

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Dissertação de Mestrado

**Geração de Casos de Teste Funcional para Aplicações de
Celulares**

Emanuela Gadelha Cartaxo

Campina Grande, Paraíba, Brasil

Outubro - 2006

Geração de Casos de Teste Funcional para Aplicações de Celulares

Emanuela Gadelha Cartaxo

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Informática da Universidade Federal de Campina Grande como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Engenharia de Software

Patrícia Duarte de Lima Machado
(Orientadora)

Campina Grande, Paraíba, Brasil
© Emanuela Gadelha Cartaxo, Outubro 2006

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

C322g Cartaxo, Emanuela Gadelha
2006 Geração de casos de teste funcional para aplicações de celulares / Emanuela Gadelha Cartaxo. — Campina Grande, 2006.
158fs.: il.

Referências.
Dissertação (Mestrado em Informática) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.
Orientadora: Patrícia Duarte de Lima Machado.

1— Testes de Caixa-Preta 2— LTS 3 — Diagrama de Seqüência 4 —
Aplicação de Celular I — Título

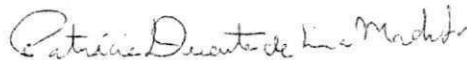
CDU 004.415.532.2



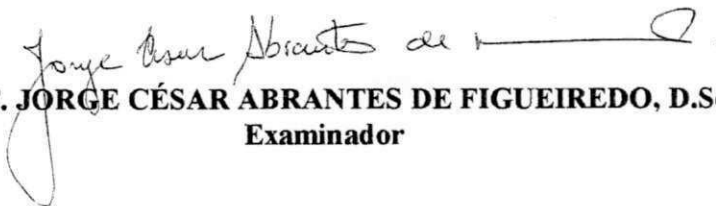
**“GERAÇÃO DE CASOS DE TESTE FUNCIONAL PARA APLICAÇÕES DE
CELULARES”**

EMANUELA GADELHA CARTAXO

DISSERTAÇÃO APROVADA EM 02.10.2006



PROF^a PATRÍCIA DUARTE DE LIMA MACHADO, Ph.D
Orientadora



PROF. JORGE CÉSAR ABRANTES DE FIGUEIREDO, D.Sc
Examinador



PROF. AUGUSTO CEZAR ALVES SAMPAIO, Ph.D
Examinador

CAMPINA GRANDE – PB

Resumo

No mercado de aplicações de celulares, o desenvolvimento baseado em *features* vem sendo considerado para a construção de aplicações. As *features* são desenvolvidas e testadas como um módulo independente. Existe uma carência por abordagens voltadas ao teste de *features* para aplicações de celulares, com relação a notações mais comumente utilizadas no mercado, como UML, e passíveis de automação. A utilização de notações já utilizadas no processo de desenvolvimento torna o processo de teste menos oneroso, uma vez que não é necessário produzir artefatos específicos para geração de casos de teste. A derivação de testes a partir de diagramas UML, mais especificamente diagramas de seqüência, e aplicação de técnicas de seleção efetivas são pontos de interesse ainda não plenamente abordados neste contexto. Neste trabalho, propomos um procedimento sistemático que realiza a geração de casos de teste funcionais de *features*, desenvolvidas para celulares, a partir de diagramas UML, mais especificamente diagrama de seqüência, como também duas estratégias de seleção de casos de teste. Uma ferramenta de suporte foi desenvolvida para automatizar suas atividades e dois estudos de caso foram realizados para demonstrar a aplicação do procedimento sistemático proposto e das duas estratégias de seleção, como também o funcionamento da ferramenta.

Abstract

In the mobile phone market, feature-based development has been considered for the applications construction. Features are developed and tested as an independent module. There is a lack for approaches directed to the feature test for mobile phone applications, with respect to notations more used in the industry, as UML, and possible of automation. The use of notations already used in the development process makes the test process less onerous, since it is not necessary to elaborate specific artifacts for test cases generation. The derivation from UML diagrams, more specifically sequence diagrams, and the application of effective selection techniques are of interest, but they are not fully adressed in this context yet. In this work, we propose a systematic procedure to generate functional test cases for features of mobile phones, as well as two strategies to test case selection. Tool suport has been developed to automate the activities of the systematic procedure and selection strategies. Two case studies were performed to demonstrate its application as well as the tool functionality.

Agradecimentos

Em primeiro lugar agradeço a Deus.

A meus pais pelo imenso amor, apoio e incentivo recebido durante todo esse período; sem isso não teria conseguido. As minhas irmãs pelo companheirismo.

Agradeço a professora Patrícia pela sua orientação e dedicação.

Aos poucos verdadeiros amigos que, mesmo diante das minhas ausências semanais a essência da verdadeira amizade.

A Laísa Helena e Francisco Neto por todo o apoio dado durante esta etapa.

A Wilkerson, Bruno Brito e Jaime por todo o companheirismo durante este etapa.

A Motorola Brasil, que através do projeto de pesquisa, ofereceu uma ambiente para experimentações, como também o suporte financeiro.

Conteúdo

Introdução	1
1.1 Objetivos do Trabalho	5
1.2 Resultados e Relevância do Trabalho.....	6
1.3 Estrutura da Dissertação	7
Fundamentação Teórica	9
2.1 Teste de Software	9
2.1.1 Casos de Teste	10
2.1.2 Tipos de Teste de <i>Software</i>	10
2.1.3 Teste Baseado em Modelo.....	11
2.1.4 Seleção de Casos de Teste	14
2.2 Aplicações para Celulares	15
2.3 Diagrama de Seqüência UML	16
2.4 LTS.....	21
2.4.1 LTS Anotado	23
2.5 Considerações Finais	24
Geração de Casos de teste funcional a partir de diagrama de seqüência UML por meio de LTS	26
3.1 Obtenção do Modelo LTS	27
3.2 Geração de Casos de Teste	34
3.3 Considerações Finais	38
Seleção de Casos de teste.....	39
4.1 Seleção de Casos de Teste Baseada em um Propósito de Teste	39
4.2 Seleção de Casos de Teste Baseada na Similaridade de Caminhos.....	44
4.3 Considerações Finais	48
Ferramenta LTS-BT	49
5.1 Visão Geral.....	49
5.2 Interface com o Usuário	52
5.2.1 TGF - AUT.....	53
5.2.2 SD - AUT	54
5.2.3 GP (<i>Generation with Purpose</i>).....	56
5.2.4 GP (<i>Generation with Purpose</i>) gráfico.....	59
5.2.5 Exibição Textual de LTS.....	60
5.2.6 Exibição Gráfica de um LTS.....	61
5.3 Projeto	62
5.4 Considerações Finais	65
Estudo de caso.....	66
6.1 <i>Feature “Hot message”</i>	66

6.1.1 Descrição da <i>Feature</i>	66
6.1.2 Geração e Seleção de Casos de Teste	67
6.2 <i>Feature</i> “Itens Embutidos em Mensagem”	72
6.2.1 Descrição da <i>Feature</i>	72
6.2.2 Geração de Casos de Teste	76
6.3 Considerações Finais	78
Conclusão	79
7.1 Contribuições	80
7.2 Trabalhos Relacionados.....	81
7.2.1 TGV.....	82
7.2.2 PTK	82
7.3 Trabalhos Futuros.....	83
Diagramas de Seqüência	87
A.1 Diagramas de seqüência relacionados à “Armazenar URL”	87
A.2 Diagramas de seqüência relacionados à “Ir para a URL”	95
A.3 Diagramas de seqüência relacionados à “Enviar mensagem ao Número do telefone”	98
A.4 Diagramas de seqüência relacionados à “Enviar mensagem ao endereço de email”	102
A.5 Diagramas de seqüência relacionados à “Armazenar Número de Telefone”	106
A.6 Diagramas de seqüência relacionados à “Armazenar endereço de email”	110
A.7 Diagramas de seqüência relacionados à “Ligar para o número do telefone”	114
A.8 Diagramas de seqüência relacionados à “Enviar uma mensagem de voz a um número de telefone”	118
Casos de Teste	122

Lista de Figuras

Figura 2.1 - Abordagem de Teste baseado em modelo.....	13
Figura 2.2 – Conjunto de <i>Features</i>	15
Figura 2.3 - Elementos do diagrama de seqüência	16
Figura 2.4 – Diagrama de Seqüência padrão	18
Figura 2.5 - Diagrama de seqüência que representa o comportamento de uma aplicação de celular quando o usuário quer armazenar um número de telefone que está embutido na mensagem	19
Figura 2.6 - Diagrama de Seqüência - Iteração	20
Figura 2.7 - Diagrama de seqüência – Fluxo alternativo	21
Figura 2.8 - Elementos do LTS	22
Figura 2.9 – LTS que representa o comportamento do sistema quando o usuário, através da aplicação mensagem, tenta visualizar a caixa de entrada e saída	23
Figura 2.10 – Exemplo de LTS anotado.....	24
Figura 3.1 - Construção inicial do LTS derivado (Figura 3.1(b)) do diagrama de Seqüência da Figura 3.1(a)	30
Figura 3.2 - Construção do LTS derivado do diagrama de Seqüência da Figura 3.1(a)– mensagens do loop	30
Figura 3.3 - LTS derivado do diagrama de seqüência Figura 3.1(a)	31
Figura 3.4 - Construção inicial do LTS derivado do diagrama de seqüência da Figura 2.7	31
Figura 3.5 - Construção do LTS derivado do diagrama de seqüência da Figura 2.7 – Fluxo alternativo 1 32	
Figura 3.6 - Construção do LTS derivado do diagrama de seqüência da Figura 2.7– Fluxo alternativo 2 33	
Figura 3.7 - Construção do LTS derivado do diagrama de seqüência da Figura 2.7– Fluxo Comum	33
Figura 3.8 - LTS derivado do diagrama de seqüência (Figura 2.7)	34
Figura 3.9 - Caminhos obtidos	36
Figura 4.1 - Fluxo da seleção de casos de teste baseada em um propósito.....	39
Figura 4.2 - Modelo LTS (I) e Propósitos de Teste	40
Figura 4.3 - Sub-modelo obtido do Modelo da Figura 4.2 (I) e o propósito da Figura 4.2 (II) (d).....	42
Figura 4.4 - Sub-modelo obtido do Modelo da Figura 4.2 (I) e o propósito da Figura 4.2 (II) (e).....	42
Figura 4.5 - Sub-modelo obtido do Modelo da Figura 4.2 (I) e o propósito da Figura 4.2 (II) (f)	43
Figura 4.6 - Sub-modelo obtido do Modelo da Figura 4.2(I) e o propósito da Figura 4.2(II) (g).....	43
Figura 4.7 - Sub-modelo obtido do Modelo da Figura 4.2(I) e o propósito da Figura 4.2 (II) (h).....	44
Figura 4.8 - Fluxo da seleção de casos de teste baseada na similaridade de caminhos	45
Figura 4.9 - Modelo LTS.....	46
Figura 4.10 - Caminhos referentes ao modelo da Figura 4.9.....	46
Figura 4.11 - Matriz de similaridade referente ao modelo da Figura 4.10	47
Figura 4.12 - Matriz de similaridade após primeira eliminação	47
Figura 4.13 - Matriz de similaridade após segunda eliminação.....	48
Figura 5.1 - Visão geral da ferramenta LTS-BT.....	51

Figura 5.2 - Ferramenta LTS-BT – Tela Principal	53
Figura 5.3 - Ferramenta LTS-BT – Tela Abrir arquivo TGF	54
Figura 5.4 - Ferramenta LTS-BT – Tela resultado de TGF-AUT	54
Figura 5.5 - Ferramenta LTS-BT – Tela Abrir Diagrama de Seqüência (MDL ou RTMDL)	55
Figura 5.6 - Ferramenta LTS-BT – Tela Resultado SD-AUT	55
Figura 5.7 - Ferramenta LTS-BT – Tela Abrir AUT	56
Figura 5.8 - Ferramenta LTS-BT – Tela Propósito de teste	57
Figura 5.9 - Ferramenta LTS-BT – Tela Propósito de teste “*” e 75% de cobertura	58
Figura 5.10 - Ferramenta LTS-BT – Tela Resultado GP	59
Figura 5.11 - Ferramenta LTS-BT – Tela Propósito de teste “*, Abrir caixa de entrada” e 100% de cobertura.....	59
Figura 5.12 - Ferramenta LTS-BT – Escolha de um arquivo (AUT ou TGF) para exibição.....	60
Figura 5.13 - Ferramenta LTS-BT – Exibição textual de um LTS.....	60
Figura 5.14 - Ferramenta LTS-BT – Exibição gráfica do LTS.....	61
Figura 5.15 - Visão Geral do projeto da ferramenta LTS-BT.....	62
Figura 5.16 - Algoritmo <i>Extractor</i>	62
Figura 5.17 – Algoritmo <i>Parser</i> TGF para AUT.....	63
Figura 5.18 – TGF (a) e AUT (b)	63
Figura 5.19 - Algoritmo MDL/RTMDL para AUT	64
Figura 6.1 - Diagrama de Seqüência - Armazenamento de mensagens favoritas	68
Figura 6.2 - LTS derivado do diagrama de seqüência da Figura 6.1	69
Figura 6.3 - Tela Propósito de teste	69
Figura 6.4 - Caminhos do LTS	70
Figura 6.5 - Tela Propósito de teste “*, <i>Hot message flex is on</i> , *, <i>accept</i> ”, 100% de cobertura de caminhos.....	71
Figura 6.6 - Tela Propósito de teste “*”, 75% cobertura de caminhos	72
Figura A.1 - Armazenar URL, que está embutida em uma mensagem de texto recebida e o usuário está visualizando, na Agenda.....	87
Figura A.2 - Armazenar URL, que está embutida em uma mensagem de texto recebida e o usuário está visualizando, nos favoritos do navegador.....	90
Figura A.3 - Armazenar URL, que está embutida em uma mensagem multimídia (MMS) recebida e o usuário está visualizando, nos favoritos do navegador.....	91
Figura A.4 - Armazenar URL, que está embutida em uma mensagem multimídia (MMS) recebida e o usuário está visualizando, na Agenda.....	92
Figura A.5 - Armazenar URL, que está embutida em uma mensagem de email recebida e o usuário está visualizando, nos favoritos do navegador.....	93
Figura A.6 - Armazenar URL, que está embutida em uma mensagem de email recebida e o usuário está visualizando, na agenda.....	94
Figura A.7 - Ir para a URL, que está embutida em uma mensagem de texto recebida, quando o usuário está visualizando a mensagem.....	95

Figura A.8 - Ir para a URL, que está embutida em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem.	96
Figura A.9 - Ir para a URL, que está embutida em uma mensagem de email recebida, quando o usuário está visualizando a mensagem.	97
Figura A.10 - Enviar mensagem ao número de telefone embutido no campo “De” de uma mensagem selecionada na caixa de entrada de mensagens.	98
Figura A.11 - Enviar mensagem ao número de telefone embutido em uma mensagem texto recebida, quando o usuário está visualizando a mensagem.	99
Figura A.12 - Enviar mensagem ao número de telefone embutido em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem.	100
Figura A.13 - Enviar mensagem ao número telefone de embutido em uma mensagem de email recebida, quando o usuário está visualizando a mensagem de email.	101
Figura A.14 - Enviar mensagem ao endereço de email embutido no campo “De” de uma mensagem selecionada na caixa de entrada de mensagens.	102
Figura A.15 - Enviar mensagem ao endereço de email embutido em uma mensagem de texto recebida, quando o usuário está visualizando a mensagem.	103
Figura A.16 - Enviar mensagem ao endereço de email embutido em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem.	104
Figura A.17 - Enviar a mensagem ao endereço de email embutido em uma mensagem de email recebida, quando o usuário está visualizando a mensagem de email.	105
Figura A.18 - Armazenar número de telefone embutido no campo “De” de uma mensagem de texto selecionada na caixa de entrada de mensagens.	106
Figura A.19 - Armazenar número de telefone embutido em uma mensagem de texto recebida, quando o usuário está visualizando a mensagem.	107
Figura A.20 - Armazenar número de telefone embutido em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem.	108
Figura A.21 - Armazenar número de telefone embutido em uma mensagem de email recebida, quando o usuário está visualizando a mensagem.	109
Figura A.22 - Armazenar endereço de email embutido no campo “De” de uma mensagem de texto selecionada na caixa de entrada de mensagens.	110
Figura A.23 - Armazenar endereço de email embutido em uma mensagem de texto recebida, quando o usuário está visualizando a mensagem.	111
Figura A.24 - Armazenar endereço de email embutido em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem.	112
Figura A.25 - Armazenar endereço de email embutido em uma mensagem de email recebida, quando o usuário está visualizando a mensagem.	113
Figura A.26 - Ligar para o número de telefone embutido no campo “De” de uma mensagem de texto selecionada na caixa de entrada de mensagens.	114
Figura A.27 - Ligar para o número de telefone embutido em uma mensagem de texto recebida, quando o usuário está visualizando a mensagem.	115

Figura A.28 - Ligar para o número de telefone embutido em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem.....	116
Figura A.29 - Ligar para o número de telefone embutido em uma mensagem de email recebida, quando o usuário está visualizando a mensagem.	117
Figura A.30 - Enviar uma mensagem de voz a um número de telefone embutido no campo “De” de uma mensagem de texto selecionada na caixa de entrada de mensagens.	118
Figura A.31 - Enviar uma mensagem de voz a um número de telefone embutido em uma mensagem de texto recebida, quando o usuário está visualizando a mensagem.	119
Figura A.32 - Enviar uma mensagem de voz a um número de telefone embutido em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem.	120
Figura A.33 - Enviar uma mensagem de voz a um número de telefone embutido em uma mensagem de email recebida, quando o usuário está visualizando a mensagem.	121

Lista de Tabelas

Tabela 3.1 - Tabela de caminho.....	35
Tabela 3.2 – Caso de teste obtido do LTS da Figura 3.3.....	36
Tabela 3.3 – Tabela de caminho referente ao LTS (Figura 3.8).....	37
Tabela 3.4 – Caso de teste 1.....	37
Tabela 3.5 - Caso de teste 2.....	37
Tabela B.1 – Caso de Teste 1 referente ao diagrama de seqüência da Figura A.1.....	122
Tabela B.2 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.1.....	122
Tabela B.3 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.2.....	123
Tabela B.4 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.2.....	123
Tabela B.5 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.3.....	123
Tabela B.6 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.3.....	124
Tabela B.7 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.4.....	124
Tabela B.8 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.4.....	124
Tabela B.9 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.5.....	125
Tabela B.10 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.5.....	125
Tabela B.11 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.6.....	125
Tabela B.12 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.6.....	126
Tabela B.13 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.7.....	126
Tabela B.14 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.7.....	126
Tabela B.15 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.8.....	127
Tabela B.16 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.8.....	127
Tabela B.17 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.9.....	127
Tabela B.18 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.9.....	128
Tabela B.19 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.10.....	128
Tabela B.20 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.10.....	128
Tabela B.21 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.11.....	129
Tabela B.22 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.11.....	129
Tabela B.23 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.12.....	129
Tabela B.24 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.12.....	130
Tabela B.25 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.13.....	130
Tabela B.26 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.13.....	131
Tabela B.27 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.14.....	131
Tabela B.28 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.14.....	131
Tabela B.29 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.15.....	132
Tabela B.30 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.15.....	132
Tabela B.31 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.16.....	133
Tabela B.32 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.16.....	133

Tabela B.33 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.17	133
Tabela B.34 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.17	134
Tabela B.35 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.18	134
Tabela B.36 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.18	134
Tabela B.37 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.19	135
Tabela B.38 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.19	135
Tabela B.39 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.20	135
Tabela B.40 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.20	136
Tabela B.41 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.21	136
Tabela B.42 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.21	136
Tabela B.43 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.22	137
Tabela B.44 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.22	137
Tabela B.45 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.23	137
Tabela B.46 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.23	138
Tabela B.47 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.24	138
Tabela B.48 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.24	138
Tabela B.49 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.25	139
Tabela B.50 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.25	139
Tabela B.51 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.26	139
Tabela B.52 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.26	140
Tabela B.53 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.27	140
Tabela B.54 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.27	140
Tabela B.55 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.28	141
Tabela B.56 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.28	141
Tabela B.57 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.29	141
Tabela B.58 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.29	142
Tabela B.59 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.30	142
Tabela B.60 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.30	143
Tabela B.61 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.31	143
Tabela B.62 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.31	144
Tabela B.63 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.32	144
Tabela B.64 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.32	145
Tabela B.65 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.33	145
Tabela B.66 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.33	146

Capítulo 1

Introdução

Um equívoco no tocante às empresas de aplicações de celulares foi pensar demasiadamente em tecnologia, em detrimento das conhecidas regras de *marketing*, em que é necessário colocar no mercado os melhores e mais inovadores produtos, traçar uma forma de incrementar a fidelidade do usuário, e reforçar a marca junto ao público final, garantindo assim a sua lucratividade [LSJ05].

Devido à disputa acirrada no mercado de aplicações de celulares, usuários têm demandado redução de custos [DJK⁺99] e um produto de boa qualidade (que atenda aos seus requisitos de forma correta) para manter seus contratos. Deste modo, empresas têm investido em pesquisas de abordagens para reduzir os custos, mantendo ou aumentando a qualidade da aplicação que está sendo desenvolvida de forma a garantir a preferência do usuário final aos seus produtos.

Um dos principais objetivos da engenharia de *software* é desenvolver aplicações de alta qualidade [GWH99]. Várias técnicas foram desenvolvidas com a finalidade de tornar o teste de *software* uma tarefa mais simples. Atualmente, o alvo é melhorar a eficiência e completude de tais técnicas de modo que haja um aumento da qualidade das aplicações de *software*.

No contexto de aplicações de celulares, o desenvolvimento de aplicações é baseado em *features*. Uma *feature* é uma estrutura funcional e com propriedades visíveis [TLW98] e um celular serve como uma plataforma para um conjunto de *features*. Cada *feature* é desenvolvida sem o conhecimento das outras *features*. Isto surgiu como um fator determinante para uma rápida construção do *software*. Para tanto, as *features* precisam ser testadas de forma a garantir a sua confiabilidade. Desta forma o teste funcional para as *features* isoladas surge como uma técnica não apenas capaz de validar os requisitos funcionais (ou serviços) demandados pelos usuários finais, mas também capaz de reduzir os custos inerentes ao processo de teste, visto que, a partir da especificação do software, casos de teste podem ser obtidos concomitantemente ao seu desenvolvimento. Casos de testes funcionais são conduzidos para validar a aplicação de

acordo com os requisitos que foram especificados, sendo gerados observando a aplicação apenas através de suas interfaces.

Muitas pesquisas têm sido desenvolvidas com o intuito de definir técnicas efetivas de derivação de casos de teste a partir da especificação de requisitos das aplicações como pode ser visto em [Bei95, BL01, BAMF04, OA99, AO00, Gra94]. Para isto, é necessário que a especificação dos requisitos do *software* seja formalmente ou semi-formalmente definida [TB02], de modo a caracterizar com exatidão o comportamento do sistema.

Testar é um processo concorrente no ciclo de vida da engenharia de *software* a fim de medir a qualidade da aplicação que está sendo considerada e contribuir para a sua melhoria [CJ02]. Em um processo manual, quanto maior a aplicação sob teste, maior o tempo gasto com esta tarefa. Dessa forma, aspira-se a automatização do esforço de teste.

Conforme visto em [LJX⁺04, TB02], o esforço de teste é dividido em três fases: a geração de casos de teste, a execução e a avaliação dos testes, sendo estas duas últimas tarefas fáceis de automatizar enquanto a geração é uma tarefa não trivial de ser feita, uma vez que requer especificações formais ou semi-formais. Neste ponto, a utilização de especificações UML (*Unified Modeling Language*), é atrativa devido ao fato de UML possuir informações relevantes para testes e estar em uso na academia e na maioria das indústrias de TI&T [HVR04] nos dias de hoje, de forma que não haveria custos extras associados a treinamento e integração no processo de desenvolvimento da aplicação.

Como em [Bin99] são levantados alguns problemas acerca da interpretação na notação de diagramas UML, especialmente diagrama de seqüência, então, como forma de amenizar este problema pode-se optar por um modelo de representação de comportamento, usualmente na forma de sistemas de transição rotulados (LTS) [Vri00] que podem ser derivados a partir de diagramas UML. Existem várias ferramentas de animação deste tipo de modelo no mercado que tornam possível a validação dos requisitos de uma aplicação. Em geral, um LTS provê uma descrição integral e global do conjunto de todos os possíveis comportamentos de um sistema; um caminho em um LTS pode ser visto como uma seqüência de teste. Logo, LTSs são modelos versáteis para geração de casos de teste.

Partindo do princípio que nem sempre é possível executar todos os casos de testes gerados para uma determinada aplicação, devido aos custos e tempo demandado, a

comunidade científica e as empresas têm investido também em estratégias de seleção de casos de teste [TB02].

Um ponto comum entre os pesquisadores é a necessidade de automação das técnicas que tratam da geração e seleção de casos de teste, de modo que os processos de geração e de seleção tenham a menor intervenção humana possível, para evitar erros. Uma vez obtida a especificação formal do comportamento da aplicação, tem-se como fazer a verificação da especificação, geração e seleção automática de casos de testes.

Embora existam várias técnicas de geração e de seleção de casos de teste já disponíveis, a maioria delas utiliza como entrada uma especificação formal para a sua viabilização, o que pode dificultar a aplicação prática do processo de teste. Desse modo, partindo da observação que tanto as empresas quanto a academia se utiliza de modelos UML para especificar as aplicações que serão desenvolvidas, e sabendo que a intenção é melhorar a qualidade de software, então resolvemos utilizar UML neste trabalho, de modo que possamos gerar casos de teste automaticamente daquele tipo de notação, sem trazer junto com a tarefa de teste, a exigência de produção de artefatos específicos para aquela. Com isto poderemos tanto gerar casos de teste como também fazer uma seleção dentre um conjunto possível de casos de teste.

Este trabalho foi desenvolvido no contexto de uma cooperação entre a Motorola, o Centro de Informática da UFPE, chamado CInBTCRD (*CIn Brazil Test Center Research and Development*), que conta com a cooperação da UFCG e cujo objetivo é definir um processo integrado de geração, seleção e avaliação de casos de testes para aplicações de celulares. O CInBTCRD aborda a criação de modelos formais não ambíguos que representam os requisitos de aplicações da Motorola. A partir destes modelos é possível extrair, de forma sistemática e automatizada, diversas informações pertinentes ao processo de testes. Tendo em vista o contexto CInBTCRD, delimitamos o nosso escopo para teste funcional de *features* isoladas partindo de diagramas de seqüência UML.

Teste de *features* e, mais particularmente, a interação de *features* tem sido investigada em diferentes trabalhos no domínio de telefonia [CKMR99]. Em [BBJ⁺03], é apresentada a ferramenta PTK. Esta ferramenta gera scripts de teste de *features* a partir de especificações baseadas em cenários usando *Message Sequence Charts* (MSC). Em [Per92], é apresentada uma metodologia para projetar e executar testes de *feature* de ISDN onde para cada *feature* é desenhada uma árvore da *feature* que representa os

requisitos da *feature*, e cada caminho da árvore representa um caso de teste. Em [Nog06], é apresentada uma proposta de geração automática orientada por propósitos de casos de teste de *features* partindo de modelos *Communicating Sequential Processes* (CSP) [RHB97]. Em [FWM06], é apresentada uma proposta de geração automática de casos de teste de interações de *features*.

Geração de casos de teste partindo de diagramas UML tem sido bastante explorada nos últimos anos. Em [Bin99] é apresentado um padrão de teste “*Round-trip Scenario Test Pattern*” que utiliza diagramas de seqüência UML para geração de casos de teste. Neste trabalho, o autor deriva um grafo de fluxo do diagrama de seqüência e, a partir dele gera os casos de teste. Esta abordagem difere da que estamos propondo, visto que do diagrama de seqüência derivamos um modelo LTS e, com isto, conforme citado anteriormente, torna-se possível a validação dos requisitos da *feature*. UMLAUT (*Unified Modeling Language All pUrposes Transformer*) é uma ferramenta para manipulação de modelos (incluindo diagramas de seqüência) UML [JLP98].

UMLAUT faz a transformação de modelos UML para LTS. Ela é utilizada junto com o a ferramenta TGV (*Test Generation with Verification technology*) [BMJ01] para tornar possível a geração de casos de teste. Como UMLAUT possui algumas restrições, tais como ausência de *self-loops*, *loops* e fluxos alternativos, para criar diagrama de seqüências, então não conseguimos representar comportamentos mais interessantes do sistema de aplicações de celulares (fluxo de controle).

Em geral, pode-se observar que existe uma carência por abordagens voltadas ao teste de *features*, com base em notações mais comumente utilizadas no mercado, como UML, e passíveis de automação. A geração de casos de teste a partir de diagramas de seqüência, apesar de já ser considerada em diferentes trabalhos, não é abordada de forma que possa ser diretamente aplicada ao nosso contexto, devido às restrições impostas aos diagramas e às limitações de interpretação de tais diagramas. O uso de LTS como modelo para geração de casos de teste também já é abordado em diferentes trabalhos. No entanto, sua derivação a partir de diagramas de seqüência e aplicação de técnicas de seleção efetivas são pontos de interesse ainda não plenamente abordados.

1.1 Objetivos do Trabalho

O objetivo deste trabalho é desenvolver um procedimento sistemático para a extração de casos de teste funcionais e estratégias de seleção, a partir especificações de *features* de aplicações para celulares. O enfoque é no teste de *features* isoladas. Cenários de uso da *feature* são especificados usando diagramas de seqüência UML. Tal procedimento e estratégias de seleção devem ser passíveis de automação.

Diagramas de seqüência UML descrevem o comportamento da *feature*, mostrando a interação entre os objetos em um nível mais detalhado [CCT02] e, como já mencionado anteriormente, eles são bastante utilizados tanto na academia quanto na indústria. Dessa forma estamos aliando o nosso objetivo de gerar casos de teste funcional, sem ter um custo adicional com artefatos de teste, considerando o reaproveitamento de diagramas de seqüência previamente construídos por desenvolvedores.

Desenvolveremos uma ferramenta, que dê suporte ao procedimento sistemático, contemplando os seguintes pontos:

- **Transformação de diagramas de seqüência UML em LTS:** Como há alguns problemas de interpretação com a notação de diagramas de seqüência UML no que diz respeito a representação de interações e fluxos alternativos [Bin99], então como forma de amenizar tais problemas decidimos derivar um modelo LTS de diagramas de seqüência. Desta forma, poderemos focar em uma única interpretação que poderá ser validada por meio de ferramentas de animação, minimizando ambigüidades e inconsistências no processo de geração de casos de teste;
- **Geração de casos de testes:** Partindo do modelo LTS obtido, deveremos fazer a geração de casos de teste;
- **Seleção de casos de teste:** dado um conjunto de casos de teste, poderemos fazer uma seleção de acordo com algum propósito ou baseado na similaridade de caminhos.

1.2 Resultados e Relevância do Trabalho

A principal contribuição deste trabalho é a disponibilização de um procedimento sistemático para geração de casos de teste funcional e de duas estratégias de seleção de casos de teste para *features* isoladas, partindo de diagramas de seqüência UML, com suporte ferramental.

Um ponto relevante deste trabalho é a não adição de custos extras associado a tarefa de testes, uma vez que não se utiliza de artefatos novos e a elevação da confiabilidade associada ao procedimento sistemático, uma vez que pode ser feita uma verificação se o modelo representa fielmente o comportamento requerido da aplicação.

Com o uso do procedimento sistemático, os custos relacionados a testes podem ser bastante reduzidos, uma vez que inconsistências na especificação podem ser detectadas no início do processo de desenvolvimento (o modelo LTS pode ser animado). Isto torna menos onerosa a correção do *software* já que erros são detectados previamente, evitando assim a propagação dos mesmos para as etapas futuras do desenvolvimento, o que tornaria mais complicada e onerosa a correção.

Uma breve descrição das contribuições relativas a cada resultado obtido é dada a seguir:

- **Procedimento sistemático para geração de casos de teste a partir de diagramas de seqüência UML** Um procedimento sistemático para geração de casos de teste poderá contribuir de forma significativa para a cobertura da especificação, uma vez que comportamentos possíveis da *feature* estão no modelo e devemos “explorar” todo o modelo. Outra contribuição é a não duplicação de casos de teste, evitando o desprendimento de esforço de execução (manual ou automático) para o mesmo caso de teste mais de uma vez.
- **Seleção de Casos de Teste** Executar todos os caminhos de uma aplicação nem sempre é possível devido ao tempo necessário. Estratégias capazes de reduzir casos de teste de uma suíte de teste têm sido pesquisadas, mas não se tem chegado a uma solução ótima. Teremos duas contribuições com seleção:

- **Seleção de casos de teste com propósito de teste** Esta estratégia já é bastante conhecida na literatura, porém as notações utilizadas são pouco utilizadas na prática. Aqui estaremos utilizando uma proposta de escrever propósito de teste bastante simples, com o fim de facilitar a sua rápida adoção;
- **Seleção de casos de teste baseada na similaridade de caminhos** Essa estratégia tem a finalidade de selecionar um conjunto de casos de teste dada uma suíte completa, e baseada em um grau de cobertura necessário, selecionar de acordo com o grau de similaridade entre eles.

1.3 Estrutura da Dissertação

Este documento está estruturado da seguinte forma:

Capítulo 2: Fundamentação Teórica: Na fundamentação teórica, são apresentados os principais conceitos relativos a testes, aplicações de celulares (*features*), diagramas de seqüência UML e LTS, que foram abordados neste trabalho. A intenção é fornecer um embasamento teórico para os leitores.

Capítulo 3: Geração de Casos de teste funcional a partir de diagramas de seqüência UML por meio de LTS: Neste capítulo, é apresentado um procedimento sistemático para a transformação de diagrama de seqüência em LTS e, a partir do LTS, é apresentado o processo de geração de casos de teste.

Capítulo 4: Seleção de Casos de Teste: Neste capítulo, são apresentadas duas estratégias de seleção de casos de testes: baseada em propósito de teste e baseada na similaridade de caminhos.

Capítulo 5: Ferramenta LTS-BT: A ferramenta LTS-BT é apresentada neste capítulo. A ferramenta foi implementada para dar suporte ao procedimento sistemático e as estratégias de seleção apresentados nos Capítulos 3 e 4. É feita uma descrição da arquitetura da ferramenta e das suas funcionalidades.

Capítulo 6: Estudo de Caso: Uma aplicação prática do procedimento sistemático e das estratégias de seleção de casos de teste propostos é demonstrada. Diagramas de seqüência, de duas *features* reais, são utilizados para aplicar o procedimento sistemático e as estratégias de seleção propostos. Por fim, os resultados obtidos serão analisados.

Capítulo 7: Conclusões: O trabalho proposto é concluído a partir de uma análise dos resultados obtidos e é feita uma descrição das perspectivas de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

O objetivo deste capítulo é fornecer embasamento teórico para os leitores acerca dos conceitos utilizados neste trabalho. São apresentados os principais conceitos relacionados a teste, ressaltando o teste baseado em modelo e a seleção de casos de teste, aplicações de celulares, digramas de sequência UML (*Unified Modelling Language*), e LTSs (*Labeled Transition System*).

2.1 Teste de Software

O teste do software é uma das fases do processo de engenharia de *software* que visa atingir um nível superior da qualidade de *software*. O objetivo desta fase é encontrar defeitos no *software*, para que estes possam ser corrigidos antes da entrega final.

A atividade de teste de software é um elemento de um tema mais amplo chamado Verificação e Validação (V&V), onde:

- **Verificação** - refere-se ao conjunto de atividades que garante que o *software* implementa corretamente uma função específica, e;
- **Validação** - refere-se ao conjunto de atividades que garante que o *software* que foi construído atende às exigências do cliente.

A definição de V&V abrange muitas das atividades às quais nos referimos como garantia da qualidade de *software* (SQA).

Uma das definições mais aceitas sobre teste diz que testar é um processo concorrente no ciclo de vida da engenharia de *software* a fim de medir e melhorar a qualidade da aplicação que está sendo testada [CJ02].

A concepção tradicional do processo de teste, como sendo uma fase final e independente do processo de desenvolvimento propriamente dito, tem se mostrado

bastante ineficiente devido aos altos custos associados com a correção de erros encontrados e manutenção do *software*. Tal fato contribuiu para a definição de métodos e técnicas sistemáticas de teste que fazem do processo de teste, um conjunto de tarefas a parte que pode ser aplicado ao longo do processo de desenvolvimento [MS01].

2.1.1 Casos de Teste

A essência do teste de *software* é determinar o conjunto de casos de teste para o *software* a ser testado. Um caso de teste é composto por [Jor95]:

- **Entradas:**
 - **Condição Inicial:** assegura a condição inicial para que o caso de teste possa ser executado;
 - **Passos:** passos a serem executados, identificados pelos métodos de teste.
- **Saídas:**
 - **Resultados esperados:** respostas esperadas do sistema, para a entrada atual;
 - **Pós-condição:** representa o estado final do sistema.

Estando o caso de teste definido, o testador deve deixar a aplicação de acordo com a condição inicial e exercitá-la com os “passos atuais”, coletando os resultados até que a pós-condição seja alcançada. Os resultados coletados são comparados com as saídas esperadas para checar se o teste passou ou não, isto é, se o software se comporta como o esperado.

O conjunto de casos de teste definido para uma dada aplicação é denominado de suíte de teste.

2.1.2 Tipos de Teste de Software

O teste de *software* pode ser classificado como [Wat00]:

- **Black-box ou funcional:** são testes planejados a partir da especificação abstrata, ou seja, não há conhecimento do código;
- **White-box ou estrutural:** são testes definidos a partir do conhecimento de detalhes da implementação;
- **Gray-box:** é um teste *black-box* baseado no conhecimento limitado de detalhes da implementação.

A abordagem funcional baseia-se em uma visão “*black-box*” da funcionalidade do *software* a ser testada, já a abordagem estrutural baseia-se em um conhecimento detalhado da implementação [Jor95]. Por esta razão, técnicas baseadas na abordagem funcional são bem mais simples de entender e aplicar comparadas às técnicas baseadas em abordagens estruturais.

Com a abordagem funcional temos a vantagem de gerar casos de teste mesmo antes de a implementação estar pronta, já que a base é a especificação. Por outro lado, temos que esta abordagem é dependente do modelo, ou seja, só serão gerados casos de testes do comportamento do *software* que estiver na especificação. Se todos os comportamentos possíveis do *software* não estiverem especificados, então os casos de teste gerados para aplicação não cobrirão todos os possíveis comportamentos. Já com a abordagem estrutural temos a desvantagem de somente conseguir gerar casos de teste depois que a implementação estiver pronta.

O foco deste trabalho é nas técnicas de teste funcional. Uma vez que o teste funcional pode ser obtido antes mesmo da fase de implementação, já que é baseado na especificação, isto pode significar para o processo de teste uma redução de custos e de tempo gasto por motivos citados anteriormente. Além disso, estando esta especificação descrita em uma notação semi-formal ou formal, podemos pensar na automatização do processo, que minimiza tanto o esforço para a aplicação da técnica quanto à probabilidade de inserção de erro humano.

2.1.3 Teste Baseado em Modelo

Teste baseado em modelo é uma abordagem *black box* para geração de teste de *software* a partir de modelos da aplicação [EW01], onde casos de testes são executados para avaliar a correspondência entre o modelo e a aplicação. Para isto, a especificação

da aplicação a ser testada precisa estar formalmente ou semi-formalmente descrita por meio de um modelo de modo a caracterizar com exatidão o seu comportamento [Bei95, DJK⁺99].

As principais atividades relacionadas ao teste baseado em modelo, mostrados na Figura 2.1 são descritas a seguir [EW01]:

- **Construir o modelo:** o modelo formal ou semi-formal é construído a partir dos requisitos da aplicação;
- **Gerar entradas:** as entradas do teste são geradas partindo do modelo. Estas entradas são passos que servirão para exercitar a aplicação que está sendo testada. Um exemplo de entrada em uma aplicação de celular seria “usuário seleciona a opção enviar mensagem”;
- **Gerar saídas esperadas:** as saídas esperadas do teste são geradas partindo do modelo formal. Aquelas indicam o comportamento esperado do sistema. Um exemplo de saída esperada em uma aplicação de celular seria “tela mensagem enviada é mostrada”;
- **Executar testes:** a aplicação é executada com as entradas geradas, gerando saídas;
- **Comparar saídas com saídas esperadas:** as saídas da aplicação que está sendo testada são comparadas com saídas esperadas geradas a partir do modelo.

O processo de teste baseado em modelo se inicia com os requisitos, isto significa que o processo de teste pode ser iniciado, assim que os requisitos da aplicação estiverem definidos. Com os requisitos definidos, o próximo passo é a construção de um modelo que retrate de forma íntegra o comportamento requisitado. A partir deste modelo construído, podemos ter um melhor entendimento da aplicação e obter casos de teste, geralmente obtidos de forma automática. A especificação dos casos de teste inclui entradas e saídas esperadas. Com as entradas, podemos executar a aplicação e observar o comportamento da aplicação, e por fim comparar as saídas obtidas com as saídas esperadas. Tal comparação servirá para avaliar a presença de defeitos na aplicação.

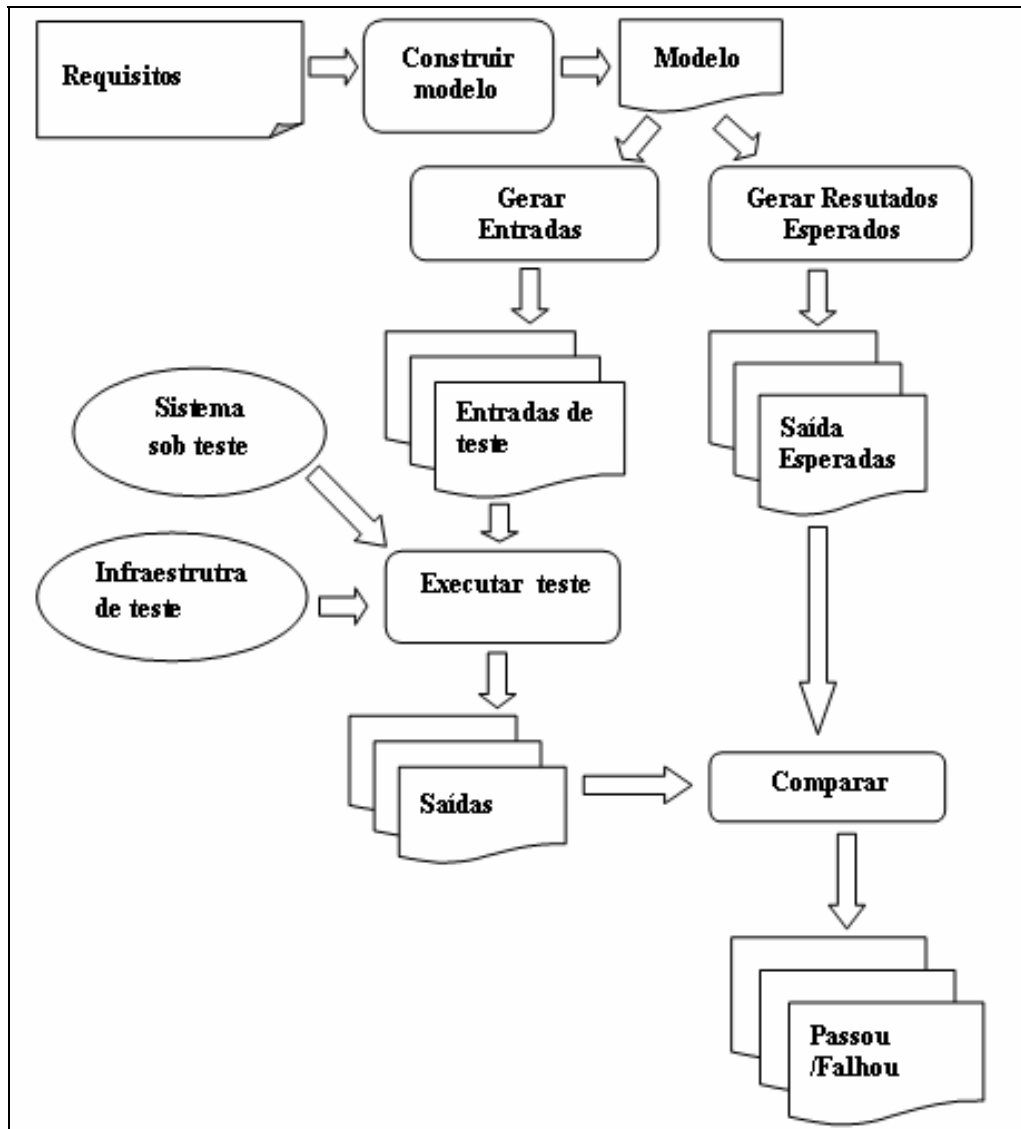


Figura 2.1 - Abordagem de Teste baseado em modelo

As vantagens de utilizar teste baseado em modelos são [EW01]:

- **Comunicação entre desenvolvedores e testadores:** Uma vez que existe um modelo do comportamento da aplicação, então este pode ser utilizado como a base de comunicação entre testadores e desenvolvedores;
- **Geração automática de testes:** com o modelo do comportamento da aplicação, a geração de casos de teste pode ser facilmente automatizada;
- **Atualização da suíte de testes:** uma vez alterado o modelo, facilmente pode ser feita a atualização da suíte de teste.

A desvantagem de utilizar teste baseado em modelos é a requisição de conhecimento da notação do modelo. O testador deve ser familiar com a notação que será utilizada, o que culmina na requisição de tempo e investimento em treinamentos, além do tempo que deve ser reservado para a obtenção do modelo [EW01]. Outra desvantagem é a dependência da existência do modelo, uma vez que toda essa abordagem parte do modelo construído.

2.1.4 Seleção de Casos de Teste

Nem sempre é possível executar todos os testes que são obtidos numa geração exaustiva de casos de teste [FGMT02, Hes06], devido à indisponibilidade de fatores como recursos e tempo. Assim, faz-se necessário aplicar estratégias que guiem a seleção de casos de teste.

Neste trabalho, iremos mostrar duas formas de aplicar as seguintes estratégias de seleção:

- **Seleção baseada em propósito de teste:** esta estratégia seleciona casos de teste baseada em uma entrada dada, o propósito de teste. Este é composto por comportamentos da aplicação que necessitam ser testados. Assim, dado o propósito de teste e o modelo da aplicação, é realizado um casamento entre eles, e só são gerados casos de teste que atendam ao propósito de teste definido. Esta seleção é baseada em [JJ04];
- **Seleção baseada em similaridade de caminhos:** esta estratégia seleciona casos de teste baseada na similaridade entre os caminhos que a aplicação (vista como modelo) possui. Esta seleção leva em consideração a porcentagem de cobertura de caminhos que se quer atingir. Assim, dado um modelo e a porcentagem de cobertura de caminho desejada, é aplicada uma função de similaridade entre os caminhos existentes e daí, vão sendo descartados os que têm maior grau de similaridade, até se chegar à porcentagem de caminhos desejada. Esta seleção é baseada em [LY01];

2.2 Aplicações para Celulares

No mercado de celulares, cada aplicação desenvolvida para celular é denominada *feature*. Uma *feature* denota uma estrutura funcional e com propriedades visíveis de um sistema de *software* [TLW98]. Uma *feature* é composta por um conjunto de requisitos que descrevem uma unidade coesa de funcionalidade, por exemplo, mensagem é uma *feature* e enviar e receber mensagens são os requisitos de mensagem.

O celular serve como uma plataforma para uma variedade de *features* [RL03] (Agenda, Mensagem, jogos, etc.). Cada *feature* é especificada sem o conhecimento de outras *features* com as quais pode ser agrupada [FN00] e pode executar concorrentemente, mas nem todas [RL03] devido a restrições de qualidade ou uso de recursos específicos de *hardware*

Veja na Figura 2.2 um conjunto de *features*. Naquela temos três *features* especificadas, A, B e C. Cada uma destas *features* são compostas por requisitos A (x, y, z), B (a, b, c, d), C(e, f). E as *features* podem ou não se relacionarem. Como no exemplo, temos que a *Feature A* se relaciona com a *Feature B*, mas a *Feature C* não se relaciona com nenhuma delas.

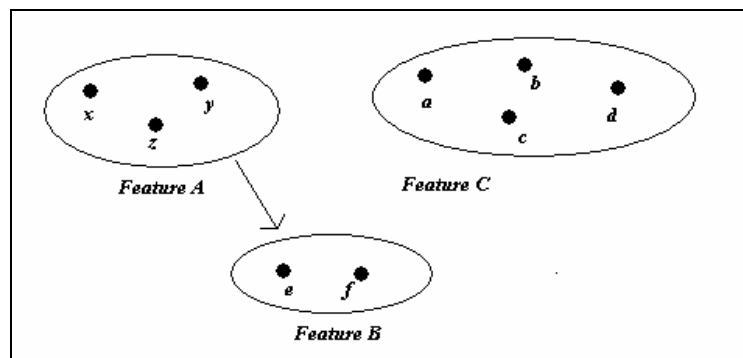


Figura 2.2 – Conjunto de *Features*

Features são tipicamente aplicações reativas, isto é, são caracterizadas pela interação com o ambiente em que elas estão inseridas. Tal reatividade é caracterizada pela interação da aplicação com o ambiente através de entradas e de suas saídas.

2.3 Diagrama de Seqüência UML

Um diagrama de seqüência descreve a seqüência de ações que ocorre em um sistema. Ele captura a invocação de métodos de cada objeto, e também a ordem em que ocorre. Isto torna diagramas de seqüência uma ferramenta muito usual para representar facilmente o comportamento dinâmico de um sistema, isto é, o comportamento dos vários objetos relacionados. Diagramas de seqüência são vistos como uma boa representação de requisito, uma vez que eles mostram o comportamento do sistema e a interface com outros subsistemas.

Um diagrama de seqüência consiste de:

- **Objeto:** em um diagrama de seqüência temos seqüências de interações entre os diferentes objetos. Um objeto é o principal elemento envolvido neste diagrama e é representado por um retângulo (Figura 2.3 (a));
- **Mensagem:** as interações entre diferentes objetos em um diagrama de seqüência são chamadas de mensagens. Uma mensagem é representada por uma seta dirigida e a representação da seta difere dependendo do tipo de mensagem. Podemos representar mensagens simples (Figura 2.3 (b)), mensagem especial para criar (Figura 2.3 (c)) ou destruir objetos (Figura 2.3 (d)), e mensagens de resposta (Figura 2.3 (e)).

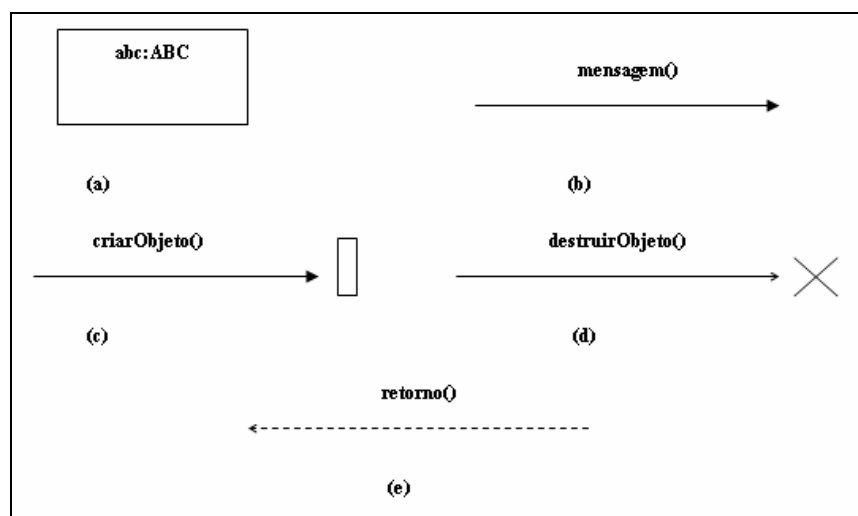
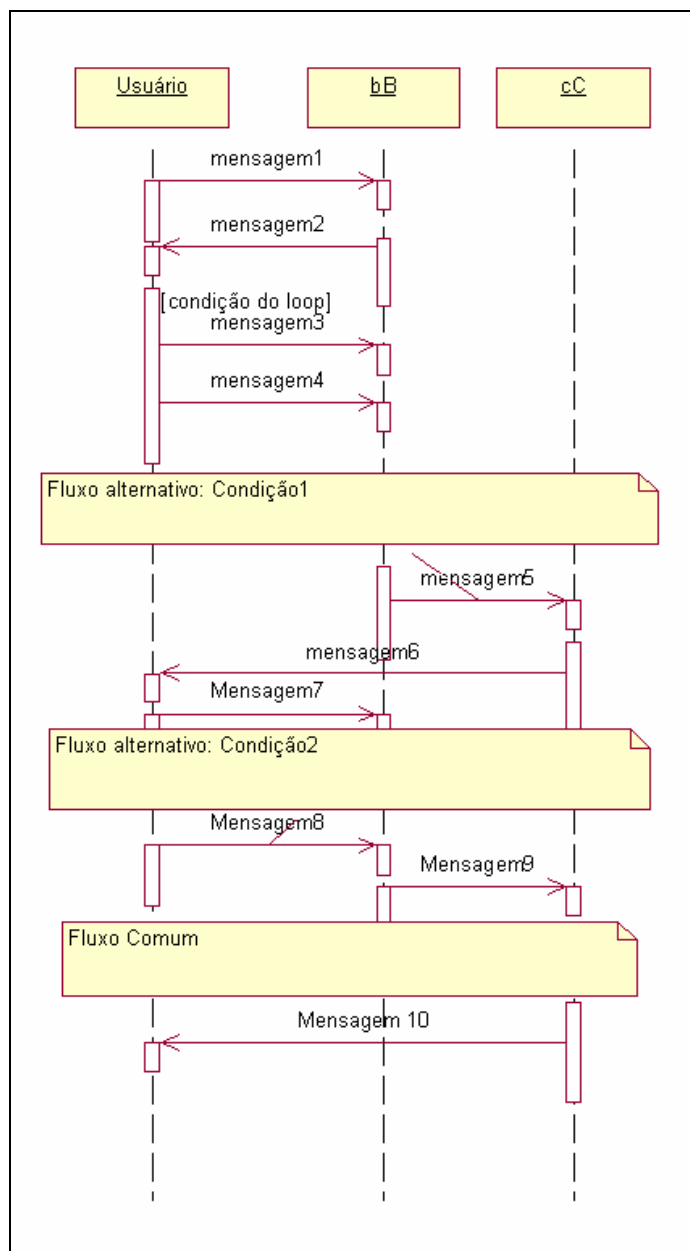


Figura 2.3 - Elementos do diagrama de seqüência

A Figura 2.4 mostra um formato geral do diagrama de seqüência que consideraremos neste trabalho. Um dos objetos do diagrama de seqüência deve representar o Usuário que interage com a aplicação. Todas as mensagens são representadas como mensagens simples. Fluxos alternativos são representados como uma anotação, e possuem uma condição associada. Neste caso, temos dois fluxos alternativos, cada um com a condição associada. A notação para *loops* pode ser vista também na Figura 2.4. Cada *loop* a ser representado tem uma condição de *loop* que é representado como uma guarda no diagrama. No caso do diagrama geral da Figura 2.4, temos um loop representado.

Na Figura 2.5, pode ser visto um diagrama de seqüência que representa como uma aplicação de um celular se comporta quando tem na sua caixa de entrada de mensagem, no mínimo uma mensagem que contenha um ou mais números de telefone embutidos, e o usuário decide armazenar o número do telefone que está embutido na mensagem.

O cenário se inicia quando um usuário vai para a caixa de mensagem. Quando esta ação é executada, o objeto *MensagemAp* popula a caixa de entrada com as mensagens existentes, para que o usuário possa visualizar. A partir daí, o usuário seleciona a mensagem que possui o(s) número(s) do(s) telefone(s) que ele quer armazenar. A partir deste ponto, o cenário pode seguir dois fluxos distintos, cuja condição é a quantidade de números embutidos na mensagem, um ou mais de um número de telefone embutido.

**Figura 2.4 – Diagrama de Seqüência padrão**

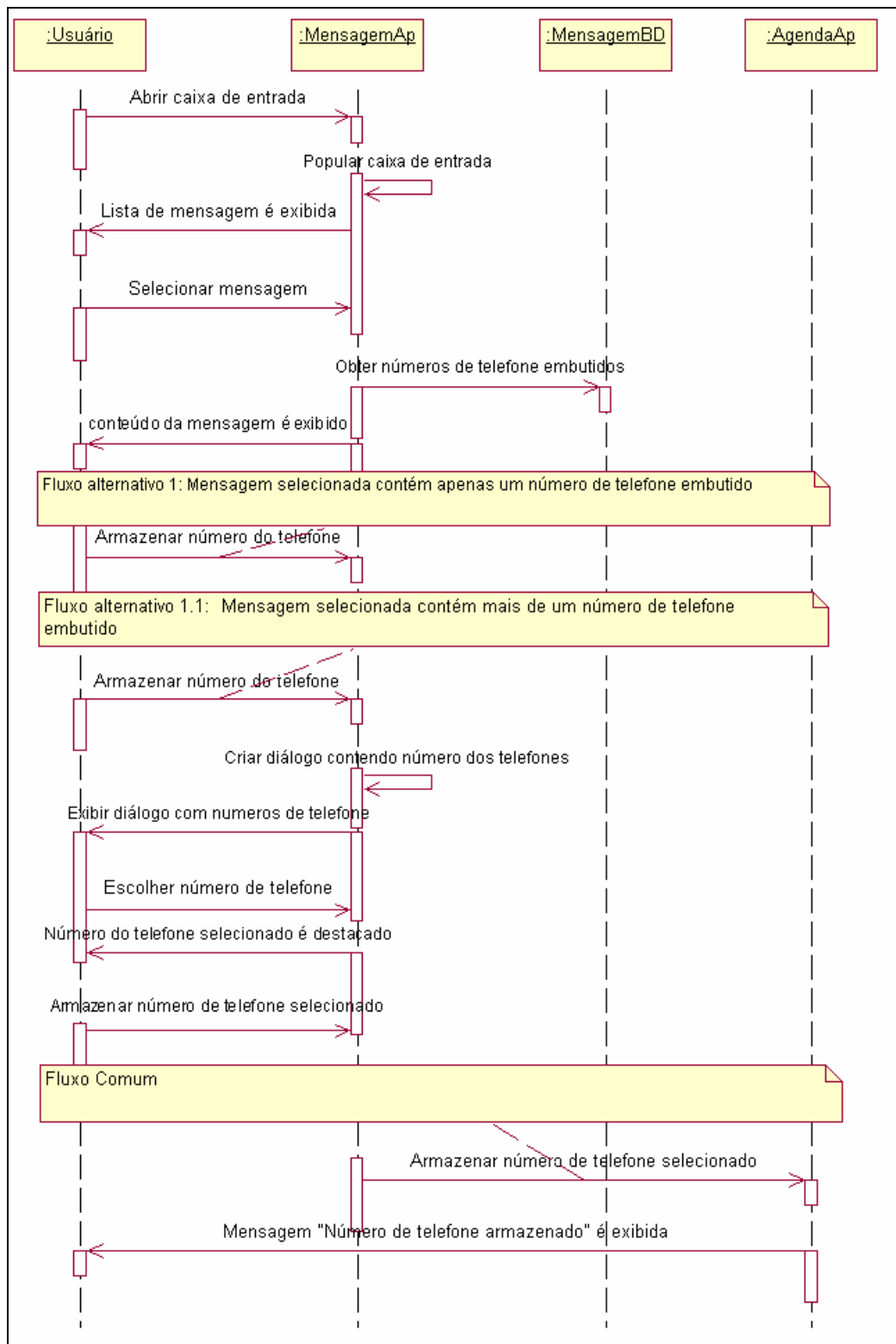


Figura 2.5 - Diagrama de seqüência que representa o comportamento de uma aplicação de celular quando o usuário quer armazenar um número de telefone que está embutido na mensagem

Como bem exposto em [Bin99], existem alguns problemas de interpretação em um diagrama de seqüência quando temos que representar controle de fluxo tais como:

- **Iteração:** a notação para iteração não é clara e isto pode trazer problemas de interpretação, que por sua vez pode impactar na tarefa de extração de casos de teste visto que será difícil decidir quando todos os caminhos possíveis forma cobertos. Na Figura 2.6 pode ser visto um exemplo de iteração em diagrama de seqüência. Esta parte do diagrama de seqüência que é mostrado, revela o comportamento que a aplicação de *chat* deve ter quando um usuário tentar efetuar *login*. Essa operação só poderá se repetir três vezes caso o usuário digite o seu nome de usuário ou a sua senha incorretamente. Veja que a interação é para as três mensagens, e isto pode não estar claro;

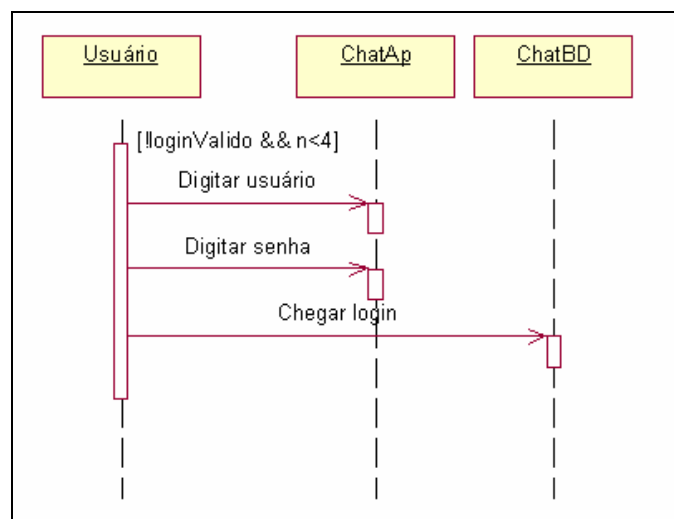


Figura 2.6 - Diagrama de Seqüência - Iteração

- **Fluxo alternativo:** a notação para fluxos alternativos pode também trazer problemas de interpretação, uma vez que o projetista de teste pode perder a noção de onde está o fluxo no momento da extração dos casos de teste. A representação de um fluxo alternativo pode ser vista na Figura 2.7, onde temos dois fluxos alternativos, ou seja, o usuário vai seguir um deles de acordo com a condição imposta neles. Se $a > 0$, então segue o primeiro fluxo e logo após vai para o fluxo comum. Entretanto, se $a < 0$, deverá ser seguido o outro fluxo e logo depois o fluxo comum.

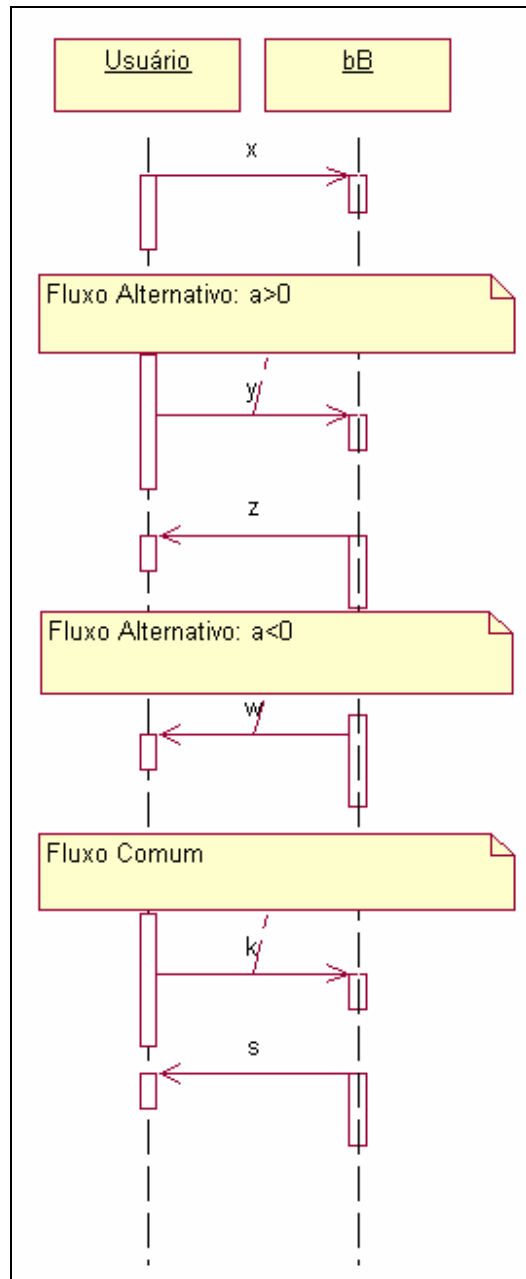


Figura 2.7 - Diagrama de seqüência – Fluxo alternativo

2.4 LTS

Em geral, um LTS (*Labeled Transition System*) provê uma descrição integral do conjunto de todos os possíveis comportamentos do sistema.

Um LTS é uma 4-tupla $S = (Q, A, T, q_0)$, onde [Vri00]:

- Q é um conjunto finito, não vazio de estados;

- A é um conjunto finito, não vazio de rótulos;
- T , relação de transição, é o subconjunto de $Q \times A \times Q$;
- q_0 é o estado inicial.

Os elementos que compõem graficamente um LTS são:

- **Estado:** representa o estado do sistema. Na Figura 2.8(a), pode ser vista a representação do estado “1” do sistema;
- **Estado inicial:** representa o estado inicial do sistema. Na Figura 2.8 (b), pode ser vista a representação do estado inicial “0” do sistema;
- **Transição rotulada:** representa uma ação que ocorre e altera o estado do sistema. Na Figura 2.8 (c), pode ser vista uma transição que possui rótulo “a”, onde “a” representa a ação que ocorre.

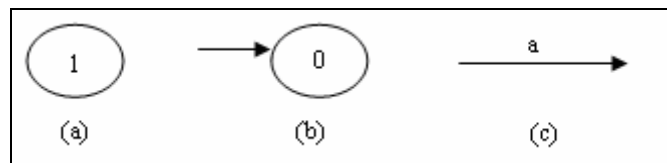


Figura 2.8 - Elementos do LTS

Na Figura 2.9, pode ser visto um LTS que representa como uma aplicação de um celular se comporta quando o usuário entra na caixa de entrada ou de saída de mensagens.

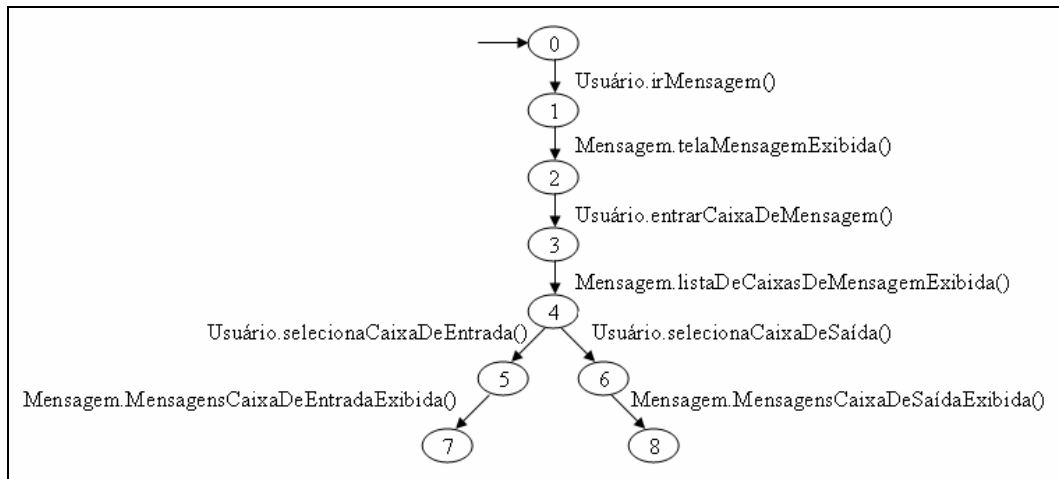


Figura 2.9 – LTS que representa o comportamento do sistema quando o usuário, através da aplicação mensagem, tenta visualizar a caixa de entrada e saída

2.4.1 LTS Anotado

Um LTS anotado é um LTS que segue as características descritas na seção anterior e que possui nas transições não somente ações, conforme descrito anteriormente, como também anotações. Tais anotações são inseridas no LTS com uma finalidade específica. Como no nosso caso, estamos interessados na geração de casos de teste funcional, teremos algumas transições do LTS com anotações inerentes a tal atividade. Na Figura 2.10 pode ser visto um LTS anotado com informações para geração de casos de teste. Neste caso, as informações inerentes ao processo de teste são: Passos, Resultados Esperados e Condições Iniciais. Estas informações inseridas servem como delimitadores, ou seja, quando houver transições com os rótulos Passos, Resultados Esperados e Condições Iniciais, isto significa que as transições que vem logo após correspondem, respectivamente, a passos, resultados esperados e condições iniciais em um caso de teste. O LTS apresentado na Figura 2.10 representa o comportamento de uma aplicação de celular quando o usuário quer armazenar um número de telefone que está embutido em uma mensagem e foi derivado do diagrama de seqüência apresentado na Figura 2.5. Os detalhes da derivação podem ser vistos no Capítulo 3.

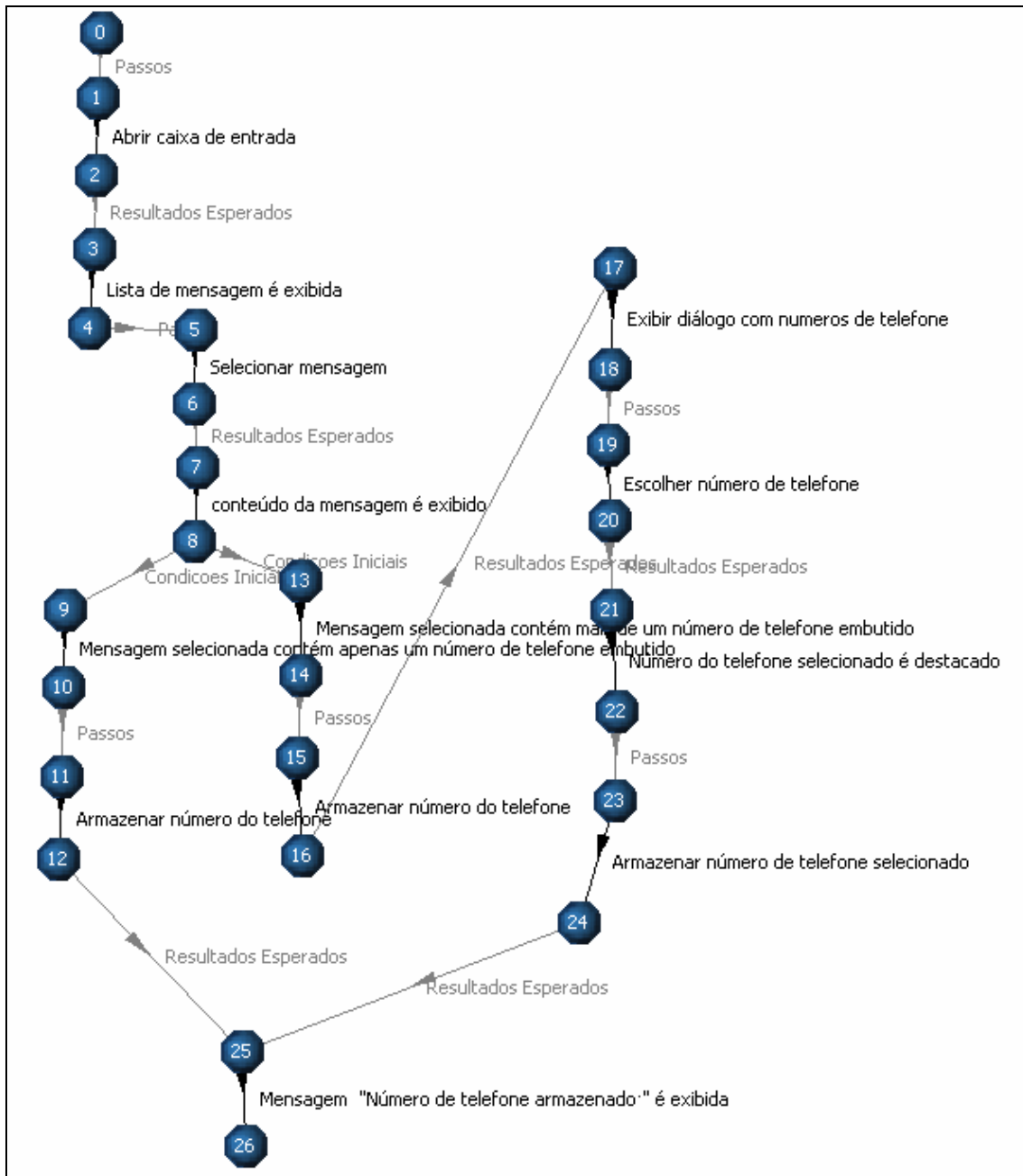


Figura 2.10 – Exemplo de LTS anotado

2.5 Considerações Finais

Neste capítulo, foi mostrada a fundamentação teórica necessária para o entendimento do trabalho proposto. Foram apresentados os principais conceitos relacionados a teste de *software*, os tipos de teste de software, abordagem de teste baseado em modelo, enumerando vantagens e desvantagens. Além disso, foi levantada a

necessidade de aplicar técnicas de seleção de casos de teste e foi mostrada uma visão geral sobre diagramas de seqüência UML e LTS.

Capítulo 3

Geração de Casos de teste funcional a partir de diagrama de seqüência UML por meio de LTS

Este Capítulo apresenta o procedimento sistemático para geração de casos de teste partindo de diagrama de seqüência. As etapas desenvolvidas para o processo de geração são apresentadas com suas justificativas. Por fim, são apresentadas considerações finais, com uma breve discussão incluindo restrições e extensões da aplicação do procedimento proposto.

O procedimento sistemático proposto com o objetivo de gerar casos de teste funcional para aplicações de celulares a partir de diagramas de seqüência UML é dividido em duas grandes tarefas, que por sua vez são divididas em subtarefas.

As duas grandes tarefas são:

- **Obtenção do modelo LTS:** A partir do diagrama de seqüências derivaremos um modelo LTS. O diagrama de seqüências considerado neste trabalho tem o formato apresentado no Capítulo 2. Tais diagramas são usados na especificação de cenários de uso da aplicação;
- **Geração dos casos de teste a partir do modelo LTS:** A partir do LTS obtido no passo anterior geraremos os casos de teste funcional.

Cada uma dessas tarefas tem procedimentos bem definidos que culminam na geração de casos de teste funcional. Tais atividades são passíveis de automação, como poderá ser visto à medida que cada uma delas for apresentada. Ressaltando que a entrada do procedimento sistemático proposto são diagramas de seqüência e a saída são casos de teste funcional. Cada caso de teste será composto de condições iniciais, passos e resultados esperados.

3.1 Obtenção do Modelo LTS

O primeiro passo do procedimento sistemático proposto é obter o modelo LTS. Temos que levar em consideração que a nossa proposta é extrair casos de teste funcional, tendo como entrada diagramas de seqüência da aplicação que se quer testar. Mas nada impede de também ser considerado a geração de casos de teste partindo de modelo LTS, já que no escopo do projeto CInBTCRD existe trabalhos que transformam requisitos em CSP [Tor06]. Do CSP obtemos o LTS [FW06].

Visto que escolhemos trabalhar com LTS e a nossa entrada é um diagrama de seqüência, então temos que mapear elementos de diagrama de seqüência em elementos de um LTS. Como nosso foco é a geração de casos de teste, o LTS será anotado, ou seja, ele terá algumas transições com anotações inerentes a atividade de geração. Neste caso, as informações inerentes ao processo de teste são: passos, resultados esperados e condições iniciais. Estas informações a serem inseridas no LTS servem como delimitadores, ou seja, quando houver transições com os rótulos passos, resultados esperados e condições iniciais, isto significa que as transições que vêm logo após correspondem, respectivamente, a passos, resultados esperados e condições iniciais em um caso de teste.

A nossa estratégia de mapeamento é baseada em um procedimento geral para gerar grafos de fluxo apresentado em [Bin99], o padrão *Round-trip Scenario Test*. Este tem como entrada um diagrama de seqüência que é mapeado para o grafo de fluxo e a partir do grafo de fluxo são gerados os casos de teste.

Algumas observações importantes para a estratégia de mapeamento são:

- Os estados do LTS devem ser numerados em ordem crescente, iniciando com o zero (0), e são criados a medida que vão sendo criadas transições;
- O estado do LTS numerado com 0 representa o estado inicial do sistema, isto é, o estado em que o sistema se encontra para ocorrer o cenário representado no diagrama de seqüência. Como dito no Capítulo 2, como é estado inicial então ele deve ter uma “seta” diferenciando-o dos demais.

Os passos do mapeamento são os seguintes:

1. Inicialmente deve-se criar o estado inicial;
2. Para cada mensagem do diagrama de seqüência, seguindo a “linha de vida” do diagrama devem ser criadas duas transições seqüenciais. Para cada transição deve ser criado um novo estado com a numeração incrementada;
 - a. O rótulo da transição do LTS deve ser criado observando a mensagem e os objetos do diagrama de seqüência. Ele deve ser da seguinte forma:
 - i. A primeira transição deve ser rotulada com:
 1. *Passos*: qualquer mensagem exceto quando o **objeto Destino** for Usuário, ou;
 2. *Resultados Esperados*: se o **objeto Destino** for Usuário.
 - ii. A segunda transição deve ser rotulada com:
 1. **objetoOrigem.ObjetoDestino.Mensagem**:
objetoOrigem representa o nome do objeto que envia a mensagem, o **objetoDestino** representa o nome do objeto que recebe a mensagem e **mensagem** representa o conteúdo da mensagem do diagrama de seqüência.

Vale ressaltar que se existir mais de uma mensagem subsequente que seja classificada como *Passos* ou *Resultados Esperados*, então a partir da segunda mensagem deve-se criar apenas uma transição rotulada ao invés de duas. Esta corresponde à **objetoOrigem.ObjetoDestino.Mensagem**, como mostrada anteriormente.

3. Se na ordem que estiverem sendo mapeados os elementos aparecer algum *loop*, então devem ser criadas duas transições e dois novos estados com a numeração incrementada. Os rótulos devem ser:
 - a. A primeira transição deve ser rotulada com:
 - i. *Condições iniciais*;
 - b. A segunda deve ser:
 1. A condição existente no diagrama de seqüência;
 - c. Criada a transição com o rótulo do *loop*, o próximo passo é observar as mensagens que estão no *loop* e então mapeá-las em

transições no LTS, lembrando que o último estado do *loop* terá uma transição com o rótulo *Condições iniciais* que aponta para o estado depois da transição que contém a transição *Condições iniciais*.

4. Se na ordem que estiverem sendo mapeados os elementos aparecer algum fluxo alternativo, então deve ser criada uma transição com o rótulo *Condições iniciais* e um estado; A partir deste estado devem ser criados duas transições e dois novos estados com a numeração incrementada. O rótulo destas transições deve ser da seguinte forma:
 - a. O conteúdo da condição do fluxo alternativo do diagrama de seqüência.

A última transição rotulada de cada fluxo alternativo deve apontar para o mesmo estado, se houver um fluxo comum no fim, ou deve apontar para estados diferentes, se não existir mais fluxo comum entre eles (os fluxos).

Seguindo os passos mencionados acima, vamos obter o LTS derivado do diagrama de seqüência mostrado na Figura 2.6 (replicada na Figura 3.1(a)).

O primeiro passo é criar o estado inicial. Feito isto, observando o diagrama temos que o próximo elemento seguindo a “linha de vida” do diagrama de seqüência é um *loop*. Sendo assim, temos que criar uma transição com o rótulo “*Condições Iniciais*” e um novo estado e em seguida uma transição com o rótulo *!loginValido* e um novo estado (ver Figura 3.1(b)).

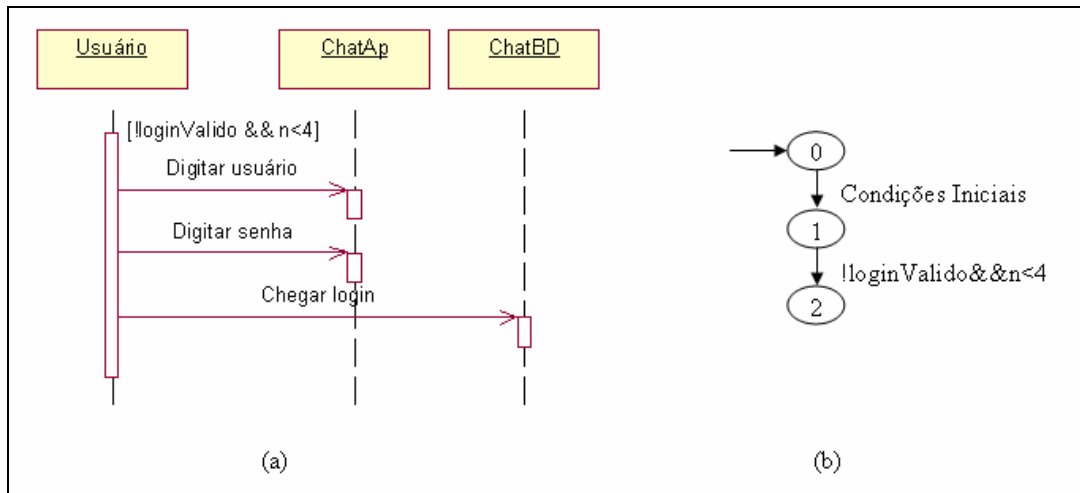


Figura 3.1 - Construção inicial do LTS derivado (Figura 3.1(b)) do diagrama de Seqüência da Figura 3.1(a)

Dentro do *loop* existem três mensagens e todas são classificadas como *Passos*. Desta forma deve ser criada para cada mensagem uma nova transição e um novo estado, exceto para a primeira, que deve ser criada uma transição anterior a ela, cujo rótulo é *Passos*. As demais transições devem ter o rótulo: *objetoOrigem.objetoDestino.mensagem*. Então teremos três transições na seguinte ordem: *Usuário.ChatAp.Digitar usuário*, *Usuário.ChatAp.Digitar Senha*, *Usuário.ChatAp.Checar login* (ver Figura 3.2).

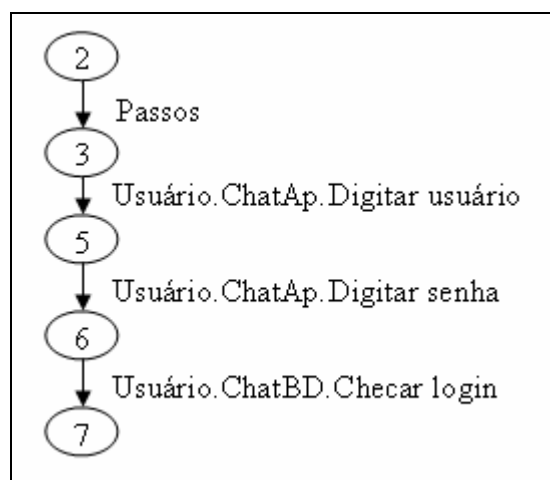


Figura 3.2 - Construção do LTS derivado do diagrama de Seqüência da Figura 3.1(a)– mensagens do loop

Como a mensagem *Checar login* é a última do *loop*, então do estado depois dela, deve ser criada uma transição que aponta para o estado anterior a mensagem inicial do diagrama de seqüência. Sendo assim a transição tem o rótulo *Condições Iniciais* (ver Figura 3.3).

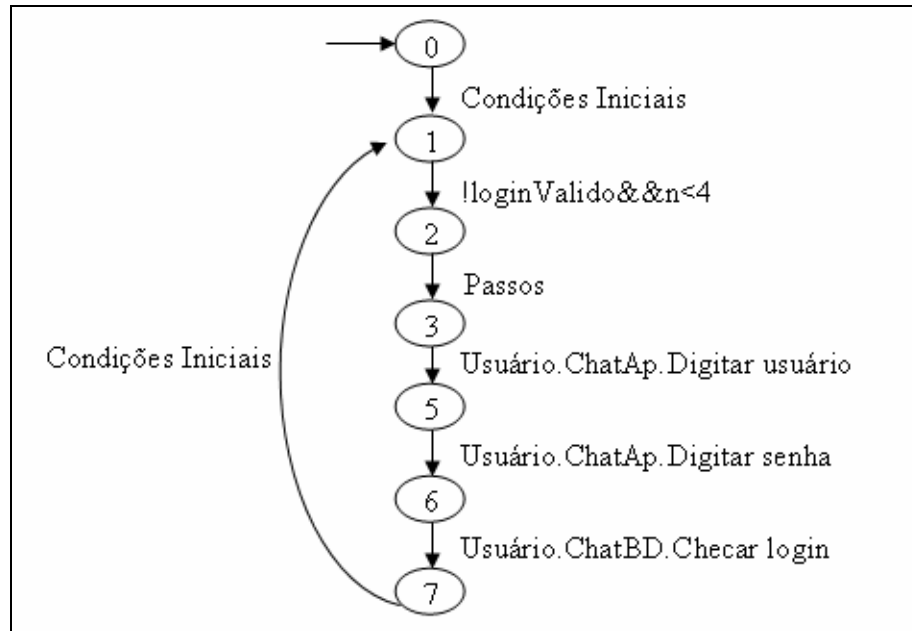


Figura 3.3 - LTS derivado do diagrama de seqüência Figura 3.1(a)

Seguindo os passos já mostrados para a construção do LTS derivado do diagrama da Figura 2.6, vamos considerar agora o diagrama de seqüência da Figura 2.7. Temos inicialmente a criação do estado inicial. Seguindo a “linha de vida” do diagrama de seqüência temos que a mensagem *x* enviada do *Usuário* para o objeto *Bb*, no qual devem ser criadas duas transições e dois novos estados. A primeira, rotulada com *Passos* e a segunda *Usuário.bB.x* (ver Figura 3.4).

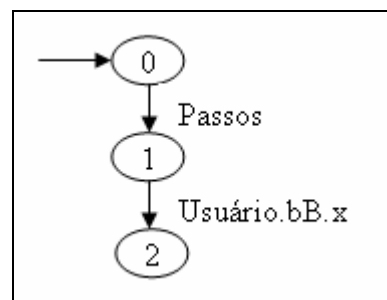


Figura 3.4 - Construção inicial do LTS derivado do diagrama de seqüência da Figura 2.7

Logo em seguida, temos um fluxo alternativo. Para o fluxo alternativo, devemos criar duas transições, a primeira com o rótulo *Condições Iniciais* e a segunda com o rótulo sendo a condição do fluxo alternativo, neste caso temos $a > 0$. Este fluxo possui duas mensagens (ver Figura 2.7):

- A mensagem y será representada por duas transições. A primeira será rotulada com *Passos*, e a segunda com $Usuário.bB.y$;
- A mensagem z será representada por duas transições. A primeira será rotulada com *Resultados Esperados*, e a segunda com $bB.Usuário.z$.

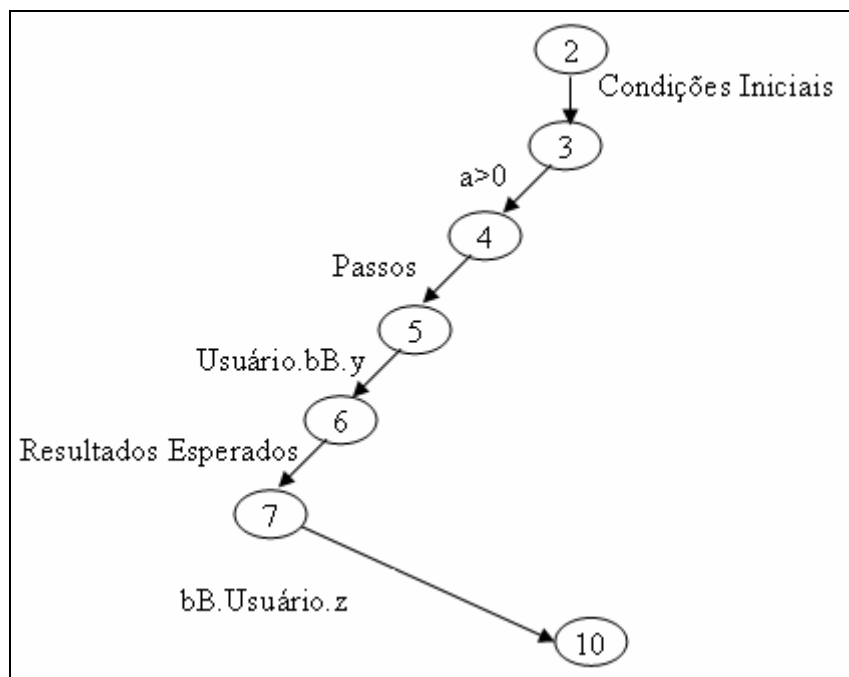


Figura 3.5 - Construção do LTS derivado do diagrama de seqüência da Figura 2.7 – Fluxo alternativo 1

Logo após existe outro fluxo alternativo, cuja transição derivada no LTS deve partir do mesmo estado do outro fluxo alternativo já mencionado, que tem o rótulo *Condições Iniciais*. Logo após deve ser criada uma transição rotulada com $a < 0$. Neste fluxo temos apenas uma mensagem, que será mapeada para duas transições rotuladas: *Resultados Esperados* e $bB.Usuário.w$, nesta ordem. Lembrando que $bB.Usuário.z$ e $bB.Usuário.w$ devem apontar para o mesmo estado já que existe um fluxo comum logo após. Veja o resultado deste passo na Figura 3.6.

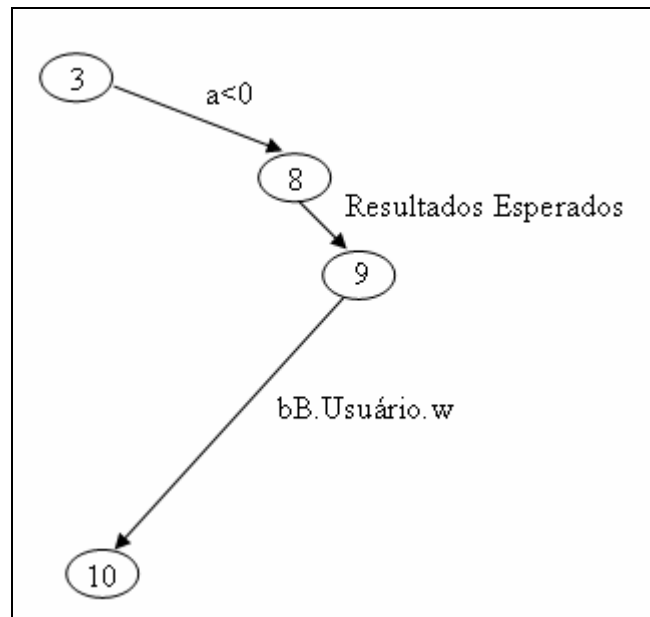


Figura 3.6 - Construção do LTS derivado do diagrama de seqüência da Figura 2.7- Fluxo alternativo 2

Seguindo o fluxo comum, temos quatro transições rotuladas, nesta ordem: *Passos*, *Usuário.bB.k*, *Resultados Esperados* e *bB.Usuário.s*. Veja Figura 3.7.

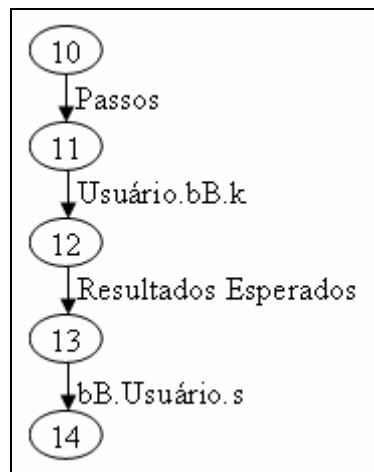


Figura 3.7 - Construção do LTS derivado do diagrama de seqüência da Figura 2.7- Fluxo Comum

Na Figura 3.8, pode ser visto o LTS derivado do diagrama de seqüência da Figura 2.7.

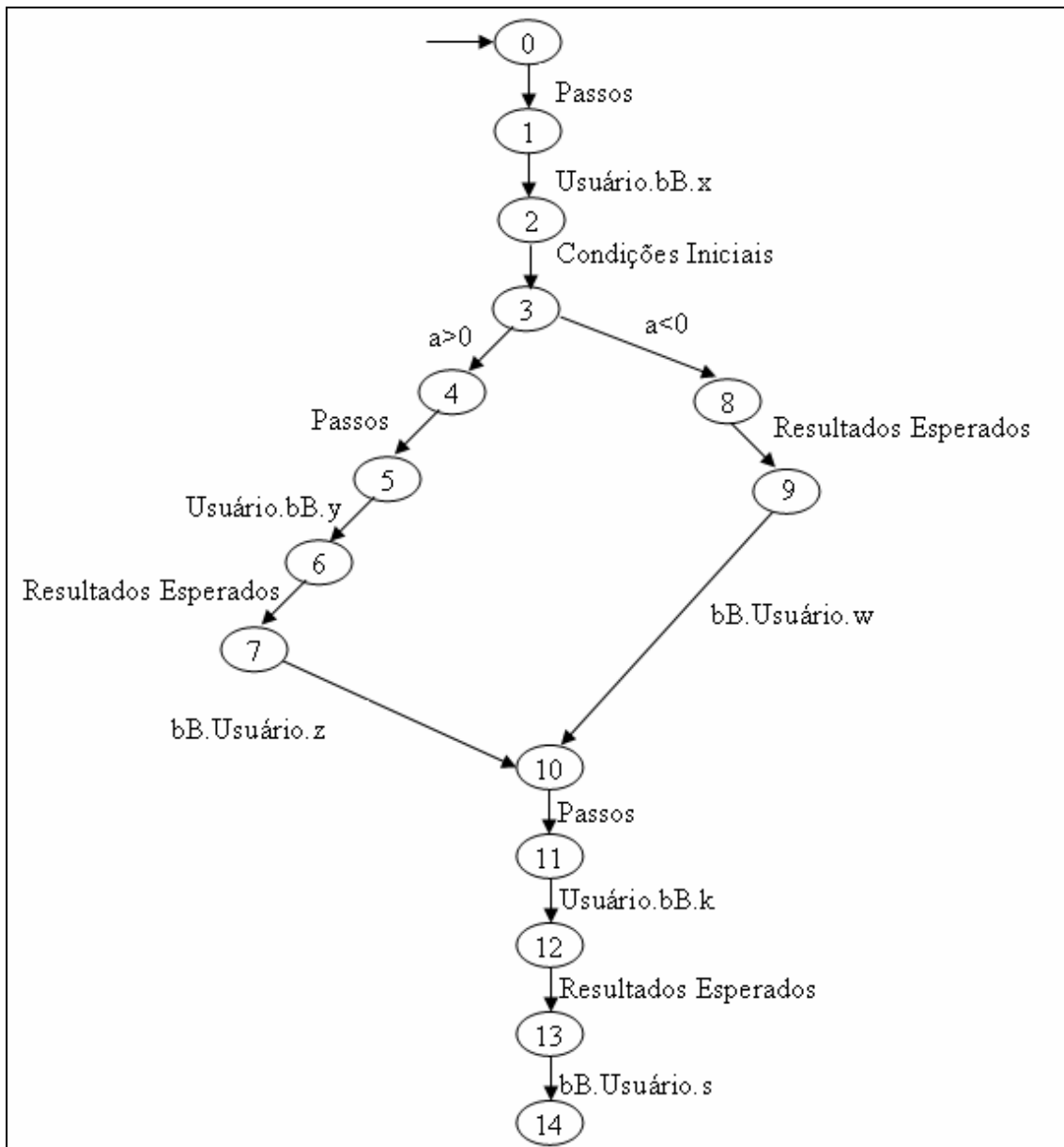


Figura 3.8 - LTS derivado do diagrama de seqüência (Figura 2.7)

3.2 Geração de Casos de Teste

Feito o mapeamento do diagrama de seqüência para LTS podemos gerar os casos de testes para a aplicação. O critério de cobertura adotado será a cobertura de caminhos, para garantir que todos os comportamentos explícitos no diagrama de seqüência possam ser testados. Um caso de teste é, portanto, um caminho no LTS, partindo do estado inicial e chegando no estado final. E os casos de testes obtidos obedecem a propriedade

de consistência (*soundness*) apresentada em [JJ04], essa propriedade é garantida pela forma com que é feito o mapeamento de diagramas de seqüência para LTS e com a forma de extração de casos de teste, que é por meio de caminhos do LTS.

Para obter todos os caminhos, vamos utilizar o método de busca em profundidade (DFS) no LTS iniciando com o estado inicial até um estado onde não tenha transições de saída. Se existir algum *loop* não condicional no modelo LTS, devemos passar por ele apenas se formar um novo caminho, isto é, se o caminho resultante tiver pelo menos uma transição diferente. Se existir algum *loop* condicional no modelo LTS, devemos passar por ele apenas uma vez.

Cada caminho obtido deve ser colocado em uma tabela de caminhos como mostrada na Tabela 3.1.

Tabela 3.1 - Tabela de caminho

Identificação do caminho	Caminho
1	
...	
N	

Como estamos interessados em teste funcional, e um diagrama de seqüência possui não somente mensagens trocadas entre um usuário e a aplicação como também entre objetos da aplicação (ações internas), nós aplicaremos um “filtro” em cima da tabela de caminhos obtida. Este servirá para eliminar informações desnecessárias para o tipo de teste que estamos nos propondo a gerar. As informações a serem descartadas correspondem a ações internas da aplicação. Neste caso, as informações que devemos manter são as trocadas entre o usuário e os objetos da aplicação. O resultado da filtragem são caminhos.

Feita a filtragem, temos que montar o caso de teste; desta forma teremos que as ações executadas pelo usuário são as que possuem *Usuário* como *objetoOrigem*, os resultados esperados da aplicação são os que possuem *Usuário* como *objetoDestino*. Já a condição inicial do teste são os rótulos das transições que vêm logo após a uma transição que tem “Condições iniciais” como rótulo. Se o rótulo “Condições iniciais” diz respeito a condição de *loop* conforme visto na Figura 3.3, então isso deve ficar

explícito no caso de teste, já que deverá ser uma tomada de decisão por parte do testador. Veja na Tabela 3.2 o caso de teste referente ao LTS da Figura 3.3.

Lembramos que ao montar o caso de teste é necessário retirar do rótulo o *objetoOrigem.objetoDestino*, já que são anotações do modelo. Então, se o rótulo for *Usuário.bB.x*, o que nos interessa é somente o *x*.

Tabela 3.2 – Caso de teste obtido do LTS da Figura 3.3

Condição Inicial:	
Condição de loop: !loginValido&&n<4	
Passos	Resultados Esperados
<i>Loop</i> Digitar Usuário Digita Senha Checar login Fim de <i>Loop</i>	

Na Figura 3.9, podemos observar a aplicação do algoritmo para extração dos casos de testes referentes ao LTS apresentado na Figura 3.8. Obtemos os dois caminhos em destaque.

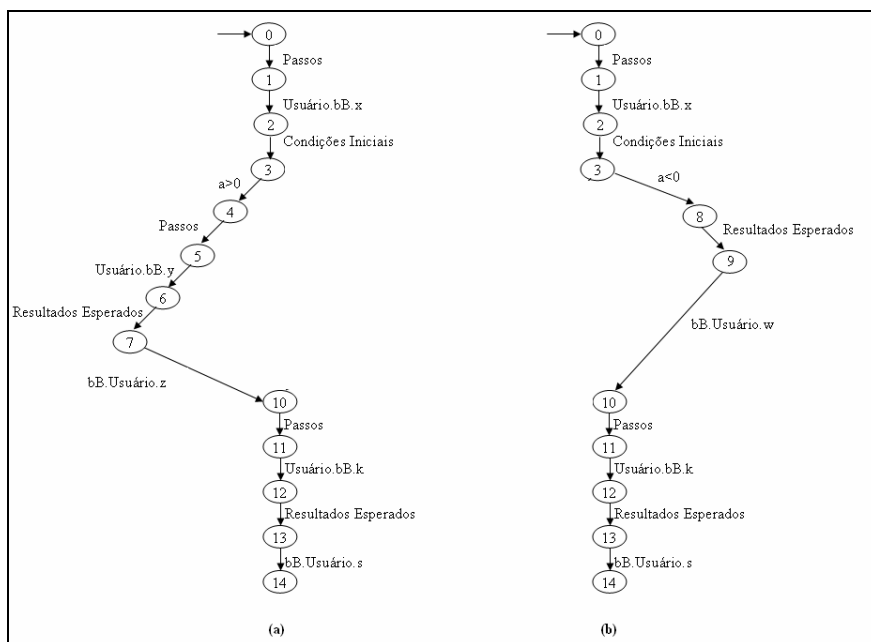


Figura 3.9 - Caminhos obtidos

A partir daí, podemos preencher a Tabela de caminhos. A Tabela de caminhos para o LTS da Figura 3.8 é a Tabela 3.3.

Tabela 3.3 – Tabela de caminho referente ao LTS (Figura 3.8Figura 3.9)

Identificação do caminho	Caminho
1	Passos, Usuário.bB.x, Condições Iniciais, $a > 0$, Passos, Usuário.bB.y, Resultados Esperados, bB.Usuário.z, Passos, Usuário.bB.k, Resultados Esperados, bB.Usuário.s
2	Passos, Usuário.bB.x, Condições Iniciais, $a < 0$, Resultados Esperados, bB.Usuário.w, Passos, Usuário.bB.k, Resultados Esperados, bB.Usuário.s

Em cada caminho é feito a filtragem, ou seja, retirada todas as mensagens trocadas entre objetos do sistema. Como para este exemplo temos apenas mensagens trocadas entre o usuário e o sistema e o sistema e o usuário, então este passo não tem nenhum resultado. A partir de agora, devemos montar os casos de teste. Conforme pode ser visto nas Tabelas 3.4 e 3.5, obtemos dois casos de teste.

Tabela 3.4 – Caso de teste 1

Condição Inicial: $a > 0$	
Passos	Resultados Esperados
x	
y	z
k	s

Tabela 3.5 - Caso de teste 2

Condição Inicial: $a < 0$	
Passos	Resultados Esperados
x	w
k	s

3.3 Considerações Finais

Neste capítulo, foi apresentado um procedimento sistemático para geração de casos de teste partindo de diagrama de seqüência, tendo como meio o LTS. Para cada passo a ser executado foi mostrada uma justificativa. A grande vantagem desde procedimento abordado é a utilização da notação UML que, por ser amplamente usada em indústrias de TI&T, pode facilitar a sua adoção, minimizando custos extras associados a treinamento e integração no processo de desenvolvimento da aplicação.

Um outro ponto importante é que, da forma como foram apresentados, podem ser gerados casos de teste funcional (aplicando a filtragem) como também casos de teste de integração, uma vez que o diagrama de seqüência traz as informações de troca de mensagens entre objetos, que geralmente são vistos como componentes da aplicação. Estes poderiam ser gerados desconsiderando o passo da filtragem.

O procedimento, apesar de ter sido apresentado como um procedimento manual foi desenvolvido tendo automação como um dos requisitos, como destacado no Capítulo 5. Com isto, justifica-se o uso de um modelo interno com o formato específico do LTS derivado.

O procedimento sistemático apresentado considera a cobertura total de caminhos, ou seja, a geração exaustiva de casos de teste do modelo, o que pode levar a geração de uma grande suíte de teste. Como nem sempre recursos e tempo para a execução de toda a suíte gerada estão disponíveis, ou até mesmo seja de interesse gerar testes apenas para comportamentos específicos, então faz-se necessário aplicar estratégias que guiem a seleção de casos de teste (veja o Capítulo 4).

Capítulo 4

Seleção de Casos de teste

Neste Capítulo, são apresentadas duas estratégias de seleção de casos de teste: uma baseada em um propósito de teste, isto é, dado um modelo, e um propósito, deverão ser selecionados casos de teste que atendam o propósito de teste; a outra baseada na similaridade de caminhos, isto é, dado um modelo e uma porcentagem de cobertura de caminhos que se deseja, deverão ser selecionados casos de teste que atendam a porcentagem seguindo critérios de similaridade.

4.1 Seleção de Casos de Teste Baseada em um Propósito de Teste

Este tipo de seleção tem como objetivo selecionar do modelo um conjunto de casos de teste que atenda a um determinado propósito. Na Figura 4.1, pode ser visto um fluxo desta atividade. Para que possamos fazer a seleção de casos de teste que obedecem a um determinado propósito, temos que ter o Modelo e o Propósito de teste. O propósito de teste serve para limitar o modelo e então é aplicada a geração. A partir destas duas entradas podemos obter os casos de teste.

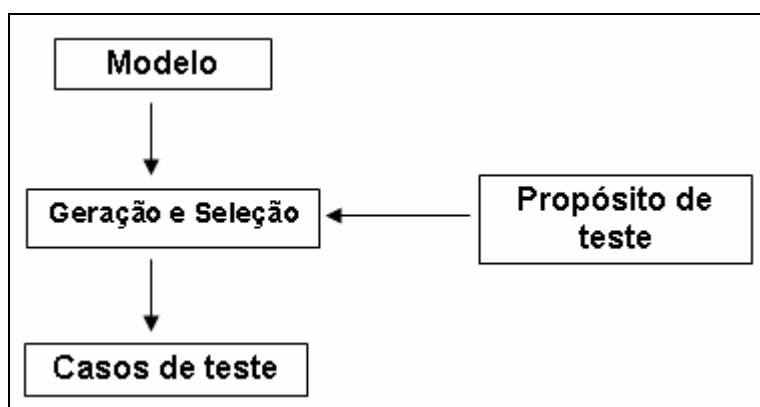


Figura 4.1 - Fluxo da seleção de casos de teste baseada em um propósito.

Para fins de automação do processo, definimos uma notação para o propósito de teste baseada na idéia do TGV mostrada em [JJ04], sendo que o TGV utiliza a notação IOLTS (*Input/Output Labeled Transition System*) como entrada para o propósito de teste e permite a junção das noções de “aceitar” e “rejeitar” no mesmo propósito. Já na notação que propomos, usamos a notação LTS e não permitimos a junção de “aceitar” e “rejeitar” no mesmo propósito.

O propósito de teste deverá ser composto por uma seqüência de rótulos do modelo (uma vez que nosso modelo é um LTS), podendo ter * (asterisco) entre os rótulos. Neste caso o * representa qualquer rótulo de transição, exceto as anotações do modelo (Passos, Resultados Esperados, Condições Iniciais), ou simplesmente nenhum rótulo. A seqüência é finalizada com Aceitar ou Rejeitar. Se no fim da seqüência aparecer:

- **Aceitar:** isto significa que se quer todos os casos de teste que satisfaçam o propósito;
- **Rejeitar:** isto quer dizer que se deseja todos os casos de teste que não satisfaçam o propósito, isto é, o que se quer é o complemento dos casos de teste que satisfazem o propósito.

Para ficar mais claro apresentamos alguns propósitos de teste para o modelo da Figura 4.2.

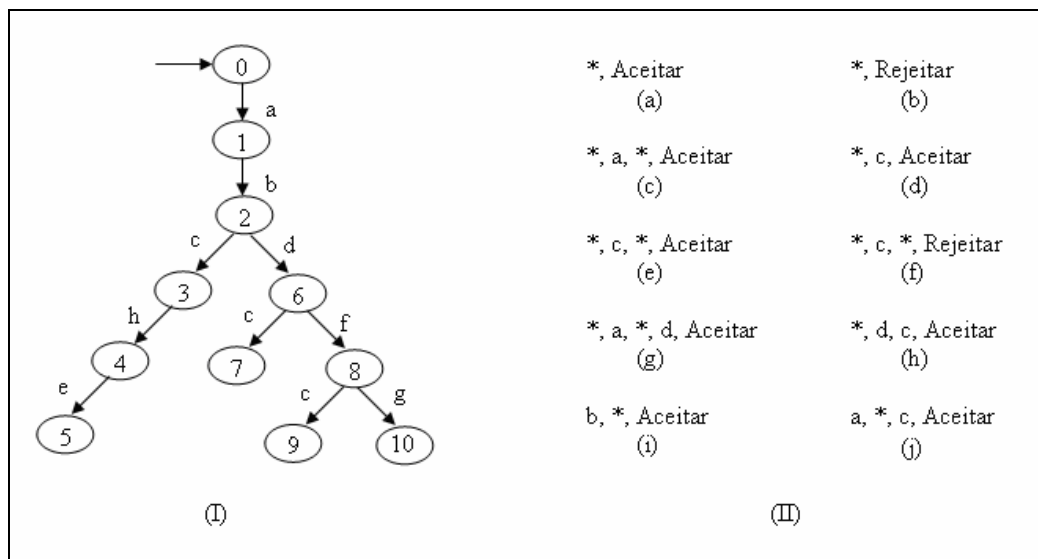


Figura 4.2 - Modelo LTS (I) e Propósitos de Teste

Exemplos de Propósitos de teste para o modelo da Figura 4.2 (I) podem ser vistos na Figura 4.2 (II).

O significado de cada um desses propósitos é o seguinte:

- *, Aceitar (Figura 4.2 (II)(a)) → Significa que se quer caminhos com qualquer transição, ou seja, todos os caminhos possíveis de serem extraídos do modelo;
- *, Rejeitar (Figura 4.2 (II) (b)) → Significa que se quer o complemento do da Figura 4.2 (II) (a). Como o objetivo do propósito da Figura 4.2 (II)(a) é o conjunto de todos os caminhos do modelo, então, o “*, Rejeitar”, tem como resultado nenhum caminho;
- *, a, *, Aceitar (Figura 4.2 (II) (c)) → Significa que se quer todos os caminhos que possuam “a”;
- *, c, Aceitar (Figura 4.2 (II) (d)) → Significa que se quer todos os caminhos que possam ser extraídos do modelo que terminem com “c”;
- *, c, *, Aceitar (Figura 4.2 (II) (e)) → Significa que se quer todos os caminhos que possuam “c”;
- *, c, *, Rejeitar (Figura 4.2 (II) (f)) → Significa que se quer todos os caminhos que não possuam “c”;
- *, a, *, d, Aceitar (Figura 4.2 (II) (g)) → Significa que se quer todos os caminhos que possuam “a” e “d”, nesta ordem, não necessariamente “a” e “d” consecutivos e terminados por “d”;
- *, d, c (Figura 4.2 (II) (h)) → Significa que se quer todos os caminhos que terminem com um “d” seguido de um “c”;
- b, *, Aceitar (Figura 4.2 (II) (i)) → Significa que se quer todos os caminhos que iniciem com “b”;
- a, *, c, Aceitar (Figura 4.2 (II) (j)) → Significa que se quer todos os caminhos que iniciem com “a” e terminem com “c”.

Dados o propósito e o modelo, o objetivo é fazer uma seleção no modelo dos caminhos que obedecem ao propósito de teste. Com isso, temos um sub-modelo que obedece o propósito de teste e a partir dele podemos gerar os casos de teste. Os sub-

modelos obtidos com os propósitos da Figura 4.2 (II) (a) e da Figura 4.2 (II) (c) são exatamente iguais ao modelo. Com o propósito da Figura 4.2 (II) (b) não obtemos nenhum sub-modelo, portanto não geraremos nenhum caso de teste.

Na Figura 4.3, podemos observar o sub-modelo obtido aplicando-se o propósito de teste da Figura 4.2 (II) (d) (*, c, Aceitar). Observem que as últimas transições do sub-modelo são rotuladas com “c”.

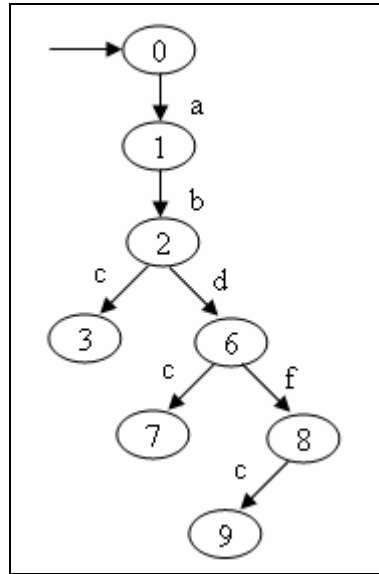


Figura 4.3 - Sub-modelo obtido do Modelo da Figura 4.2 (I) e o propósito da Figura 4.2 (II) (d)

Na Figura 4.4, podemos observar o sub-modelo obtido aplicando-se o propósito de teste da Figura 4.2 (II) (e) (*, c, * Aceitar).

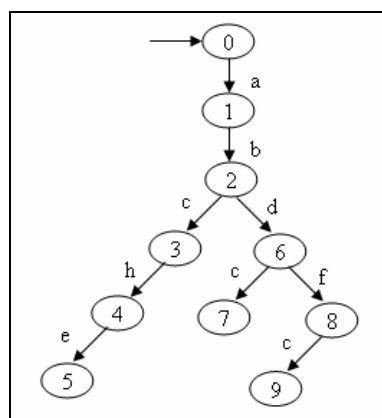


Figura 4.4 - Sub-modelo obtido do Modelo da Figura 4.2 (I) e o propósito da Figura 4.2 (II) (e)

O sub-modelo obtido com o propósito de teste da Figura 4.2 (II) (f) é complementar ao sub-modelo (visto na Figura 4.4); observe isto na Figura 4.5.

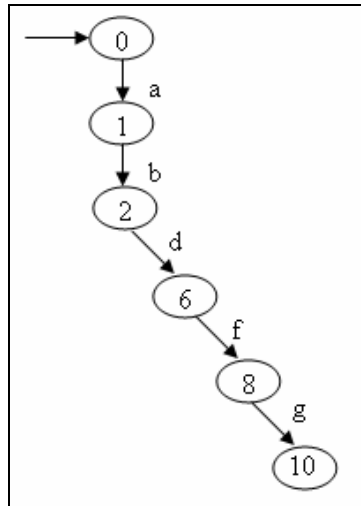


Figura 4.5 - Sub-modelo obtido do Modelo da Figura 4.2 (I) e o propósito da Figura 4.2 (II) (f)

O sub-modelo obtido com o propósito *, a, *, d, Aceitar da Figura 4.2 (II) (g), observe que os caminhos são compostos por “a” e tem um “d” no final (veja Figura 4.6).

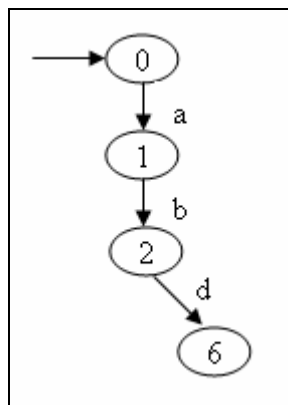


Figura 4.6 - Sub-modelo obtido do Modelo da Figura 4.2(I) e o propósito da Figura 4.2(II) (g)

Na Figura 4.7, podemos observar o sub-modelo obtido aplicando-se o propósito de teste da Figura 4.2 (II) (h). Observe que o caminho passa por “d” e em seguida “c”.

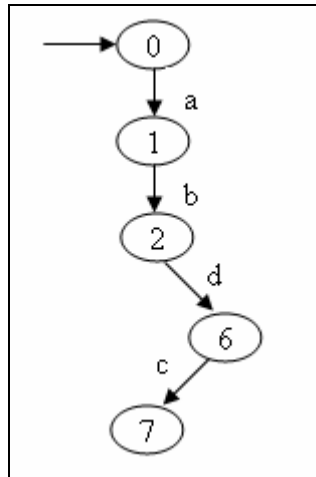


Figura 4.7 - Sub-modelo obtido do Modelo da Figura 4.2(I) e o propósito da Figura 4.2 (II) (h)

Com o propósito de teste $b, *$, Aceitar da Figura 4.2 (II) (i), não obtemos nenhum sub-modelo, uma vez que não temos caminhos que iniciam com “ b ”. Já com o propósito da Figura 4.2 (II) (j) ($a, *, c$, Aceitar), obtemos o sub-modelo mostrado na Figura 4.3. Veja que os caminhos iniciam com “ a ” e terminam com “ c ”.

Com esta estratégia de seleção conseguimos gerar apenas casos de teste que atendam a um propósito de teste, ou seja, que testam determinados comportamentos. Apesar da estratégia apresentada já reduzir o tamanho da suíte de teste, em algum momento pode ser que seja necessário diminuir ainda mais, tendo em vista recursos e tempo reservados para tal tarefa. Pensando nisto, na próxima seção será apresentada outra estratégia de seleção que pode ser usada junto com esta proposta.

4.2 Seleção de Casos de Teste Baseada na Similaridade de Caminhos

Este tipo de seleção tem como objetivo reduzir a quantidade de casos de teste de uma suíte de teste dado uma porcentagem, que é o nível de cobertura de caminhos exigida (veja Figura 4.8). Desse modo, dado um modelo é feita a geração de casos de teste e, baseado na porcentagem de cobertura exigida, é feita uma seleção dos teste gerados.

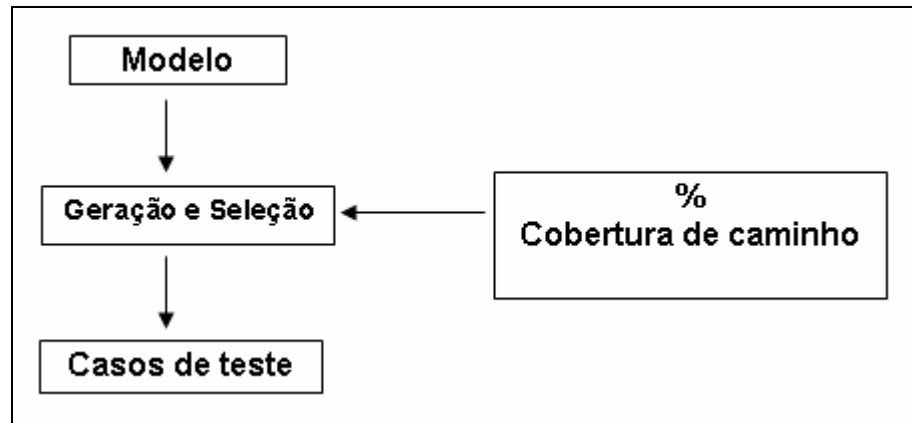


Figura 4.8 - Fluxo da seleção de casos de teste baseada na similaridade de caminhos

A redução da quantidade de casos de teste é feita observando a similaridade entre cada dois caminhos do modelo. Para isto, dado um modelo, devemos montar uma matriz de similaridade de caminhos. Esta matriz deve ser:

- $n \times n$, onde n é a quantidade de caminhos e cada n representa um caminho;
- $a_{ij} = \text{FunçãoSimilaridade}(i,j)$;
 - $\text{FunçãoSimilaridade}(i,j) \rightarrow$ A função de similaridade entre i e j é calculada observando a quantidade de transições idênticas de i e j . Uma transição é idêntica quando o estado origem, a transição rotulada e o estado destino são iguais.

Na matriz só consideraremos a triangular superior (sem a diagonal principal), uma vez que se fôssemos preencher a matriz completa, a triangular inferior seria igual a superior, isto é, a função de similaridade para os caminhos i e j é o mesmo para os caminhos j e i .

Estando montada a matriz de similaridade, devemos começar eliminando os valores mais altos. Vale ressaltar que se o valor mais alto for a_{xy} , então significa que devemos eliminar um dos caminhos x ou y .

Neste momento de escolha devemos observar qual dos dois caminhos é menor (x ou y), ou seja, o que possui o menor número de transições, visto que pode significar um menor número de funcionalidades e, desta forma eliminá-lo. Se o número de transições do caminho x é igual ao número de transições do caminho y , então se deve escolher um

dos dois aleatoriamente e eliminar. Eliminar o x, por exemplo, significa eliminar a linha x e a coluna x. Apresentamos a seguir um exemplo, para ficar mais claro esta técnica.

Na Figura 4.9, é apresentado um modelo, com o qual trabalharemos. Suponha que só queremos 25% da cobertura de caminho.

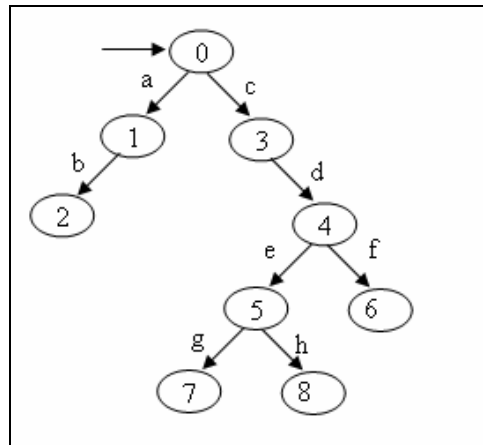


Figura 4.9 - Modelo LTS

Dado o modelo (Figura 4.9), com a técnica de geração apresentada na Seção 3, obtemos 4 caminhos (veja Figura 4.10a-d).

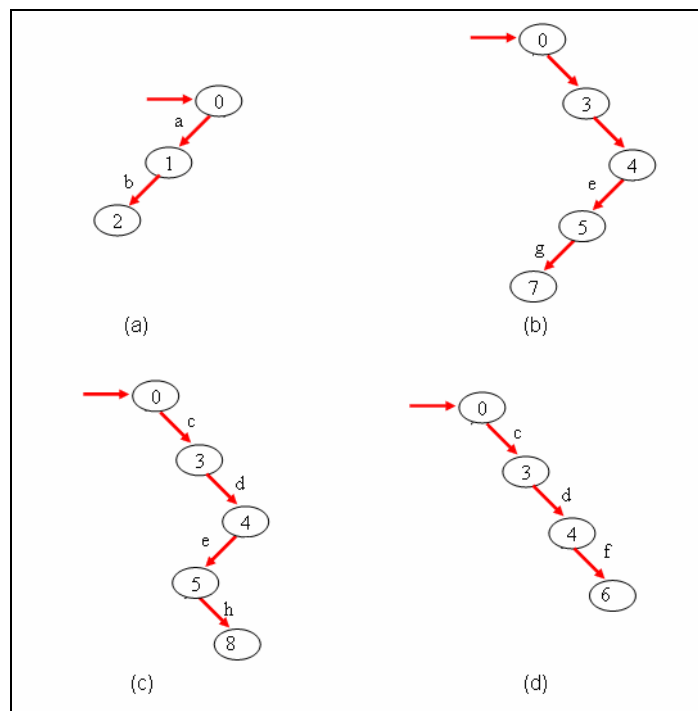


Figura 4.10 - Caminhos referentes ao modelo da Figura 4.9

Vamos montar a matriz de similaridade. Como já mencionado, temos 4 caminhos (a , b , c e d), logo teremos uma matriz 4x4; veja na Figura 4.11.

	a	b	c	d
a		0	0	0
b			3	2
c				2
d				

Figura 4.11 - Matriz de similaridade referente ao modelo da Figura 4.10

Vamos calcular também o tamanho de cada um dos caminhos:

- $|a| = 2$;
- $|b| = 4$;
- $|c| = 4$;
- $|d| = 3$.

Dado que a porcentagem de cobertura de caminhos é 25% e temos 4 caminhos, então devemos eliminar 3 dos 4 caminhos. A primeira eliminação é a do maior valor da matriz. Este está na linha b, coluna c. Logo, teremos que eliminar um dos dois caminhos. O primeiro critério de eliminação é o tamanho dos caminhos, mas podemos observar que $|b| = |c| = 4$, logo, devemos escolher um dos dois aleatoriamente. Vamos eliminar então b. Veja a matriz (Figura 4.12) depois da eliminação.

	a	c	d
a		0	0
c			2
d			

Figura 4.12 - Matriz de similaridade após primeira eliminação

A próxima eliminação está entre os caminhos c e d, como o $|c| > |d|$ então vemos eliminar o caminho d. A matriz resultante pode ser vista na Figura 4.13.

	a	c
a		0
c		

Figura 4.13 - Matriz de similaridade após segunda eliminação

Agora como temos que eliminar mais um caminho, tendo apenas dois para escolha, devemos comparar o tamanho dos caminhos. Sabendo que $|a| < |c|$, então devemos eliminar a.

Neste caso, são eliminados 3 caminhos: a, b e d e apenas um deles não é eliminado, já que a porcentagem de cobertura de caminhos exigida era de 25%.

4.3 Considerações Finais

Neste capítulo foram apresentadas duas técnicas de seleção de casos de teste. A seleção de casos de teste baseada em propósito de teste tende a ser uma seleção mais inteligente que a seleção baseada na similaridade de caminhos, uma vez que a primeira faz eliminações semânticas, já a última faz eliminações sintático-estruturais.

Tanto a seleção de casos de teste baseada em propósito de teste quanto a baseada na similaridade de caminhos, da forma como foram apresentadas podem ser automatizadas e combinadas. Desta forma podemos a partir de um modelo, gerar casos de teste baseados em um determinado propósito de teste e requerer uma determinada porcentagem de cobertura.

Capítulo 5

Ferramenta LTS-BT

O objetivo deste capítulo é apresentar a ferramenta LTS-BT. Esta foi desenvolvida para dar suporte à geração automática de casos de teste apresentada no Capítulo 3 e as estratégias de seleção de casos de teste apresentadas no Capítulo 4. Inicialmente são mostradas uma visão geral da ferramenta, destacando as características para as quais o projeto da ferramenta foi dirigido. Logo após, será apresentada uma visão geral do projeto da ferramenta.

5.1 Visão Geral

A ferramenta LTS-BT (*Labeled Transition System-Based Testing*) foi desenvolvida para dá suporte ferramental ao procedimento sistemático proposto no Capítulo 3 e as duas estratégias de seleção de casos de teste apresentadas no Capítulo 4. Assim sendo, ela tem como objetivo gerar casos de teste a partir de diagramas de seqüência UML, como também selecionar um subconjunto de casos de teste. Como dito no Capítulo 1, este trabalho está inserido no CInBTCRD, que utiliza diagramas de seqüência projetados nas ferramentas IBM Rational Rose¹ e IBM Rational Rose Real Time (Rose RT)².

Entre as características gerais da ferramenta, podemos destacar:

- **Integração com as ferramentas de modelagem UML:** Rational Rose e Rose RT;
- **Geração de casos de teste partindo de LTS:** os casos de teste podem ser gerados diretamente de um modelo LTS, sem necessariamente a fonte ser diagramas de seqüência;
- **Disponibilização dos modelos:** uma vez feita a transformação, os modelos resultantes ficam disponíveis para possível utilização posterior;

¹ <http://www.rational.com>

² <http://www.rational.com>

- **Disponibilização dos testes gerados:** uma vez gerados os casos de testes, estes ficam disponíveis para utilização posterior.

Foram considerados dois requisitos não-funcionais:

- **Interoperabilidade:** projetamos a ferramenta de forma que ela fosse independente de plataforma, para isto utilizamos a linguagem de programação JAVA.
- **Independência:** Em uma ferramenta de automação, o número de interações humanas necessárias deve ser minimizado.

Na Figura 5.1, é apresentada uma visão geral da ferramenta. Como proposto nos Capítulos 3 e 4, foi implementada a geração de casos de teste partindo de diagramas de Seqüência UML, isto inclui:

- A transformação de diagramas de seqüência UML em LTS's;
- Geração dos casos de teste partindo do LTS resultante;
- Seleção de casos de teste, que pode ser:
 - baseada em um propósito de teste;
 - baseada na similaridade de caminhos, dada uma porcentagem de cobertura de caminhos que são fornecidos pelo usuário;
 - combinação das duas estratégias acima;

Como pode ser visto na Figura 5.1, a ferramenta dá suporte a quatro tipos de entrada:

- **LTS no formato TGF³ (*Trivial Graph Format*):** o TGF é uma notação para descrever um LTS textualmente. Esta notação foi escolhida por existir no contexto do projeto CInBTCRD uma proposta de geração de especificações formais a partir de linguagem natural; para maiores detalhes consultar [Tor06, FW06];

³ <http://www.yworks.com/products/yfiles/doc/developers-guide/tgf.html>

- **LTS no formato AUT⁴:** o formato *Aldebaran* (formato AUT) é, assim como o TGF, uma forma de descrever um LTS textualmente. Esta notação foi escolhida por ser uma notação bastante conhecida e de fácil manuseio. Ela é utilizada pela ferramenta TGV [JJ04];
- **Diagrama de seqüência no formato MDL:** MDL são arquivos gerados no IBM Rational Rose. Esta notação foi escolhida por ser bastante utilizada na Motorola em geral para a especificação de cenários de uso;
- **Diagrama de seqüência no formato RTMDL:** RTMDL são arquivos gerados no IBM Rational Rose RealTime. Esta notação foi escolhida por existir no contexto CInBTCRD um proposta de geração automática de diagramas de seqüência partindo de um modelo CSP [Fer06];

Como forma de unificar o modelo com o qual trabalharíamos na geração e seleção de casos de teste foi escolhido o formato AUT, já que é bastante conhecido. Desta forma, foram implementados dois *parsers*:

- TGF para AUT;
- MDL/RTMDL para AUT.

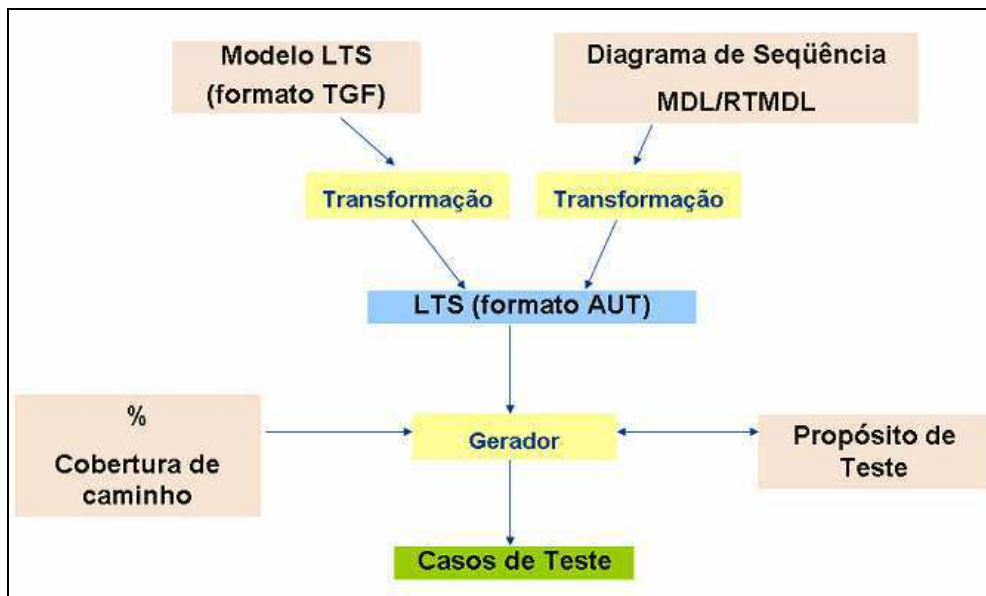


Figura 5.1 - Visão geral da ferramenta LTS-BT

⁴ <http://www.inrialpes.fr/vasy/cadp/man/aldebaran.html>

Em linhas gerais, a ferramenta funciona de acordo a entrada dada (que pode ser uma das quatro acima citadas). Se a entrada for:

- **Diagrama de seqüência no formato MDL ou RTMDL:** será feita uma transformação (seguindo os passos apresentados no Capítulo 3) e aí obteremos o LTS correspondente, já no formato AUT;
- **LTS (formato TGF):** será feita uma transformação de TGF para AUT.

Tendo o modelo LTS (no formato AUT), a geração de casos de teste pode ser iniciada. Neste passo, são requisitadas ao usuário duas entradas:

- **Propósito de teste:** Determinado pelo usuário da ferramenta. O propósito de teste irá selecionar caminhos no modelo que atendam ao propósito passado pelo usuário;
- **Percentual de cobertura de caminhos:** Assim como o propósito de teste é uma entrada determinada pelo usuário da ferramenta, que vai indicar qual o percentual de cobertura de caminhos se quer atingir.

5.2 Interface com o Usuário

A interface gráfica foi desenvolvida buscando centralizar o acesso às funcionalidades suportadas pela ferramenta LTS-BT. Na Figura 5.2, pode ser vista a tela principal da ferramenta. Nela podemos ver 9 botões, sendo os quatro primeiros das funcionalidades que foram propostas:

- **TGF – AUT** → transforma um modelo LTS na notação TGF em um na notação AUT;
- **SD – AUT** → transforma um diagrama de seqüência com extensão mdl ou rtmdl em um modelo LTS na notação AUT;
- **GP (*Generation with Purpose*)** → Executa a geração e seleção de casos de teste. O primeiro botão de geração com propósito faz a geração tendo em vista somente as transições disponíveis (Veja Seção 5.3.3). Já o segundo faz a geração tendo em vista a notação gráfica do LTS, onde o usuário pode

selecionar diretamente as transições onde ele deseja que o caso de teste passe (aceitar) ou não (rejeitar).

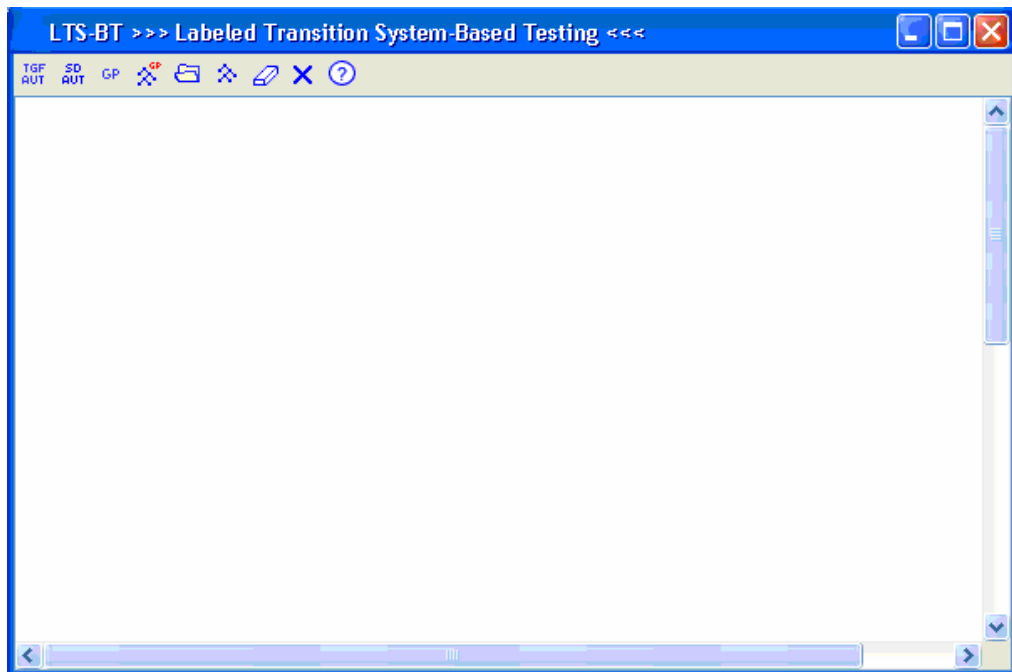


Figura 5.2 - Ferramenta LTS-BT – Tela Principal

O quinto botão serve para visualização do conteúdo dos arquivos no formato AUT ou TGF. O quinto botão serve para visualizar um modelo LTS no formato AUT ou TGF. O sexto, no formato de uma borracha, serve para limpar a tela. O sétimo, encerra a aplicação e o oitavo é a ajuda da ferramenta.

5.2.1 TGF - AUT

Na Figura 5.3, pode ser vista a tela referente à escolha do arquivo TGF que se quer colocar no formato AUT. Já na Figura 5.4, é mostrado o resultado na tela principal, isto é, o arquivo no formato AUT equivalente ao TGF apontado pelo usuário da ferramenta. Por questão de persistência e até mesmo para o usuário poder usar o arquivo AUT para outros fins, já que é bastante conhecida/utilizada essa notação, é criado um arquivo no formato AUT no mesmo diretório do TGF.

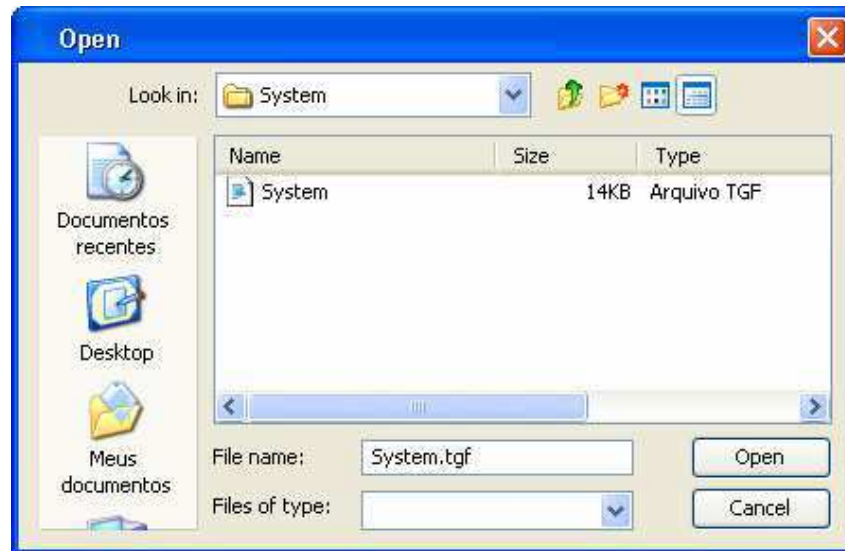


Figura 5.3 - Ferramenta LTS-BT – Tela Abrir arquivo TGF

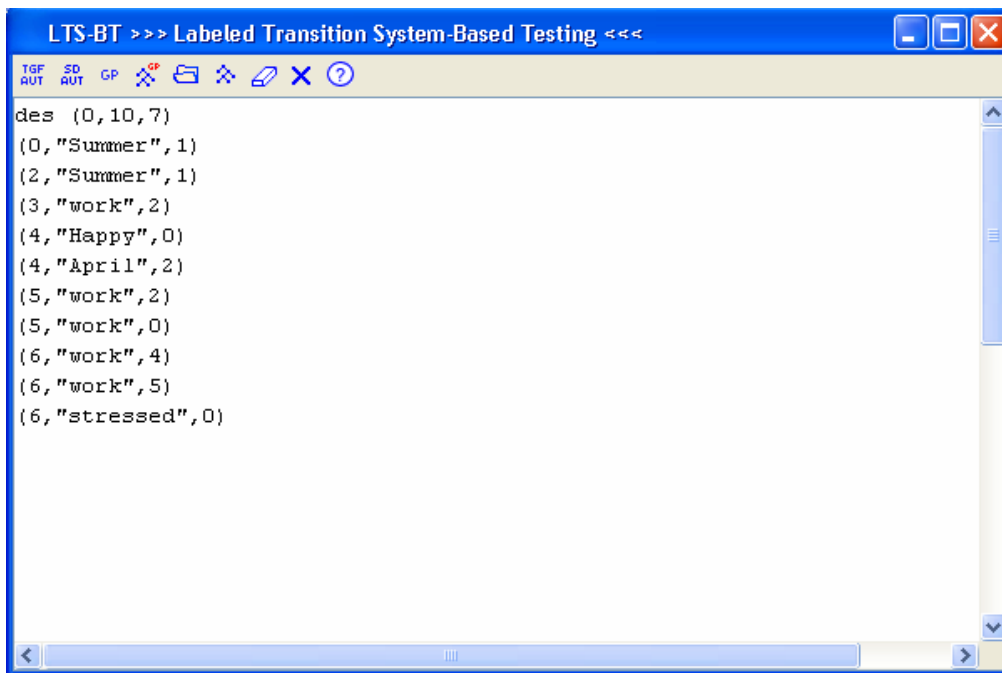


Figura 5.4 - Ferramenta LTS-BT – Tela resultado de TGF-AUT

5.2.2 SD - AUT

Na Figura 5.5, pode ser visto a tela referente à escolha do arquivo MDL/RTMDL que se quer colocar no formato AUT. Já na Figura 5.6, é mostrado o resultado na tela principal, isto é, o arquivo no formato AUT equivalente ao digrama de seqüência

apontado pelo usuário da ferramenta. Por questões já citadas anteriormente, é criado uma arquivo no formato AUT no mesmo diretório do TGF.

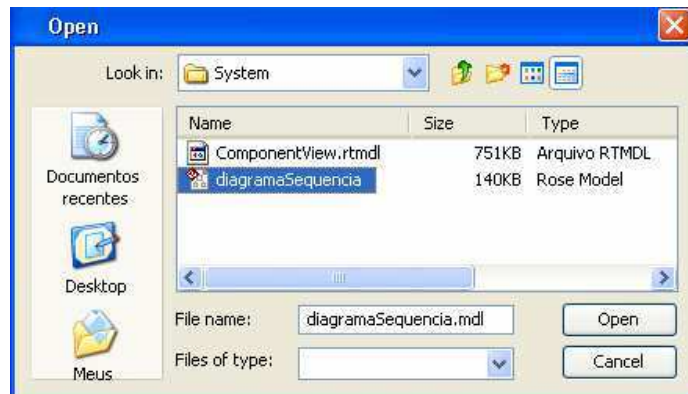


Figura 5.5 - Ferramenta LTS-BT – Tela Abrir Diagrama de Seqüência (MDL ou RTMDL)

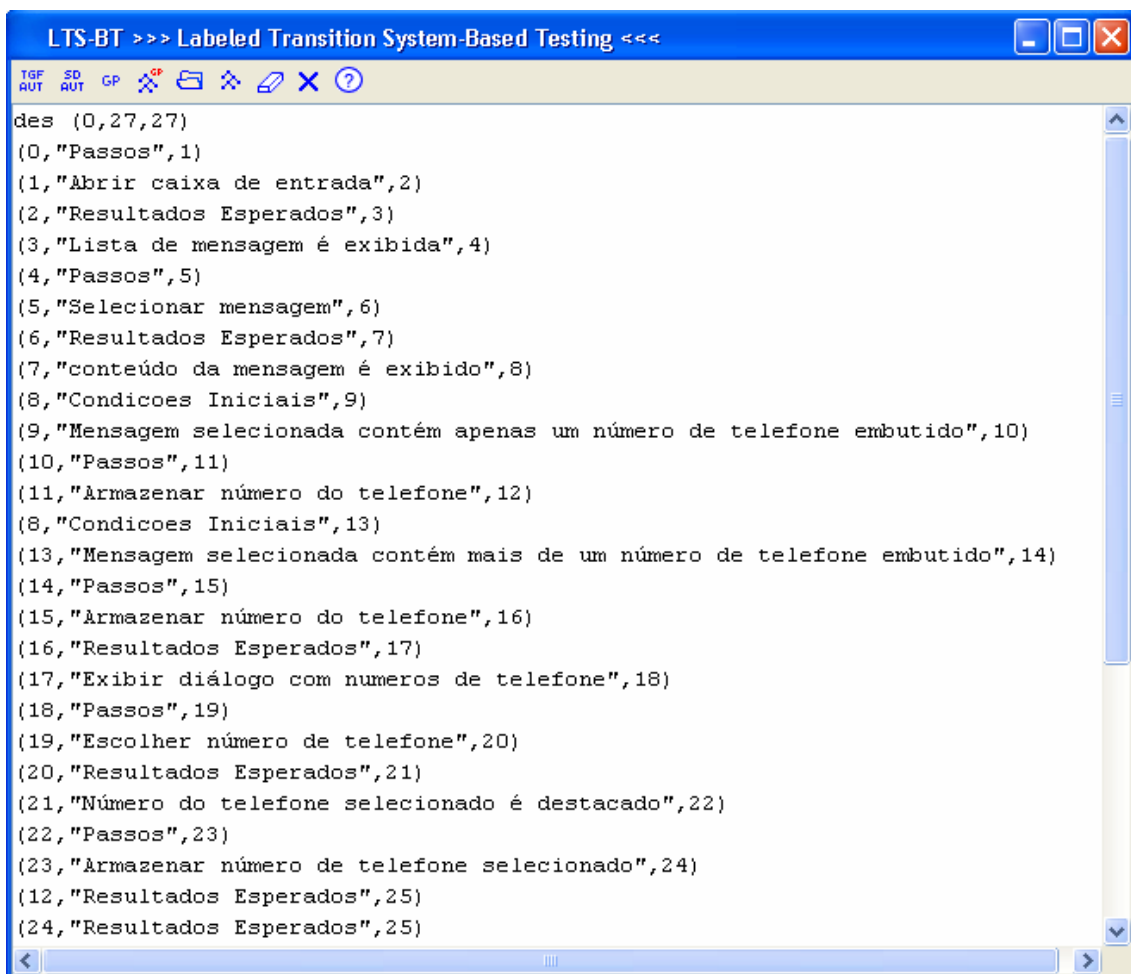


Figura 5.6 - Ferramenta LTS-BT – Tela Resultado SD-AUT

5.2.3 GP (*Generation with Purpose*)

Ao clicar no botão de geração é pedido um modelo LTS, no formato AUT ou TGF, ou um diagrama de seqüência, no formato MDL ou RTMDL (veja Figura 5.7).

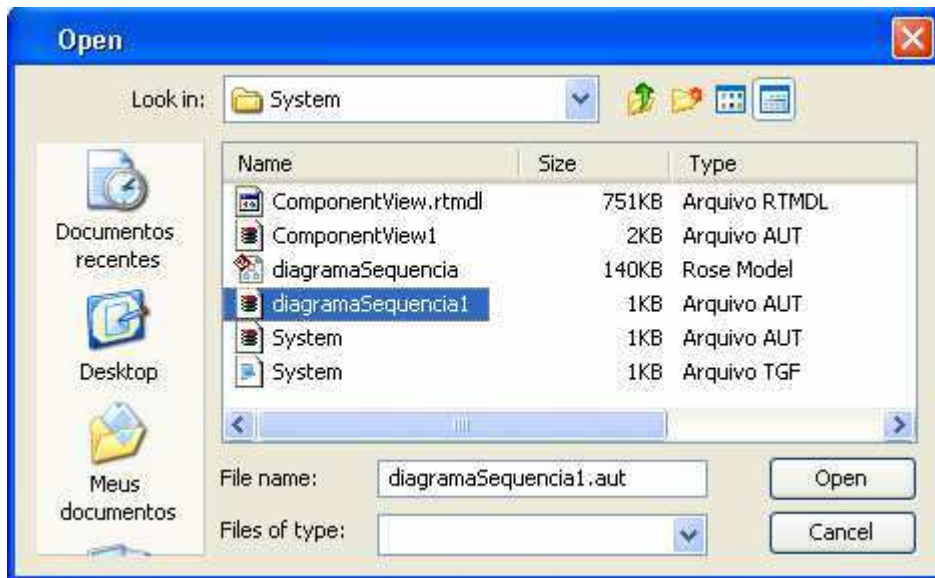


Figura 5.7 - Ferramenta LTS-BT – Tela Abrir AUT

A partir do modelo escolhido pelo usuário, teremos a geração. Para esta atividade, como já dito anteriormente, são passadas três entradas:

- Modelo;
- Propósito de teste;
- Porcentagem de cobertura de caminhos.

Procurando propor uma ferramenta amigável e menos favorável à inserção de erros pelo usuário, o propósito de teste é escolhido pelo usuário na própria interface, isto é, o usuário não é obrigado a escrever o propósito de teste em uma notação específica, a ferramenta oferece um recurso simples e se encarrega de gerar um propósito formal de acordo com a escolha do usuário. Assim, a ferramenta carrega todos os possíveis estados do sistema e ações do usuário que estão no modelo. Veja a tela de propósito de teste na Figura 5.8.

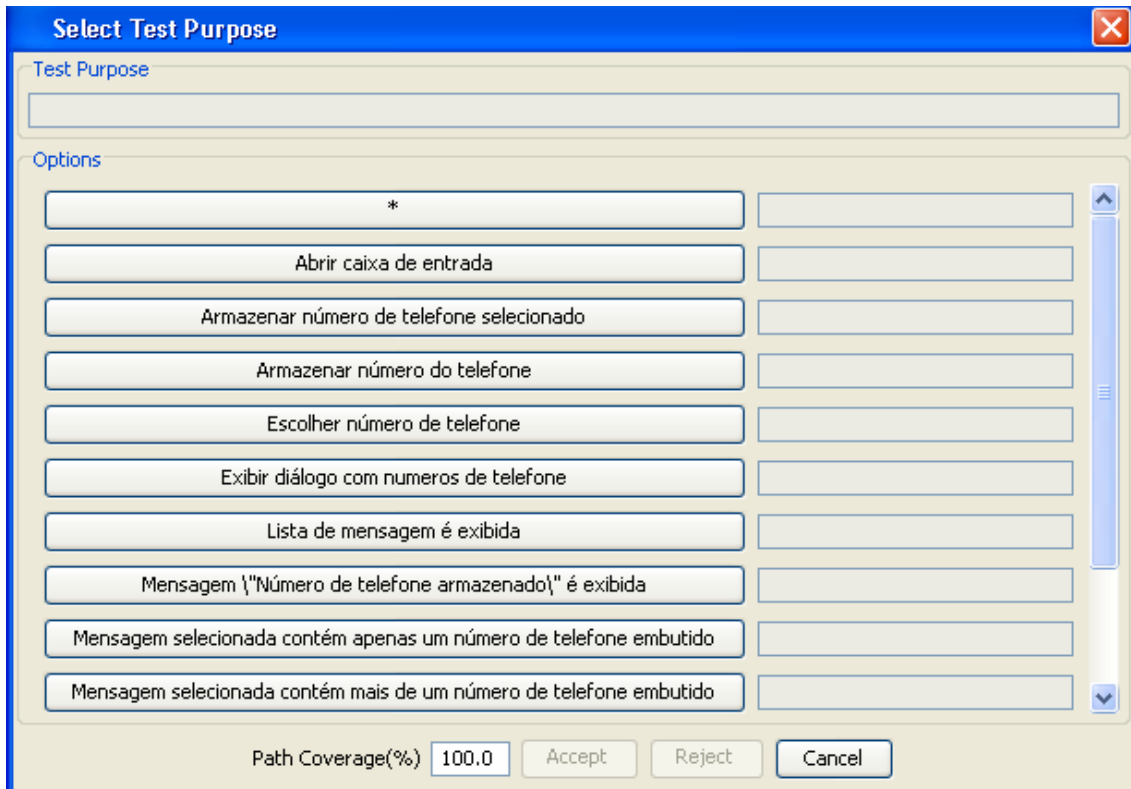


Figura 5.8 - Ferramenta LTS-BT – Tela Propósito de teste

Na Figura 5.8, o campo *Test Purpose* indica a seqüência escolhida pelo usuário. Este campo não pode ser editado. À medida que o usuário vai escolhendo o propósito, vai aparecendo a seqüência no campo. Em *Options*, temos as escolhas que podem ser feitas pelo usuário, à medida que vai clicando, vai sendo exibida a ordem (ao lado da escolha) que foi escolhida. *Path Coverage (%)* é a porcentagem de cobertura de caminhos que o usuário pode escolher, o *default* é 100%. Escolhido o propósito de teste e a porcentagem de cobertura de caminhos, o usuário pode escolher por *Accept* ou *Reject*. O *Accept* significa que o usuário deseja os caminhos que atendam o propósito de teste e o *Reject*, os caminhos que não atendam o propósito de teste escolhido.

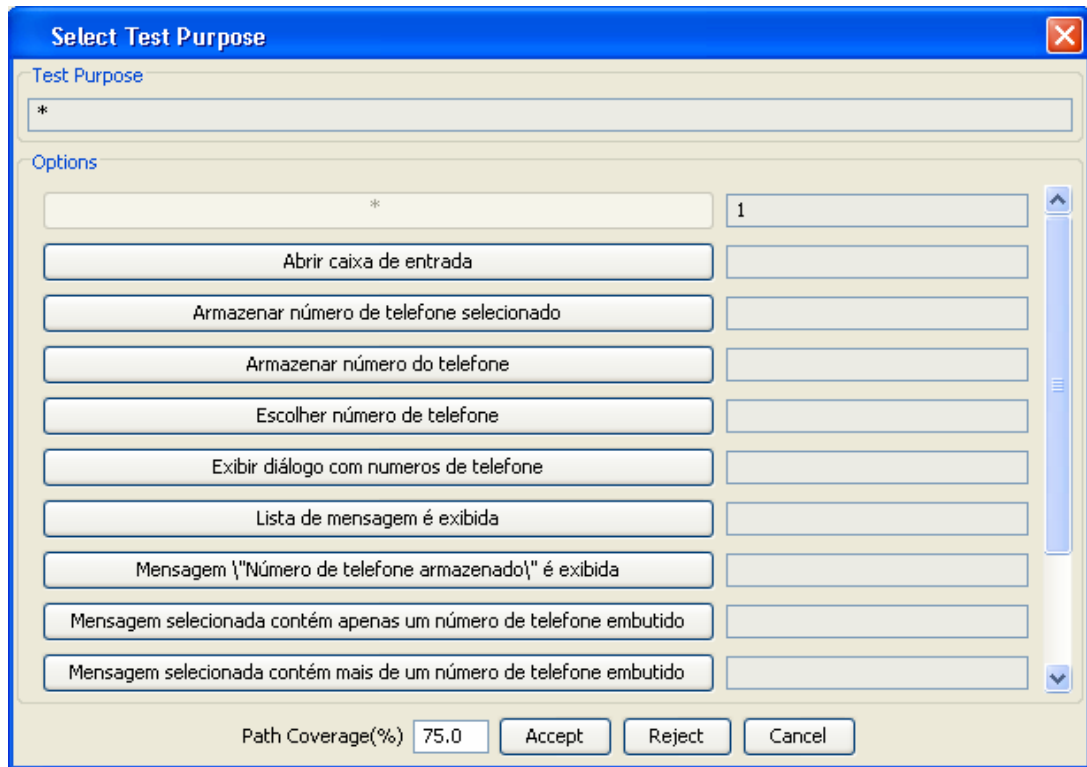


Figura 5.9 - Ferramenta LTS-BT – Tela Propósito de teste “*” e 75% de cobertura

Na Figura 5.9, temos a tela de propósito de teste já com o propósito escolhido (*) e com a porcentagem de cobertura de caminhos determinada 75%. Neste caso serão gerados todos os casos de teste no modelo e, seguindo os critérios de similaridade de caminhos apresentado no Capítulo 4, serão selecionados 75% dos possíveis. Veja a tela de resultado na Figura 5.10.

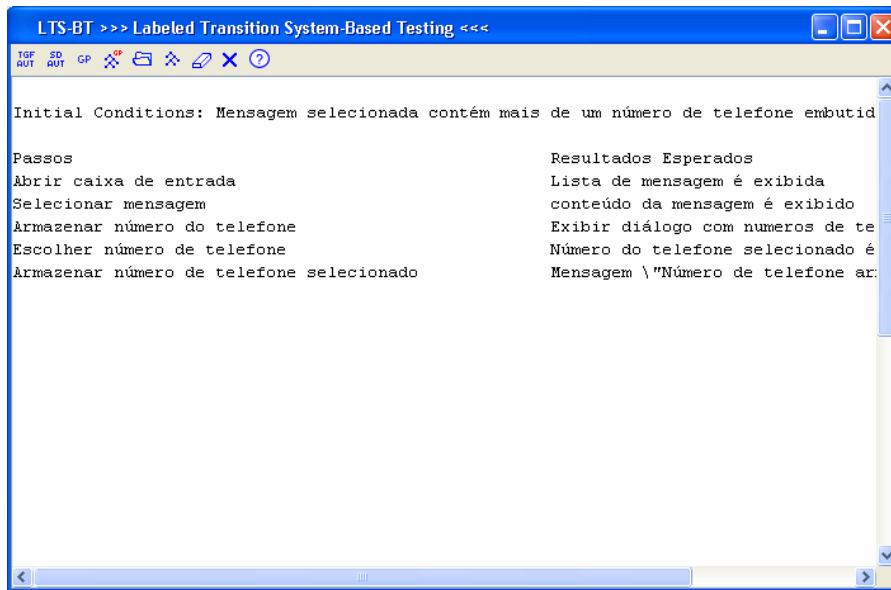


Figura 5.10 - Ferramenta LTS-BT – Tela Resultado GP

5.2.4 GP (*Generation with Purpose*) gráfico

Ao clicar no botão de geração é pedido um modelo LTS, no formato AUT ou TGF, ou um diagrama de seqüência, no formato MDL ou RTMDL (veja Figura 5.7). Na Figura 5.11, veja o propósito de teste selecionado. A diferença deste, para o apresentado anteriormente é que, neste se tem a idéia de fluxo mais clara.

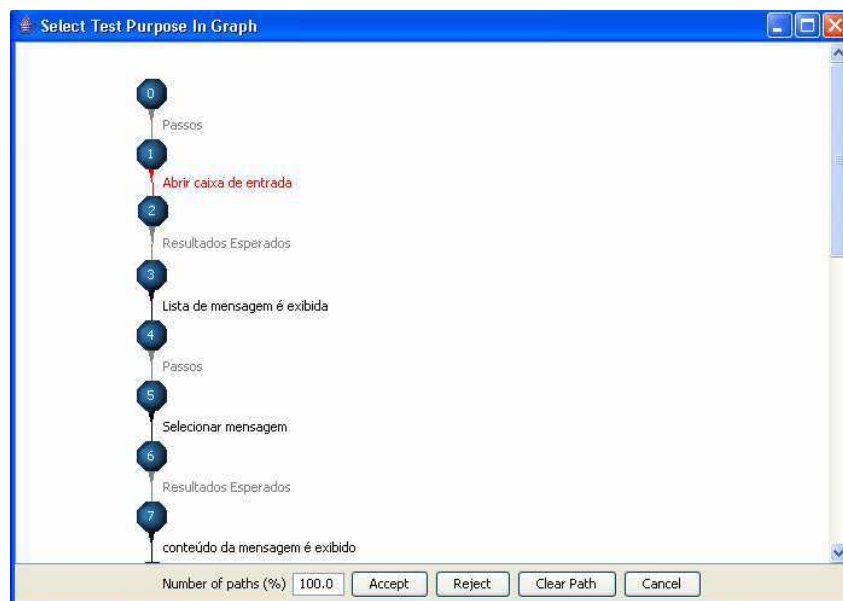


Figura 5.11 - Ferramenta LTS-BT – Tela Propósito de teste “*, Abrir caixa de entrada” e 100% de cobertura

5.2.5 Exibição Textual de LTS

Ao clicar no botão de exibição de conteúdo de arquivos, é pedido um arquivo, formato AUT ou TGF (veja Figura 5.12).

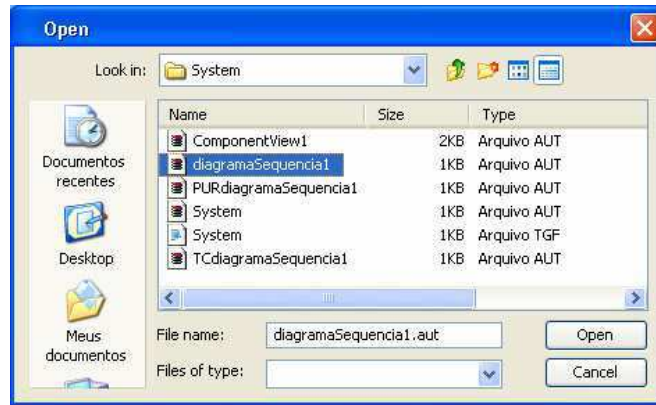


Figura 5.12 - Ferramenta LTS-BT – Escolha de um arquivo (AUT ou TGF) para exibição

Na Figura 5.13, pode ser vista a exibição textual do LTS selecionado (veja Figura 5.12).

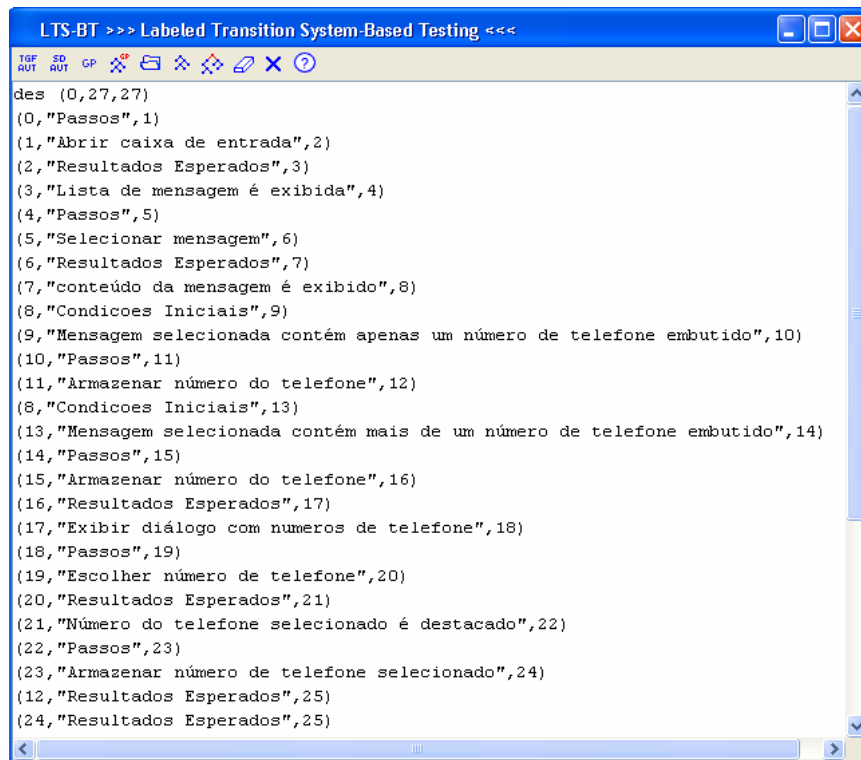


Figura 5.13 - Ferramenta LTS-BT – Exibição textual de um LTS

5.2.6 Exibição Gráfica de um LTS

Na Figura 5.14, pode ser visto a exibição gráfica do LTS. Os formatos aceitos pela ferramenta para exibição, são TGF e AUT.

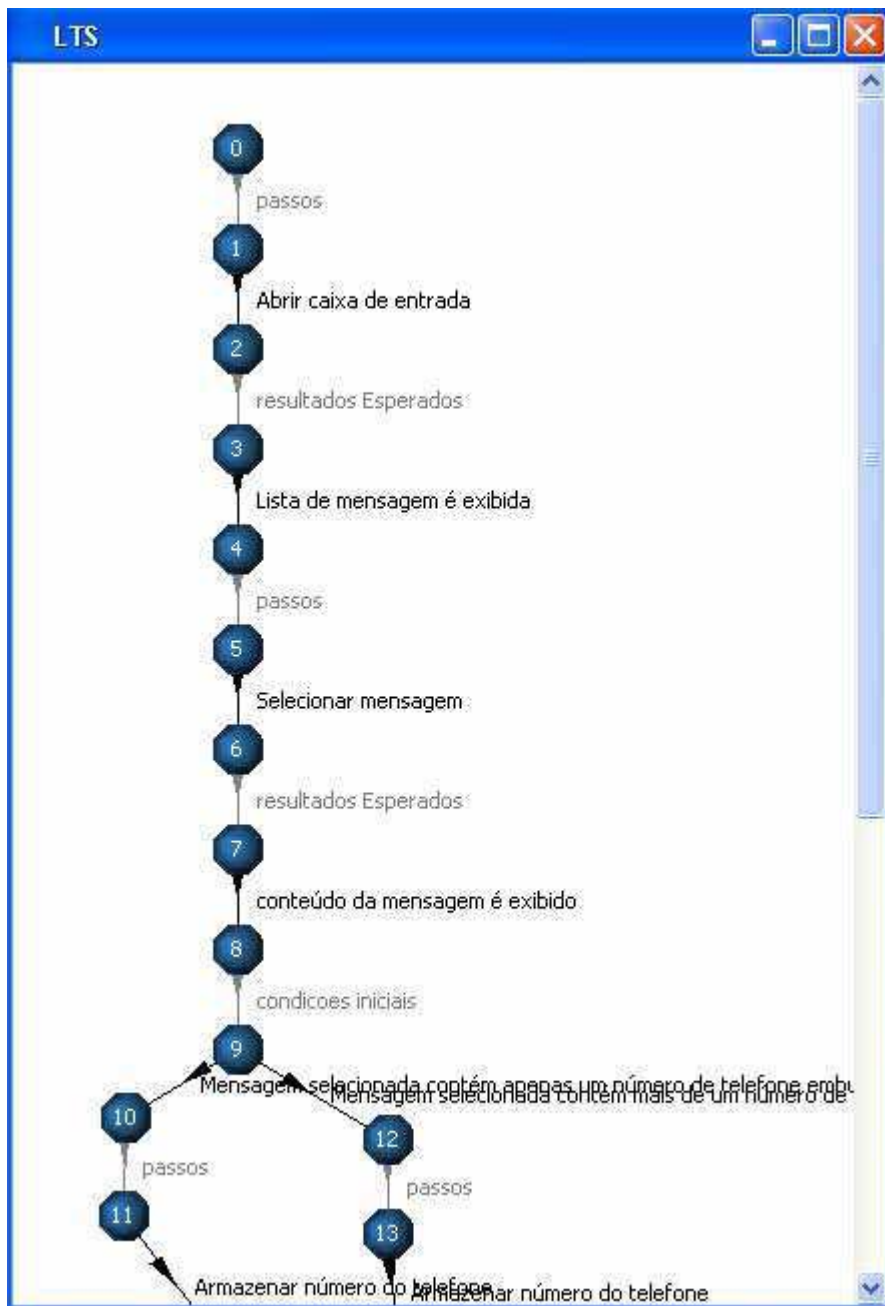


Figura 5.14 - Ferramenta LTS-BT – Exibição gráfica do LTS

5.3 Projeto

A ferramenta foi projetada para facilitar a sua expansão. Isto pode ser visto na Figura 5.15, onde é mostrada uma visão geral do projeto da ferramenta LTS-BT. Para facilitar a sua integração, fizemos uso do padrão de projeto *Facade* [JHJV95].

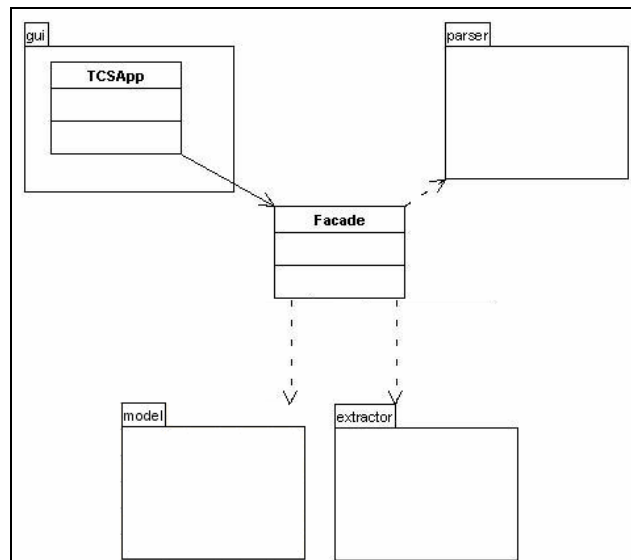


Figura 5.15 - Visão Geral do projeto da ferramenta LTS-BT

A classe *Facade* é a classe que centraliza todas as funcionalidades do *software*, isto é, dá acesso à lógica da aplicação e tem dependência com todos os pacotes. A lógica da aplicação se encontra nos pacotes *parser* e *extractor*. A interface com o usuário é feita através do pacote *gui*. O pacote *model* tem as classes que representam os modelos de uso que trabalhamos (diagrama de seqüência e LTS).

Como já mencionado no Capítulo 3, o algoritmo de geração segue o método DFS no LTS iniciando com o estado inicial até um estado onde não tenha transições de saída. O primeiro passo é selecionar um submodelo do modelo, que atenda ao propósito de teste fornecido, logo em seguida é feita a geração dos casos de teste, veja Figura 5.16.

```

submodeloAUT = selecionaSubmodelo (modeloAUT,propósitoTeste);
geraCasosTeste(submodeloAUT)
  
```

Figura 5.16 - Algoritmo *Extractor*

Os *parsers* implementados, como já dito anteriormente, foram dois:

- TGF para AUT: é feito apenas uma transformação entre formatos, já que os dois são formatos de LTS textual. Veja o algoritmo na Figura 5.17 e um exemplo de TGF e o seu AUT correspondente na Figura 5.18;

```

arquivoAUT= criarArquivoAUT();
root = arquivoTGF.root();
nEstados = contarEstados(ArquivoTGF);
nTransições = contarTransições(ArquivoTGF);
arquivoAUT.imprime "des (root,nTransições,nEstados)";
para i = 1 até ((arquivoTGF.transições()).tamanho())
    transiçãoAtual = arquivoTGF.transições().get(i);
    arquivoAUT.imprime "(transiçãoAtual.nodoOrigem(),"transiçãoAtual.rótulo()",
                        transiçãoAtual.nodoDestino())"

```

Figura 5.17 – Algoritmo *Parser* TGF para AUT

0	root	
1	1	
2	2	
3	3	
4	4	des (0,10,7)
5	5	(0,"Summer",1)
6	6	(2,"Summer",1)
	#	(3,"work",2)
0	1 Summer	(4,"Happy",0)
2	1 Summer record	(4,"April",2)
3	2 work	(5,"work",2)
4	0 Happy New Year!	(5,"work",0)
4	2 April Fools Day	(6,"work",4)
5	2 work	(6,"work",5)
5	0 work and vacation	(6,"stressed",0)
6	4 work	
6	5 work	
6	0 stressed	
	(a)	(b)

Figura 5.18 – TGF (a) e AUT (b)

- MDL/RTMDL para AUT: Neste *parser*, são aplicados todos os procedimentos explicitados no Capítulo 3. A única ressalva é que, como estamos interessados apenas em testes funcionais, ao invés de considerar

todas as mensagens do diagrama de seqüência para a construção do LTS, por questão de melhor desempenho, estamos considerando aqui apenas as que interessam para o teste funcional, isto é, mensagens que partem ou que chegam em Usuário; Veja o algoritmo na Figura 5.19, que foi explicado no Capítulo 3.

```

arquivoAUT = criarArquivoAUT();
para i = 1 até ((arquivoRose.getElementos()).tamanho()) {

tipoAtual = (arquivoRose.getElementos().get(i)).getTipo();
contadorEstado = 0;
se tipoAtual == "mensagem" {
objetoOrigem = (arquivoRose.getElementos().get(i)).getOrigem();
objetoDestino = (arquivoRose.getElementos().get(i)).getDestino();
rótulo = (arquivoTGF.getRose().get(i)).getRotulo();

se objetoOrigem == "usuário"
arquivoAUT.imprime "(contadorEstado,"passos",contadorEstado +1)"
++contadorEstado
arquivoAUT.imprime "(contadorEstado,"rótulo",contadorEstado +1)"
++contadorEstado

se objetoDestino == "usuário"
arquivoAUT.imprime "(contadorEstado,"Resultados Esperados",contadorEstado +1)"
++contadorEstado
arquivoAUT.imprime "(contadorEstado,"rótulo",contadorEstado +1)"
++contadorEstado

se mensagem.éFimLoop()
arquivoAUT.imprime "(contadorEstado,"Condições Iniciais",estadoInicioLoop)"
}
se tipoAtual == "loop"
arquivoAUT.imprime "(contadorEstado,"Condições Iniciais",contadorEstado +1)"
++contadorEstado
rótulo = (arquivoRose.getElementos().get(i)).getRotulo()
arquivoAUT.imprime "(contadorEstado,"rótulo",contadorEstado +1)"
++contadorEstado

se tipoAtual == "loop"
arquivoAUT.imprime "(contadorEstado,"Condições Iniciais",contadorEstado +1)"
++contadorEstado
estadoInicioLoop = contadorEstado
condição = (arquivoRose.getElementos().get(i)).getCondição()
arquivoAUT.imprime "(contadorEstado,"condição",contadorEstado +1)"
++contadorEstado

se tipoAtual == "Fluxo Alternativo"
se tipoAtual.éFluxoAlternativoParalelo()
fimFluxoAlternativo.adiciona(contadorEstado);
arquivoAUT.imprime "(estadoInicioFluxoAlternativo,"Condições
Iniciais",contadorEstado +1)"
++contadorEstado
senão
arquivoAUT.imprime "(contadorEstado,"Condições Iniciais",
contadorEstado+1)"
++contadorEstado
estadoInicioFluxoAlternativo = contadorEstado

condição = (arquivoRose.getElementos().get(i)).getCondição()
arquivoAUT.imprime "(contadorEstado,"condição",contadorEstado +1)"
++contadorEstado

se tipoAtual == "Fluxo Comum"
se (((arquivoRose.getElementos().get(i+1)).getTipo() == "mensagem") e
((arquivoRose.getElementos().get(i+1))).objetoOrigem = "Usuário"))
rótulo = "passos"
se (((arquivoRose.getElementos().get(i+1)).getTipo() == "mensagem") e
((arquivoRose.getElementos().get(i+1))).objetoDestino = "Usuário"))
rótulo = "Resultados Esperados"
se ((arquivoRose.getElementos().get(i+1)).getTipo() == "Fluxo Alternativo"
rótulo = "Condições Iniciais"

para i = 1 até fimFluxoAlternativo.tamanho()
arquivoAUT.imprime "(fimFluxoAlternativo.get(i),"rótulo",contadorEstado +1)"
++contador
}

```

Figura 5.19 - Algoritmo MDL/RTMDL para AUT

5.4 Considerações Finais

Neste capítulo, apresentamos as características e funcionalidades da ferramenta LTS-BT, deixando claras algumas características e requisitos não funcionais que foram traçados no projeto da ferramenta. A visão geral da ferramenta foi apresentada com o intuito de deixar claro o fluxo a ser seguido na ferramenta como também mostrar todas as funcionalidades suportadas. Mostramos, também, uma visão geral do projeto da ferramenta LTS-BT, ressaltando o uso do padrão de projeto Facade que facilita a expansão da ferramenta. Temos também a minimização da interação com o usuário, diminuindo a probabilidade de inserção de falhas.

O projeto final da ferramenta apresenta:

- 56 classes;
- 5379 linhas de código (SLOC).

Capítulo 6

Estudo de caso

O objetivo deste capítulo é apresentar o estudo de caso feito para avaliar o procedimento sistemático para geração de casos de teste, as estratégias de seleção de casos de teste e a ferramenta LTS-BT. Como dito anteriormente, a ferramenta LTS-BT foi desenvolvida para dar suporte ferramental ao procedimento sistemático proposto no Capítulo 3 e às duas estratégias de seleção de casos de teste apresentadas no Capítulo 4. Iremos apresentar duas *features*: para a primeira será detalhado o passo a passo do uso da ferramenta e será mostrado apenas um cenário de uso; já para a segunda, como se trata de uma *feature* maior serão mostrados todos os cenários de uso possíveis e apenas os resultados obtidos. Dessa forma, as duas *features* escolhidas serão submetidas à ferramenta com o objetivo de, a partir de indícios reais, demonstrar a aplicabilidade do procedimento sistemático e das estratégias de seleção propostos.

6.1 *Feature* “*Hot message*”

6.1.1 Descrição da *Feature*

A *feature* “*Hot message*” selecionada consiste em um conjunto de funcionalidades que se relacionam com o armazenamento de mensagens favoritas em uma pasta específica chamada “*Hot message*”. O usuário pode querer salvar uma determinada mensagem (SMS ou MMS) na pasta de mensagens favoritas. Para salvar esta mensagem na pasta do telefone “*Hot message*” esta *feature* precisa estar ativada (o *flex*⁵ tem que estar ativado e a mensagem pode ser do tipo SMS ou MMS (*non Class 2 Message*)).

Na Figura 6.1, pode ser visto o diagrama de seqüência que modela o comportamento desta *feature*. O usuário requisita a leitura de uma mensagem na caixa de entrada e abre a mensagem. Depois de aberta a mensagem, o usuário resolve salvar

⁵ *Flex* é um bit referente a uma determinada *feature*. Se o *flex* estiver ativo, a *feature* está ativa.

esta mensagem na pasta de mensagens favoritas (*Hot message*); a partir daí podemos ter dois caminhos diferentes a serem seguidos dependendo se o *flex* “*Hot message*” estiver ativado ou não. Se o *flex* estiver ativado significa que a *feature* “*Hot message*” está disponível no celular, caso contrário, a *feature* não está disponível.

Se o *flex*:

- Não estiver ativo → então o usuário não tem a opção “*move to hot message*” no *Context Sensitive Menu*.
- Estiver ativo → então o usuário tem a opção “*move to hot message*” no *Context Sensitive Menu* e pode pedir para mover a mensagem para a pasta “*Hot message*”. Se a memória do telefone:
 - Não estiver cheia → A mensagem selecionada é movida com sucesso;
 - Estiver cheia → será requisitado ao usuário apagar algumas mensagens para liberar espaço para a que se quer armazenar.

6.1.2 Geração e Seleção de Casos de Teste

Ao iniciarmos a funcionalidade de geração e seleção de casos de teste da ferramenta LTS-BT, é requisitado o diagrama de seqüência a partir do qual é derivado o LTS. Veja o LTS derivado do diagrama de seqüência apresentado na Figura 6.1, na Figura 6.2.

Em seguida, é exibida a tela apresentada na Figura 6.3. Nesta tela indicamos o propósito de teste e a porcentagem de cobertura de caminhos. Como pode ser visto na figura, são carregadas as mensagens do diagrama de seqüência escolhido que impactarão nos testes funcionais. Tais mensagens podem ser escolhidas pelo usuário da ferramenta para gerar testes.

Considerando o caso para geração exaustiva de casos de teste, isto é, com o propósito “*, *accept*” e com 100% de cobertura de caminhos, temos como resultado três caminhos conforme pode ser visto na Figura 6.4(a, b, c).

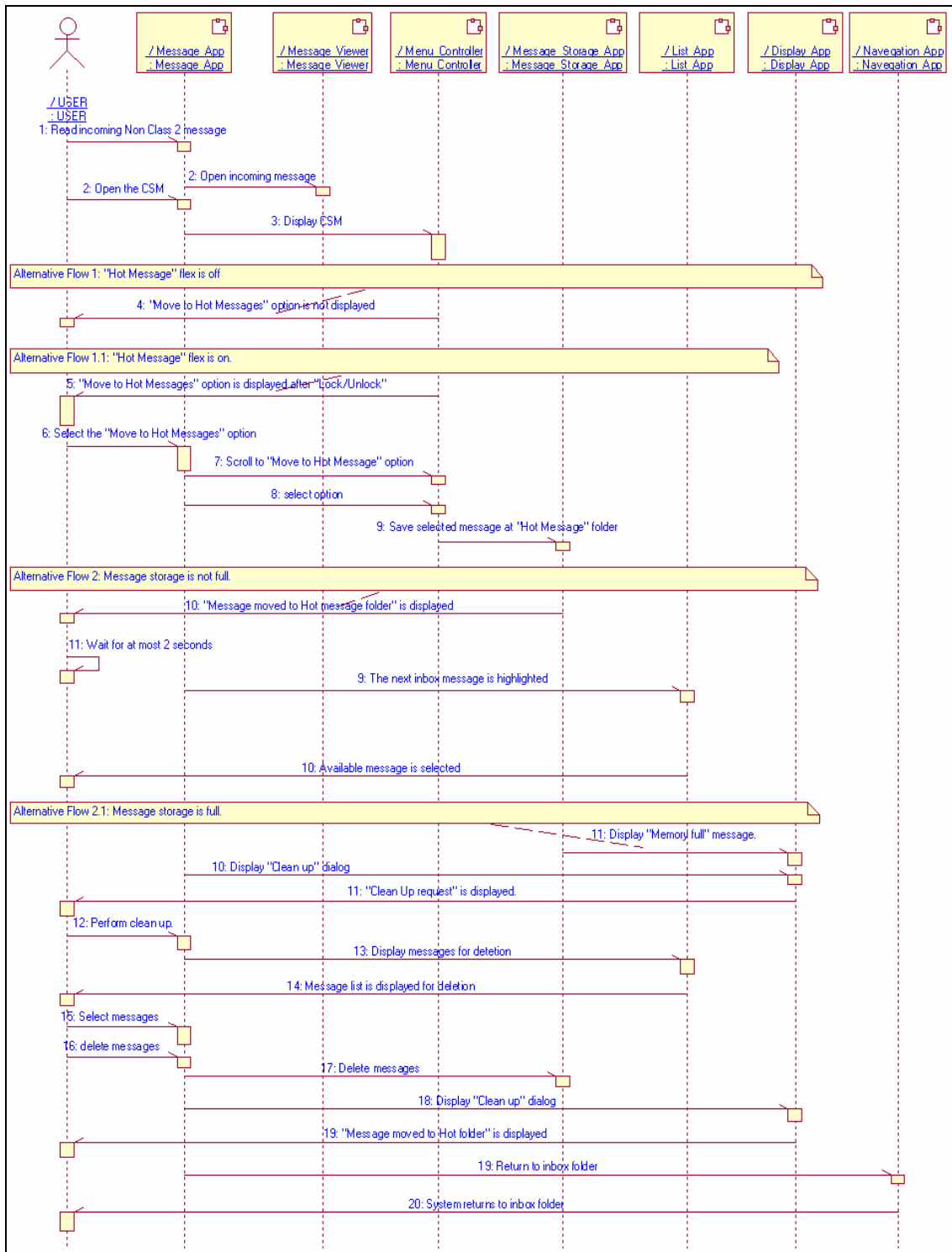


Figura 6.1 - Diagrama de Seqüência - Armazenamento de mensagens favoritas

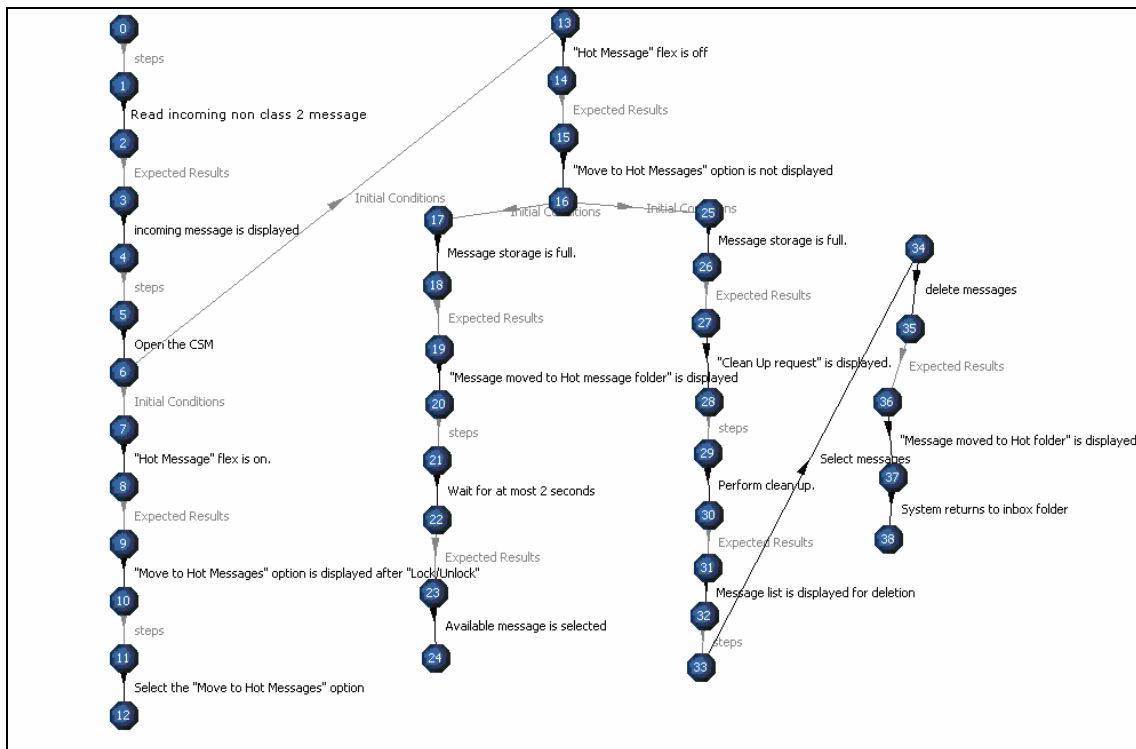


Figura 6.2 - LTS derivado do diagrama de seqüência da Figura 6.1

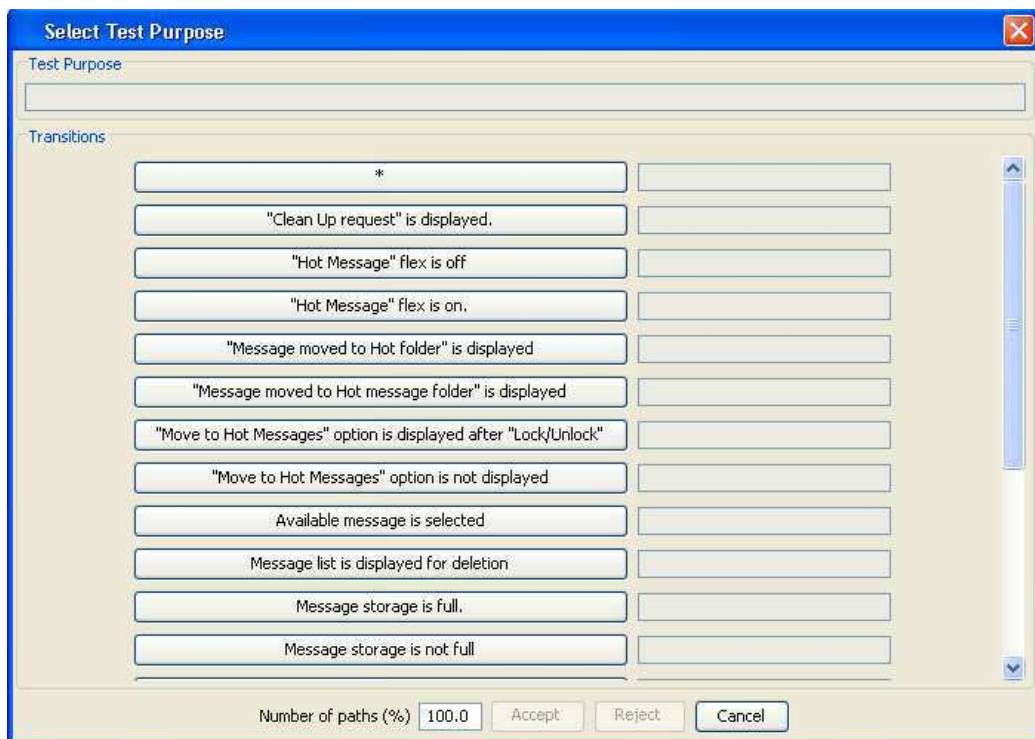


Figura 6.3 - Tela Propósito de teste

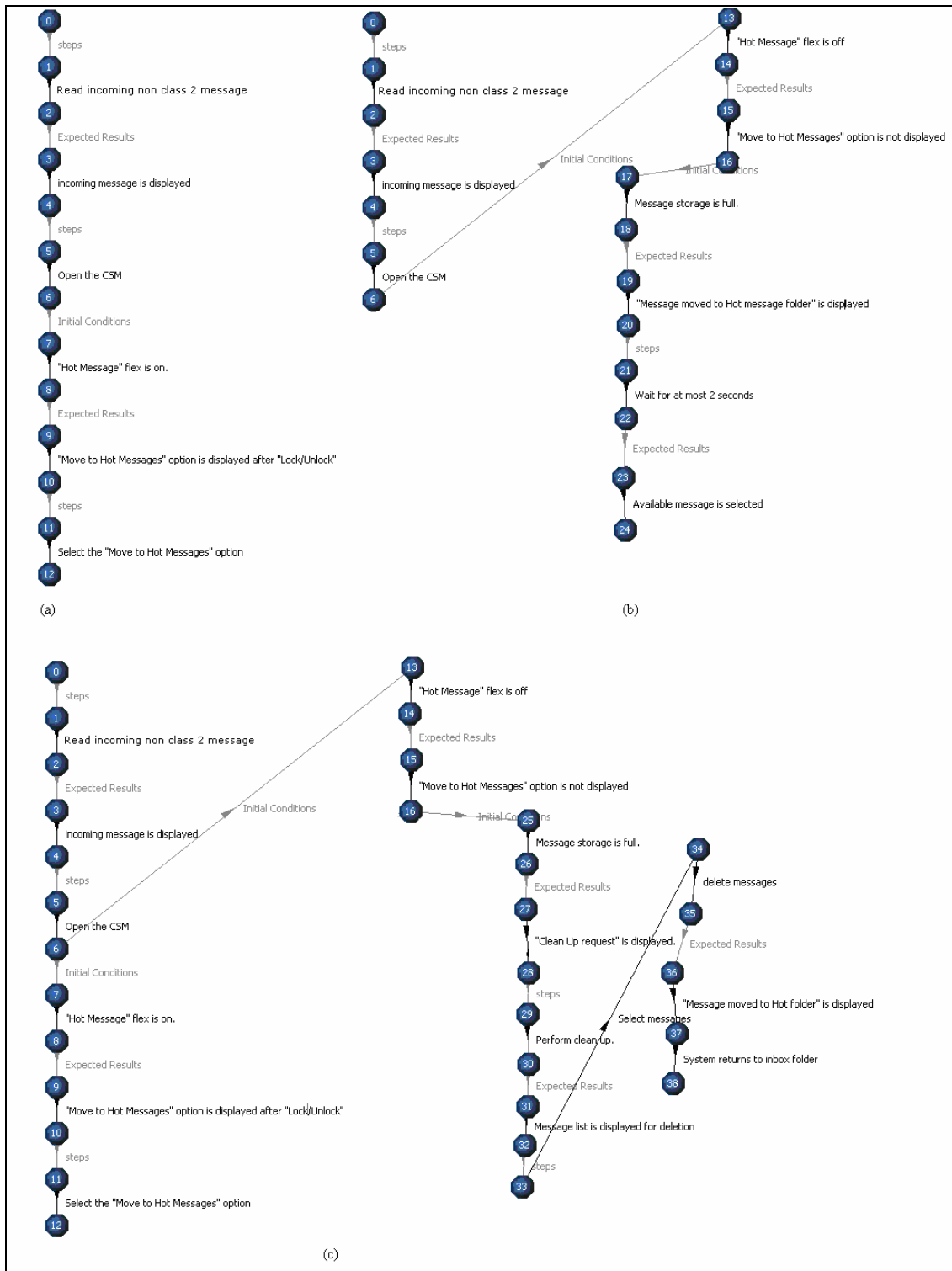


Figura 6.4 - Caminhos do LTS

Se quisermos gerar apenas os casos de teste que tenha a *feature* ativada, ou seja, “*Hot message flex is on*”, então devemos escolher o propósito de teste “*, *Hot message flex is on*, *, *accept*” (veja Figura 6.5). Neste exemplo estamos considerando que

estamos querendo todos os casos de teste que casem com esse propósito, ou seja, 100% de cobertura de caminhos para este propósito.

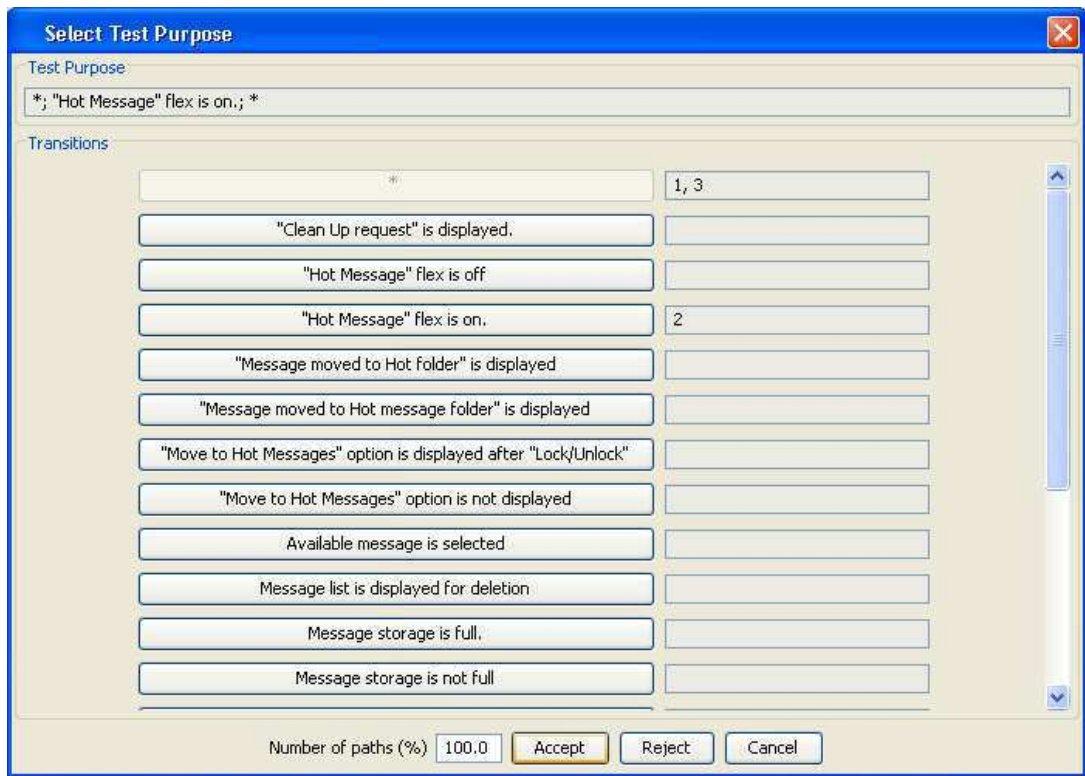


Figura 6.5 - Tela Propósito de teste “*; Hot message flex is on, *, accept”, 100% de cobertura de caminhos

Neste caso, os casos de teste gerados são os que se referem a *feature* quando o *flex* estiver ativado. Assim, obtemos dois casos de teste que se referem à *feature* ativada. Os caminhos que darão origem aos dois casos de teste podem ser vistos na Figura 6.4 (b, c).

Se começarmos a variar a porcentagem de cobertura de caminhos, então podemos notar que é feita uma seleção em cima de todos os possíveis casos de testes gerados para um determinado propósito seguindo a estratégia baseada na similaridade de caminhos apresentada no Capítulo 4.

Na Figura 6.6, é apresentada a tela de propósito de teste, onde estamos querendo gerar casos de teste com qualquer propósito, com 75% de cobertura de caminhos. Neste caso, como já mostrado anteriormente, teremos 3 possíveis casos de teste (veja Figura 6.4), mas só estamos querendo 75% de cobertura. Seguindo as regras de seleção baseada em similaridade de caminhos, devemos eliminar 1 dos 3 possíveis. Assim devemos eliminar o que tiver o maior grau de similaridade. Como os que têm maior

grau de similaridade são os apresentados na Figura 6.4 (b) e (c), será eliminado o menor deles, nesse caso o da Figura 6.4 (b).

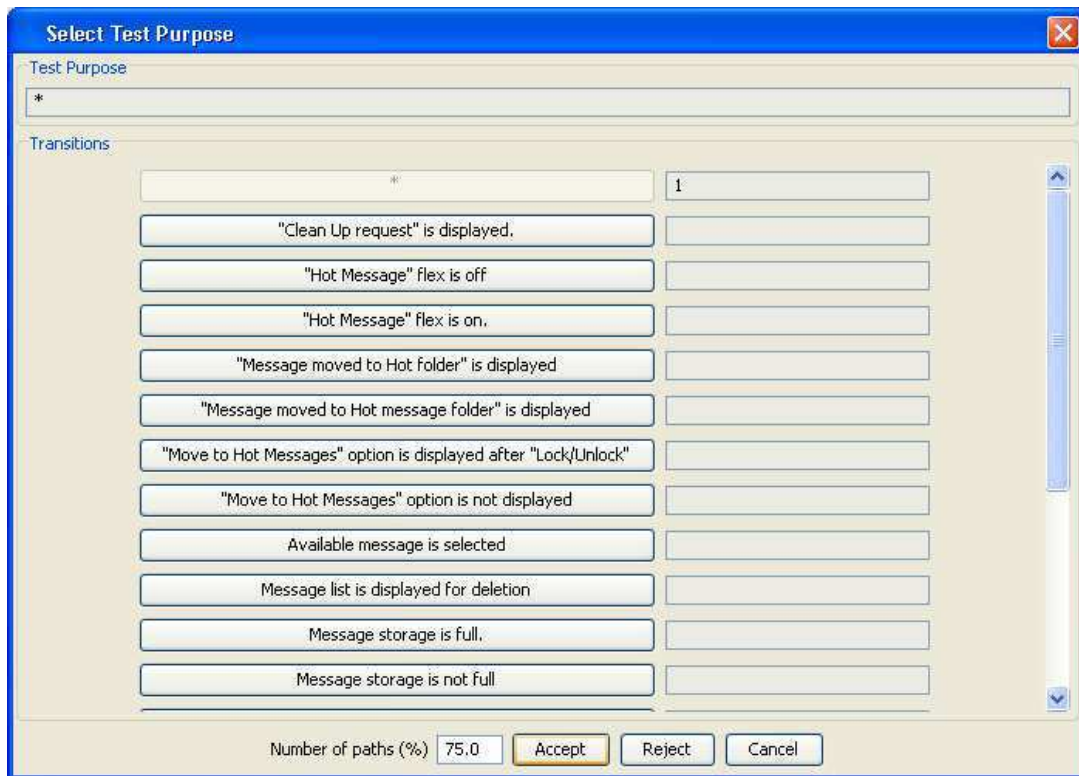


Figura 6.6 - Tela Propósito de teste “*”, 75% cobertura de caminhos

Comparando a suite de teste obtida com a suíte de teste existente na Motorola (16 casos de testes), temos que conseguimos cobrir 50% dos casos de teste existentes. Esta diferença foi devido à falta de informação no diagrama de seqüência. Mensagens do tipo *Non class 2 message* pode ser tanto MMS como SMS, portanto dobraríamos os nosso casos de testes e conseguiríamos a mesma cobertura.

6.2 Feature “Itens Embutidos em Mensagem”

6.2.1 Descrição da Feature

Esta feature permite que o usuário execute ações baseadas em itens embutidos. Um item embutido pode ser:

- Número de telefone;
- Endereço de email;
- URL.

Baseado no tipo do item embutido há diferentes ações disponíveis para o usuário executar e cenários de uso diferente para cada ação:

- **Armazenar URL:**
 - Armazenar URL, que está embutida em uma mensagem de texto recebida e o usuário está visualizando, na Agenda;
 - Armazenar URL, que está embutida em uma mensagem de texto recebida e o usuário está visualizando, nos favoritos do navegador;
 - Armazenar URL, que está embutida em uma mensagem multimídia (MMS) recebida e o usuário está visualizando, na Agenda;
 - Armazenar URL, que está embutida em uma mensagem multimídia (MMS) recebida e o usuário está visualizando, nos favoritos do navegador;
 - Armazenar URL, que está embutida em uma mensagem de email recebida e o usuário está visualizando, na agenda;
 - Armazenar URL, que está embutida em uma mensagem de email recebida e o usuário está visualizando, nos favoritos do navegador.
- **Ir para a URL:**
 - Ir para a URL, que está embutida em uma mensagem de texto recebida, quando o usuário está visualizando a mensagem;
 - Ir para a URL, que está embutida em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem;
 - Ir para a URL, que está embutida em uma mensagem de email recebida, quando o usuário está visualizando a mensagem.
- **Enviar mensagem ao Número do telefone:**

- Enviar mensagem ao número de telefone embutido no campo “De” de uma mensagem selecionada na caixa de entrada de mensagens;
 - Enviar mensagem ao número de telefone embutido em uma mensagem texto recebida, quando o usuário está visualizando a mensagem;
 - Enviar mensagem ao número de telefone embutido em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem;
 - Enviar mensagem ao número de telefone embutido em uma mensagem de email recebida, quando o usuário está visualizando a mensagem de email.
- **Enviar mensagem ao endereço de email:**
 - Enviar mensagem ao endereço de email embutido no campo “De” de uma mensagem selecionada na caixa de entrada de mensagens;
 - Enviar mensagem ao endereço de email embutido em uma mensagem de texto recebida, quando o usuário está visualizando a mensagem;
 - Enviar mensagem ao endereço de email embutido em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem;
 - Enviar a mensagem ao endereço de email embutido em uma mensagem de email recebida, quando o usuário está visualizando a mensagem de email.
- **Armazenar Número do telefone:**
 - Armazenar número de telefone embutido no campo “De” de uma mensagem de texto selecionada na caixa de entrada de mensagens;
 - Armazenar número de telefone embutido em uma mensagem de texto recebida, quando o usuário está visualizando a mensagem;
 - Armazenar número de telefone embutido em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem;

- Armazenar número de telefone embutido em uma mensagem de email recebida, quando o usuário está visualizando a mensagem.

- **Armazenar Endereço de email:**
 - Armazenar endereço de email embutido no campo “De” de uma mensagem de texto selecionada na caixa de entrada de mensagens;
 - Armazenar endereço de email embutido em uma mensagem de texto recebida, quando o usuário está visualizando a mensagem;
 - Armazenar endereço de email embutido em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem;
 - Armazenar endereço de email embutido em uma mensagem de email recebida, quando o usuário está visualizando a mensagem.

- **Ligar para o número do telefone:**
 - Ligar para o número de telefone embutido no campo “De” de uma mensagem de texto selecionada na caixa de entrada de mensagens;
 - Ligar para o número de telefone embutido em uma mensagem de texto recebida, quando o usuário está visualizando a mensagem;
 - Ligar para o número de telefone embutido em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem;
 - Ligar para o número de telefone embutido em uma mensagem de email recebida, quando o usuário está visualizando a mensagem.

- **Enviar uma mensagem de voz a um número de telefone:**
 - Enviar uma mensagem de voz a um número de telefone embutido no campo “De” de uma mensagem de texto selecionada na caixa de entrada de mensagens;

- Enviar uma mensagem de voz a um número de telefone embutido em uma mensagem de texto recebida, quando o usuário está visualizando a mensagem;
- Enviar uma mensagem de voz a um número de telefone embutido em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem;
- Enviar uma mensagem de voz a um número de telefone embutido em uma mensagem de email recebida, quando o usuário está visualizando a mensagem.

6.2.2 Geração de Casos de Teste

Para cada um destes cenários de uso da *Feature*, temos um diagrama de seqüência correspondente que pode ser visto no Apêndice A e que foram projetados pelos desenvolvedores da *Feature* descrita na Seção 6.2.1.

Como temos 33 cenários de uso, temos também 33 diagramas de seqüência que especificam os cenários de uso desta feature. Utilizando a Ferramenta LTS-BT, que implementa o procedimento sistemático para seleção de casos de teste, foram gerados casos de teste exaustivos e com isso obtemos 66 casos de teste. Os casos de teste podem ser vistos no Apêndice B.

Como um primeiro propósito de teste, desejamos gerar todos os casos de teste (100% de cobertura de caminhos) em que o Usuário armazena uma URL. Neste caso, temos 6 cenários de uso envolvidos:

- Armazenar URL, que está embutida em uma mensagem de texto recebida e o usuário está visualizando, na Agenda;
- Armazenar URL, que está embutida em uma mensagem de texto recebida e o usuário está visualizando, nos favoritos do navegador;
- Armazenar URL, que está embutida em uma mensagem multimídia (MMS) recebida e o usuário está visualizando, na Agenda;
- Armazenar URL, que está embutida em uma mensagem multimídia (MMS) recebida e o usuário está visualizando, nos favoritos do navegador;

- Armazenar URL, que está embutida em uma mensagem de email recebida e o usuário está visualizando, na agenda;
- Armazenar URL, que está embutida em uma mensagem de email recebida e o usuário está visualizando, nos favoritos do navegador.

Com esses cenários de uso, conseguimos gerar 12 casos de teste. Utilizando o mesmo propósito, e reduzindo a cobertura de caminhos para 50% obteremos apenas 6 casos de teste, neste caso apenas os maiores (que possuem a maior quantidade de transições) de cada diagrama de seqüência.

Outro caso bem próximo da realidade seria: queremos saber quais são os casos de teste que lidam com mais de um número embutido quando o usuário está visualizando uma mensagem multimídia recebida. Neste caso, temos 5 cenários envolvidos:

- Enviar uma mensagem de voz a um número de telefone embutido em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem;
- Enviar mensagem ao número de telefone embutido em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem;
- Armazenar número de telefone embutido em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem;
- Ligar para o número de telefone embutido em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem;
- Enviar uma mensagem de voz a um número de telefone embutido em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem.

Como queremos apenas os que lidam com mais de um número embutido na mensagem multimídia, então temos 5 casos de teste. Se quisermos reduzir a cobertura de caminhos, neste caso não temos como, uma vez que cada caso de teste gerado é de um diagrama de seqüência diferente e conseqüentemente de um modelo LTS diferente, e desta forma não temos como calcular o grau de similaridade, já que pertencem a modelos LTS diferentes.

6.3 Considerações Finais

Neste capítulo, apresentamos a aplicação da ferramenta LTS-BT na geração e seleção de casos de teste de duas *features* projetadas para celulares Motorola. A primeira apresentada diz respeito ao armazenamento de mensagens favoritas por usuários em uma pasta chamada “*Hot message*”, a segunda, diz respeito a itens embutidos em mensagens. De acordo com os diagramas de seqüência utilizados para especificação do comportamento dinâmico das *features*, foram gerados, com o auxílio da ferramenta LTS-BT, casos de teste exaustivos e com alguns propósitos interessantes em cima dos diagramas de seqüência que foram formulados. O diagrama da primeira feature (ao armazenamento de mensagens favoritas) foi obtida automaticamente através do trabalho mostrado em [Fer06] e os diagramas referentes à segunda *feature* (itens embutidos em mensagens) foram obtidos em documentos elaborados pelos desenvolvedores. Os últimos precisaram de alguns ajustes para a utilização na ferramenta.

Embora tenha sido realizada em estudos de casos simples, a utilização do procedimento sistemático e das estratégias de seleção propostos foi facilitada pelo uso da ferramenta desenvolvida, tornando as tarefas propostas mais simples e menos sujeita à inserção de erros, já que todo o processo é automatizado.

Capítulo 7

Conclusão

O trabalho aqui apresentado propõe um procedimento sistemático para geração de casos de teste funcional para aplicações de celulares, partindo de diagramas de seqüência UML, tendo como meio, o modelo LTS derivado daqueles. Além do procedimento sistemático para geração de casos de teste, foram apresentadas duas estratégias de seleção de casos de teste. Como suporte ao procedimento sistemático e às estratégias de seleção propostas, foi implementada uma ferramenta de suporte para a realização do que foi proposto, a ferramenta LTS-BT (*Labeled Transition System-Based Testing*).

O artefato utilizado para a geração de casos de teste, como já dito anteriormente, são os diagramas de seqüência UML. Estes são criados segundo o formato apresentado no Capítulo 2, e podem ser utilizados assim que são desenvolvidos, o que possibilita a aplicação do processo de teste de forma paralela ao processo de desenvolvimento e sem custo adicional para produção de artefatos específicos para teste. A partir dos diagramas de seqüência UML produzidos, podemos transformá-los em modelos LTSs e a partir daí fazer a geração. Para uma seleção de casos de teste, propomos duas estratégias, a primeira baseada em um propósito, cujo formato específico foi também apresentado; a segunda, baseada na similaridade de caminhos. As duas estratégias podem tanto ser aplicadas em separado, como também em conjunto. A ferramenta LTS-BT, cujo projeto é descrito no Capítulo 5 deste documento, realiza a automação da geração e seleção de casos de teste partindo de diagramas de seqüência.

Foram desenvolvidos dois estudos de casos, sobre duas *features* distintas desenvolvidas para celulares da Motorola, “*Hot message*” e “Itens embutidos em mensagem”, para demonstrar o funcionamento do procedimento, das estratégias propostas e da ferramenta desenvolvida para dá suporte às duas últimas. Os resultados obtidos foram apresentados neste documento. Nos estudos de caso, obtivemos todos os diagramas de seqüência, projetados pelos desenvolvedores, relacionados às *features* escolhidas e aplicamos, de forma automática, a geração e seleção de casos de teste, a

partir da ferramenta LTS-BT. Os resultados observados trouxeram indícios reais da confiabilidade e fácil uso da ferramenta.

As desvantagens observadas da utilização da ferramenta com a aplicação do estudo de caso são:

- Os diagramas de seqüência que são utilizados como entrada para a ferramenta têm que estar no formato Rose ou Rose-RT que são duas ferramentas cujas licenças são dispendiosas;
- Os diagramas de seqüência que são utilizados possuem um padrão. Isto pode ser visto por quem está projetando os cenários de uso como uma limitação. Entretanto, no escopo do projeto CInBTCRD existe um trabalho que gera diagramas de seqüência, já no formato requerido pela ferramenta LTS-BT, diretamente de modelos CSP [Fer06].
- A não composição dos LTSs nos leva a executar várias vezes a ferramenta para a mesma *Feature*. Neste caso, como este trabalho está envolvido no escopo do CInBTRD, este ponto fraco está sendo coberto em [Sou06], onde é feita a composição de modelos LTS.
- Como toda abordagem baseada em modelos, a apresentada neste trabalho é dependente da disponibilização das especificações na forma de diagramas de seqüência, ou modelos no formato LTS. A sua aplicação é, portanto, restrita a ambientes onde tais modelos são construídos e mantidos como parte do processo de desenvolvimento. É importante observar que, dentro do contexto do projeto CInBTCRD, existem trabalhos para gerar tais modelos automaticamente a partir de documentos em linguagem natural [Lei06].

7.1 Contribuições

Há poucos métodos/técnicas/procedimentos sistemáticos para geração de casos de teste funcional para aplicações de celulares, considerando o desenvolvimento baseado em *Features*. A vantagem do trabalho proposto aqui, com relação a outros trabalhos relacionados é que este é adaptado para testes de *feature* de aplicações de celular, incluindo loops e fluxos alternativos que são bastante utilizados em diagramas de seqüência que representam o comportamento de *feature*.

Neste trabalho, definimos um procedimento de teste funcional aplicado à *feature* de aplicações de celulares, cujas atividades são quase totalmente automatizadas pela ferramenta de suporte desenvolvida. O procedimento proposto utiliza especificações UML como entrada. Como dito anteriormente, UML é uma linguagem bastante utilizada na indústria e na academia, portanto a aplicação do procedimento, manual ou através da ferramenta de suporte, não traria um custo adicional de treinamento para uma notação específica. Além disso, como o artefato utilizado é produzido antes da fase de implementação, através da derivação do modelo LTS partindo dos diagramas de seqüência, podemos fazer uma verificação se a especificação está refletindo os requisitos do cliente, através de ferramentas de animação de modelos LTS. Dessa forma, temos a possibilidade dos defeitos serem descobertos mais cedo, reduzindo assim, o custo de correção em fases posteriores.

Durante o desenvolvimento da ferramenta LTS-BT, foi utilizado o padrão de projeto Facade [GHJV95], que facilita a utilização de outros algoritmos. Um exemplo que podemos citar é a utilização da ferramenta TGV [JJ04] para geração de casos de teste, ao invés da utilização do nosso procedimento por completo. Neste caso seria utilizada a primeira parte do procedimento – obtenção do modelo LTS e a partir daí, seria utilizado o TGV.

7.2 Trabalhos Relacionados

Como dito anteriormente, existe uma carência por abordagens voltadas ao teste de *features*, com base em notações mais comumente utilizadas no mercado, como UML, e passíveis de automação. A geração de casos de teste a partir de diagramas de seqüência, já é considerada em diferentes trabalhos. O uso de LTS como modelo para geração de casos de teste também já é abordado em diferentes trabalhos. No entanto, sua derivação a partir de diagramas de seqüência e aplicação de técnicas de seleção efetivas são pontos de interesse ainda não abordados. Nesta seção estamos apresentando duas ferramentas bastante conhecidas tanto na indústria quanto na academia: TGV e PTK.

7.2.1 TGV

TGV (*Test Generation with Verification Technology*) é uma ferramenta de suporte a testes de conformidade projetada especificamente para testar sistemas reativos não-deterministas. TGV baseia-se na teoria ioco apresentada em [Tre96] cuja hipótese é que a IUT (*Implementation Under Test*) possa ser modelada via um IOLTS.

Como entrada, TGV recebe um modelo do comportamento do sistema e um propósito de teste, ambos fornecidos como uma variante de sistemas de transições rotuladas (LTS), chamado sistemas de transições rotuladas de entrada e saída (IOLTS). Um IOLTS provê distinção entre os eventos do sistema que são controláveis e aqueles que são observáveis pelo ambiente, i.e. entradas e saídas, respectivamente. Ações internas do sistemas também podem ser representadas. As entradas e saídas são representadas respectivamente por: ? e !.

Existe claramente um ponto comum entre TGV e nossa abordagem no que diz respeito ao modelo de execução de teste. Em nossa abordagem é utilizado um LTS anotado. Dado o LTS anotado, onde cada caminho representa um caso de teste consistente (*sound*).

7.2.2 PTK

PTK [BBJ⁺03] faz a geração de testes de conformidade a partir de especificações descritas em MSC. Diagramas MSC são empregados para definir um conjunto de comportamentos aceitos pela IUT. A ferramenta PTK interpreta o MSC, tornando explícitas as seqüências de teste e observações baseadas no comportamento externo que pode ser controlado e aspectos visíveis do MSC, e esconde as ações internas e sua ordem interna. PTK também provê mensagens de dados, baseadas na PDU (Protocol Data Unit) das especificações, para criar múltiplos casos de teste de um único MSC. As saídas de PTK são conjuntos de teste descritos em TTCN e SDL, mas também possui um gerador de código que pode ser estendido para produzir scripts em outras linguagens.

7.3 Trabalhos Futuros

Com a finalização do trabalho, tivemos a possibilidade de fazer uma análise do mesmo, a qual resultou no seguinte conjunto de propostas para a sua continuidade:

- **Complementar o procedimento proposto:** o procedimento sistemático proposto é limitado à geração de casos de teste de *Features* isoladas. Um procedimento completo deveria considerar também a possibilidade de realização de testes de interação de *Feature* [FN00, CKMR99, KK98], assim estaríamos também com o foco em um dos problemas relatados na literatura com relação a aplicações de celulares.
- **Submeter o trabalho a outros estudos de caso:** desenvolvemos dois estudos de caso que nos forneceram uma visão prática da aplicação do procedimento sistemático para geração de casos de teste e das estratégias de seleção, e do uso da ferramenta LTS-BT. Tendo como objetivo obter uma maior confiança com relação ao procedimento e estratégias propostas, precisamos aplicá-lo a um maior número de estudos de caso e realizar um estudo experimental;
- **Evolução da ferramenta:** embora já tenhamos uma primeira versão da ferramenta, como propomos, como trabalho futuro, o desenvolvimento de um método mais abrangente, é necessário que a ferramenta LTS-BT também seja evoluída para incorporar funcionalidades referentes ao teste de integração de *Features*;
- **Avaliação da cobertura e eficácia do conjunto de testes gerado:** é necessário executar o conjunto de casos de testes gerados almejando a avaliação da cobertura e a eficácia dos casos de testes gerados;
- **Avaliação da adoção em linha de produção na Motorola:** é necessário uma avaliação dos impactos causados com a adoção da ferramenta LTS-BT no processo de desenvolvimento existente.

Bibliografia

- [AO00] A. Abdurazik and J. Offutt. Using UML Collaboration Diagrams for Static Checking and Test Generation. In The Third International Conference on the Unified Modeling Language (UML'00), pags 383-395, York, UK, Outubro 2000.
- [BAMF04] D. L. Barbosa, W. L. Andrade, P. D. L. Machado, J. C. A. Figueiredo. SPACES - Uma Ferramenta para Teste Funcional de Componentes In: XI Sessão de Ferramentas - SBES 2004, Brasília. XI Sessão de Ferramentas. Porto Alegre: Sociedade Brasileira de Computação, 2004. pags 55 – 61
- [BBJ⁺03] P. Baker, P. Bristow, C. Jervis, D. King, B. Mitchell. Automatic Generation of Conformance Tests From Message Sequence Charts, Telecommunications and Beyond: The Broader Applicability of MSC and SDL, pp 170-198, LNCS 2599, 2003.
- [Bei95] B. Beizer: Black-Box Testing: Techniques for Functional Testing of Software and Systems. John Wiley & Sons, 1995.
- [Bin99] R. Binder. Testing Object-Oriented Systems: Models, Patterns, and Tools. Addison-Wesley, 1999.
- [BL01] L. Briand, and Y. Labiche. A UML-based approach to system testing. Lecture Notes in Computer Science, 2185:60 – 70, 2001.
- [BMJ01] L. du Bousquet, H. Martin, J.-M. Jézéquel, “Conformance Testing from UML Specifications”, Experience Report, 2001. 44-55.
- [CCT02] W. K. Chan, T.Y. Chen, and T.H. Tse. "An overview of integration testing techniques for object-oriented programs". in *Proceedings of the 2nd ACIS Annual International Conference on Computer and Information Science (ICIS 2002)*, International Association for Computer and Information Science, Mt. Pleasant, Michigan, pp. 696-701 (2002).
- [CJ02] R. D. Craig and S. P. Jaskiel. Systematic Software Testing. Boston: Artech House, 2002.

- [CKMR99] M. Calder, M. Kolberg, E. H. Magill, and S. Reiff-Marganiec, “Feature interaction: a critical review and considered forecast”, *Computer Networks*, Amsterdam – Holand, 1999, 41(1), pp. 115–141.
- [DJK⁺99] S. R. Dalal, A. Jain, N. Karunanithi, J. M. Leaton, C. M. Lott, G. C. Patton and B. M. Horowitz, *Model-Based Testing in Practice*. Proceedings of the ICSE’99. Maio 1999.
- [EW01] I. K. El-Far and J. A. Whittaker, *Model-Based Software Testing*. Encyclopedia of Software Engineering (edited by J. J. Marciniak). Wiley, 2001.
- [Fer06] P. M. Ferreira: *Geração Automática de Diagramas UML-RT a partir de Especificações CSP*. Master’s thesis, Centro de Informática - Universidade Federal de Pernambuco, 2006.
- [FGMT02] L. M. G. Feijs, N. Goga, S. Mauw, J. Tretmans. Test Selection. Trace Distance and Heuristics. In I. Schieferdecker, H. Käonig, and A. Wolisz, editors, *Testing of Communicating Systems XIV*, pages 267{282. Kluwer Academic Publishers, 2002.
- [FN00] A P. Felty, K.S. Namjoshi, *Feature Specification and Automated Conflict Detection, Feature Interactions*. Workshop, IOS Press, 2000.
- [FWM06] A. Figueiredo, W. Andrade , P. Machado, *Generating Integration Test Cases for Mobile Phone Systems from Use Case Specifications*. A-MOST’06.
- [FW06] L. Freitas and J. Woodcock. *FDR explorer*. Refine 2006 - International Refinement Workshop.
- [GHJV95] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software* Addison Wesley. 1995.
- [Gra94] J. Grabowski. *The Generation of TTCN Test Cases from MSCs*. Technical Report IAM-94-004, Universität Bern, Maio 1994.
- [GWH99] M. S. Gimenes, G. M. Weis, E. H. M. Huzita. *Um Padrão para Definição de um Gerenciador de Processos de Software*. Proceedings of the 2nd Workshop IberoAmericano de Engenharia de Requisitos Y Ambientes

- Software. San Jose, Costa Rica, Ideas'99 Memorias, 1999, vol. 1, San Jose: Instituto Tecnológico de Costa Rica, 1999; 30-46.
- [Hes06] A. Hessel. Master's thesis. Timing analysis of an SDL subset in Uppaal. Division of Computer Systems, Department of Information Technology, Uppsala University, Uppsala, Sweden, Março 2006.
- [HVR04] J. Hartmann, M. Vieira, A. Ruder. "UML-based Test Generation and Execution". In Proceedings of the 21st Workshop on Software Test, Analyses and Verification (GI-FG TAV), Junho 2004 Berlin.
- [JJ04] C. Jard, T. Jéron, "TGV: theory, principles and algorithms, A tool for the automatic synthesis of conformance test cases for non-deterministic reactive systems", Software Tools for Technology Transfer (STTT), Outubro 2004.
- [JLP98] J.-M. Jézéquel, A. LeGuennec, and F. Pennaneach, "Validating distributed software modelled with UML", In Proc. Int. Workshop UML98, Mulhouse, France, Junho 1998.
- [Jor95] P. C. Jorgensen. Software Testing – a Craftsman Approach. CRC Press, 1995.
- [KK98] D. O. Keck and P. J. Kuehn, "The Feature and Service Interaction Problem in Telecommunications Systems: A Survey", in IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 24. NO. 10, Outubro, 1998.
- [Lei06] D. A. Leitão: NLFORSPEC: Uma Ferramenta para Geração de uma Ferramenta para Geração de Especificações Formais a partir de Casos de Teste em Linguagem Natural. Master's thesis, Centro de Informática - Universidade Federal de Pernambuco, 2006.
- [LJX⁺04] W. Linzhang, Y. Jiesong, Y. Xiaofeng, L. Xuadong and Z. Guoliang. Generating Test Cases from UML Activity Diagram based on Gray-Box Method. In: Proceedings of the 11th Asia-Pacific Software Engineering Conference (APSEC'04), 2004.

- [LSJ05] L. Li, W. Su, J. Jiang: The influencing factors and marketing strategies of developing telecommunication industry customer loyalty: based on analytic hierarchy process. ICEC 2005: 784-786.
- [LY01] Lin, J. and Yeh, P. (2001). Automatic test data generation for path testing using GAs. *Information Sciences*, 131(1-4):47-64.
- [MS01] J. D. McGregor and D. A. Sykes. *A Practical Guide to Testing Object-Oriented Software*. Addison-Wesley, 2001.
- [Nog06] S. C. Nogueira. *Geração Automática de Casos de Teste CSP Orientada por Propósitos*. Master's thesis, Centro de Informática - Universidade Federal de Pernambuco, 2006.
- [OA99] J. Offutt and A. Abdurazik. Generating tests from UML specifications. In *Proceedings of the Second IEEE International Conference on the Unified Modelling Language (UML99)*, pages 416-429, Fort Collins, CO, Outubro 1999. IEEE Computer Society Press.
- [Per92] W. B. Perkinson. *A Methodology for Designing and Executing ISDN Feature Tests, Using Automated Test Systems*, IEEE GLOBECOM'92. Orlando, Florida, Dezembro 1992.
- [RHR97] A. W. Roscoe, C. A. R. Hoare, and Richard Bird. *The Theory and Practice of Concurrency*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1997.
- [RL03] A. Ran, R. Lencevicius, *Making Sense of Runtime Architecture for Mobile Phone Software*, to appear in *Proceedings of the joint European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2003.
- [Sou06] C. Souza: *A Strategy for Composing and Unifying System Behavioural Models*. Master's thesis (in progress), Informatics Center of Federal University of Pernambuco, 2006.
- [TB02] J. Tretmans, E. Brinksma. *Côte de Resyste – Automated Model Based Testing*. In M. Schweizer, editor, *Proceedings of Progress 2002 - 3rd Workshop on Embedded Systems*, STW Technology Foundation, Utrecht, The Netherlands, pages 246-255, 2002.

- [Tor06] Dante Gama Torres. Specnl: Uma ferramenta para gerar descrições em linguagem natural a partir de especificações de casos de teste. Master's thesis, Centro de Informática - Universidade Federal de Pernambuco, 2006.
- [TLW98] R. Turner, A. Fuggetta, L. Lavazza and A. L. Wolf, "Feature Engineering", From the Proc. Of the 9th Inter. Workshop on Software Specification and Design, Ise-shina, Japan, Abril 18-18, 1998.
- [Tre96] J. Tretmans. Software Concepts and Tools, 17(3):103.120, 1996.
- [Vri00] R. G. de Vries, J. Tretmans: On-the-fly Conformance Testing using SPIN, 2000.
- [Wat00] J. Watkins, Testing IT: an off-the-shelf software testing process, Cambridge University Press, New York, NY, 2000.

Apêndice A

Diagramas de Seqüência

Neste apêndice serão mostrados todos os diagramas relacionados à *feature* “Itens Embutidos em Mensagem”.

A.1 Diagramas de seqüência relacionados à “Armazenar URL”

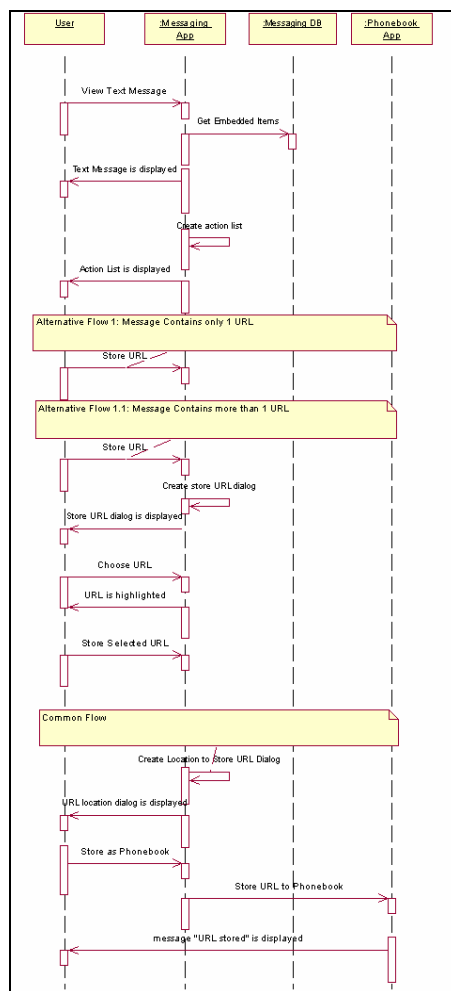


Figura A.1 - Armazenar URL, que está embutida em uma mensagem de texto recebida e o usuário está visualizando, na Agenda.

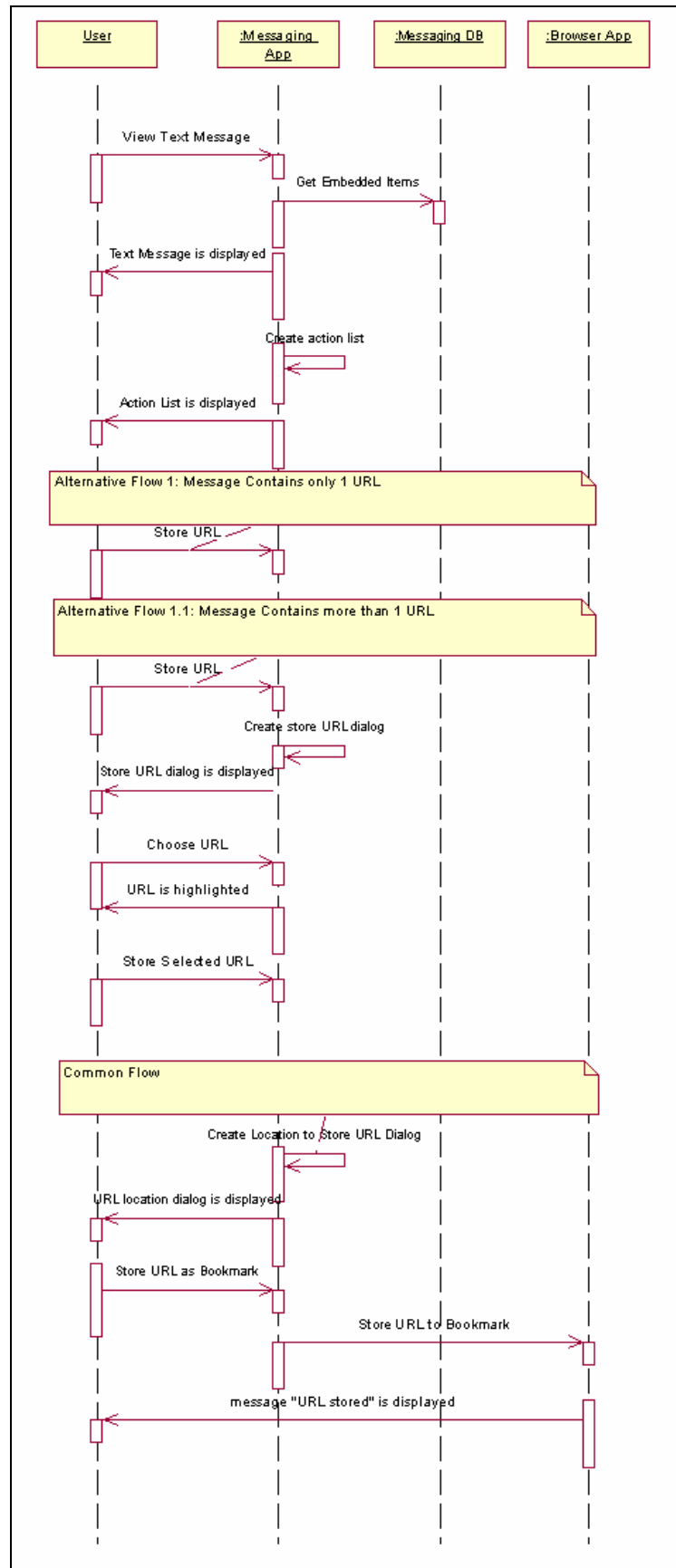


Figura A.2 - Armazenar URL, que está embutida em uma mensagem de texto recebida e o usuário está visualizando, nos favoritos do navegador.

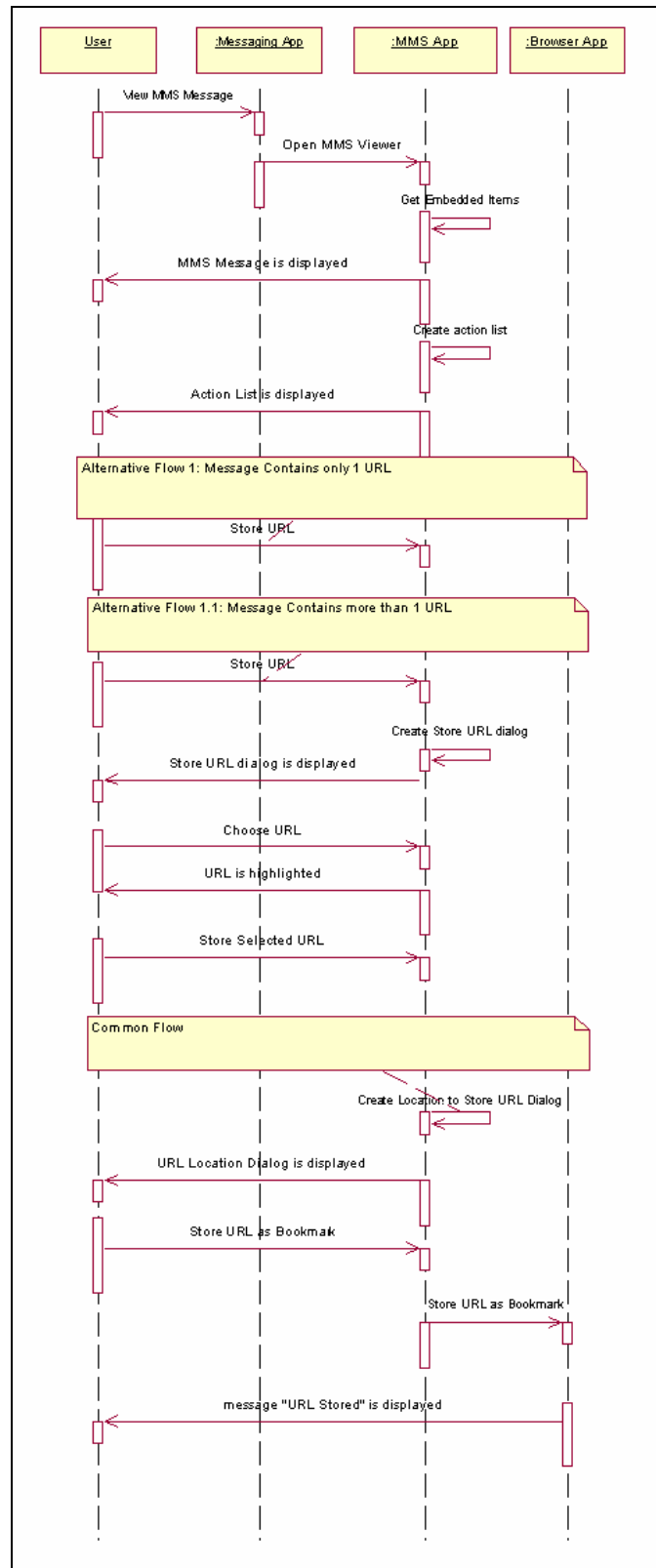


Figura A.3 - Armazenar URL, que está embutida em uma mensagem multimídia (MMS) recebida e o usuário está visualizando, nos favoritos do navegador

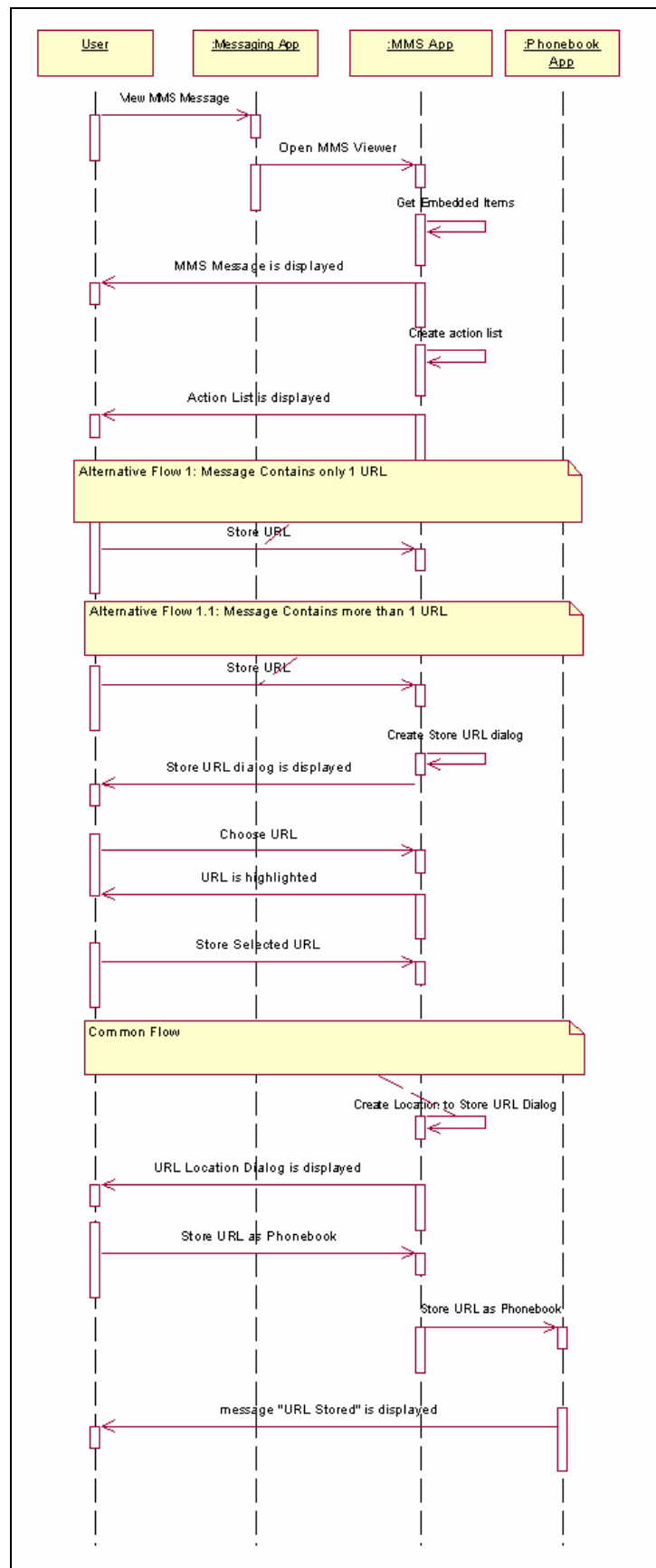


Figura A.4 - Armazenar URL, que está embutida em uma mensagem multimídia (MMS) recebida e o usuário está visualizando, na Agenda

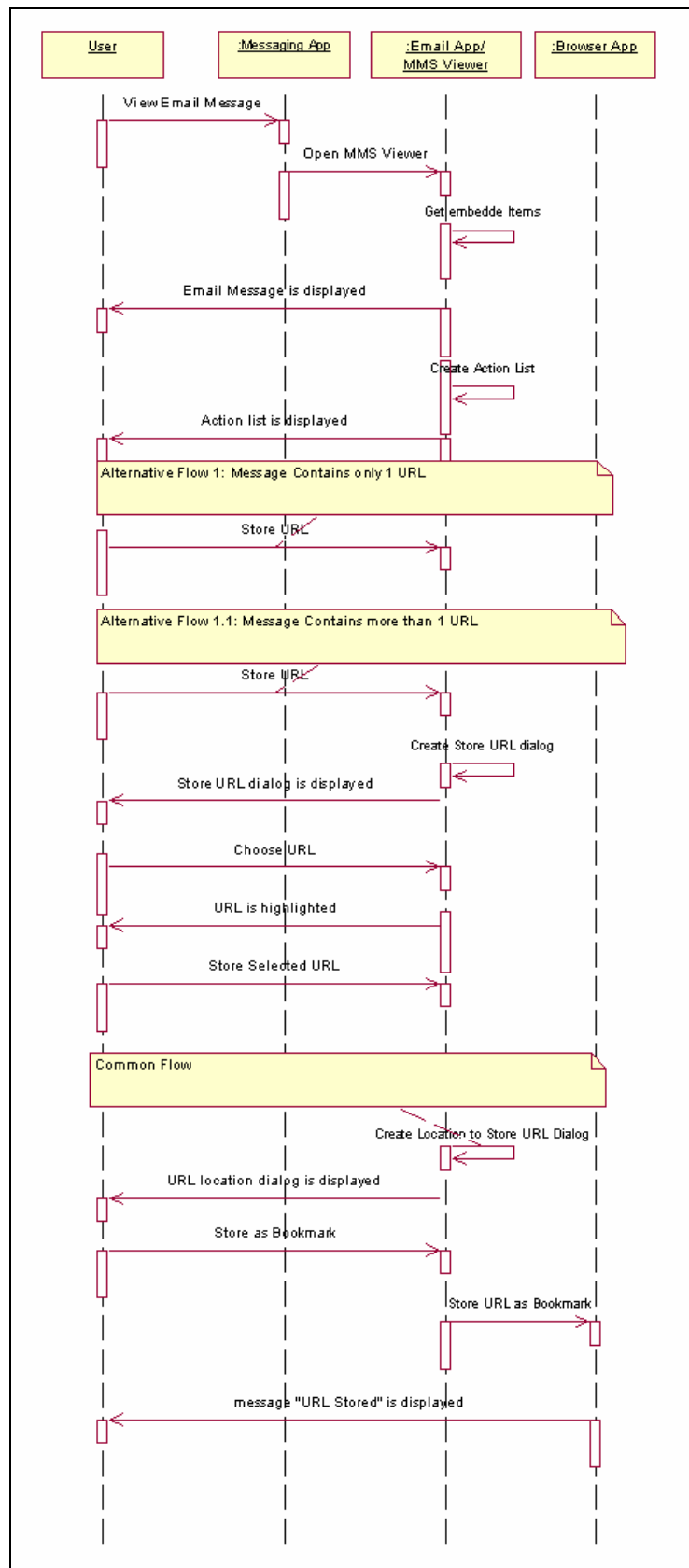


Figura A.5 - Armazenar URL, que está embutida em uma mensagem de email recebida e o usuário está visualizando, nos favoritos do navegador

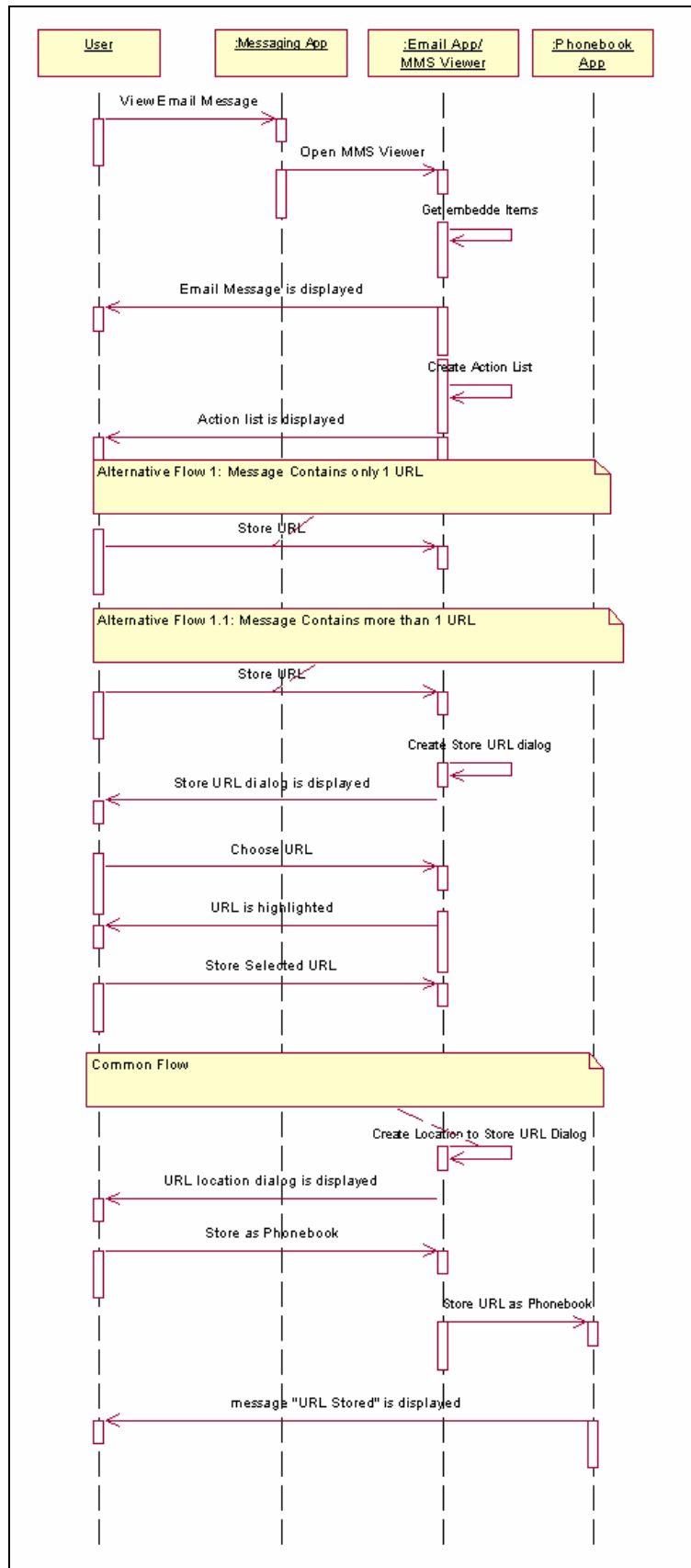


Figura A.6 - Armazenar URL, que está embutida em uma mensagem de email recebida e o usuário está visualizando, na agenda

A.2 Diagramas de seqüência relacionados à “Ir para a URL”

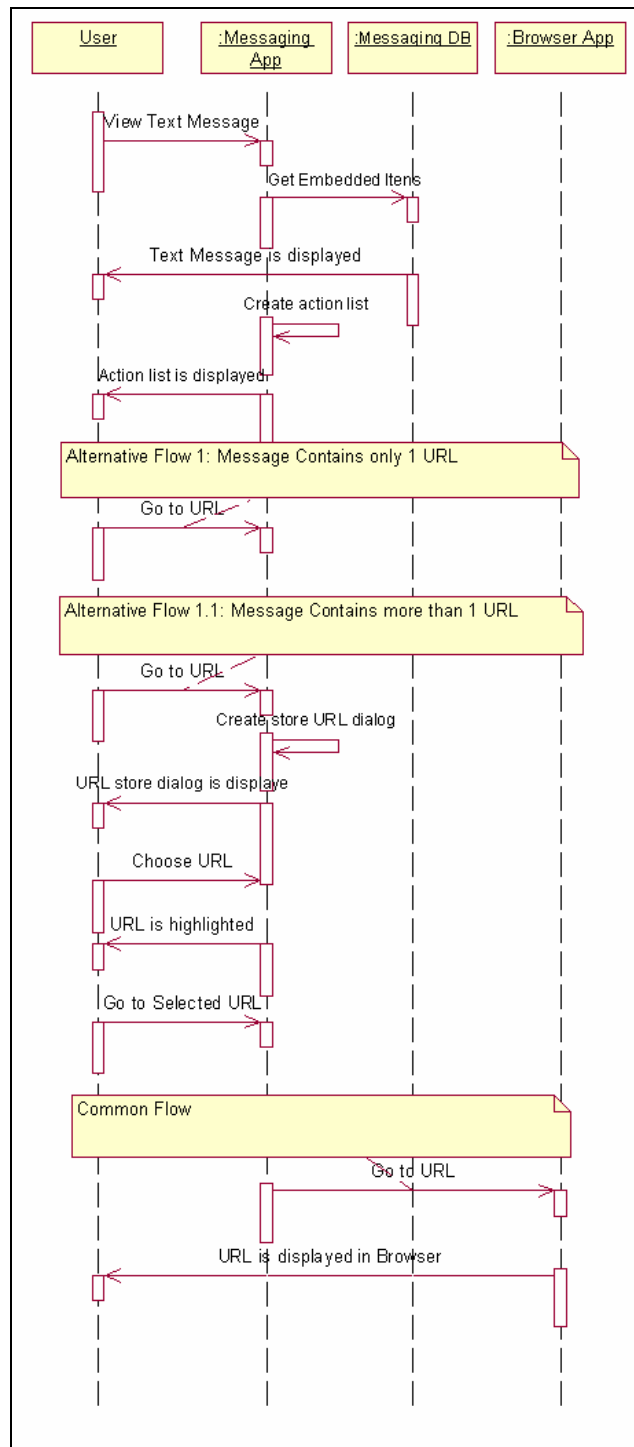


Figura A.7 - Ir para a URL, que está embutida em uma mensagem de texto recebida, quando o usuário está visualizando a mensagem.

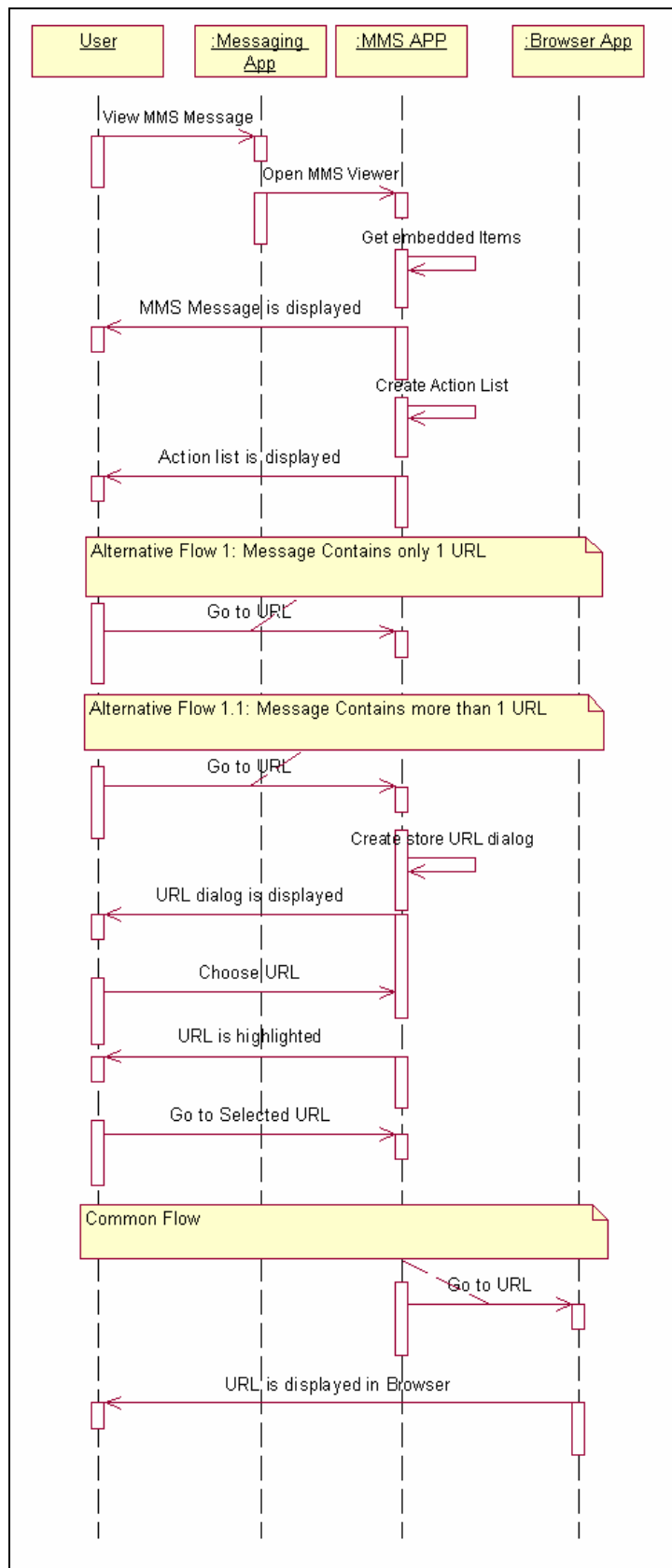


Figura A.8 - Ir para a URL, que está embutida em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem.

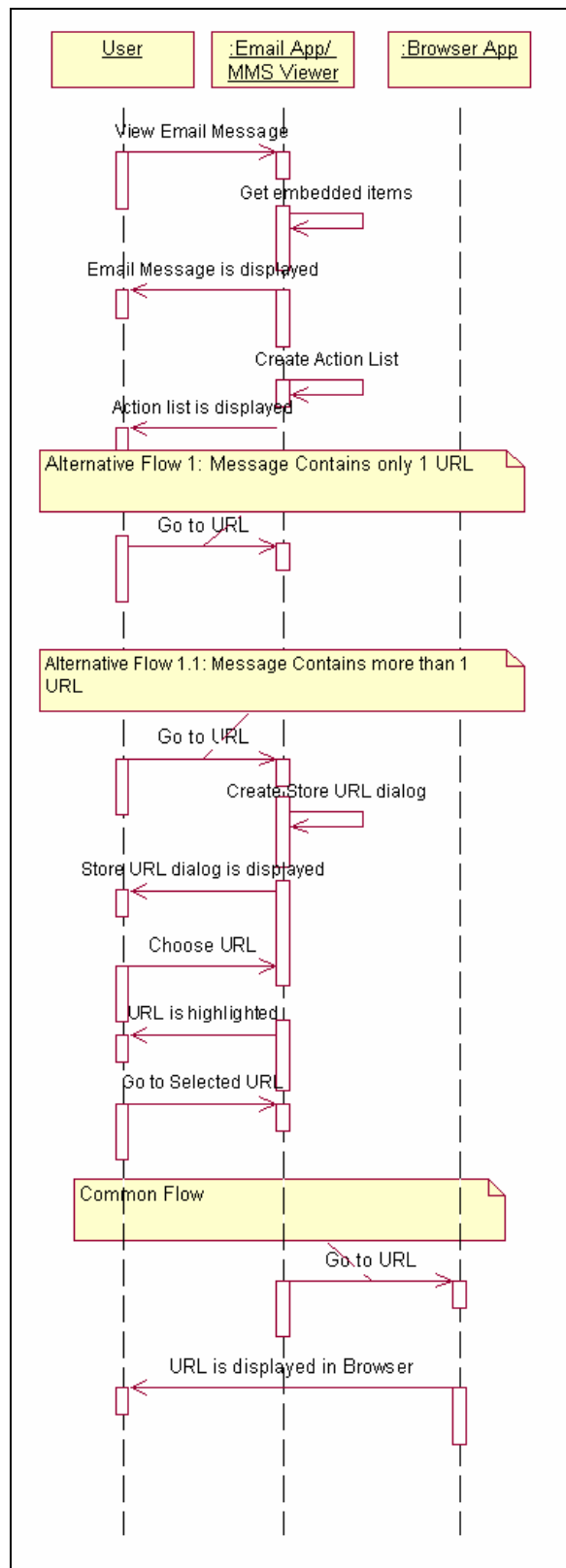


Figura A.9 - Ir para a URL, que está embutida em uma mensagem de email recebida, quando o usuário está visualizando a mensagem.

A.3 Diagramas de seqüência relacionados à “Enviar mensagem ao Número do telefone”

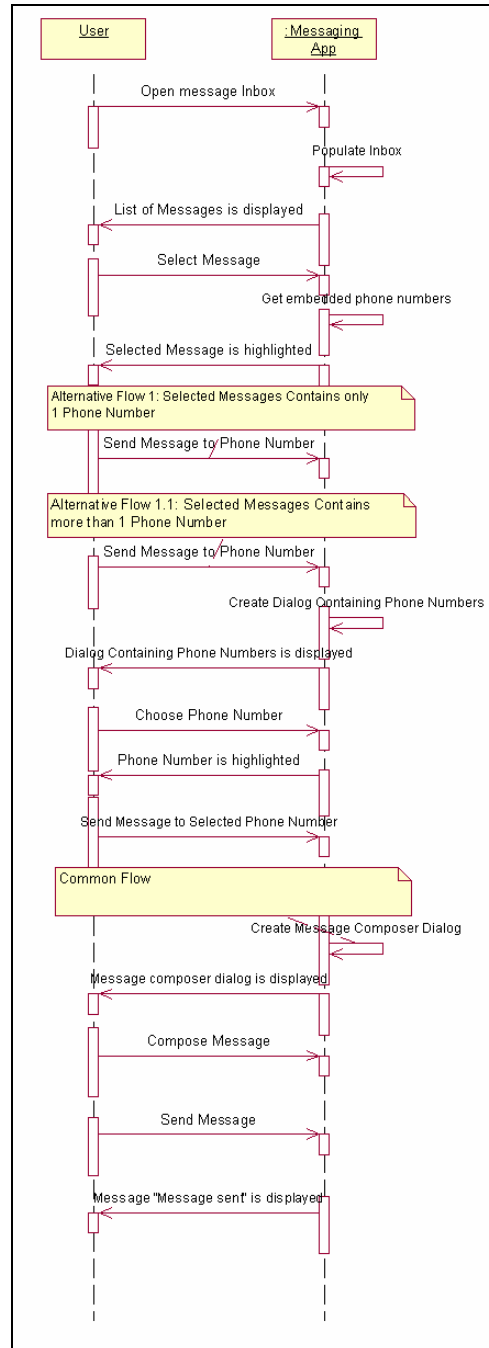


Figura A.10 - Enviar mensagem ao número de telefone embutido no campo “De” de uma mensagem selecionada na caixa de entrada de mensagens.

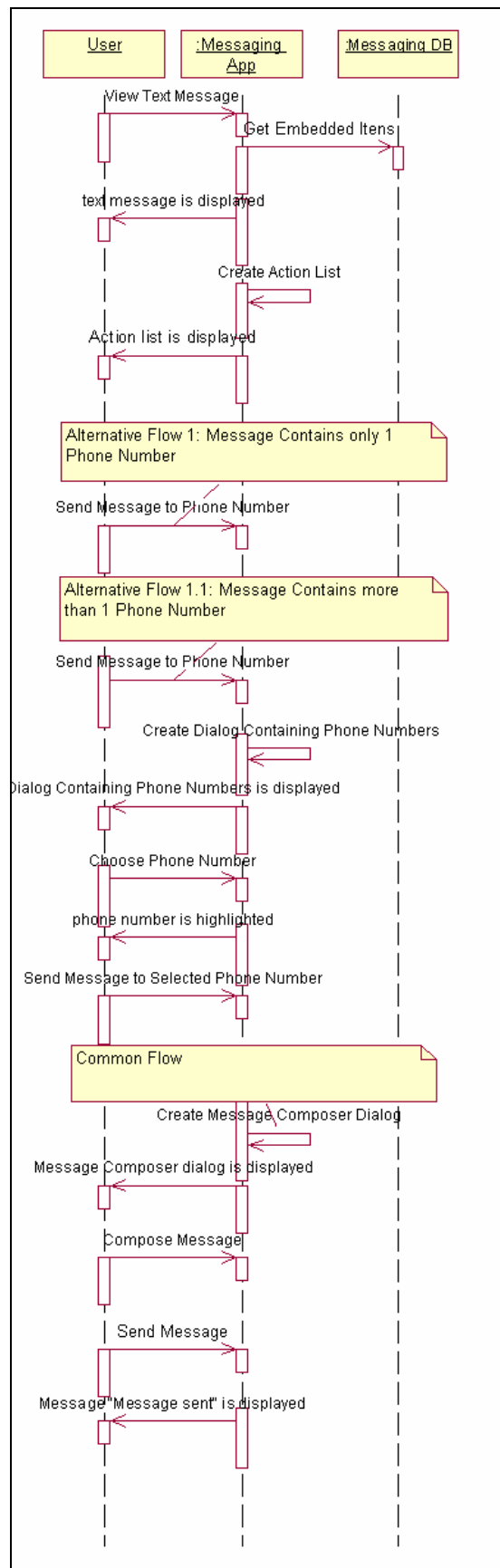


Figura A.11 - Enviar mensagem ao número de telefone embutido em uma mensagem texto recebida, quando o usuário está visualizando a mensagem.

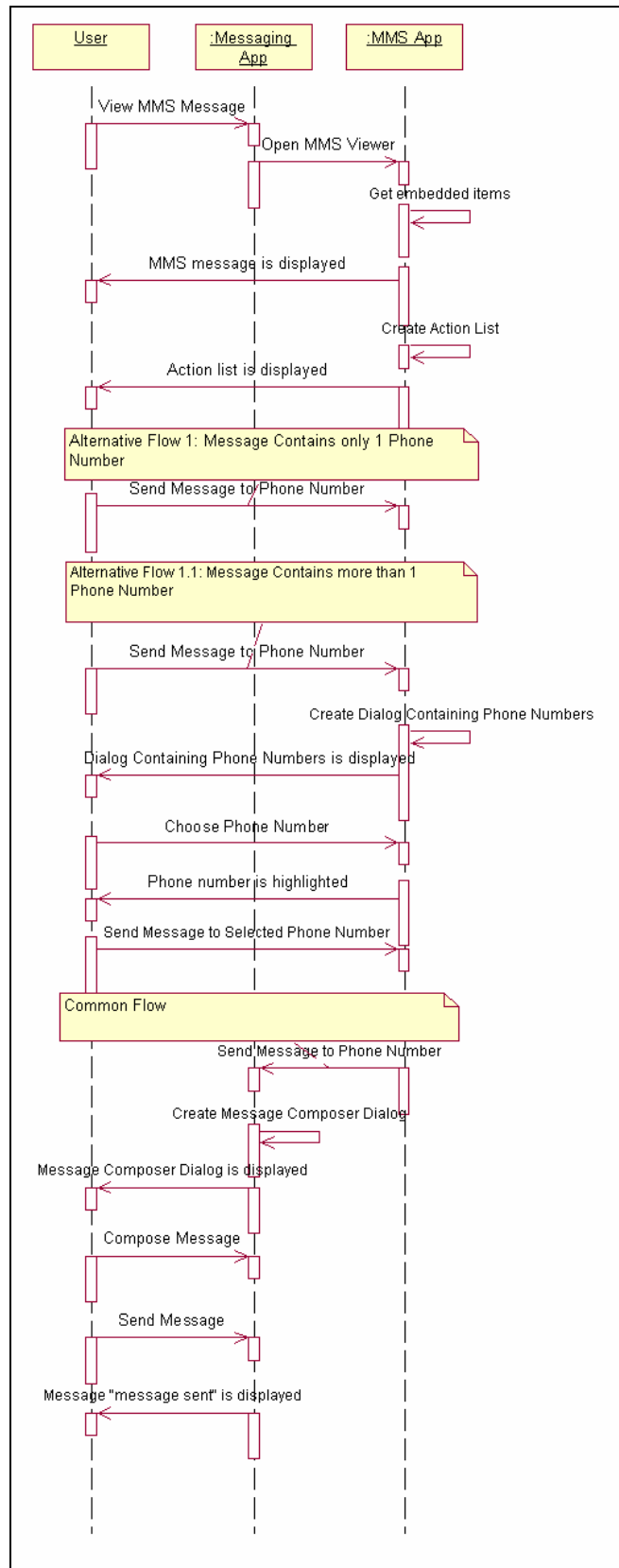


Figura A.12 - Enviar mensagem ao número de telefone embutido em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem.

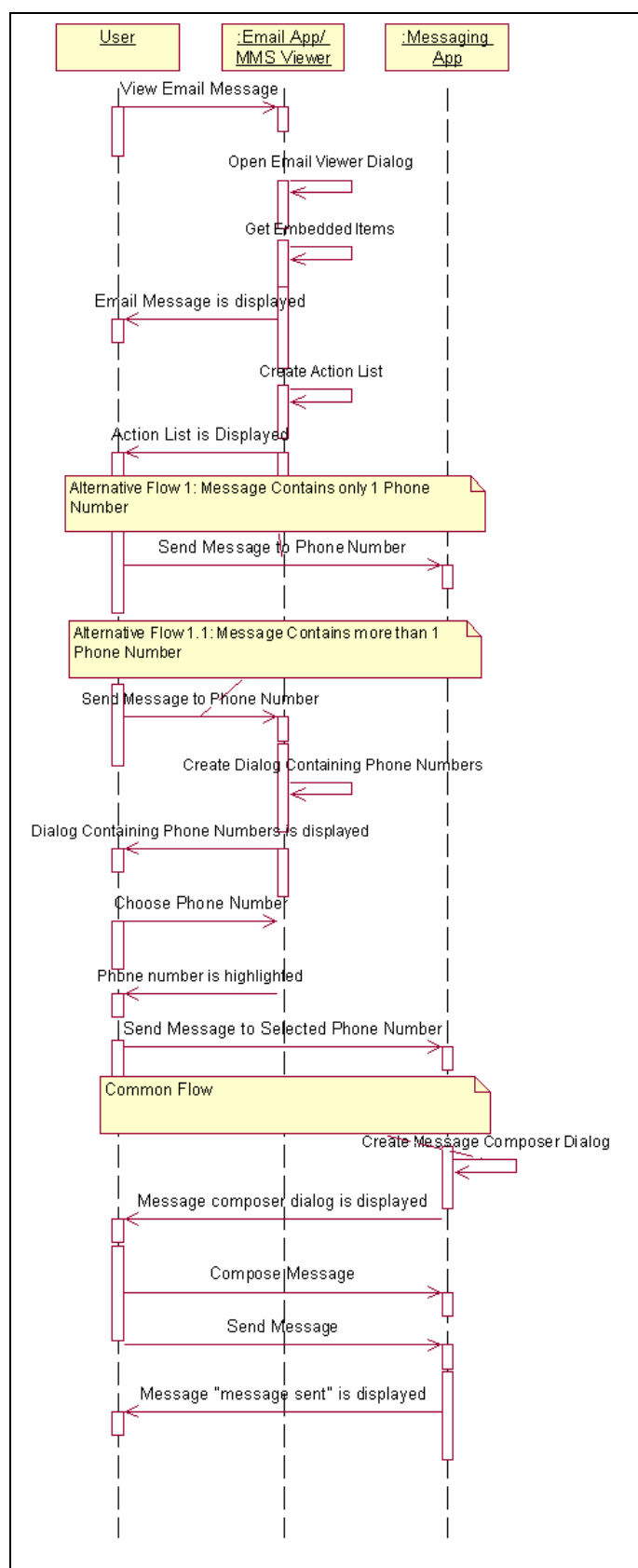


Figura A.13 - Enviar mensagem ao número telefone de embutido em uma mensagem de email recebida, quando o usuário está visualizando a mensagem de email.

A.4 Diagramas de seqüência relacionados à “Enviar mensagem ao endereço de email”

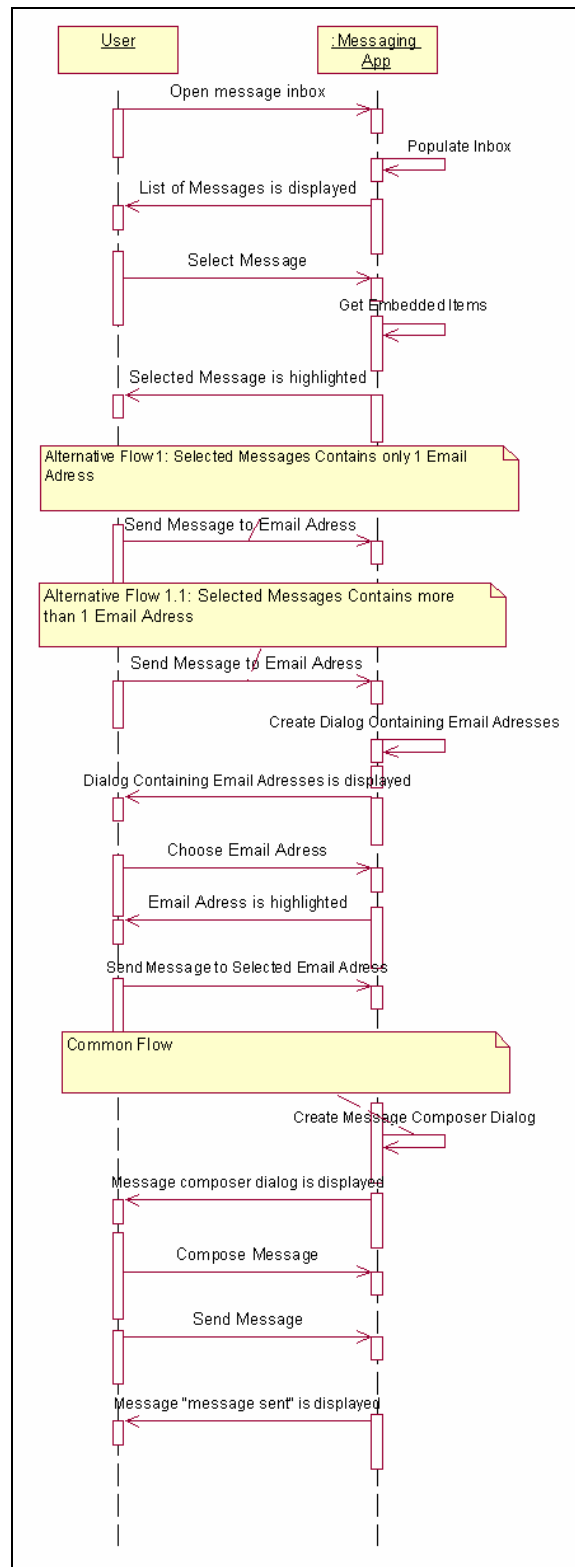


Figura A.14 - Enviar mensagem ao endereço de email embutido no campo “De” de uma mensagem selecionada na caixa de entrada de mensagens.

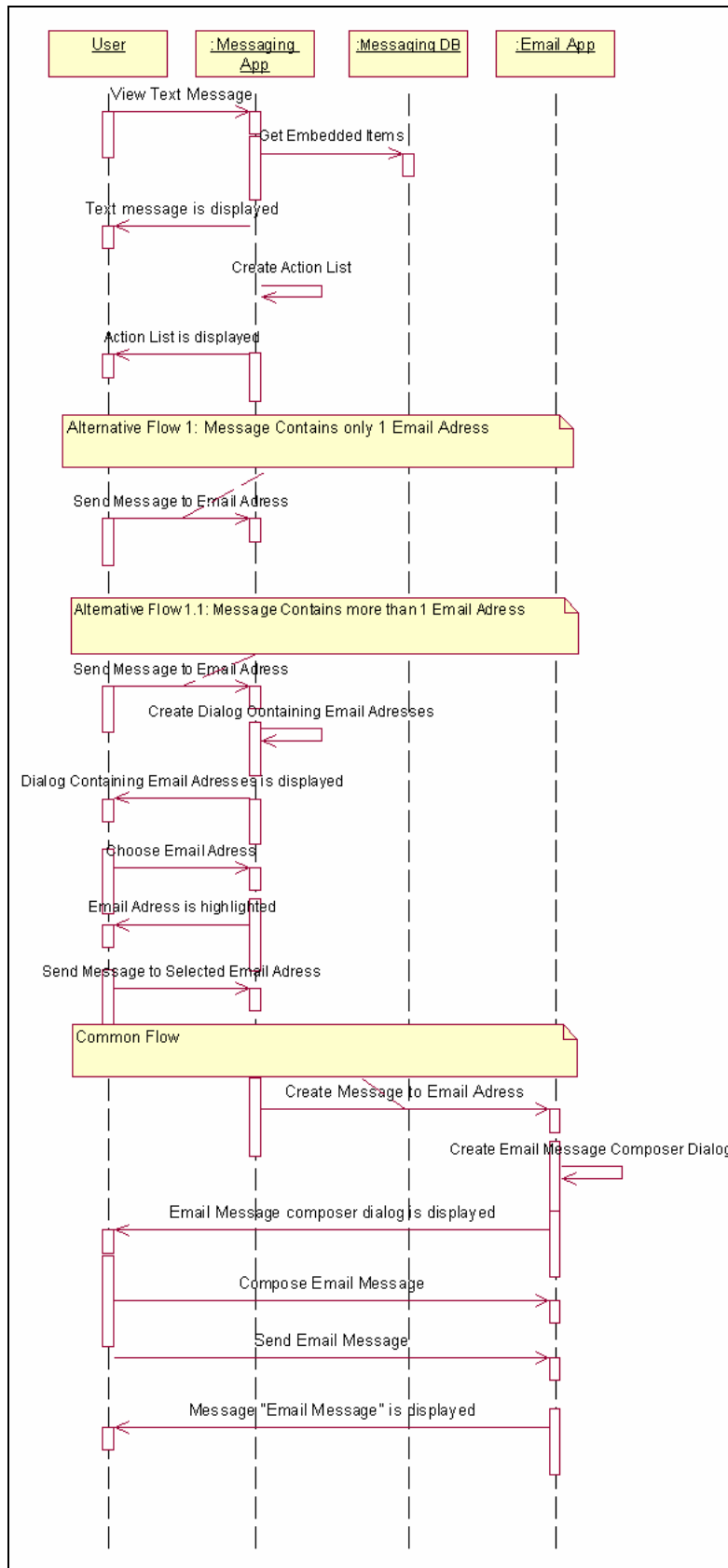


Figura A.15 - Enviar mensagem ao endereço de email embutido em uma mensagem de texto recebida, quando o usuário está visualizando a mensagem.

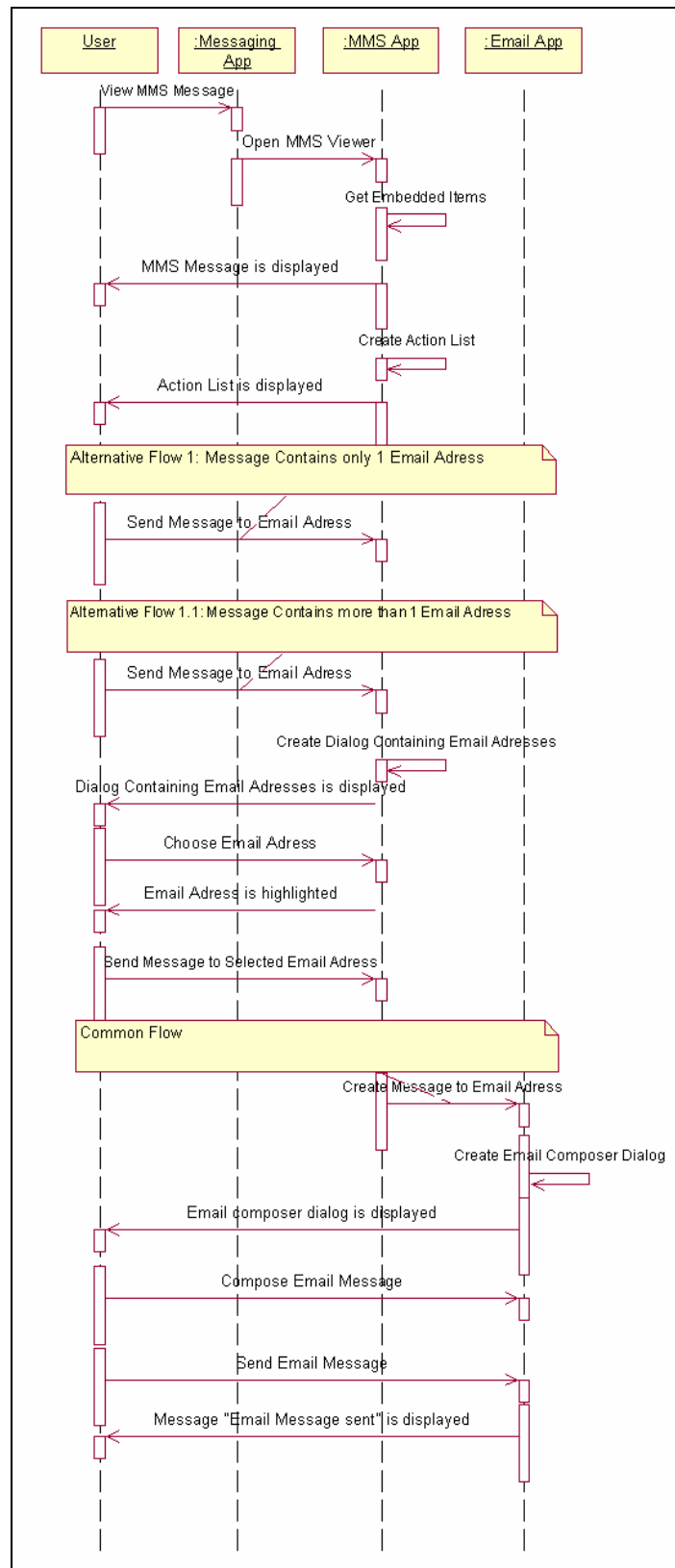


Figura A.16 - Enviar mensagem ao endereço de email embutido em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem.

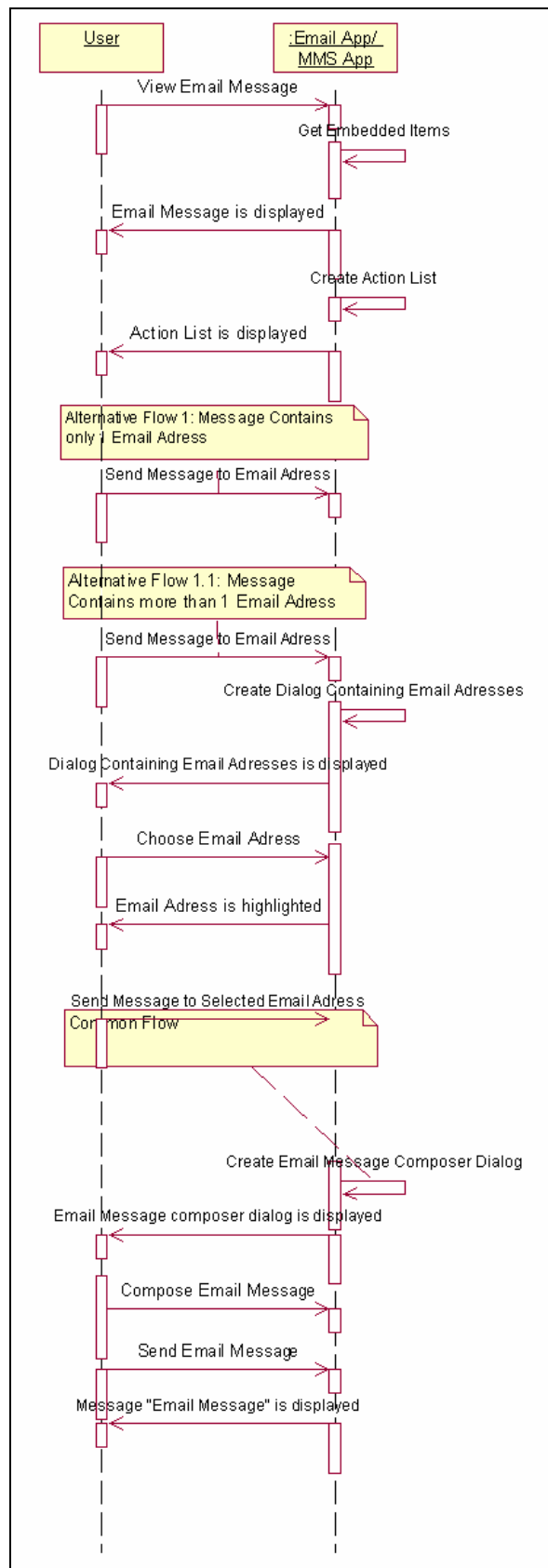


Figura A.17 - Enviar a mensagem ao endereço de email embutido em uma mensagem de email recebida, quando o usuário está visualizando a mensagem de email.

A.5 Diagramas de seqüência relacionados à “Armazenar Número de Telefone”

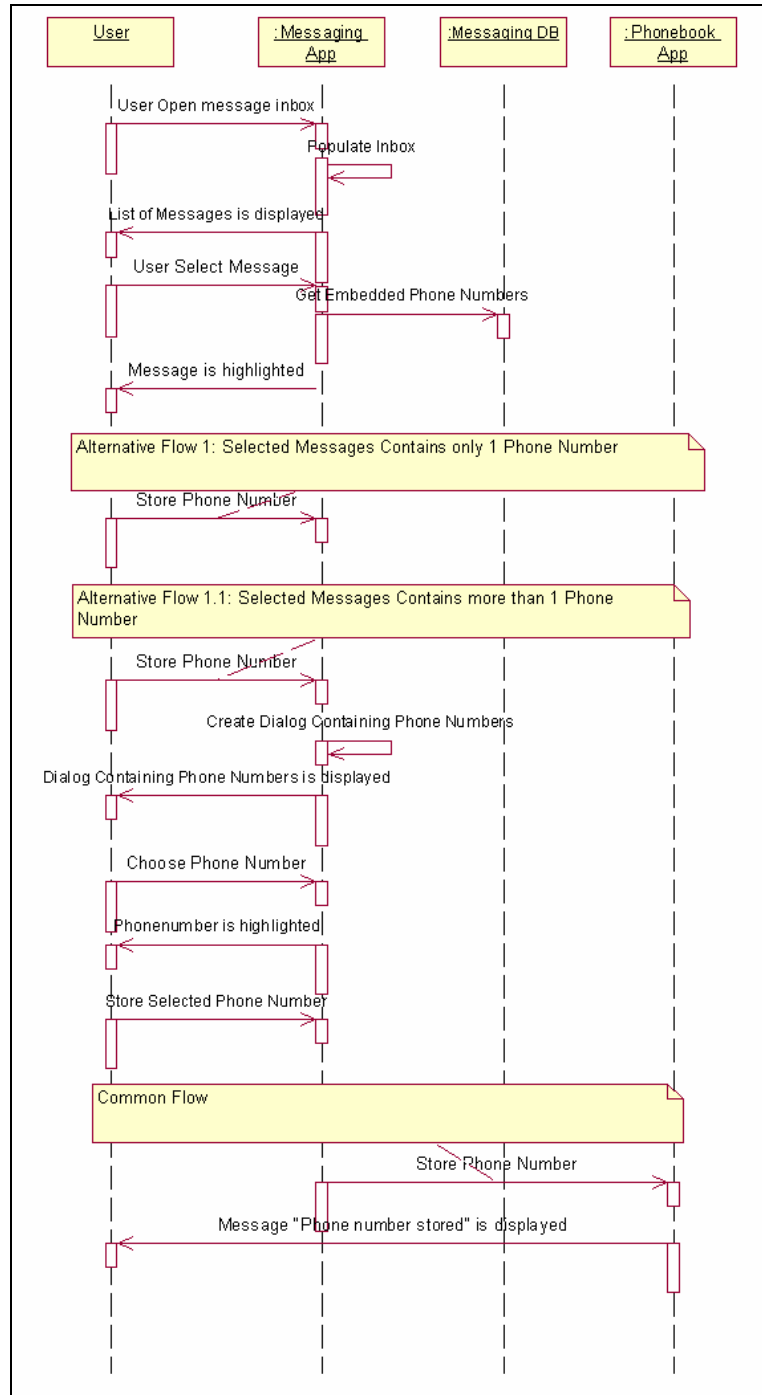


Figura A.18 - Armazenar número de telefone embutido no campo “De” de uma mensagem de texto selecionada na caixa de entrada de mensagens.

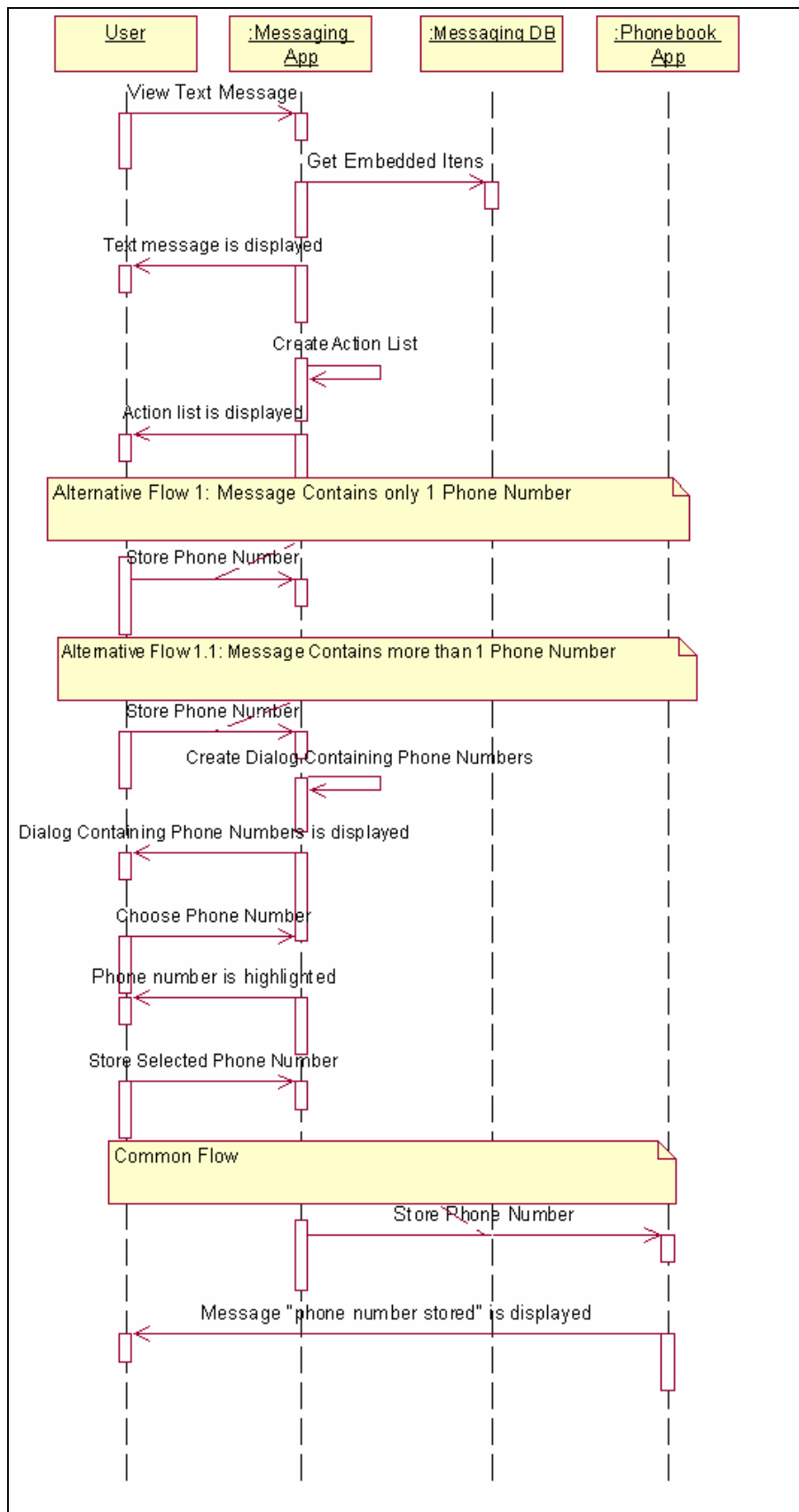


Figura A.19 - Armazenar número de telefone embutido em uma mensagem de texto recebida, quando o usuário está visualizando a mensagem.

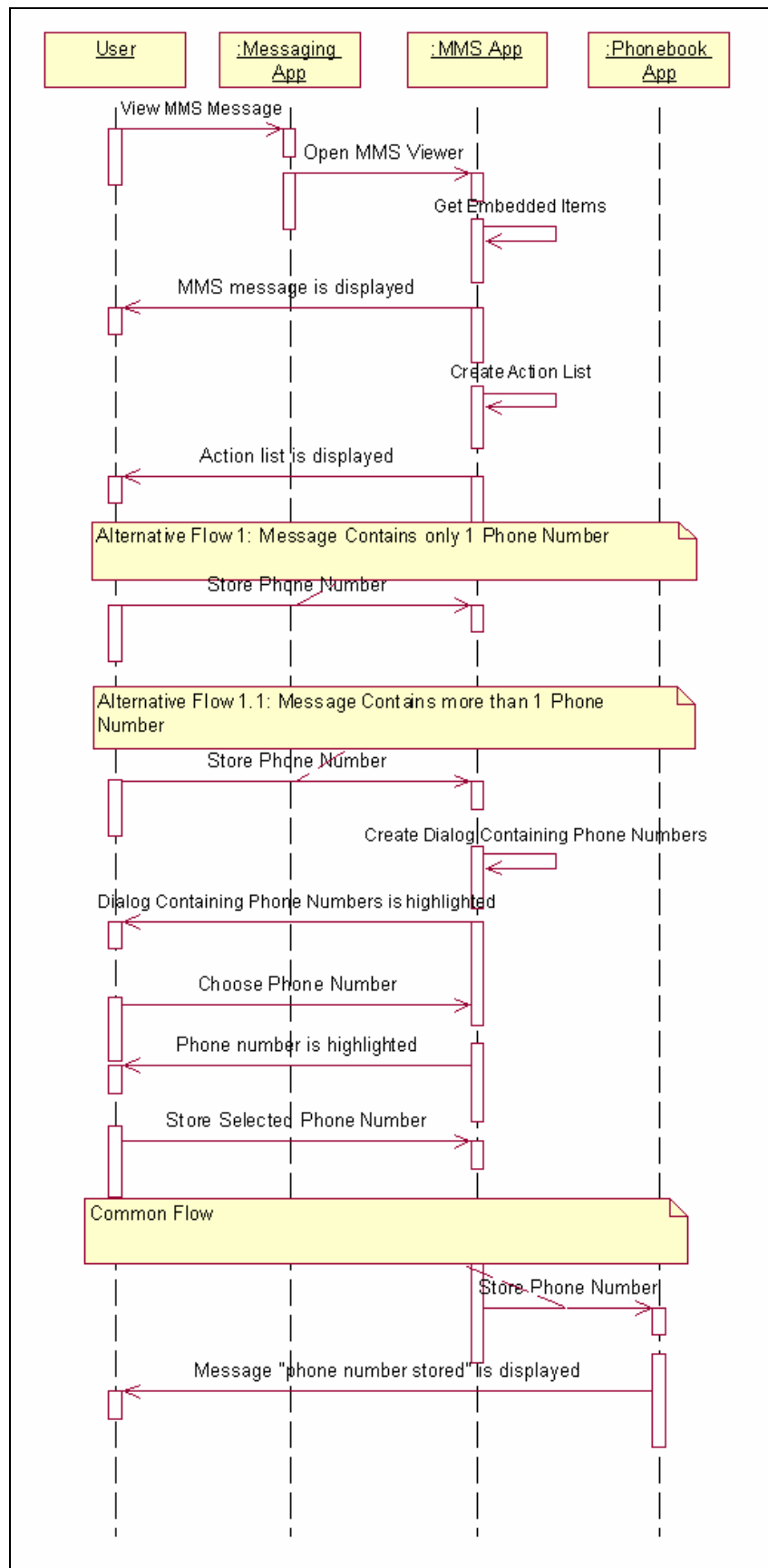


Figura A.20 - Armazenar número de telefone embutido em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem.

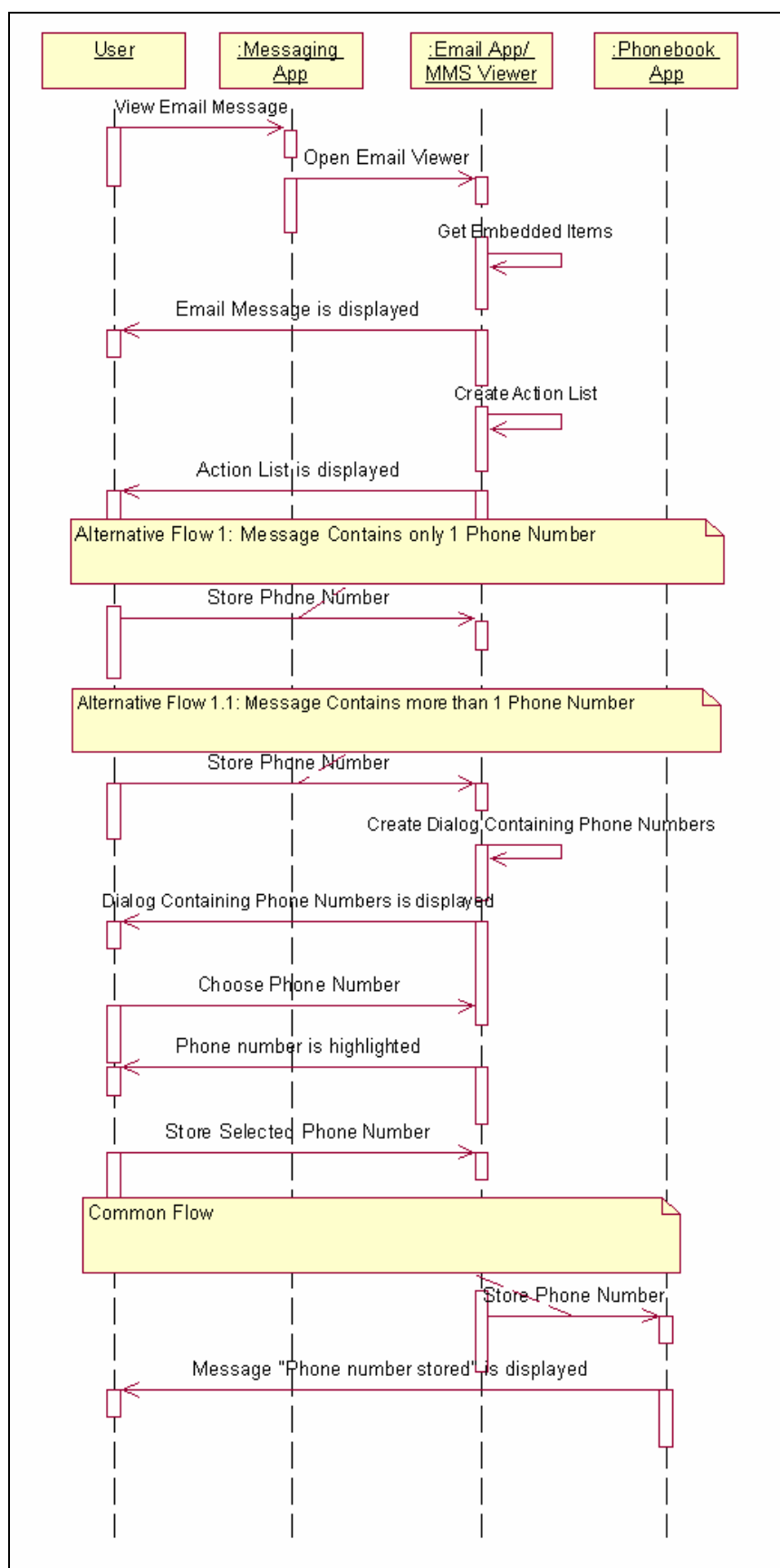


Figura A.21 - Armazenar número de telefone embutido em uma mensagem de email recebida, quando o usuário está visualizando a mensagem.

A.6 Diagramas de seqüência relacionados à “Armazenar endereço de email”

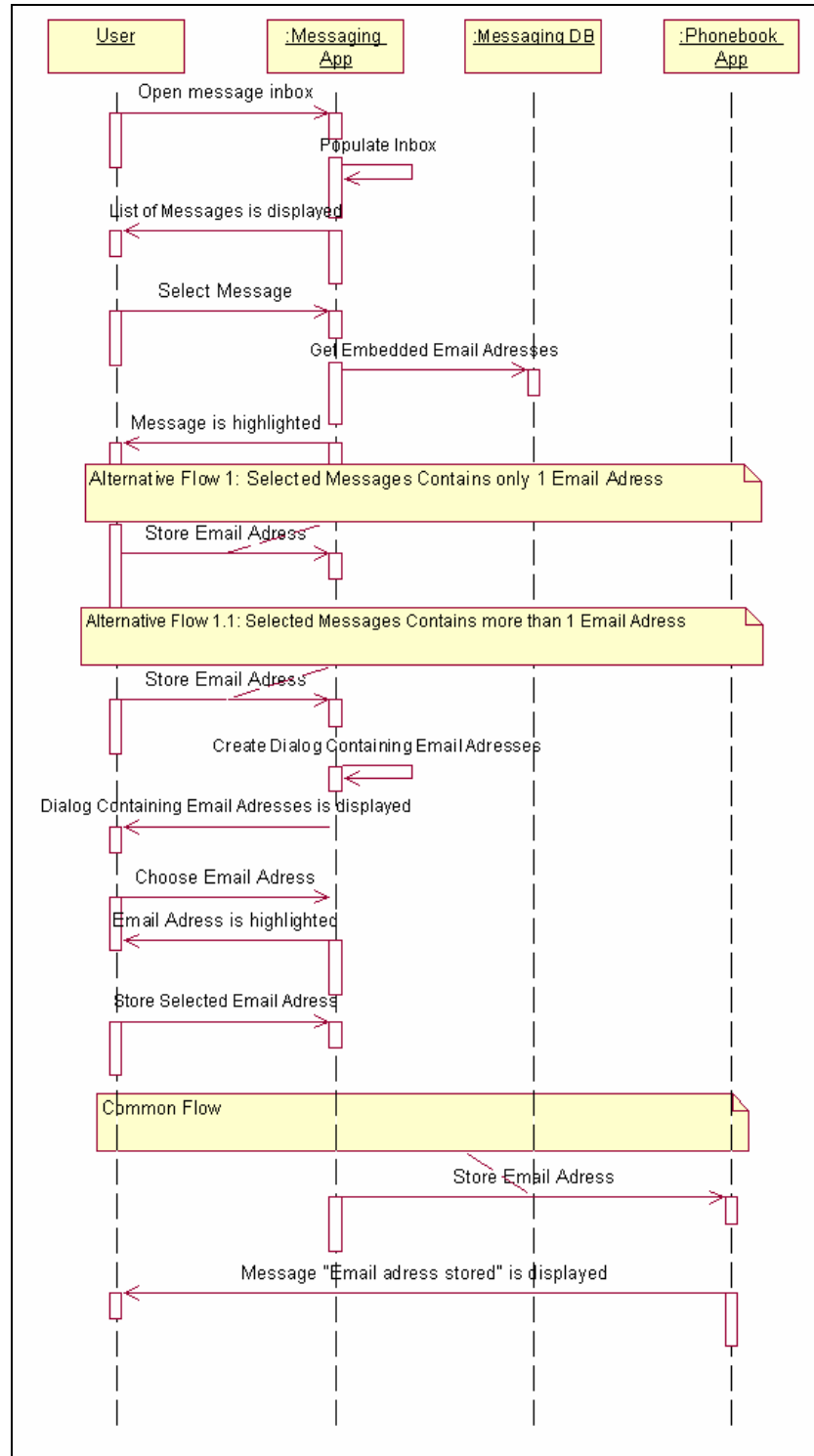


Figura A.22 - Armazenar endereço de email embutido no campo “De” de uma mensagem de texto selecionada na caixa de entrada de mensagens.

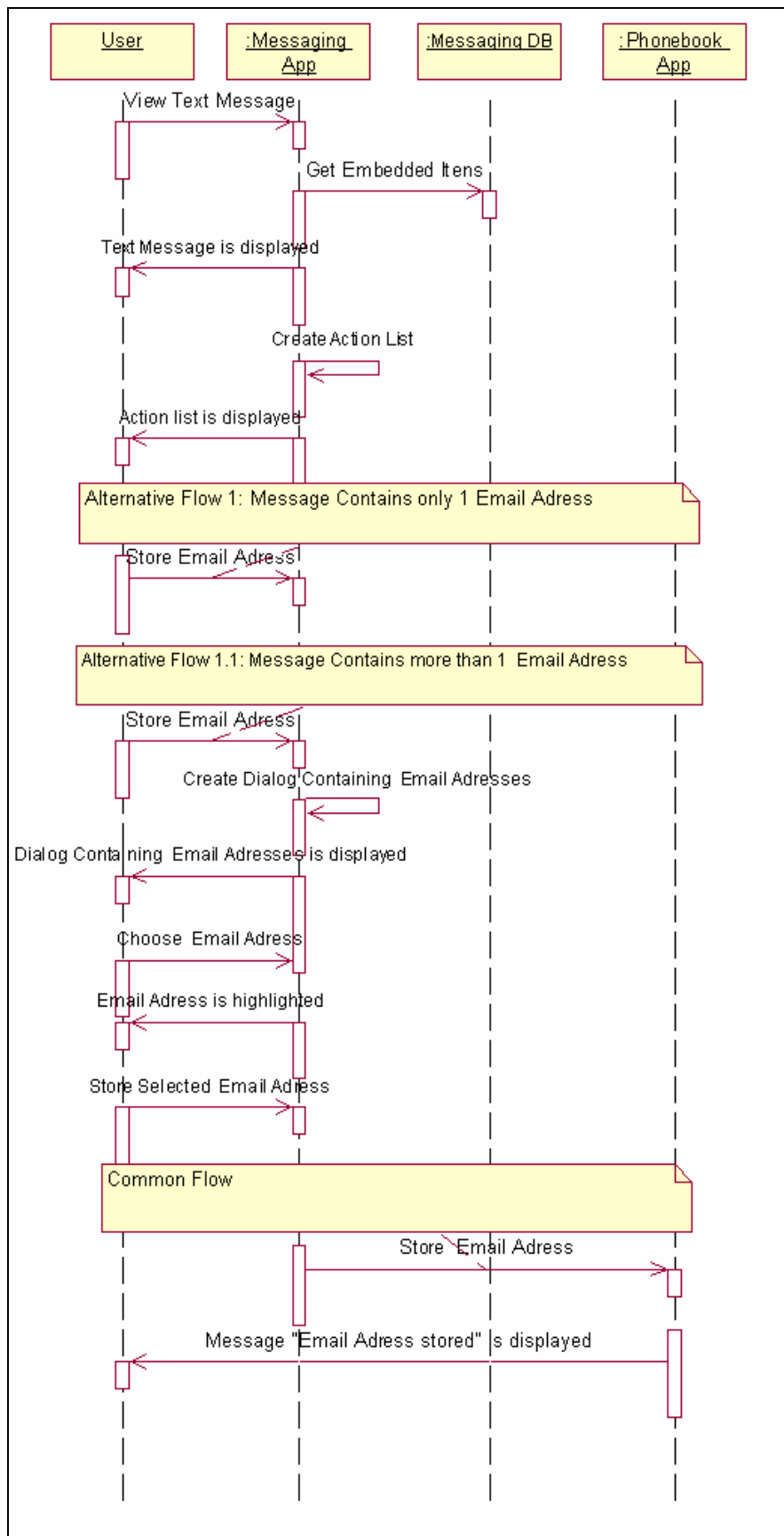


Figura A.23 - Armazenar endereço de email embutido em uma mensagem de texto recebida, quando o usuário está visualizando a mensagem.

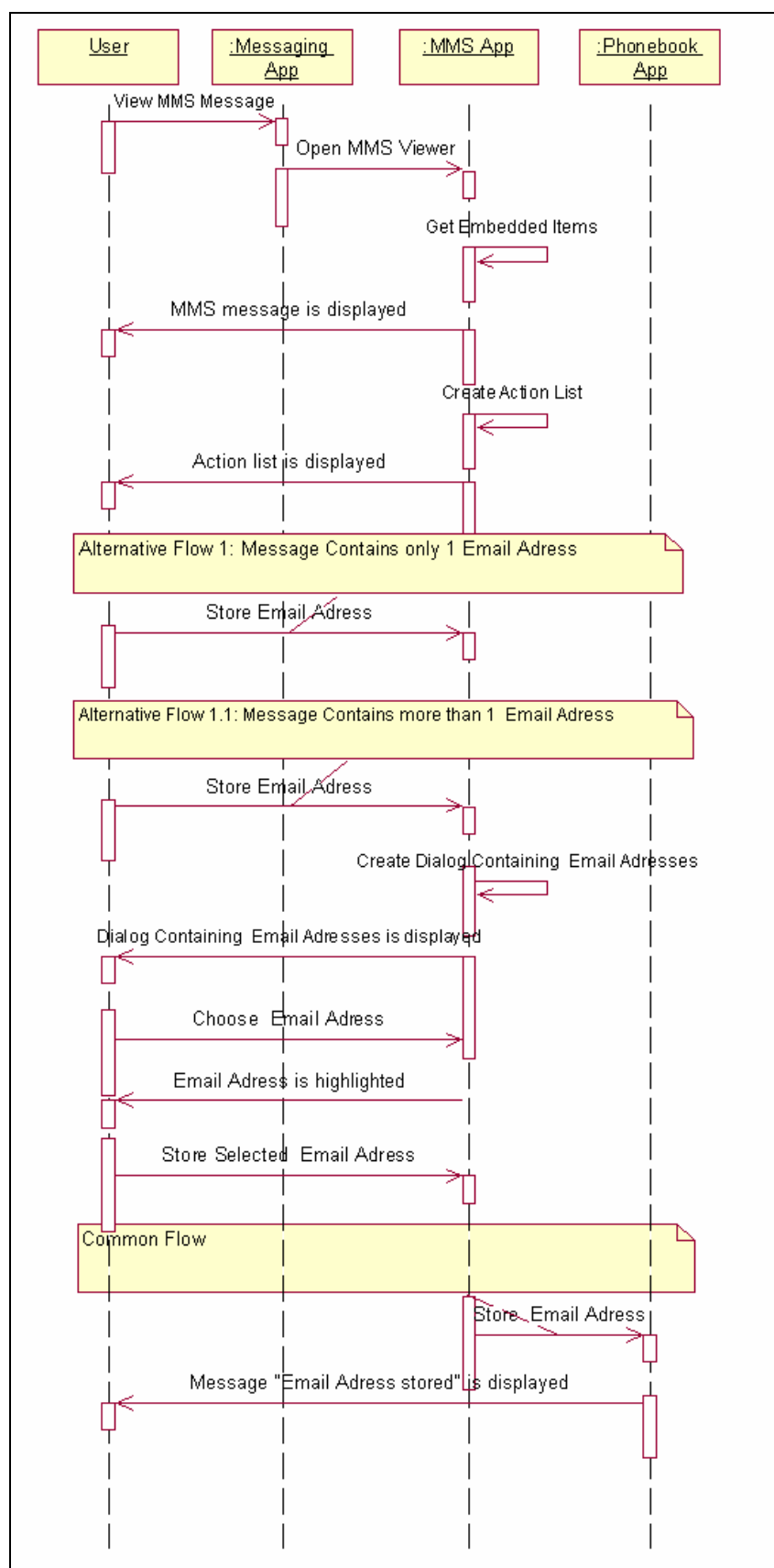


Figura A.24 - Armazenar endereço de email embutido em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem.

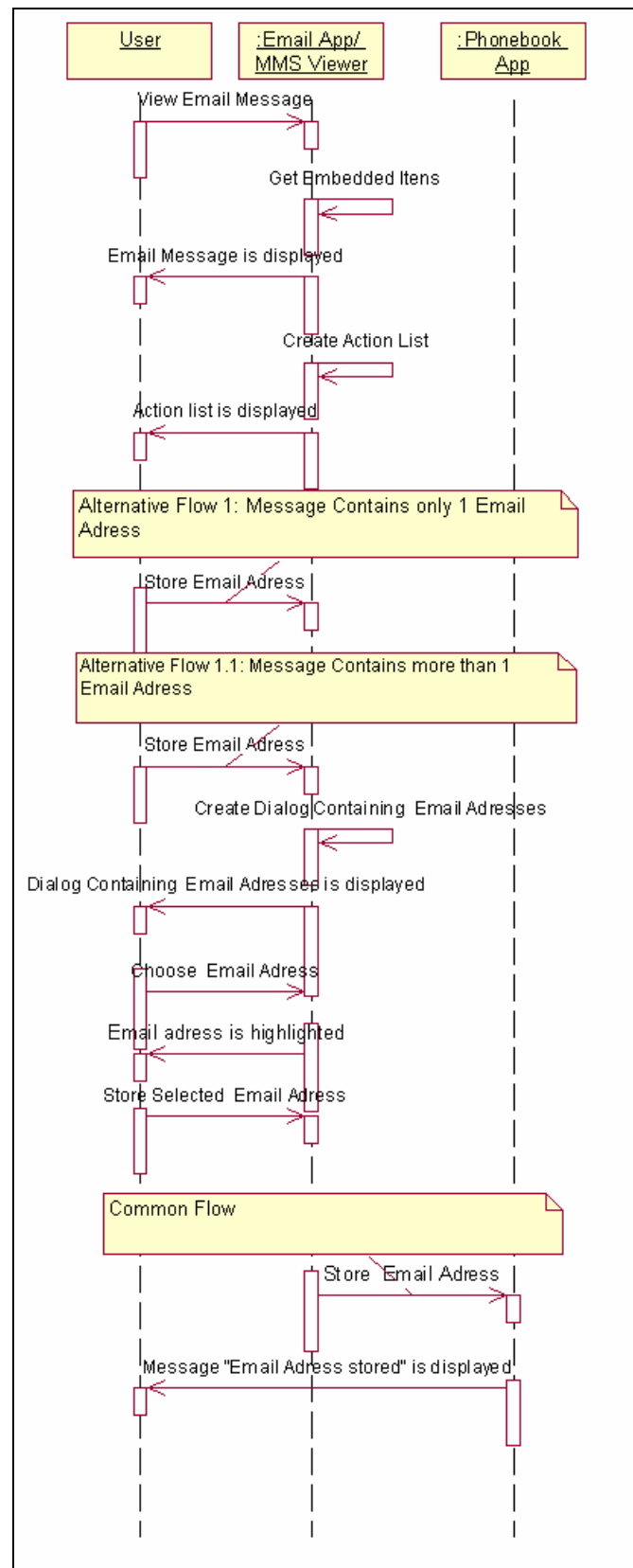


Figura A.25 - Armazenar endereço de email embutido em uma mensagem de email recebida, quando o usuário está visualizando a mensagem.

A.7 Diagramas de seqüência relacionados à “Ligar para o número do telefone”

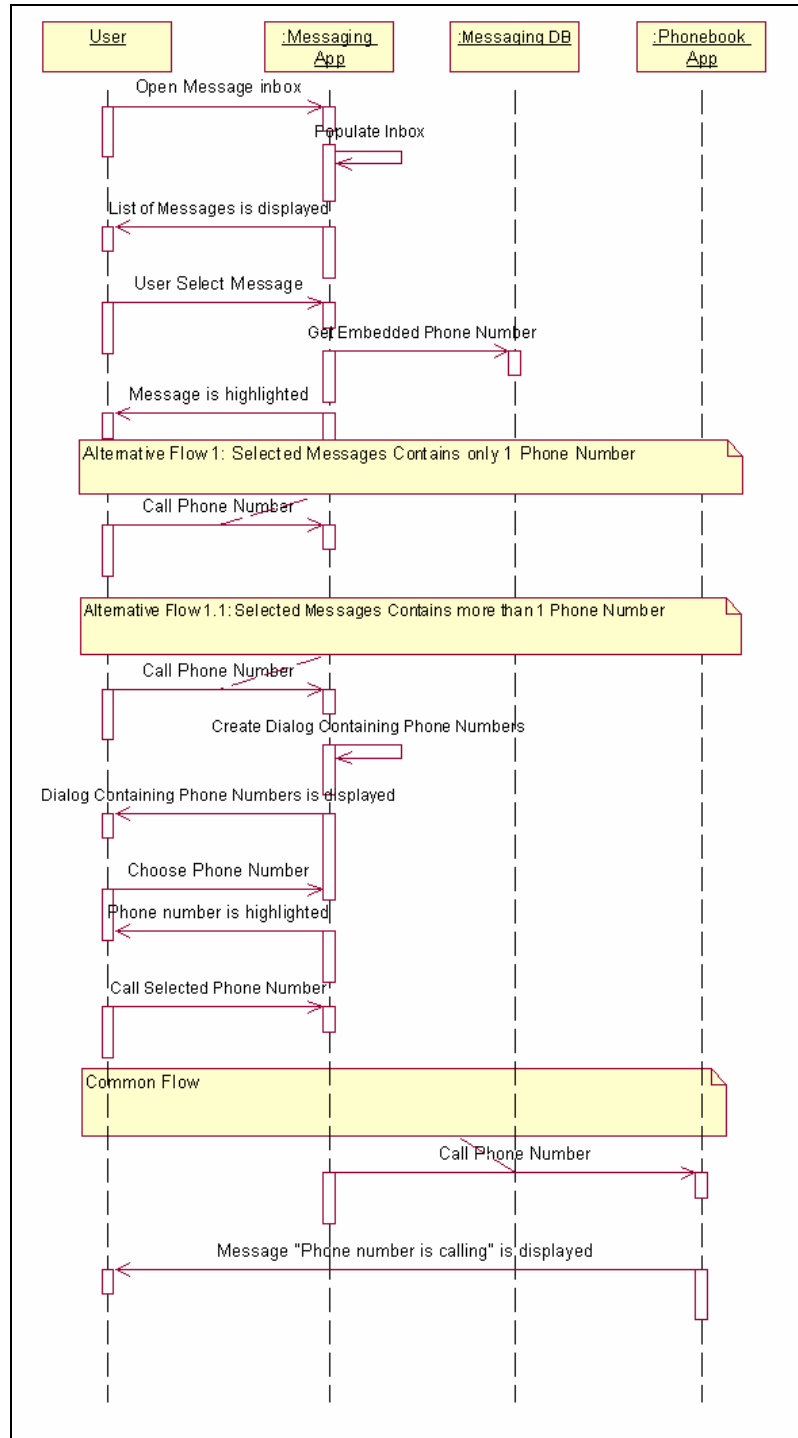


Figura A.26 - Ligar para o número de telefone embutido no campo “De” de uma mensagem de texto selecionada na caixa de entrada de mensagens.

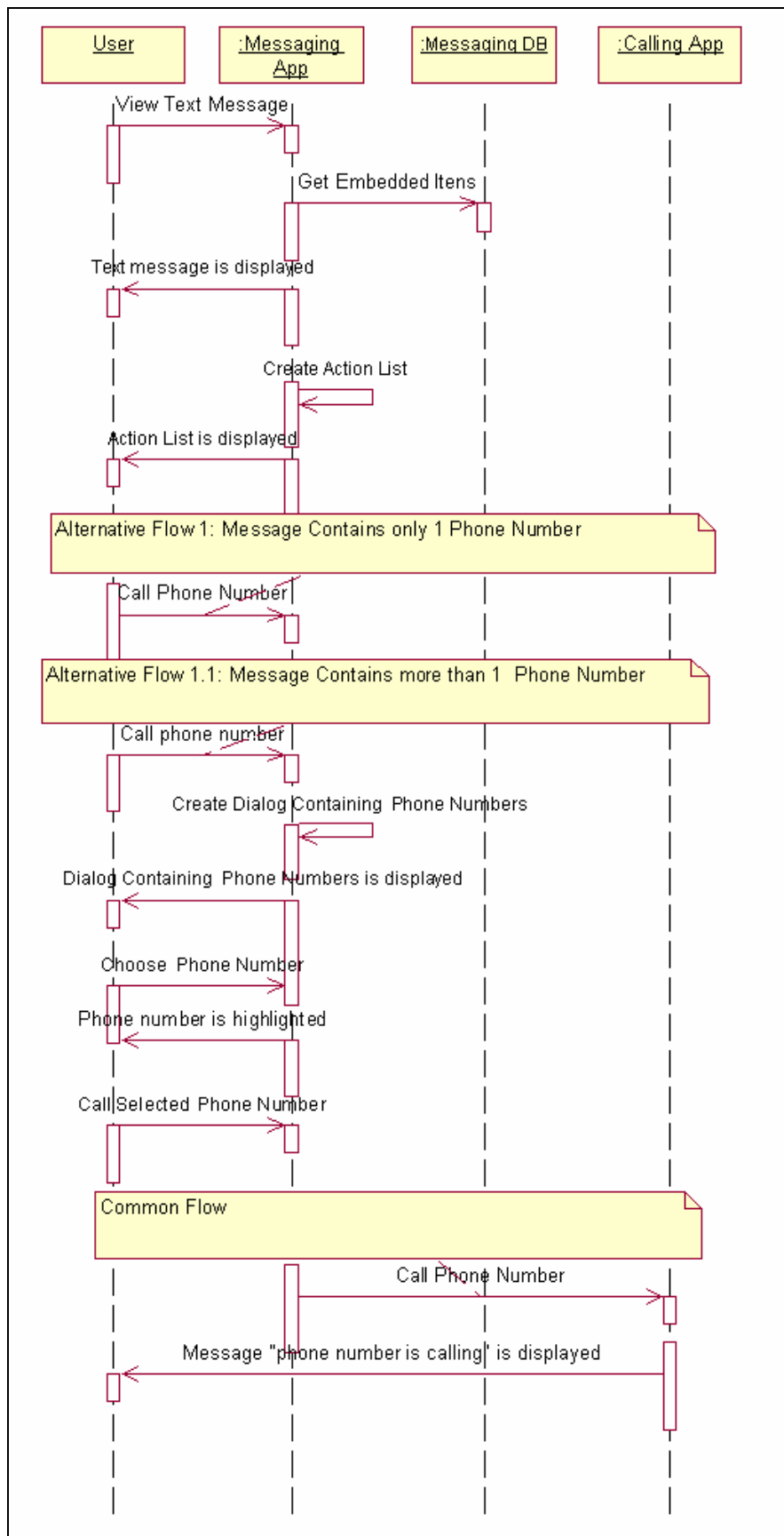


Figura A.27 - Ligar para o número de telefone embutido em uma mensagem de texto recebida, quando o usuário está visualizando a mensagem.

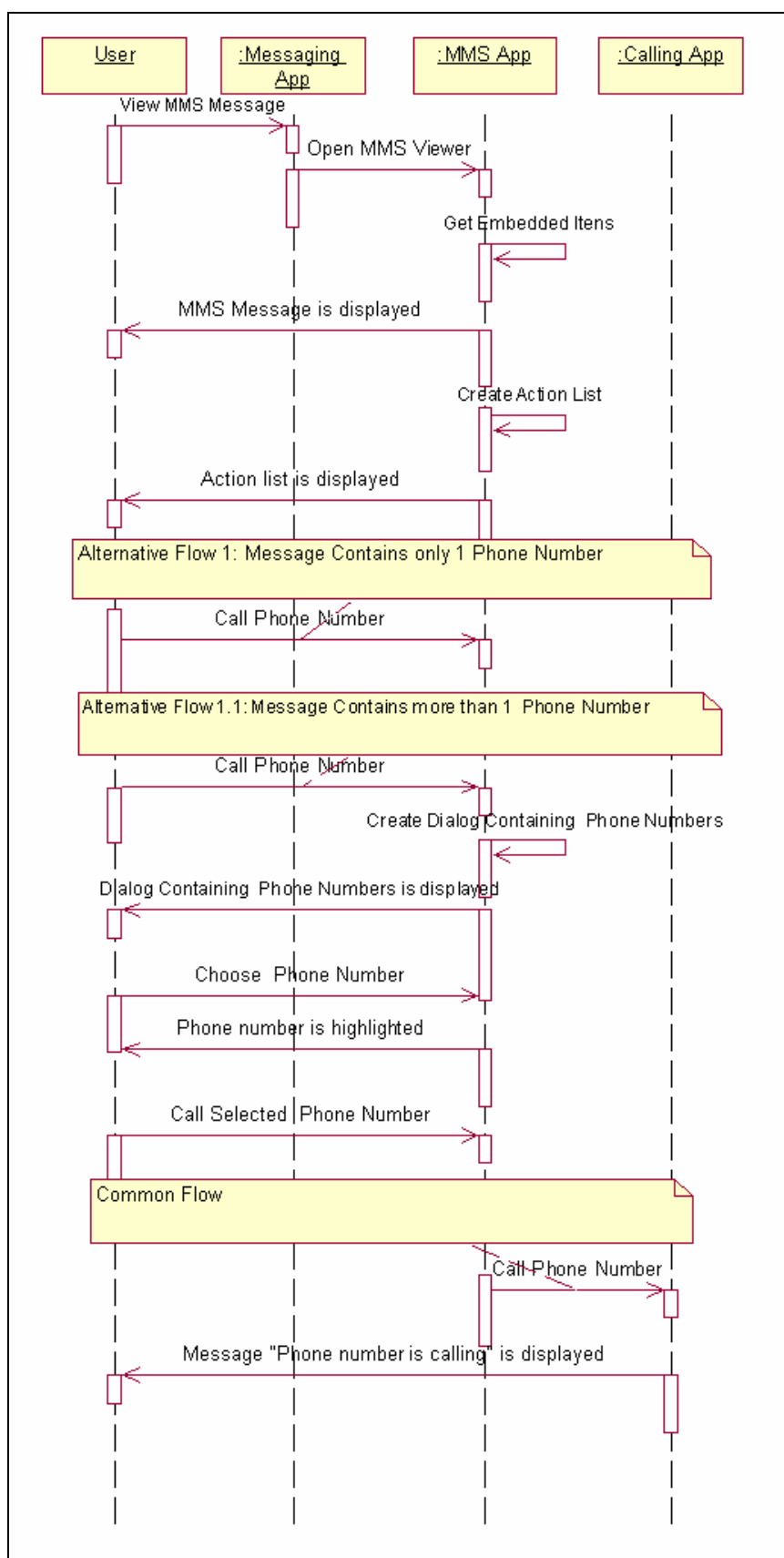


Figura A.28 - Ligar para o número de telefone embutido em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem.

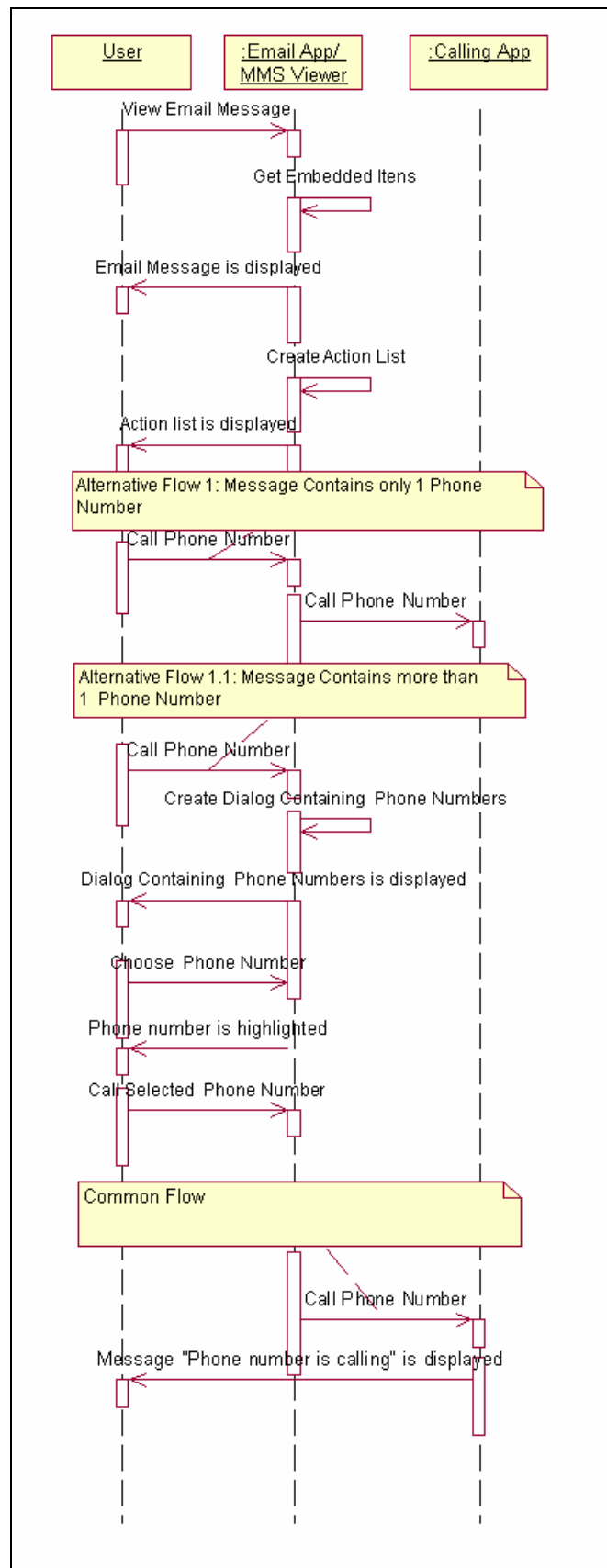


Figura A.29 - Ligar para o número de telefone embutido em uma mensagem de email recebida, quando o usuário está visualizando a mensagem.

A.8 Diagramas de seqüência relacionados à “Enviar uma mensagem de voz a um número de telefone”

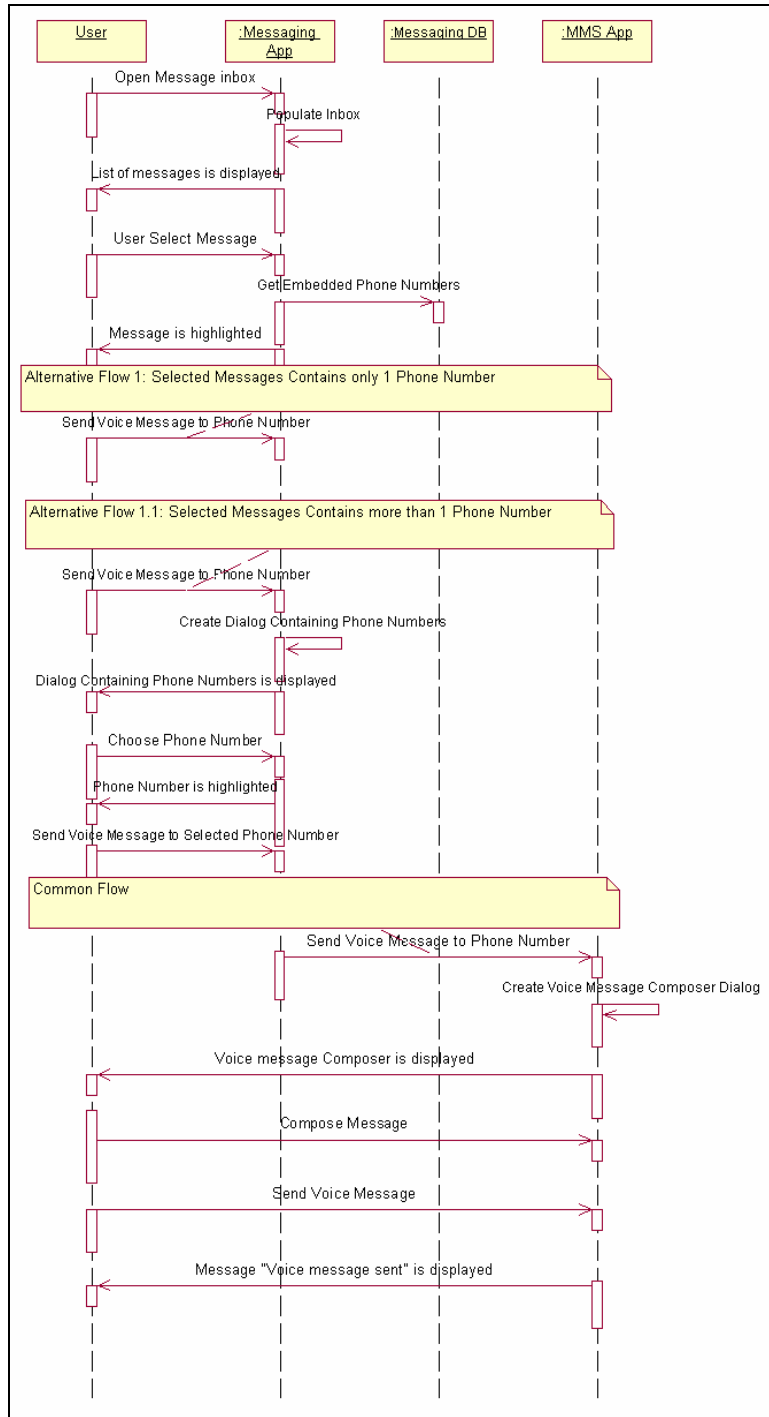


Figura A.30 - Enviar uma mensagem de voz a um número de telefone embutido no campo “De” de uma mensagem de texto selecionada na caixa de entrada de mensagens.

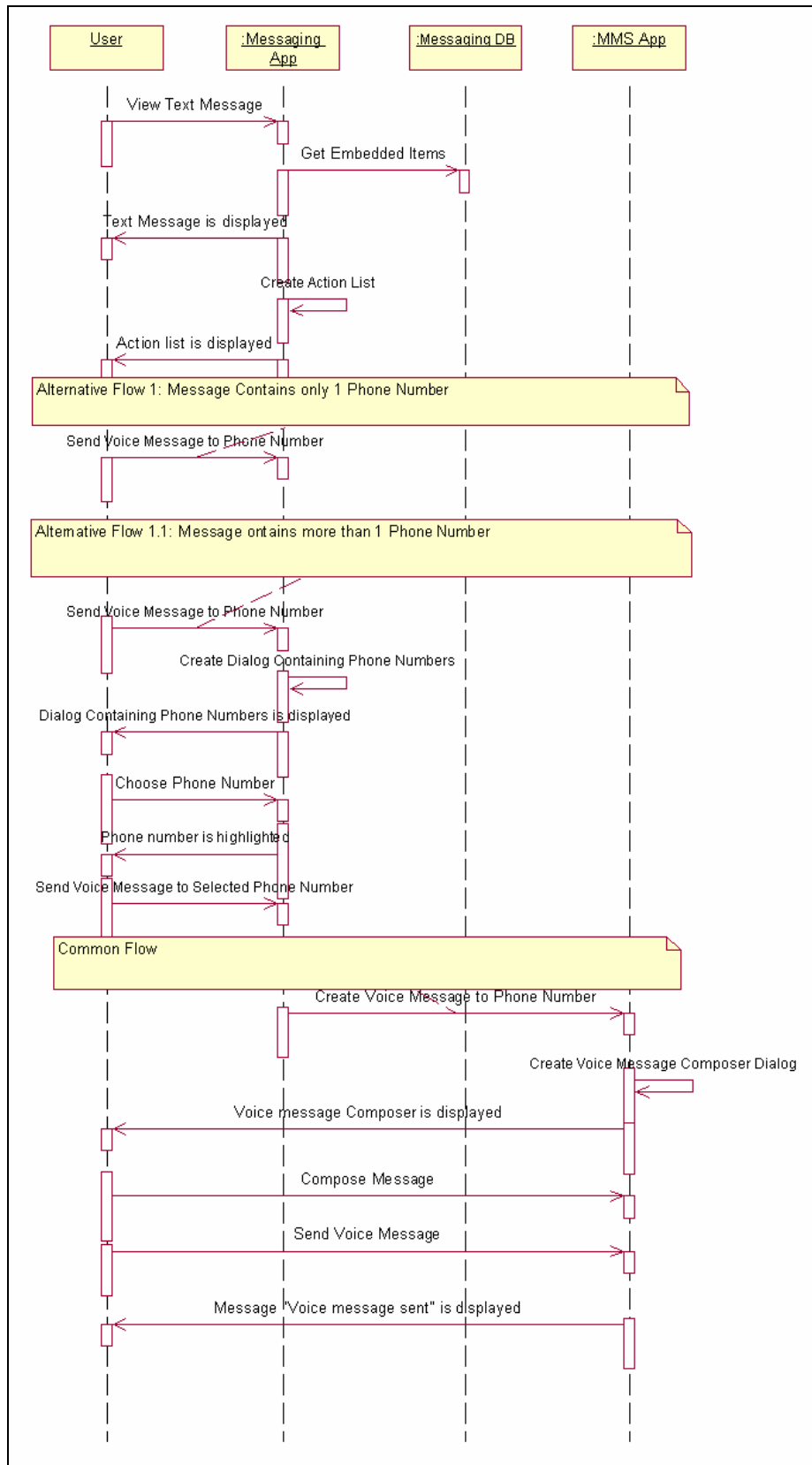


Figura A.31 - Enviar uma mensagem de voz a um número de telefone embutido em uma mensagem de texto recebida, quando o usuário está visualizando a mensagem.

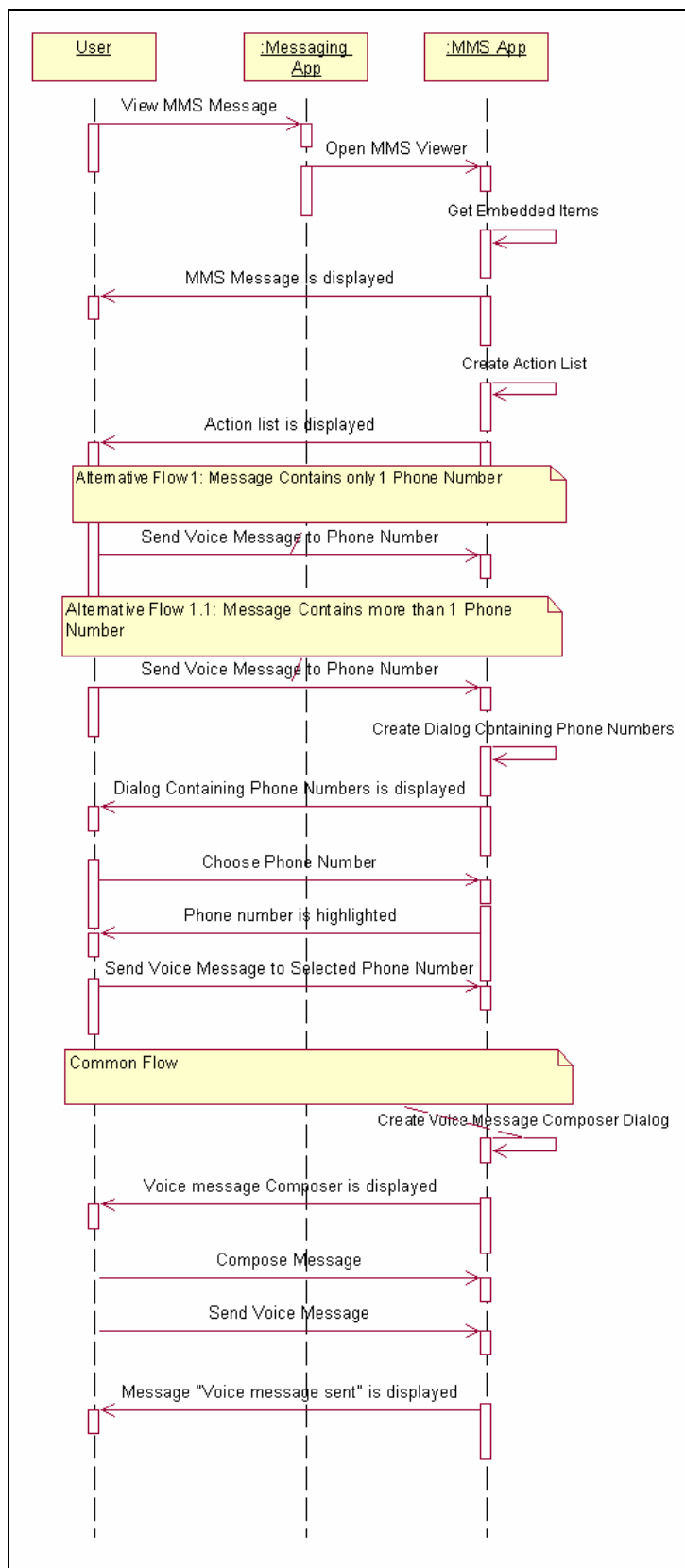


Figura A.32 - Enviar uma mensagem de voz a um número de telefone embutido em uma mensagem multimídia (MMS) recebida, quando o usuário está visualizando a mensagem.

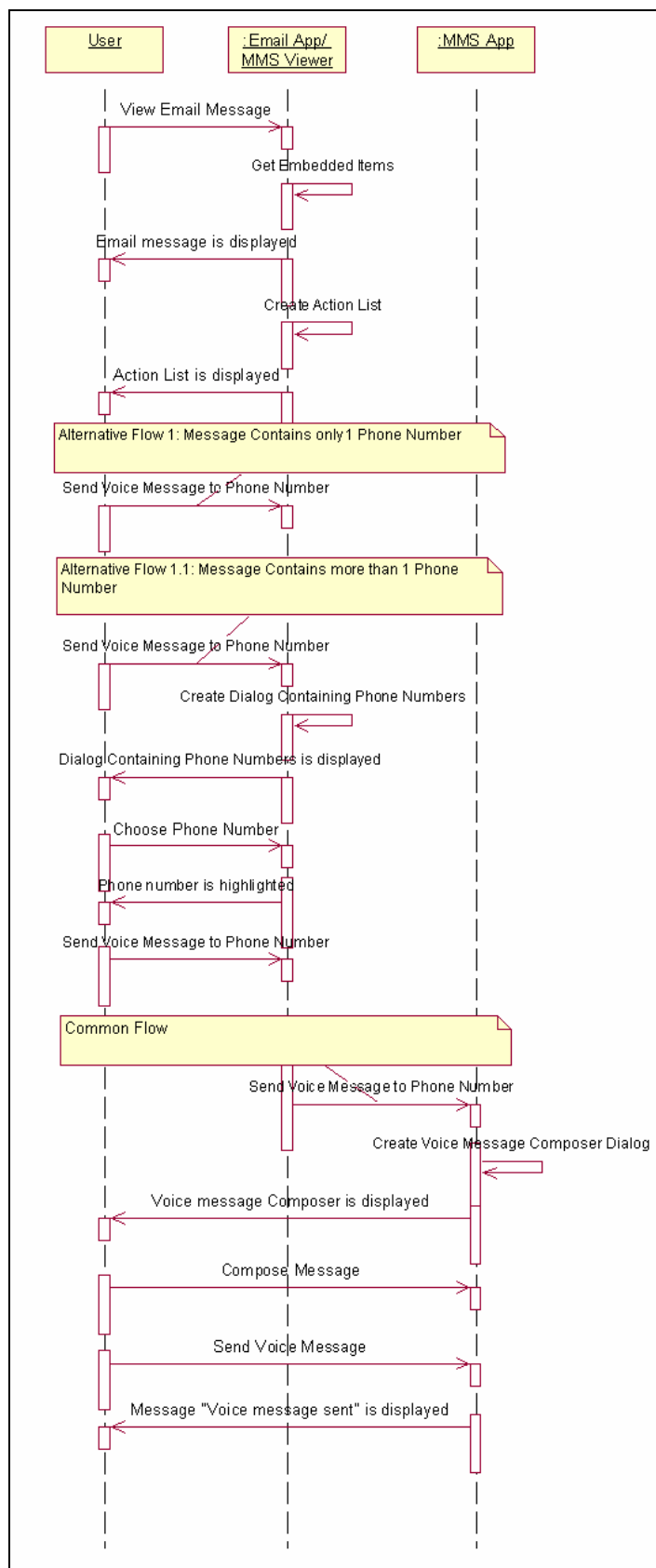


Figura A.33 - Enviar uma mensagem de voz a um número de telefone embutido em uma mensagem de email recebida, quando o usuário está visualizando a mensagem.

Apêndice B

Casos de Teste

Neste apêndice serão mostrados todos os casos de teste relacionados à *feature* “Itens Embutidos em Mensagem”.

Tabela B.1 – Caso de Teste 1 referente ao diagrama de seqüência da Figura A.1

Initial Conditions: Message Contains only 1 URL	
steps	Expected Results
View Text Message	Text Message is displayed Action List is displayed
Store URL	URL location dialog is displayed
Store as Phonebook	message "URL stored" is displayed

Tabela B.2 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.1

Initial Conditions: Message Contains more than 1 URL	
steps	Expected Results
View Text Message	Text Message is displayed Action List is displayed
Store URL	URL location dialog is displayed
Choose URL	URL is highlighted
Store Selected URL	URL location dialog is displayed
Store as Phonebook	message "URL stored" is displayed

Tabela B.3 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.2

Initial Conditions: Message Contains only 1 URL	
steps	Expected Results
View Text Message	Text Message is displayed Action List is displayed
Store URL	URL location dialog is displayed
Store URL as Bookmark	message "URL stored" is displayed

Tabela B.4 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.2

Initial Conditions: Message Contains more than 1 URL	
steps	Expected Results
View Text Message	Text Message is displayed Action List is displayed
Store URL	Store URL dialog is displayed
Choose URL	URL is highlighted
Store Selected URL	URL location dialog is displayed
Store URL as Bookmark	message "URL stored" is displayed

Tabela B.5 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.3

Initial Conditions: Message Contains only 1 URL	
steps	Expected Results
View MMS Message	MMS Message is displayed Action List is displayed
Store URL	URL Location Dialog is displayed
Store URL as Bookmark	message "URL Stored" is displayed

Tabela B.6 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.3

Initial Conditions: Message Contains more than 1 URL	
steps	Expected Results
View MMS Message	MMS Message is displayed Action List is displayed
Store URL	Store URL dialog is displayed
Choose URL	URL is highlighted
Store Selected URL	URL location dialog is displayed
Store URL as Bookmark	message "URL stored" is displayed

Tabela B.7 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.4

Initial Conditions: Message Contains only 1 URL	
steps	Expected Results
View MMS Message	MMS Message is displayed Action List is displayed
Store URL	URL Location Dialog is displayed
Store URL as Phonebook	message "URL Stored" is displayed

Tabela B.8 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.4

Initial Conditions: Message Contains more than 1 URL	
steps	Expected Results
View MMS Message	MMS Message is displayed Action List is displayed
Store URL	Store URL dialog is displayed
Choose URL	URL is highlighted
Store Selected URL	URL location dialog is displayed
Store URL as Phonebook	message "URL stored" is displayed

Tabela B.9 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.5

Initial Conditions: Message Contains only 1 URL	
steps	Expected Results
View Email Message	Email Message is displayed Action List is displayed
Store URL	URL Location Dialog is displayed
Store URL as Bookmark	message "URL Stored" is displayed

Tabela B.10 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.5

Initial Conditions: Message Contains more than 1 URL	
steps	Expected Results
View Email Message	Email Message is displayed Action List is displayed
Store URL	Store URL dialog is displayed
Choose URL	URL is highlighted
Store Selected URL	URL location dialog is displayed
Store URL as Bookmark	message "URL stored" is displayed

Tabela B.11 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.6

Initial Conditions: Message Contains only 1 URL	
steps	Expected Results
View Email Message	Email Message is displayed Action List is displayed
Store URL	URL Location Dialog is displayed
Store URL as Phonebook	message "URL Stored" is displayed

Tabela B.12 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.6

Initial Conditions: Message Contains more than 1 URL	
steps	Expected Results
View Email Message	Email Message is displayed Action List is displayed
Store URL	Store URL dialog is displayed
Choose URL	URL is highlighted
Store Selected URL	URL location dialog is displayed
Store URL as Phonebook	message "URL stored" is displayed

Tabela B.13 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.7

Initial Conditions: Message Contains only 1 URL	
steps	Expected Results
View Text Message	Text Message is displayed Action List is displayed
Go URL	URL is displayed in Browser

Tabela B.14 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.7

Initial Conditions: Message Contains more than 1 URL	
steps	Expected Results
View Text Message	Text Message is displayed Action List is displayed
Go URL	URL store dialog is displayed
Choose URL	URL is highlighted
Go to Selected URL	URL is displayed in Browse

Tabela B.15 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.8

Initial Conditions: Message Contains only 1 URL	
steps	Expected Results
View MMS Message	MMS Message is displayed Action List is displayed
Go URL	URL is displayed in Browser

Tabela B.16 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.8

Initial Conditions: Message Contains more than 1 URL	
steps	Expected Results
MMS Text Message	MMS Message is displayed Action List is displayed
Go URL	URL dialog is displayed
Choose URL	URL is highlighted
Go to Selected URL	URL is displayed in Browser

Tabela B.17 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.9

Initial Conditions: Message Contains only 1 URL	
steps	Expected Results
View Email Message	Email Message is displayed Action List is displayed
Go URL	URL is displayed in Browser

Tabela B.18 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.9

Initial Conditions: Message Contains more than 1 URL	
steps	Expected Results
View Email Message	Email Message is displayed Action List is displayed
Go URL	Store URL dialog is displayed
Choose URL	URL is highlighted
Go to Selected URL	URL is displayed in Browser

Tabela B.19 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.10

Initial Conditions: Selected Messages Contains only 1 Phone Number	
steps	Expected Results
Open message Inbox	List of Messages is displayed
Select Message	Selected Message is highlighted
Send Message to Phone Number	Message composer dialog is displayed
Compose Message Send Message	Message "Message sent" is displayed

Tabela B.20 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.10

Initial Conditions: Selected Messages Contains more than 1 Phone Number	
steps	Expected Results
Open message Inbox	List of Messages is displayed
Select Message	Selected Message is highlighted
Send Message to Phone Number	Dialog Containing Phone Numbers is displayed
Choose Phone Number	Phone Number is highlighted
Send Message to Selected Phone Number	Message composer dialog is displayed
Compose Message Send Message	Message "Message sent" is displayed

Tabela B.21 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.11

Initial Conditions: Message Contains only 1 Phone Number	
steps	Expected Results
View Text Message	text message is displayed Action list is displayed
Send Message to Phone Number	Message Composer dialog is displayed
Compose Message Send Message	Message "Message sent" is displayed

Tabela B.22 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.11

Initial Conditions: Message Contains more than 1 Phone Number	
steps	Expected Results
View Text Message	text message is displayed Action list is displayed
Send Message to Phone Number	Dialog Containing Phone Numbers is displayed
Choose Phone Number	phone number is highlighted
Send Message to Selected Phone Number	Message Composer dialog is displayed
Compose Message Send Message	Message "Message sent" is displayed

Tabela B.23 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.12

Initial Conditions: Message Contains only 1 Phone Number	
steps	Expected Results
View MMS Message	MMS message is displayed Action list is displayed
Send Message to Phone Number	Message Composer Dialog is displayed
Compose Message Send Message	Message "Message sent" is displayed

Tabela B.24 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.12

Initial Conditions: Message Contains more than 1 Phone Number	
steps	Expected Results
View MMS Message	MMS message is displayed Action list is displayed
Send Message to Phone Number	Dialog Containing Phone Numbers is displayed
Choose Phone Number	phone number is highlighted
Send Message to Selected Phone Number	Message Composer dialog is displayed
Compose Message Send Message	Message "Message sent" is displayed

Tabela B.25 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.13

Initial Conditions: Message Contains only 1 Phone Number	
steps	Expected Results
View Email Message	Email message is displayed Action list is displayed
Send Message to Phone Number	Message composer dialog is displayed
Compose Message Send Message	Message "Message sent" is displayed

Tabela B.26 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.13

Initial Conditions: Message Contains more than 1 Phone Number	
steps	Expected Results
View Email Message	Email message is displayed Action list is displayed
Send Message to Phone Number	Dialog Containing Phone Numbers is displayed
Choose Phone Number	phone number is highlighted
Send Message to Selected Phone Number	Message Composer dialog is displayed
Compose Message Send Message	Message "Message sent" is displayed

Tabela B.27 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.14

Initial Conditions: Selected Messages Contains only 1 Email Adress	
steps	Expected Results
Open message inbox	List of Messages is displayed
Select Message	Selected Message is highlighted
Send Message to Email Adress	Message Composer dialog is displayed
Compose Message Send Message	Message "Message sent" is displayed

Tabela B.28 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.14

Initial Conditions: Selected Messages Contains more than 1 Email Adress	
steps	Expected Results
Open message inbox	List of Messages is displayed
Select Message	Selected Message is highlighted
Send Message to Email Adress	Dialog Containing Email Adress is displayed
Choose Email Adress	Email Adress is highlighted

Send Message to Selected Email Adress	Message composer dialog is displayed
Compose Message Send Message	Message "Message sent" is displayed

Tabela B.29 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.15

Initial Conditions: Message Contains only 1 Email Adress	
steps	Expected Results
View Text Message	Text message is displayed Action List is displayed
Send Message to Email Adress	Email Message composer dialog is displayed
Compose Message Send Message	Message "Email Message sent" is displayed

Tabela B.30 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.15

Initial Conditions: Message Contains more than 1 Email Adress	
steps	Expected Results
View Text Message	Text message is displayed Action List is displayed
Send Message to Email Adress	Dialog Containing Email Adresses is displayed
Choose Email Adress	Email Adress is highlighted
Send Message to Selected Email Adress	Email Message composer dialog is displayed
Compose Message Send Message	Message "Email Message sent" is displayed

Tabela B.31 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.16

Initial Conditions: Message Contains only 1 Email Adress	
steps	Expected Results
View MMS Message	MMS message is displayed Action List is displayed
Send Message to Email Adress	Email composer dialog is displayed
Compose Email Message Send Email Message	Message "Email Message sent" is displayed

Tabela B.32 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.16

Initial Conditions: Message Contains more than 1 Email Adress	
steps	Expected Results
View MMS Message	MMS message is displayed Action List is displayed
Send Message to Email Adress	Dialog Containing Email Adresses is displayed
Choose Email Adress	Email Adress is highlighted
Send Message to Selected Email Adress	Email Message composer dialog is displayed
Compose Email Message Send Email Message	Message "Email Message sent" is displayed

Tabela B.33 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.17

Initial Conditions: Message Contains only 1 Email Adress	
steps	Expected Results
View Email Message	Email message is displayed Action List is displayed
Send Message to Email Adress	Email Message composer dialog is displayed
Compose Email Message Send Email Message	Message "Email Message sent" is displayed

Tabela B.34 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.17

Initial Conditions: Message Contains more than 1 Email Adress	
steps	Expected Results
View Email Message	Email message is displayed Action List is displayed
Send Message to Email Adress	Dialog Containing Email Adresses is displayed
Choose Email Adress	Email Adress is highlighted
Send Message to Selected Email Adress	Email Message composer dialog is displayed
Compose Email Message Send Email Message	Message "Email Message sent" is displayed

Tabela B.35 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.18

Initial Conditions: Selected Messages Contains only 1 Phone Number	
steps	Expected Results
User Open message inbox	List of Messages is displayed
User Select Message	Message is highlighted
Store Phone Number	Message "Phone number stored" is displayed

Tabela B.36 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.18

Initial Conditions: Selected Messages Contains more than 1 Phone Number	
steps	Expected Results
User Open message inbox	List of Messages is displayed
User Select Message	Message is highlighted
Store Phone Number	Dialog Containing Phone Numbers is displayed
Choose Phone Number	Phone Number is highlighted
Store Phone Number	Message "Phone number stored" is displayed

Tabela B.37 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.19

Initial Conditions: Message Contains only 1 Phone Number	
steps	Expected Results
View Text Message	Text message is displayed Action List is displayed
Store Phone Number	Message "phone number stored" is displayed

Tabela B.38 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.19

Initial Conditions: Message Contains more than 1 Phone Number	
steps	Expected Results
View Text Message	Text message is displayed Action List is displayed
Store Phone Number	Dialog Containing Phone Numbers is displayed
Choose Phone Number	Phone number is highlighted
Store Selected Phone Number	Message "phone number stored" is displayed

Tabela B.39 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.20

Initial Conditions: Message Contains only 1 Phone Number	
steps	Expected Results
View MMS Message	MMS message is displayed Action List is displayed
Store Phone Number	Message "phone number stored" is displayed

Tabela B.40 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.20

Initial Conditions: Message Contains more than 1 Phone Number	
steps	Expected Results
View MMS Message	MMS message is displayed Action List is displayed
Store Phone Number	Dialog Containing Phone Numbers is displayed
Choose Phone Number	Phone number is highlighted
Store Selected Phone Number	Message "phone number stored" is displayed

Tabela B.41 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.21

Initial Conditions: Message Contains only 1 Phone Number	
steps	Expected Results
View Email Message	Email message is displayed Action List is displayed
Store Phone Number	Message "phone number stored" is displayed

Tabela B.42 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.21

Initial Conditions: Message Contains more than 1 Phone Number	
steps	Expected Results
View Email Message	Email message is displayed Action List is displayed
Store Phone Number	Dialog Containing Phone Numbers is displayed
Choose Phone Number	Phone number is highlighted
Store Selected Phone Number	Message "phone number stored" is displayed

Tabela B.43 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.22

Initial Conditions: Selected Messages Contains only 1 Email Adress	
steps	Expected Results
Open message inbox	List of Messages is displayed
Select Message	Message is highlighted
Store Email Adress	Message " Email Adress stored" is displayed

Tabela B.44 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.22

Initial Conditions: Selected Messages Contains more than 1 Email Adress	
steps	Expected Results
Open message inbox	List of Messages is displayed
Select Message	Message is highlighted
Store Email Adress	Dialog Containing Email Adresses is displayed
Choose Email Adress	Email Adress is highlighted
Store Email Adress	Message " Email Adress stored" is displayed

Tabela B.45 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.23

Initial Conditions: Message Contains only 1 Email Adress	
steps	Expected Results
View Text Message	Text message is displayed Action List is displayed
Store Phone Number	Message " Email Adress stored" is displayed

Tabela B.46 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.23

Initial Conditions: Message Contains more than 1 Email Adress	
steps	Expected Results
View Text Message	Text message is displayed Action List is displayed
Store Email Adress	Dialog Containing Email Adresses is displayed
Choose Email Adress	Email Adress is highlighted
Store Phone Number	Message " Email Adress stored" is displayed

Tabela B.47 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.24

Initial Conditions: Message Contains only 1 Email Adress	
steps	Expected Results
View MMS Message	MMS message is displayed Action List is displayed
Store Email Adress	Message " Email Adress stored" is displayed

Tabela B.48 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.24

Initial Conditions: Message Contains more than 1 Email Adress	
steps	Expected Results
View MMS Message	MMS message is displayed Action List is displayed
Store Email Adress	Dialog Containing Email Adresses is displayed
Choose Email Adress	Email Adress is highlighted
Store Email Adress	Message " Email Adress stored" is displayed

Tabela B.49 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.25

Initial Conditions: Message Contains only 1 Email Adress	
steps	Expected Results
View Email Message	Email message is displayed Action List is displayed
Store Email Adress	Message "Email Adress stored" is displayed

Tabela B.50 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.25

Initial Conditions: Message Contains more than 1 Email Adress	
steps	Expected Results
View Email Message	Email message is displayed Action List is displayed
Store Email Adress	Dialog Containing Email Adresses is displayed
Choose Email Adress	Email Adress is highlighted
Store Email Adress	Message "Email Adress stored" is displayed

Tabela B.51 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.26

Initial Conditions: Selected Messages Contains only 1 Phone Number	
steps	Expected Results
Open message inbox	List of Messages is displayed
User Select Message	Message is highlighted
Call Phone Number	Message "Phone number is calling" is displayed

Tabela B.52 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.26

Initial Conditions: Selected Messages Contains more than 1 Phone Number	
steps	Expected Results
Open message inbox	List of Messages is displayed
User Select Message	Message is highlighted
Call Phone Number	Dialog Containing Phone Numbers is displayed
Choose Phone Number	Phone number is highlighted
Call Phone Number	Message "Phone number is calling" is displayed

Tabela B.53 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.27

Initial Conditions: Message Contains only 1 Phone Number	
steps	Expected Results
View Text Message	Text message is displayed Action List is displayed
Call Phone Number	Message "phone number is calling" is displayed

Tabela B.54 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.27

Initial Conditions: Message Contains more than 1 Phone Number	
steps	Expected Results
View Text Message	Text message is displayed Action List is displayed
Call phone number	Dialog Containing Phone Numbers is displayed
Choose Phone Number	Phone number is highlighted
Call Phone Number	Message "phone number is calling" is displayed

Tabela B.55 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.28

Initial Conditions: Message Contains only 1 Phone Number	
steps	Expected Results
View MMS Message	MMS message is displayed Action List is displayed
Call Phone Number	Message "phone number is calling" is displayed

Tabela B.56 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.28

Initial Conditions: Message Contains more than 1 Phone Number	
steps	Expected Results
View MMS Message	MMS message is displayed Action List is displayed
Call Phone Number	Dialog Containing Phone Numbers is displayed
Choose Phone Number	Phone number is highlighted
Call Phone Number	Message "phone number is calling" is displayed

Tabela B.57 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.29

Initial Conditions: Message Contains only 1 Phone Number	
steps	Expected Results
View Email Message	Email message is displayed Action List is displayed
Call Phone Number	Message "phone number is calling" is displayed

Tabela B.58 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.29

Initial Conditions: Message Contains more than 1 Phone Number	
steps	Expected Results
View Email Message	Email message is displayed Action List is displayed
Call Phone Number	Dialog Containing Phone Numbers is displayed
Choose Phone Number	Phone number is highlighted
Call Phone Number	Message "phone number is calling" is displayed

Tabela B.59 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.30

Initial Conditions: Selected Messages Contains only 1 Phone Number	
steps	Expected Results
Open message inbox	List of Messages is displayed
User Select Message	Message is highlighted
Send Voice Message to Phone Number	Voice message Composer is displayed
Compose Message Send Voice Message	Message "Voice message sent" is displayed

Tabela B.60 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.30

Initial Conditions: Selected Messages Contains more than 1 Phone Number	
steps	Expected Results
Open message inbox	List of Messages is displayed
User Select Message	Message is highlighted
Send Voice Message to Phone Number	Dialog Containing Phone Numbers is displayed
Choose Phone Number	Phone Number is highlighted
Send Voice Message to Phone Number	Voice message Composer is displayed
Compose Message Send Voice Message	Message "Voice message sent" is displayed

Tabela B.61 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.31

Initial Conditions: Message Contains only 1 Phone Number	
steps	Expected Results
View Text Message	Text Message is displayed Action List is displayed
Send Voice Message to Phone Number	Voice message Composer is displayed
Compose Message Send Voice Message	Message "Voice message sent" is displayed

Tabela B.62 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.31

Initial Conditions: Message ontains more than 1 Phone Number	
steps	Expected Results
View Text Message	Text Message is displayed Action List is displayed
Send Voice Message to Phone Number	Dialog Containing Phone Numbers is displayed
Choose Phone Number	Phone number is highlighted
Send Voice Message to Phone Number	Voice message Composer is displayed
Compose Message Send Voice Message	Message "Voice message sent" is displayed

Tabela B.63 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.32

Initial Conditions: Message Contains only 1 Phone Number	
steps	Expected Results
View MMS Message	MMS Message is displayed Action List is displayed
Send Voice Message to Phone Number	Voice message Composer is displayed
Compose Message Send Voice Message	Message "Voice message sent" is displayed

Tabela B.64 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.32

Initial Conditions: Message Contains more than 1 Phone Number	
steps	Expected Results
View MMS Message	MMS Message is displayed Action List is displayed
Send Voice Message to Phone Number	Dialog Containing Phone Numbers is displayed
Choose Phone Number	Phone number is highlighted
Send Voice Message to Selected Phone Number	Voice message Composer is displayed
Compose Message Send Voice Message	Message "Voice message sent" is displayed

Tabela B.65 - Caso de Teste 1 referente ao diagrama de seqüência da Figura A.33

Initial Conditions: Message Contains only 1 Phone Number	
steps	Expected Results
View Email Message	Email Message is displayed Action List is displayed
Send Voice Message to Phone Number	Voice message Composer is displayed
Compose Message Send Voice Message	Message "Voice message sent" is displayed

Tabela B.66 - Caso de Teste 2 referente ao diagrama de seqüência da Figura A.33

Initial Conditions: Message Contains more than 1 Phone Number	
steps	Expected Results
View Email Message	Email Message is displayed Action List is displayed
Send Voice Message to Phone Number	Dialog Containing Phone Numbers is displayed
Choose Phone Number	Phone number is highlighted
Send Voice Message to Selected Phone Number	Voice message Composer is displayed
Compose Message Send Voice Message	Message "Voice message sent" is displayed