertação de Mestrado. Universidade Federal de Fluminense - UFF. 2012.

SANTOS, J. A. F. dos; BRAGA, C.; MUCHALUAT-SAADE, D. C. (2013). Automating the analysis of NCL documents with a model-driven approach. In Proceedings of the 19th

Brazilian symposium on Multimedia and the web (WebMedia '13). ACM, New York, NY, USA, 193-200. DOI=10.1145/2526188.2526214 http://doi.acm.org/10.1145/2526188.2526214

SANTOS, J. A. F. dos; BRAGA, C.; MUCHALUAT-SAADE, D.C. (2013). An executable semantics for a multimedia authoring language. In Formal Methods: Foundations and Applications, Springer, 2013, pp. 67-82.

SANTOS, R. C. M.; CERQUEIRA NETO, J. R.; SOARES NETO, C. S.; TEIXEIRA, M. M. (2012). Incremental validation of digital TV applications in nested context language. Proceedings of the 10th European Conference on Interactive Tv and Video - EuroiTV '12, 203, 2012. http://doi.org/10.1145/2325616.2325657

SANTOS, R. C. M.; NETO, J. R. C.; SOARES NETO, C. S.; TEIXEIRA, M. M. (2013). Incremental validation of NCL hypermedia documents. Journal of the Brazilian Computer Society, 19(3), 235–256, 2013. http://doi.org/10.1007/s13173-013-0110-1.

SCHUR, M.; ROTH, A.; ZELLER, A. (2013). Mining behavior models from enterprise web applications. In Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2013). ACM, New York, NY, USA, 422-432, 2013. DOI=10.1145/2491411.2491426 http://doi.acm.org/10.1145/2491411.2491426

SOKOLOVA, Marina; LAPALME, G. (2009). A systematic analysis of performance measures for classification tasks. Inf. Process. Manage. 45, 4 (July 2009), 427-437. DOI=10.1016/j.ipm.2009.03.002 http://dx.doi.org/10.1016/j.ipm.2009.03.002

VÉRAS, Douglas; PROTA, T.; BISPO, A.; PRUDÊNCIO, R.; FERRAZ, C. (2015). A literature review of recommender systems in the television domain, Expert Systems with Applications. Available online 4 July 2015, ISSN 0957-4174, 2015. http://dx.doi.org/10.1016/j.eswa.2015.06.052.

W3C (2002). HTML4 Test Suite Documentation. 2002. Disponível em: <http://www.w3.org/MarkUp/Test/HTML401/current/htmltestdocumentation.html. Acesso em: 20 de Setembro de 2016.

W3C (2008). Synchronized Multimedia Integration Language (SMIL 3.0) - W3C Recommendation. 2008. Disponível em: <http://www.w3.org/TR/2008/RECSMIL3-20081201/. Acesso em: 20 de Setembro de 2016.

W3C (2010). XHTML™ 2.0 - W3C Recommendation. 2010. Disponível em: <http://www.w3.org/TR/2010/NOTE-xhtml2-20101216. Acesso em: 20 de Setembro de 2016.

W3C (2011). Scalable Vector Graphics (SVG 1.1) - Second Edition - W3C Recommendation. 2011. Disponível em: <http://www.w3.org/TR/2011/REC-SVG11-20110816. Acesso em: 20 de Setembro de 2016.

W3C (2014). HTML5 - W3C Recommendation. 2014. Disponível em: <http://www.w3.org/TR/2014/REC-html5-20141028/. Acesso em: 20 de Setembro de 2016.

WAGNER, F.; SCHMUKI, R.; WAGNER, T.; WOLSTENHOLME, P. (2006). Modeling Software with Finite State Machines: A Practical Approach. CRC Press, NY (2006)

WALKINSHAW, N.; BOGDANOV, K. (2013). Automated Comparison of State-Based Software Models in Terms of Their Language and Structure. ACM Transactions on Software Engineering and Methodology, 22(2), 1–37, 2013. http://doi.org/10.1145/2430545.2430549.

WALKINSHAW, N.; TAYLOR, R.; DERRICK, J. (2013). Inferring Extended Finite State Machine models from software executions. In Reverse Engineering (WCRE), 2013 20th Working Conference on , vol., no., pp.301-310, 14-17 Oct. 2013. DOI=10.1109/WCRE.2013.6671305"

WOHLIN, C; RUNESON, P.; HÖST, M.; OHLSSON, M. C.; REGNELL, B. (2012). Experimentation in Software Engineering. 2012. Editora Springer. 2012.

ZELLER, A (2009). Why Programs Fail: A Guide to Systematic Debugging. Morgan Kaufmann. 2nd Edition. ISBN-9780123745156. 2009.

Apêndice A

Este apêndice tem por objetivo apresentar os artefatos produzidos pelos experimentos, gerados durante as rodadas realizadas, bem como os scripts utilizados para analisar os dados.

Primeira Rodada

Na primeira rodada do experimento, foram geradas Matrizes de Confusões originadas pelo tratamento dos conflitos. A Figura A.1 apresenta as Matrizes de Confusão resultantes do tratamento de conflitos entre o Modelo Comportamental e o Astrobox. Enquanto que, a Figura A.2 apresenta as referentes ao Modelo Comportamental e o EiTV smartBox.

Na etapa da análise foram calculados os indicadores relevantes para o estudo: *Precision*; *Recall / Sensitivity* ; *F-Measure*; *Specificity*; *Balanced Classification Rate (BCR)*. O

Quadro A.1 apresenta os indicadores para cada aplicação referentes ao Astrobox, enquanto o Quadro A.2 apresenta os referentes ao EiTV smartBox.

Aplicação 01		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	26	1
	Rejeita	3	27

Aplicação 01		Máquina Real Inferida	
		Aceita	Rejeita
Astrobox	Aceita	28	0
	Rejeita	1	28

Aplicação 02		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	16	0
	Rejeita	0	91

Aplicação 02		Máquina Real Inferida	
		Aceita	Rejeita
Astrobox	Aceita	8	0
	Rejeita	8	91

Aplicação 03		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	78	0
	Rejeita	0	108

Aplicação 03		Máquina Real Inferida	
		Aceita	Rejeita
Astrobox	Aceita	0	0
	Rejeita	78	108

Aplicação 04		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	58	0
	Rejeita	0	134

Aplicação 04		Máquina Real Inferida	
		Aceita	Rejeita
Astrobox	Aceita	0	0
	Rejeita	58	134

Aplicação 05		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	82	0
	Rejeita	0	135

Aplicação 05		Máquina Real Inferida	
		Aceita	Rejeita
Astrobox	Aceita	0	0
	Rejeita	82	135

Aplicação 06		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	56	0
	Rejeita	0	77

Aplicação 06		Máquina Real Inferida	
		Aceita	Rejeita
Astrobox	Aceita	0	0
	Rejeita	56	77

Aplicação 07		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	9	0
	Rejeita	0	27

Aplicação 07		Máquina Real Inferida	
		Aceita	Rejeita
Astrobox	Aceita	0	0
	Rejeita	9	27

Aplicação 08		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	55	0
	Rejeita	0	77

Aplicação 08		Máquina Real Inferida	
		Aceita	Rejeita
Astrobox	Aceita	0	0
	Rejeita	55	77

Aplicação 09		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	29	16
	Rejeita	0	120

Aplicação 09		Máquina Real Inferida	
		Aceita	Rejeita
Astrobox	Aceita	2	0
	Rejeita	27	136

Aplicação 10		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	390	0
	Rejeita	0	148

Aplicação 10		Máquina Real Inferida	
		Aceita	Rejeita
Astrobox	Aceita	18	0
	Rejeita	372	148

Aplicação 11		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	1	274
	Rejeita	0	180

Aplicação 11		Máquina Real Inferida	
		Aceita	Rejeita
Astrobox	Aceita	0	0
	Rejeita	1	454

Aplicação 12		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	65	0
	Rejeita	0	46

Aplicação 12		Máquina Real Inferida	
		Aceita	Rejeita
Astrobox	Aceita	9	0
	Rejeita	56	46

Figura A.1. Matrizes de Confusão da primeira rodada referentes ao Astrobox

Aplicação 01		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	26	1
	Rejeita	3	27

Aplicação 01		Máquina Real Inferida	
		Aceita	Rejeita
EiTV smartBox	Aceita	29	0
	Rejeita	0	28

Aplicação 02		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	16	0
	Rejeita	0	91

Aplicação 02		Máquina Real Inferida	
		Aceita	Rejeita
EiTV smartBox	Aceita	2	0
	Rejeita	14	91

Aplicação 03		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	78	0
	Rejeita	0	108

Aplicação 03		Máquina Real Inferida	
		Aceita	Rejeita
EiTV smartBox	Aceita	69	58
	Rejeita	9	50

Aplicação 04		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	58	0
	Rejeita	0	134

Aplicação 04		Máquina Real Inferida	
		Aceita	Rejeita
EiTV smartBox	Aceita	57	53
	Rejeita	1	81

Aplicação 05		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	82	0
	Rejeita	0	135

Aplicação 05		Máquina Real Inferida	
		Aceita	Rejeita
EiTV smartBox	Aceita	72	57
	Rejeita	10	78

Aplicação 06		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	56	0
	Rejeita	0	77

Aplicação 06		Máquina Real Inferida	
		Aceita	Rejeita
EiTV smartBox	Aceita	53	31
	Rejeita	3	46

Aplicação 07		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	9	0
	Rejeita	0	27

Aplicação 07		Máquina Real Inferida	
		Aceita	Rejeita
EiTV smartBox	Aceita	8	0
	Rejeita	1	27

Aplicação 08		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	55	0
	Rejeita	0	77

Aplicação 08		Máquina Real Inferida	
		Aceita	Rejeita
EiTV smartBox	Aceita	41	29
	Rejeita	14	48

Aplicação 09		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	29	16
	Rejeita	0	120

Aplicação 09		Máquina Real Inferida	
		Aceita	Rejeita
EiTV smartBox	Aceita	5	7
	Rejeita	24	122

Aplicação 10		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	390	0
	Rejeita	0	148

Aplicação 10		Máquina Real Inferida	
		Aceita	Rejeita
EiTV smartBox	Aceita	15	0
	Rejeita	375	148

Aplicação 11		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	1	274
	Rejeita	0	180

Aplicação 11		Máquina Real Inferida	
		Aceita	Rejeita
EiTV smartBox	Aceita	1	0
	Rejeita	0	454

Aplicação 12		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	65	0
	Rejeita	0	46

Aplicação 12		Máquina Real Inferida	
		Aceita	Rejeita
EiTV smartBox	Aceita	13	0
	Rejeita	52	46

Figura A.2. Matrizes de Confusão da primeira rodada referentes ao EiTV smartBox

Quadro A.1. Indicadores de confiabilidade da primeira rodada referentes ao Astrobox

Aplicação	Tratamento	*Precision*	*Recall*	*FMeasure*	*Specificity*	*BCR*
Aplicação 01	Modelo Comportamental	0,9630	0,8966	0,9286	0,9000	0,8983
Aplicação 01	Astrobox	1,0000	0,9655	0,9825	0,9655	0,9655
Aplicação 02	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 02	Astrobox	1,0000	0,5000	0,6667	0,9192	0,7096
Aplicação 03	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 03	Astrobox	0,0000	0,0000	0,0000	0,5806	0,2903
Aplicação 04	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 04	Astrobox	0,0000	0,0000	0,0000	0,6979	0,3490
Aplicação 05	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 05	Astrobox	0,0000	0,0000	0,0000	0,6221	0,3111
Aplicação 06	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 06	Astrobox	0,0000	0,0000	0,0000	0,5789	0,2895
Aplicação 07	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 07	Astrobox	0,0000	0,0000	0,0000	0,7500	0,3750
Aplicação 08	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 08	Astrobox	0,0000	0,0000	0,0000	0,5833	0,2917
Aplicação 09	Modelo Comportamental	0,6444	1,0000	0,7838	1,0000	1,0000
Aplicação 09	Astrobox	1,0000	0,0690	0,1290	0,8344	0,4517
Aplicação 10	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 10	Astrobox	1,0000	0,0462	0,0882	0,2846	0,1654
Aplicação 11	Modelo Comportamental	0,0036	1,0000	0,0072	1,0000	1,0000
Aplicação 11	Astrobox	0,0000	0,0000	0,0000	0,9978	0,4989
Aplicação 12	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 12	Astrobox	1,0000	0,1385	0,2432	0,4510	0,2947

Quadro A.2. Indicadores de confiabilidade da primeira rodada referentes ao EiTV smartBox

Aplicação	Tratamento	*Precision*	*Recall*	*FMeasure*	*Specificity*	*BCR*
Aplicação 01	Modelo Comportamental	0,9630	0,8966	0,9286	0,9000	0,8983
Aplicação 01	EiTV smartBox	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 02	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 02	EiTV smartBox	1,0000	0,1250	0,2222	0,8667	0,4958
Aplicação 03	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 03	EiTV smartBox	0,5433	0,8846	0,6732	0,8475	0,8660
Aplicação 04	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000

Aplicação 04	EiTV smartBox	0,5182	0,9828	0,6786	0,9878	0,9853
Aplicação 05	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 05	EiTV smartBox	0,5581	0,8780	0,6825	0,8864	0,8822
Aplicação 06	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 06	EiTV smartBox	0,6310	0,9464	0,7571	0,9388	0,9426
Aplicação 07	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 07	EiTV smartBox	1,0000	0,8889	0,9412	0,9643	0,9266
Aplicação 08	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 08	EiTV smartBox	0,5857	0,7455	0,6560	0,7742	0,7598
Aplicação 09	Modelo Comportamental	0,6444	1,0000	0,7838	1,0000	1,0000
Aplicação 09	EiTV smartBox	0,4167	0,1724	0,2439	0,8356	0,5040
Aplicação 10	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 10	EiTV smartBox	1,0000	0,0385	0,0741	0,2830	0,1607
Aplicação 11	Modelo Comportamental	0,0036	1,0000	0,0072	1,0000	1,0000
Aplicação 11	EiTV smartBox	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 12	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 12	EiTV smartBox	1,0000	0,2000	0,3333	0,4694	0,3347

Segunda Rodada

Na segunda rodada do experimento, foram geradas Matrizes de Confusões originadas pelo tratamento dos conflitos. A Figura A.3 apresenta as Matrizes de Confusão resultantes do tratamento de conflitos entre o Modelo Comportamental e o Astrobox. Enquanto que, a Figura A.4 apresenta as referentes ao Modelo Comportamental e o EiTV smartBox.

Na etapa da análise foram calculados os indicadores relevantes para o estudo: *Precision*; *Recall / / Sensitivity* ; *F-Measure*; *Specificity*; *Balanced Classification Rate (BCR)*. O

Quadro A.3 apresenta os indicadores para cada aplicação referentes ao Astrobox, enquanto o

Quadro A.4 apresenta os referentes ao EiTV smartBox.

Aplicação 01		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	96	0
	Rejeita	0	92

Aplicação 01		Máquina Real Inferida	
		Aceita	Rejeita
Astrobox	Aceita	90	0
	Rejeita	6	92

Aplicação 02		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	60	0
	Rejeita	0	10

Aplicação 02		Máquina Real Inferida	
		Aceita	Rejeita
Astrobox	Aceita	45	0
	Rejeita	15	10

Aplicação 03		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	123	23
	Rejeita	0	307

Aplicação 03		Máquina Real Inferida	
		Aceita	Rejeita
Astrobox	Aceita	123	0
	Rejeita	0	330

Aplicação 04		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	11	0
	Rejeita	0	9

Aplicação 04		Máquina Real Inferida	
		Aceita	Rejeita
Astrobox	Aceita	4	0
	Rejeita	7	9

Aplicação 05		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	106	0
	Rejeita	0	110

Aplicação 05		Máquina Real Inferida	
		Aceita	Rejeita
Astrobox	Aceita	1	0
	Rejeita	105	110

Aplicação 06		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	110	1
	Rejeita	0	434

Aplicação 06		Máquina Real Inferida	
		Aceita	Rejeita
Astrobox	Aceita	10	0
	Rejeita	100	435

Aplicação 07		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	11	0
	Rejeita	0	4

Aplicação 07		Máquina Real Inferida	
		Aceita	Rejeita
Astrobox	Aceita	4	0
	Rejeita	7	4

Aplicação 08		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	11	0
	Rejeita	0	9

Aplicação 08		Máquina Real Inferida	
		Aceita	Rejeita
Astrobox	Aceita	4	0
	Rejeita	7	9

Aplicação 09		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	11	0
	Rejeita	0	9

Aplicação 09		Máquina Real Inferida	
		Aceita	Rejeita
Astrobox	Aceita	4	0
	Rejeita	7	9

Aplicação 10		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	22	0
	Rejeita	0	88

Aplicação 10		Máquina Real Inferida	
		Aceita	Rejeita
Astrobox	Aceita	5	2
	Rejeita	17	86

Aplicação 11		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	40	18
	Rejeita	0	0

Aplicação 11		Máquina Real Inferida	
		Aceita	Rejeita
Astrobox	Aceita	2	0
	Rejeita	38	18

Aplicação 12		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	33	0
	Rejeita	0	10

Aplicação 12		Máquina Real Inferida	
		Aceita	Rejeita
Astrobox	Aceita	1	0
	Rejeita	32	10

Figura A.3. Matrizes de Confusão da segunda rodada referentes ao Astrobox

Aplicação 01		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	96	0
	Rejeita	0	92

Aplicação 01		Máquina Real Inferida	
		Aceita	Rejeita
EiTV smartBox	Aceita	90	0
	Rejeita	6	92

Aplicação 02		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	60	0
	Rejeita	0	10

Aplicação 02		Máquina Real Inferida	
		Aceita	Rejeita
EiTV smartBox	Aceita	41	0
	Rejeita	19	10

Aplicação 03		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	123	23
	Rejeita	0	307

Aplicação 03		Máquina Real Inferida	
		Aceita	Rejeita
EiTV smartBox	Aceita	123	0
	Rejeita	0	330

Aplicação 04		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	11	0
	Rejeita	0	9

Aplicação 04		Máquina Real Inferida	
		Aceita	Rejeita
EiTV smartBox	Aceita	1	0
	Rejeita	10	9

Aplicação 05		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	106	0
	Rejeita	0	110

Aplicação 05		Máquina Real Inferida	
		Aceita	Rejeita
EiTV smartBox	Aceita	86	0
	Rejeita	20	110

Aplicação 06		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	110	1
	Rejeita	0	434

Aplicação 06		Máquina Real Inferida	
		Aceita	Rejeita
EiTV smartBox	Aceita	0	0
	Rejeita	110	435

Aplicação 07		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	11	0
	Rejeita	0	4

Aplicação 07		Máquina Real Inferida	
		Aceita	Rejeita
EiTV smartBox	Aceita	1	0
	Rejeita	10	4

Aplicação 08		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	11	0
	Rejeita	0	9

Aplicação 08		Máquina Real Inferida	
		Aceita	Rejeita
EiTV smartBox	Aceita	1	0
	Rejeita	10	9

Aplicação 09		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	11	0
	Rejeita	0	9

Aplicação 09		Máquina Real Inferida	
		Aceita	Rejeita
EiTV smartBox	Aceita	1	0
	Rejeita	10	9

Aplicação 10		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	22	0
	Rejeita	0	88

Aplicação 10		Máquina Real Inferida	
		Aceita	Rejeita
EiTV smartBox	Aceita	4	3
	Rejeita	18	85

Aplicação 11		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	40	18
	Rejeita	0	0

Aplicação 11		Máquina Real Inferida	
		Aceita	Rejeita
EiTV smartBox	Aceita	40	0
	Rejeita	0	18

Aplicação 12		Máquina Real Inferida	
		Aceita	Rejeita
Modelo Comportamental	Aceita	33	0
	Rejeita	0	10

Aplicação 12		Máquina Real Inferida	
		Aceita	Rejeita
EiTV smartBox	Aceita	28	0
	Rejeita	5	10

Figura A.4. Matrizes de Confusão da segunda rodada referentes ao EiTV smartBox

Quadro A.3. Indicadores de confiabilidade da segunda rodada referentes ao Astrobox

Aplicação	Alvo	*Precision*	*Recall*	*FMeasure*	*Specificity*	*BCR*
Aplicação 01	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 01	Astrobox	1,0000	0,9375	0,9677	0,9388	0,9381
Aplicação 02	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 02	Astrobox	1,0000	0,7500	0,8571	0,4000	0,5750
Aplicação 03	Modelo Comportamental	0,8425	1,0000	0,9145	1,0000	1,0000
Aplicação 03	Astrobox	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 04	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 04	Astrobox	1,0000	0,3636	0,5333	0,5625	0,4631
Aplicação 05	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 05	Astrobox	1,0000	0,0094	0,0187	0,5116	0,2605
Aplicação 06	Modelo Comportamental	0,9910	1,0000	0,9955	1,0000	1,0000
Aplicação 06	Astrobox	1,0000	0,0909	0,1667	0,8131	0,4520
Aplicação 07	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 07	Astrobox	1,0000	0,3636	0,5333	0,3636	0,3636
Aplicação 08	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 08	Astrobox	1,0000	0,3636	0,5333	0,5625	0,4631
Aplicação 09	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 09	Astrobox	1,0000	0,3636	0,5333	0,5625	0,4631
Aplicação 10	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 10	Astrobox	0,7143	0,2273	0,3448	0,8350	0,5311
Aplicação 11	Modelo Comportamental	0,6897	1,0000	0,8163	0,0000	0,5000
Aplicação 11	Astrobox	1,0000	0,0500	0,0952	0,3214	0,1857
Aplicação 12	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 12	Astrobox	1,0000	0,0303	0,0588	0,2381	0,1342

Quadro A.4. Indicadores de confiabilidade da segunda rodada referentes ao EiTV smartBox

Aplicação	Alvo	*Precision*	*Recall*	*FMeasure*	*Specificity*	*BCR*
Aplicação 01	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 01	EiTV smartBox	1,0000	0,9375	0,9677	0,9388	0,9381
Aplicação 02	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 02	EiTV smartBox	1,0000	0,6833	0,8119	0,3448	0,5141

Aplicação 03	Modelo Comportamental	0,8425	1,0000	0,9145	1,0000	1,0000
Aplicação 03	EiTV smartBox	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 04	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 04	EiTV smartBox	1,0000	0,0909	0,1667	0,4737	0,2823
Aplicação 05	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 05	EiTV smartBox	1,0000	0,8113	0,8958	0,8462	0,8287
Aplicação 06	Modelo Comportamental	0,9910	1,0000	0,9955	1,0000	1,0000
Aplicação 06	EiTV smartBox	0,0000	0,0000	0,0000	0,7982	0,3991
Aplicação 07	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 07	EiTV smartBox	1,0000	0,0909	0,1667	0,2857	0,1883
Aplicação 08	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 08	EiTV smartBox	1,0000	0,0909	0,1667	0,4737	0,2823
Aplicação 09	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 09	EiTV smartBox	1,0000	0,0909	0,1667	0,4737	0,2823
Aplicação 10	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 10	EiTV smartBox	0,5714	0,1818	0,2759	0,8252	0,5035
Aplicação 11	Modelo Comportamental	0,6897	1,0000	0,8163	0,0000	0,5000
Aplicação 11	EiTV smartBox	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 12	Modelo Comportamental	1,0000	1,0000	1,0000	1,0000	1,0000
Aplicação 12	EiTV smartBox	1,0000	0,8485	0,9180	0,6667	0,7576

Análise

A análise formal foi realizada através do software R, utilizando os scripts apresentados na Figura A.5 e na Figura A.6. As figuras: Figura A.7, Figura A.8, Figura A.9 e Figura A.10 apresentam os resultados da execução da análise da primeira rodada, enquanto que as figuras: Figura A.11, Figura A.12, Figura A.13 e Figura A.14 apresentam os da segunda.

```
exp1.dat = read.table(file="fase01_projeto.txt", header = T)

amostraMC = subset(exp1.dat, Tratamento=="1");
amostraA = subset(exp1.dat, Tratamento=="2");
amostraES = subset(exp1.dat, Tratamento=="3");

#FMeasure A x MC
t.test(amostraA$FMeasure,
        amostraMC$FMeasure,
        alternative="less",
        paired=TRUE,
        conf.level=0.95)

#FMeasure ES x MC
t.test(amostraES$FMeasure,
        amostraMC$FMeasure,
        alternative="less",
        paired=TRUE,
        conf.level=0.95)
```

Figura A.5. Script R utilizado na análise formal dos dados para a medida *F-Measure*

```
exp1.dat = read.table(file="fase01_projeto.txt", header = T)

amostraMC = subset(exp1.dat, Tratamento=="1");
amostraA = subset(exp1.dat, Tratamento=="2");
amostraES = subset(exp1.dat, Tratamento=="3");

#BCR A x MC
t.test(amostraA$BCR,
        amostraMC$BCR,
        alternative="less",
        paired=TRUE,
        conf.level=0.95)

#BCR ES x MC
t.test(amostraES$BCR,
        amostraMC$BCR,
        alternative="less",
        paired=TRUE,
        conf.level=0.95)
```

Figura A.6. Script R utilizado na análise formal dos dados para a medida *BCR*

```
data:  amostraA$FMeasure and amostraMC$FMeasure
t = -6.1975, df = 11, p-value = 3.37e-05
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
       -Inf -0.5095843
sample estimates:
mean of the differences
                -0.7175
```

Figura A.7. Resultado da análise formal da primeira rodada referente aos dados do Astrobox para a medida *F-Measure*

```
data:  amostraES$FMeasure and amostraMC$FMeasure
t = -2.0204, df = 11, p-value = 0.03418
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
        -Inf -0.03201703
sample estimates:
mean of the differences
              -0.288125
```

Figura A.8. Resultado da análise formal da primeira rodada referente aos dados do EiTV smartBox para a medida *F-Measure*

```
data:  amostraA$BCR and amostraMC$BCR
t = -8.1419, df = 11, p-value = 2.762e-06
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
       -Inf -0.4485533
sample estimates:
mean of the differences
             -0.5754917
```

Figura A.9. Resultado da análise formal da primeira rodada referente aos dados do Astrobox para a medida *BCR*

```
data:  amostraES$BCR and amostraMC$BCR
t = -2.9323, df = 11, p-value = 0.006817
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
        -Inf -0.09819997
sample estimates:
mean of the differences
             -0.2533833
```

Figura A.10. Resultado da análise formal da primeira rodada referente aos dados do EiTV smartBox para a medida *BCR*

```
data:  amostraA$FMeasure and amostraMC$FMeasure
t = -5.1083, df = 11, p-value = 0.0001698
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
       -Inf -0.3287644
sample estimates:
mean of the differences
             -0.5070083
```

Figura A.11. Resultado da análise formal da segunda rodada referente aos dados do Astrobox para a medida *F-Measure*

```
data:  amostraES$FMeasure and amostraMC$FMeasure
t = -3.3944, df = 11, p-value = 0.002994
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
       -Inf -0.2036862
sample estimates:
mean of the differences
             -0.4325167
```

Figura A.12. Resultado da análise formal da segunda rodada referente aos dados do EiTV smartBox para a medida *F-Measure*

```
data:  amostraA$BCR and amostraMC$BCR
t = -6.5353, df = 11, p-value = 2.11e-05
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
       -Inf -0.3426877
sample estimates:
mean of the differences
             -0.4725417
```

Figura A.13. Resultado da análise formal da segunda rodada referente aos dados do Astrobox para a medida *BCR*

```
data:  amostraES$BCR and amostraMC$BCR
t = -3.3371, df = 11, p-value = 0.003314
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
       -Inf -0.1741043
sample estimates:
mean of the differences
              -0.376975
```

Figura A.14. Resultado da análise formal da segunda rodada referente aos dados do EiTV smartBox para a medida *BCR*