Desenvolvimento de um leitor e representador gráfico de diagramas de estado

Victor Wilvert Antunes
Centro Tecnológico de Joinville
Universidade Federal de Santa Catariana
Joinville, Brasil
Email:victorwilvertantunes@gmail.com

I. Introdução

O presente relatório tem como objetivo apresentar o desenvolvimento de uma aplicação capaz de interpretar diagramas de estado contidos em arquivos do tipo Scxml, linguagem de marcação derivada do XML, e realizar a construção de uma representação gráfica deste diagrama após ter efetivado a avaliação dos elementos a serem exibidos nele. Além disso, também possibilita a animação das transições entre os estados a partir de lista dos eventos obtida de um arquivo de texto. A aplicação foi desenvolvida através da linguagem C++ e do o uso da biblioteca Qt5, tendo ênfase na aplicação os diversos conceitos da programação orientada a objetos em sua criação, principalmente do polimorfismo, em razão da necessidade de abstração em tempo de execução.

Ao longo do desenvolvimento foram encontradas diversos obstáculos, sendo os principais a abstração e construção dos itens suportados pelo programa e a geração de uma comunicação eficiente entre a lista de eventos e o diagrama sem gerar uma grande interdependência entre os dois.

II. Desenvolvimento

A. SCXML

Diagramas de estados são usados para modelagem de diversos tipos de problemas, dentre os quais podem ser problemas advindos de automação de design eletrônico, projeto de protocolo de comunicação, análise e outras aplicações. Uma das formas de descrever um diagrama de estados se encontra através da linguagem Scxml, acrônimo para **S**tate **C**hart **XML**, a qual é usada para abstração de notação de máquinas de estados. Scxml é uma linguagem de marcação desenvolvida pela W3C, acrônimo de World **W**ide **W**eb **C**onsortium, a qual tem objetivo de de gerar um padrão das notações para diagramas de estado que são usados em outros contextos de XML, linguagem da qual Scxml deriva.

O programa tem a capacidade de interpretar especificamente essa linguagem de marcação e representar os diagramas de estados descritos por ela. Devido a existência diversas notações presentes no padrão Scxml, foram apenas desenvolvidos como itens gráficos apenas algumas dessas notações, as quais são o **<scxml>**, **<state>**, **<transition>**, **<initial>**, **<final>** e **<history>**. os outros elementos da linguagem são representados na forma de texto inseridos em conjunto com a descrição do item que o possui.

B. Implementação

A implementação do programa está divida em algumas partes, sendo separadas em abstração das notações do Scxml, feita com do uso de polimorfismo e de uma derivação do padrão Composite, leitura do arquivo e construção dos itens do diagrama, feito com o uso de bibliotecas de XML presentes no Qt5, representação gráfica do diagrama e a comunicação das informações de transição e animação das passagens de estado, feito usando o padrão Chain Of Responsability.

1) Abstração das Notações do Scxml: Os elementos presentes na linguagem de marcação suportada pela aplicação, são apenas desenvolvidos as exibições gráficas das notações os mais representativas. Para realizar a construção dos itens gráficos, todos os elementos foram divididos através de uma abstração gerada a partir de uma derivação do padrão Composite, o qual é o padrão de projeto utilizado para representar um objeto formado pela composição de objetos similares fazendo uso do polimorfismo, implementado de uma forma priorizando a uniformidade, onde

a composto e a folha possuem todos os métodos do componente. Uma modificação em relação ao padrão está presente no fato de todos os componentes possuem acesso ao componente parente e a folha, representada pela Seta, dispõe de dois parentes, simbolizando os extremos das transições. Neste ponto encontra-se a derivação em relação ao padrão Composite, já que uma folha não pode deter acesso a filhos e o segundo parente se assemelha a um filho.

No padrão usado a classe que representa o componente é a Item, que apresenta todos os métodos de acesso a informações a serem usadas externamente, como acesso aos filhos, aos parentes e a informações dos atributos e tipo do objeto. O composto é representado pela classe Bloco, que representa o **<state>**, que é derivada para as classes Bola, representando **<initial>** e **<final>**, e Base, que representa o **<scxml>**, a classe Bola também é derivada para a classe Historia, que representa o **<history>**. A folha é representada pela classe Seta, que representa o **<transition>**. Todas essas classe tentam reproduzir da forma mais completa as notações do Scxml a qual correspondem, dando acesso a todas as informações necessárias para que possa ser realizada uma validação, caso apresente-se necessário. Na Figura 1 é demonstrado como cada notação é representada graficamente.

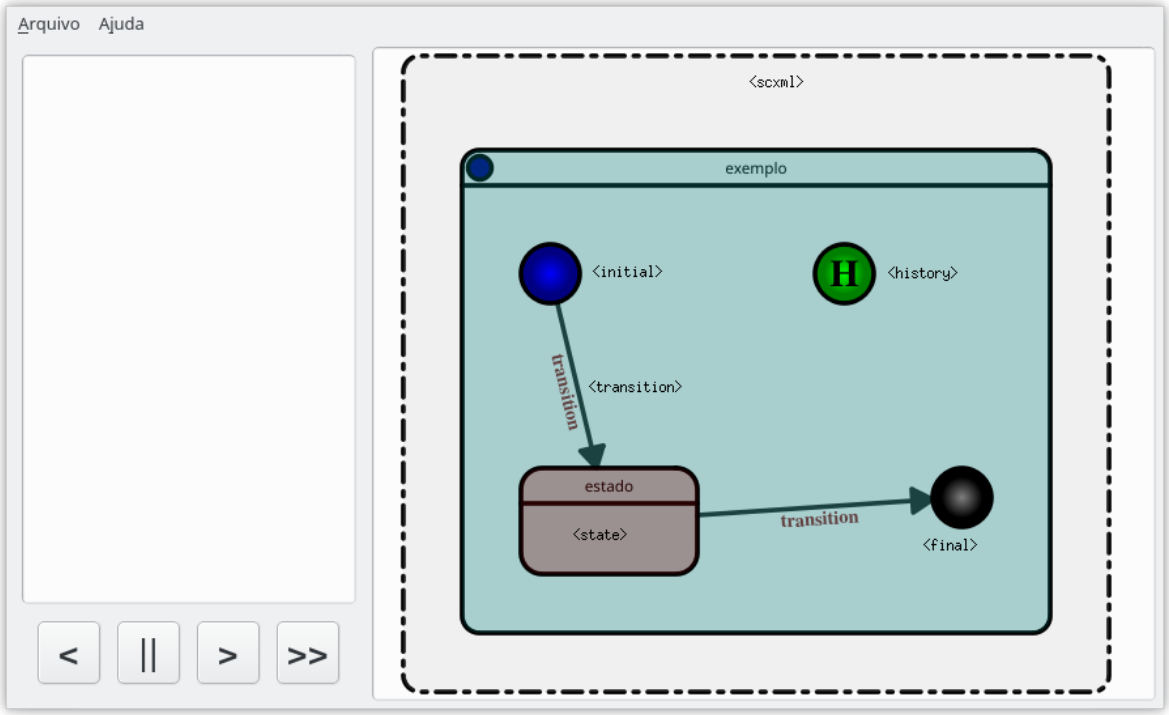

Figura 1: Exemplo dos elementos representados

2) Leitura do arquivo e construção do diagrama: A leitura do arquivo é feita através de uma classe do Qt5, a QDomDocument, a qual lê arquivos do XML ou derivados. Essa biblioteca analisa o arquivo e o organiza separando seus elementos em uma árvore, tornando mais acessível o acesso desses componentes de forma direta segundo a organização da linguagem de marcação em relação a se a organização ocorresse de forma linear.

A construção do diagrama de estados na forma de itens é realizada pela classe Construtor, a qual através de um objeto QDomDocument gera cada um dos itens gráficos respectivos as notações presentes nos nós da árvore do objeto e também são realizadas as ligações entre os diversos elementos. Posteriormente a criação de todos os itens realiza-se a validação de cada um deles, removendo-o caso esteja fora dos padrões propostos pelo Scxml.

3) Representação gráfica: Toda a representação gráfica do programa ocorre através da classe Janela, nela residem as duas partes responsáveis pela demonstração de informações, sendo a classe VistaLista responsável pela exibição da lista de eventos, que será usada para a animação, e a classe VistaDiagrama a responsável pela apresentação do diagrama de estados. Cada uma dessas classes herda uma classe gráfica e adiciona ou remove funcionalidades,

na VistaLista que herda a classe QListWidgets, ela remove o controle da lista através do mouse e o passa para métodos, e no caso da VistaDiagrama que herda a classe QGraphicsView, só está presente a sobrecarga do evento de mouse para possibilitar ampliação e redução do tamanho da imagem.

Na Figura 2 há uma demonstração da interface completa com o diagrama e a lista possuindo itens e do painel de acesso das informações de um estado, o elemento atual é representado em vermelho.

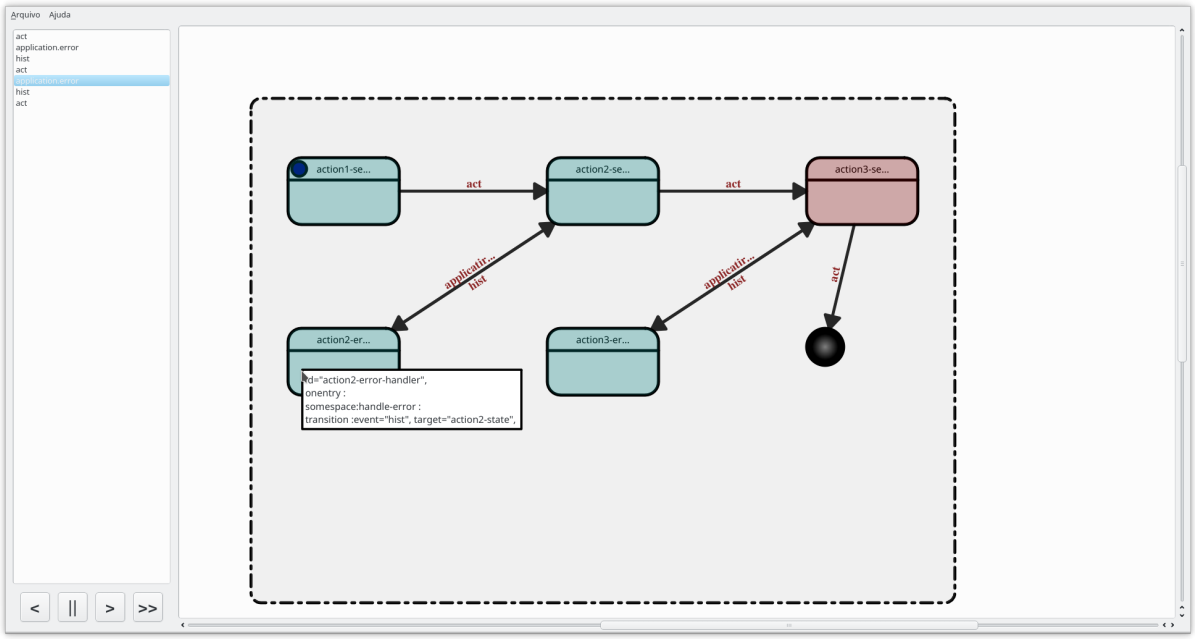

Figura 2: Exemplo da interface completa

4) Animação: A animação do diagrama de estados é realizada por meio de uma lista de eventos, os quais são passados ao programa através de um arquivo do tipo *.txt, em que em cada linha deve estar disposto o nome de um evento. A animação ocorre por intermédio da passagem das informações da lista para o diagrama, que foi desenvolvida com o uso do padrão Chain Of Responsability, que tem como objetivo evitar a dependência entre um objeto receptor e um objeto solicitante, onde utiliza-se a classe Interface herdada pelas classes Tabela e Diagrama. Na caso presente a cadeia de responsabilidade fica organizada da forma em que a lista antes de avançar um elemento, necessite efetuar um pedido ao diagrama que realize a transição proposta pelo evento do elemento atual da lista, caso a transição seja válida, a lista ganha permissão para avançar ao próximo elemento

Na Figura 3 mostra-se um diagrama que teve um evento executado.

(a) Primeiro estado

(b) Segundo estado

Figura 3: Exemplo de transição de estado

III. Resultados e Discussão

Para realização do programa foi necessário a implementação de diversas classes muitas vezes abstratas, em razão de maior parte das decisões que o programa deve tomar acontecerem em tempo de execução, logo torna-se vantajoso o uso de polimorfismo no programa.

Entre os obstáculos encontrados no desenvolvimento, os que apresentaram maior complexidade para serem resolvidos foram a forma de validação dos itens após a construção da árvore do diagrama, dado que além das diversas condições, também existe a necessidade de remover o elemento e todas suas referências da árvore do diagrama. Outro impedimento encontrado foi a comunicação do diagrama com a lista de eventos, que foi resolvido usando a logica proposta pelo padrão de projeto Chain Of Responsability.

Porém, após ultrapassados esse impedimentos foi possível produzir uma ferramenta de representação e animação para diagramas de estado significativamente completa, levando em conta que as notações não implementadas poderiam ser adicionadas futuramente sem grandes mudanças ao código devido a extensibilidade gerada pela presença do polimorfismo na definição dos itens.

IV. Conclusão

Foi possível desenvolver uma aplicação satisfatória de representação de diagramas de estado a partir da interpretação de arquivos Scxml, que inclusive possui um sistema de animação simples de transição de estados. Além disso, mesmo sem apresentar todos as notações do padrão elas podem ser facilmente inseridos no programa futuramente devido a forma que foram implementado os itens gráficos. Desta Forma foi possível agregar grande conhecimento relativo a programação orientada a objetos através do desenvolvimento do programa e também o conhecimento da implementação de uma interface gráfica usando a biblioteca Qt.