# Geração de Casos de Teste de Interrupção para Aplicações de Celulares*

**Wilkerson L. Andrade**[1], **Francisco G. O. Neto**[1], **Patrícia D. L. Machado**[1]

[1] Departamento de Sistemas e Computação
Universidade Federal de Campina Grande (UFCG) – Campina Grande, PB – Brasil

{wilker,netojin,patricia}@dsc.ufcg.edu.br

***Abstract.*** *The mobile phone market has become even more competitive, demanding high quality standards. In this context, applications are developed as sets of functionalities, called features. There are scenarios where features can interrupt the execution of the other features and due to the fact that the features are usually developed in an isolated way, the tests of their interactions in such scenarios are compromised. In this work, we present an approach to generating and selecting interruption test cases from use case specifications. Use cases are translated into LTS models from which test cases can be selected according to a selection strategy basing on test purposes. The presented approach is supported by the LTS-BT tool.*

***Resumo.*** *O mercado de telefonia celular tem se mostrado cada vez mais competitivo, demandando um padrão de qualidade cada vez maior. Neste contexto, aplicações são desenvolvidas como conjuntos de funcionalidades, chamados de features. Existem cenários onde as features interrompem a execução de outras features e devido ao fato de as features serem usualmente desenvolvidas e testadas de forma isolada, os testes de suas interações nos cenários de uso das aplicações ficam comprometidos. Neste trabalho, apresentamos uma abordagem para a geração e seleção de casos de teste de interrupção a partir de casos de uso. Casos de uso são transformados em modelos LTS, a partir dos quais casos de teste podem ser obtidos de acordo com uma estratégia de seleção baseada em propósitos de teste. A abordagem apresentada é apoiada pela ferramenta LTS-BT.*

## 1. Introdução

O mercado de telecomunicações tem crescido cada vez mais no decorrer dos anos, principalmente o mercado de celulares. O aumento do número de pessoas usando esse tipo de aparelho de comunicação móvel implica no aumento da complexidade do software embutido nesses aparelhos. Esse aumento da complexidade se dá por dois motivos: a concorrência gerada pelo crescimento do mercado e a necessidade de acompanhar a evolução das infra-estruturas de rede e de hardware. Nesse contexto, a necessidade de produtos de qualidade é eminente, visto que, o mercado está cada vez mais competitivo. Por outro lado, a complexidade dos sistemas nos remete a um cenário onde atingir tal qualidade requer ferramentas e técnicas também evoluídas.

*Projeto financiado por FAPESQ/CNPq (Projeto 060/03 e CNPq Proc. 550466/2005-3). Cooperação entre Motorola/UFPE/UFCG.

Uma importante característica das aplicações para celulares é o de seu desenvolvimento se dar de maneira evolutiva, onde novos conjuntos de funcionalidades (conhecidas por *features*) são adicionados a versões previamente desenvolvidas. Uma *feature* é um conjunto de requisitos individuais que descreve uma unidade de funcionalidade coesa e identificável [Turner et al. 1998]. Como um exemplo de *feature* em aplicações para celulares pode-se citar a *feature* Mensagem que possui, dentre outros requisitos, os requisitos de enviar e receber mensagens. Em geral, as novas *features* são desenvolvidas e testadas de maneira isolada ou pelo menos com algumas outras *features* das quais ela depende [Calder et al. 2003].

No entanto, esta forma isolada de tratar as *features* pode acarretar alguns problemas, como os causados pelas interações entre as *features*. O termo interação entre *features* (do Inglês *feature interaction*) tem sido usado na comunidade para expressar os problemas causados por funcionalidades adicionadas aos sistemas que geram conflitos com outras já existentes [Calder et al. 2003, Keck and Kuehn 1998]. Por exemplo, uma *feature* que redireciona para o aparelho C uma chamada originada de B quando o telefone A está ocupado, pode entrar em conflito com uma outra *feature* que mostra o número do telefone B para o usuário de A quando A está ocupado. Diz-se que, neste caso, ocorre interação entre *features* uma vez que isoladamente as *features* funcionam, porém um mesmo sistema não poderá implementar ambas as *features*.

No contexto deste trabalho, são chamados de interações de *features* os cenários onde ocorrem interrupções. Ao contrário do que já vem sendo pesquisado pela comunidade, estamos interessados na verificação de cenários compostos por várias *features* onde interrupções são permitidas. O seguinte cenário apresenta um exemplo de interrupção: quando o usuário está compondo uma mensagem de texto, uma ligação pode chegar em seu dispositivo, causando uma interrupção da *feature* de ligação na *feature* de enviar mensagens de texto.

Aplicações para celulares são basicamente aplicações reativas, ou seja, aplicações que interagem com o ambiente no qual estão inseridas. Nesse contexto, uma das formas mais conhecidas para especificar requisitos de aplicações reativas se dá através de casos de uso. Caso de uso é um conceito amplamente difundido e muito utilizado para a documentação e o desenvolvimento de requisitos. Com relação a especificação de aplicações para celulares, Cabral et al [Cabral and Sampaio 2006] apresentam *templates* definidos para especificar casos de uso de *features* e Figueiredo et al [Figueiredo et al. 2006] juntamente com Andrade [Andrade 2007] apresentam uma extensão dos mesmos *templates* para a especificação de interações entre *features*.

Testes de *feature* e de interação entre *features* podem ser obtidos através dos *templates*. Para isso, modelos comportamentais que representam o sistema devem ser extraídos dos mesmos. Sistemas de Transições Rotuladas (*Labelled Transition Systems* - LTS) é um formalismo que tem sido muito utilizado na definição de semânticas de especificações comportamentais [Jard and Jéron 2005, de Vries and Tretmans 2000]. Em geral, LTS provê uma descrição precisa do comportamento de um sistema, e mais, caminhos em modelos LTS podem ser vistos como seqüencias de testes. Logo, modelos LTS são muito úteis na geração de casos de teste.

No contexto das aplicações para celulares, nem sempre o teste de interação per-

mite a execução de todos os possíveis casos de teste gerados. Isso ocorre devido a existência de fortes restrições com relação ao tempo e a recursos necessários a execução. Por isso, a comunidade científica e a indústria têm investido muito em estratégias de seleção de casos de teste [Jard and Jéron 2005].

O objetivo deste artigo é apresentar uma estratégia para dar suporte ao teste de interação entre *features* em aplicações para celulares. A partir das especificações das *features*, através de casos de uso, um modelo comportamental LTS é gerado e, a partir desse modelo, casos de teste são obtidos automaticamente. O processo final de construção da suíte de testes é apoiado por técnicas de seleção que visam a redução do número de casos de teste com o objetivo de focar o teste em interações específicas. Com relação à automação, a ferramenta LTS-BT (*Labelled Transition System-Based Testing*), fruto do trabalho de Cartaxo [Cartaxo 2006], foi estendida para dar suporte à estratégia apresentada.

Este artigo está estruturado da seguinte forma. Na Seção 2, é apresentado o contexto no qual a nossa proposta está inserida. Nas Seções 3, 4 e 5 apresentamos nossa proposta mostrando, respectivamente, a especificação das interações em casos de uso, o modelo comportamental gerado a partir desses casos de uso e a estratégia para a geração e seleção de casos de teste. Em seguida, na Seção 6, um estudo de caso realizado é apresentado. Os principais trabalhos relacionados são discutidos na Seção 7. Por fim, as considerações finais são mostradas na Seção 8.

## 2. Contexto

Em geral, o processo de teste no contexto desse trabalho se dá com a especificação dos requisitos das *features* e suas interações em casos de uso. Em seguida, o modelo comportamental LTS é gerado a partir dos casos de uso. Por fim, combinando o modelo LTS gerado com propósitos de teste, casos de teste de interrupção podem ser obtidos. A geração de casos de teste de interrupção utiliza propósitos de teste com o intuito de manter o teste focado em interações específicas (as interrupções), considerando que a verificação de *features* isoladas é coberta pelo teste de *feature*. Uma visão geral desse processo de teste é apresentada na Figura 1.

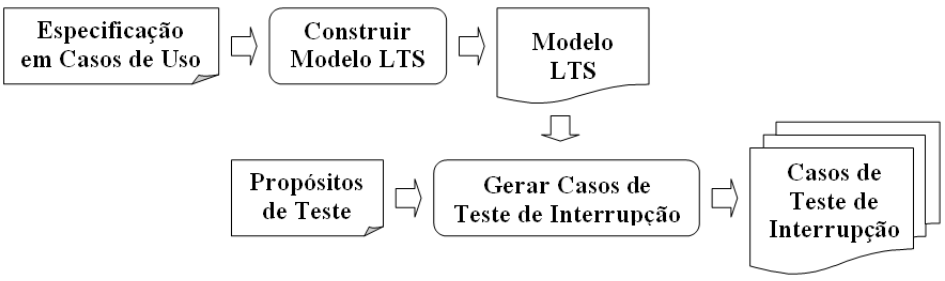


**Figure 1. Visão Geral do Processo de Teste de Interrupção**

Nas próximas seções, mostraremos, através de exemplos, como se dá cada uma das etapas do processo. No primeiro exemplo, uma nova pasta para armazenar as mensagens prediletas do usuário é adicionada ao telefone. O usuário poderá mover mensagens para esta pasta através de atalhos, bem como realizar outras operações referentes a essa nova pasta.

## 3. Especificação em Casos de Uso

As *features* são especificadas com casos de uso, através de *templates*, como mostra o exemplo da Figura 2 [Andrade 2007, Cabral and Sampaio 2006, Figueiredo et al. 2006]. Neste caso de uso, vemos que uma tabela contendo a descrição do cenário principal do caso de uso é apresentada, porém, uma outra tabela contendo os cenários alternativos (Figura 3) também está contida no caso de uso. Os cenários são descritos através de passos que compreendem uma ação do usuário e a respectiva resposta do sistema. Por exemplo, o passo "5M" possui como ação do usuário a seleção da opção "Mover para Mensagens Prediletas", e a respectiva resposta do sistema é exibir uma mensagem dizendo que a mensagem foi movida. Além da ação do usuário e da resposta do sistema, cada passo possui uma condição (Estado do Sistema) que determina se a resposta do sistema será realizada ou não. Caso a condição não seja satisfeita, um cenário alternativo deverá ser especificado, como temos o exemplo do passo "5M" do cenário principal, que possui um cenário alternativo (passos "1A" e "2A" da Figura 3).

**Cenário Principal**
Descrição: A mensagem é movida para a pasta de mensagens prediletas
De: INICIO
Para: FINAL

| Id | Ação do Usuário | Estado do Sistema | Resposta do Sistema |
|---|---|---|---|
| 1M | Ir até a Central de Mensagens | | A pasta "Mensagens Prediletas" é exibida |
| 2M | Ir até a "Caixa de Entrada" | | Todas as mensagens da "Caixa de Entrada" são exibidas |
| 3M | Ir até uma mensagem | | A mensagem é destacada |
| 4M | Ir até o "Menu de Contexto" | | A opção "Mover para Mensagens Prediletas" é exibida |
| 5M | Selecionar a opção "Mover para Mensagens Prediletas" | A memória não está cheia | "Mensagem movida para a pasta Mensagens Prediletas" é exibida |

**Figure 2. Exemplo de Caso de Uso - Cenário Principal**

**Cenário Alternativo**
Descrição: A mensagem não é movida para a pasta Mensagens Prediletas porque a memória está cheia
De: 5M
Para: FINAL

| Id | Ação do Usuário | Estado do Sistema | Resposta do Sistema |
|---|---|---|---|
| 1A | Selecionar a opção "Mover para Mensagens Prediletas" | A memória está cheia | Caixa de diálogo "Memória Insuficiente" é exibida |
| 2A | Confirmar a caixa de diálogo de informação da memória | | O conteúdo da mensagem é exibido |

**Figure 3. Exemplo de Caso de Uso - Cenário Alternativo**

Com relação às interrupções, para que as mesmas estejam presentes no modelo comportamental e, conseqüentemente, possam existir casos de teste que as exercitem, é preciso que tanto o cenário que interrompe seja especificado em casos de uso como o cenário que é interrompido. Dessa forma, sempre que uma *feature* possuir uma interrupção, ou seja, um cenário que é executado mesmo quando outro cenário esteja executando, um caso de uso deve ser especificado para tal interrupção. Por exemplo, a *feature*

Mensagens possui a interrupção Chegada de um Aviso, então um caso de uso deve ser especificado para a interrupção de chegada de um aviso. Essa nova *feature* especifica o comportamento de uma interrupção causada pela chegada de um tipo específico de mensagem, onde o texto aparece numa caixa de diálogo para o usuário e desaparece em seguida.

Os casos de uso de interrupções são especificados através das mesmas seções do template, mostradas nas Figuras 2 e 3. Na especificação de interrupções, só existe uma regra que a diferencia da especificação de *features* simples: o campo "Estado do Sistema" do primeiro passo do cenário principal deve ser preenchido com o nome da *feature* responsável pela interrupção. Na Figura 4, temos um exemplo de um caso de uso para a interrupção de chegada de um aviso.

**Cenário Principal**
Descrição: O aviso é exibido numa caixa de diálogo
De: INICIO
Para: FINAL

| Id | Ação do Usuário | Estado do Sistema | Resposta do Sistema |
|---|---|---|---|
| 1M | Enviar um aviso de um outro celular para o celular sob teste | Chegada de um Aviso | O aviso é exibido numa caixa de diálogo |
| 2M | Selecionar a opção "Ok" | | Volta para a aplicação que estava executando anteriormente |

**Figure 4. Caso de Uso para a Interrupção Chegada de um Aviso**

O caso de uso da Figura 4 é especificado quando a *feature* relativa à interrupção for desenvolvida, e servirá como comportamento padrão para todas as outras *features* para quando essa interrupção ocorrer. O restante do fluxo é semelhante aos casos de uso normais, podendo, inclusive, possuir cenários alternativos.

Uma vez que esse cenário de interrupção esteja especificado, assume-se que o mesmo pode ser executado a qualquer instante da execução de um outro cenário de caso de uso qualquer. Com essa estratégia de especificação de interrupções, os comportamentos das diversas interrupções são definidos durante o desenvolvimento de cada *feature* fazendo com que a forma com que as *features* são desenvolvidas, isto é, de forma isolada, não seja alterada. Além disso, não é necessário especificar todos os possíveis pontos onde as interrupções podem ocorrer. Essa informação só precisa estar representada no modelo LTS, que é gerado automaticamente.

## 4. Transformação de Casos de Uso em Modelos LTS

Esta seção apresenta a principal contribuição desse trabalho que é a estrutura do modelo comportamental LTS capaz de representar as possíveis interações entre as *features*. LTS é um formalismo que tem sido muito utilizado na definição de semânticas de especificações comportamentais [Jard and Jéron 2005]. Modelos LTS são representados por grafos onde os estados representam as possíveis configurações do sistema e as arestas representam o movimento entre essas configurações através da ocorrência de ações.

Os comportamentos das *features* são representados através de um tipo de LTS, chamado LTS Anotado [Andrade 2007, Cartaxo 2006], que segue basicamente a mesma definição apresentada em [Tretmans 1996]. A única diferença está nos rótulos das transições. Um LTS Anotado é uma 4-tupla $\langle Q, R, T, q_0 \rangle$, onde:

- $Q$ é um conjunto finito e não-vazio de estados;
- $R = A \cup N$ é um conjunto finito de rótulos, onde $A$ é um conjunto finito de ações e $N$ é um conjunto finito de anotações;
- $T \subseteq Q \times R \times Q$ é a relação de transição;
- $q_0 \in Q$ é o estado inicial.

Anotações são inseridas no LTS com uma finalidade específica. No caso deste trabalho, estamos interessados na inserção de anotações com o propósito de facilitar o processo de geração de casos de teste. Além disso, essa variação de LTS também permite a representação de condições associadas às ações.


**Figure 5. LTS da *Feature* Mensagens Prediletas**

A Figura 5 mostra o modelo LTS obtido a partir da especificação da *feature* Mensagens Prediletas (Figuras 2 e 3). Como podemos observar, o modelo obtido é um LTS Anotado, onde as colunas do *template* "Ação do Usuário", "Estado do Sistema" e "Resposta do Sistema" foram mapeadas, respectivamente, para as seguintes anotações no modelo LTS: "Passos", "Condições Iniciais" e "Resultados Esperados". No modelo LTS gerado, cada anotação é seguida pela sua respectiva informação, que está no *template*. Por exemplo, para o passo "1M" da Figura 2, tem-se a seguinte seqüencia de rótulos no modelo LTS: "Passos", "Ir até a Central de Mensagens", "Resultados Esperados" e "A pasta "Mensagens Prediletas" é mostrada". Já a seqüencia dos vértices é definida através

dos campos “De” e “Para” de cada tabela da especificação. Como outro exemplo, A Figura 6 apresenta o modelo LTS obtido a partir da especificação da interrupção Chegada de um Aviso (Figura 4).

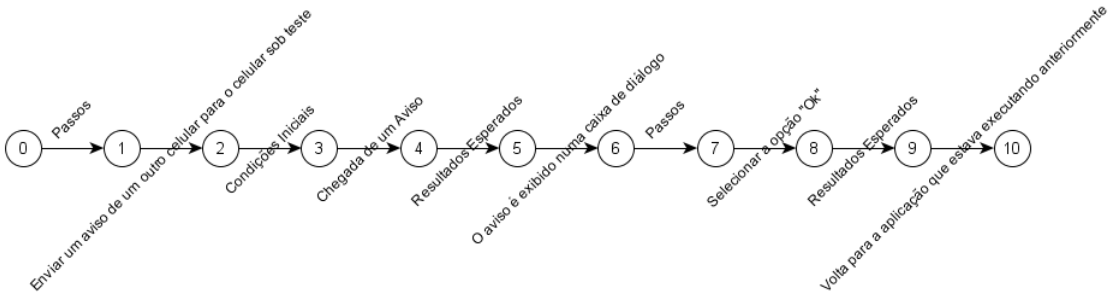


**Figure 6. LTS da Interrupção Chegada de um Aviso**

O modelo LTS Anotado usado para representar comportamentos onde interrupções podem ocorrer segue a mesma definição apresentada anteriormente, com uma simples modificação: o conjunto de anotações é acrescido de mais dois tipos de anotações, onde uma é usada para indicar o início e a outra é usada para indicar o final de uma interrupção.

O próximo passo é unir o modelo LTS da *feature* que pode ser interrompida com o modelo da *feature* que interrompe a partir dos vértices onde a interrupção pode ocorrer. Como as interrupções só podem ser observadas pelo testador se ocorrerem entre os passos do caso de uso, ou seja, sempre depois da resposta do sistema e antes da próxima ação do usuário, então a união dos modelos LTS se dará sempre através dos nós seguintes às respostas do sistema. Através das especificações das *features* Mensagens Prediletas (Figuras 2 e 3) e Chegada de um Aviso (Figura 4) é gerado o LTS apresentado na Figura 7. Os vértices numerados de 0 a 29 representam a *feature* principal (Mensagens Prediletas) e os vértices de 30 a 40 representam a interrupção (Chegada de um Aviso).

Como podemos ver, na Figura 7, duas novas anotações foram inseridas no modelo LTS: *Início da Interrupção X* e *Final da interrupção X*, onde *X* é um contador do tipo inteiro. Essas anotações são usadas para memorizar em que ponto a *feature* principal, Mensagens Prediletas, foi interrompida. Dessa forma, o fluxo de execução da *feature* principal pode continuar a partir do mesmo ponto em que houve a interrupção. Por exemplo, se a interrupção iniciar através do rótulo *Início da Interrupção* 0, então o fluxo de execução da interrupção deve terminar através do rótulo *Final da Interrupção* 0; outro exemplo, se a interrupção iniciar através do rótulo *Início da Interrupção* 1, então o fluxo de execução deve terminar através do rótulo *Final da Interrupção* 1; e assim por diante.

Note que, como se trata de um modelo de testes, estamos considerando apenas as *features* de interesse e não quaisquer *features* existentes. Dessa forma, temos um modelo de comportamento parcial. Do ponto de vista do testador, somente o comportamento especificado é observado, e com isso, todos os outros comportamentos não são observados durante o teste, inclusive, outras possíveis interrupções.

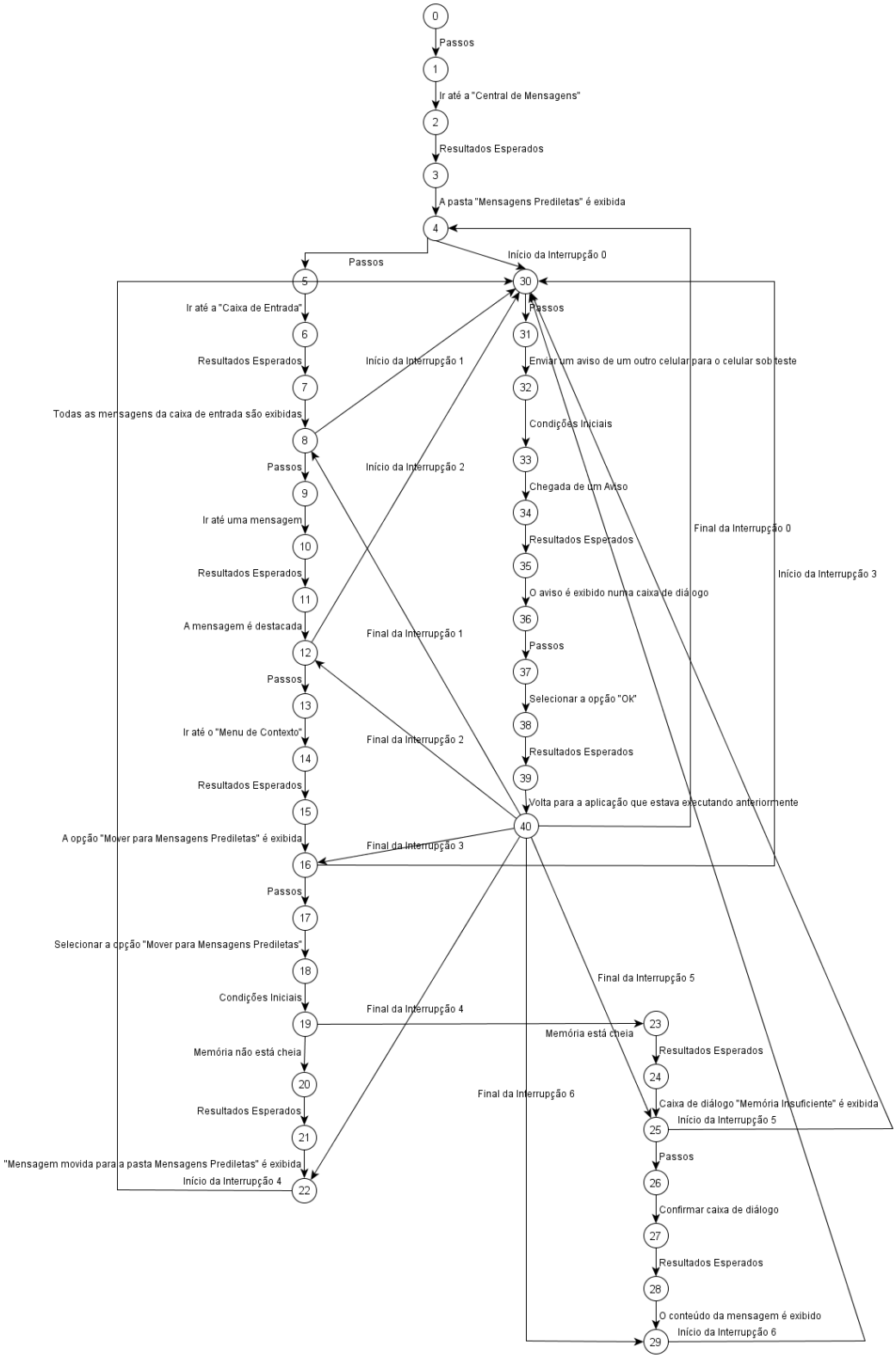


**Figure 7. Modelo LTS com Interrupções**

## 5. Geração e Seleção de Casos de Teste

Esta seção apresenta mais uma contribuição deste trabalho, que é o desenvolvimento de um algoritmo visando a geração de casos de teste de interrupção a partir de um modelo

LTS que segue a estrutura apresentada na seção anterior. Em seguida, uma estratégia de seleção de casos de teste de interrupção baseada em propósitos é apresentada, visto que, o contexto das aplicações para celulares não permite a execução de toda a suíte de teste obtida a partir de uma geração exaustiva devido a restrições de tempo e recursos.

O algoritmo de geração de casos de teste difere dos algoritmos já existentes somente pela forma como o modelo LTS é percorrido. Um caso de teste continua sendo um caminho que parte de um vértice inicial e termina em um vértice final do modelo. De maneira geral, a idéia do algoritmo é percorrer o modelo LTS usando um busca em profundidade modificada. A adaptação no algoritmo de busca em profundidade está no momento em que a seleção de um caminho passa por uma anotação *Início da Interrupção X*, isto é, o caminho selecionado conterá uma interrupção. Nesse caso, quando o encaminhamento chegar ao último vértice que representa o comportamento da interrupção, a próxima aresta a ser escolhida deve ser obrigatoriamente a aresta com o rótulo *Final da Interrupção X*, ou seja, após a interrupção, a *feature* que estava executando continuará sua execução a partir do ponto onde a mesma foi interrompida.

A realização de uma geração exaustiva de casos de teste de interrupção é impraticável devido a grande quantidade de casos de teste gerada e as características associadas ao contexto das aplicações de celulares, onde a grande maioria dos casos de teste são executados manualmente. Nesse cenário, técnicas de seleção de casos de teste precisam ser aplicadas. A técnica empregada para a redução da suíte de testes é a seleção baseada em propósitos. Essa técnica foca na cobertura de um critério de seleção, o propósito de teste, visando testar uma funcionalidade particular do sistema. Considerando a utilização de propósitos de teste, os casos de teste de interação obtidos são consistentes no sentido de que um veredito é negativo somente se a implementação não satisfaz o propósito definido [Fernandez et al. 2003]. E mais, para se manter a consistência, considera-se que nenhuma interrupção não prevista no teste ocorra durante a execução do mesmo.

LTS-BT utiliza uma notação bem simples para a definição de propósitos de teste, onde os mesmos são definidos através de seqüencias de transições. Nessa seqüencia, um "*" (asterisco) indica que naquele ponto pode ocorrer qualquer transição. Um propósito de teste sempre termina com uma transição chamada "Aceitar" (indicando que todos os casos de teste precisam estar em conformidade com o propósito) ou com uma transição chamada "Rejeitar" (caso contrário). Como exemplo de um propósito usado por LTS-BT, vamos usar o LTS apresentado na Figura 5 com o objetivo de definir um propósito de teste para o cenário onde uma mensagem não é movida para pasta de mensagens prediletas porque não há memória suficiente. Para esse cenário, o seguinte propósito poderia ser definido: "*;Caixa de diálogo "Memória Insuficiente" é exibida;*;Aceitar".

A notação utilizada por LTS-BT é uma simplificação da notação utilizada por TGV (*Test Generation with Verification technology*) [Jard and Jéron 2005], que é uma ferramenta de suporte a testes de conformidade que provê geração e seleção de casos de teste para sistemas reativos e não-determinísticos de forma automática. TGV utiliza propósitos de teste definidos através de modelos LTS. Dado que o testador precisa manipular modelos LTS na definição de propósitos de teste em TGV, essa notação não é muito útil na prática. Modelos LTS são usados por LTS-BT como modelos extraídos dos casos de uso e como os propósitos são definidos em alto nível, o testador não precisa manipular nenhum modelo LTS.

A Figura 8 mostra o fluxo da atividade de geração de casos de teste usando propósitos. Para que a seleção de casos de teste possa ser realizada obedecendo a um determinado propósito de teste, o modelo LTS e o propósito devem ser dados como entrada. O propósito de teste é utilizado para limitar o modelo, depois o algoritmo de geração é aplicado, para então, gerar os casos de teste.

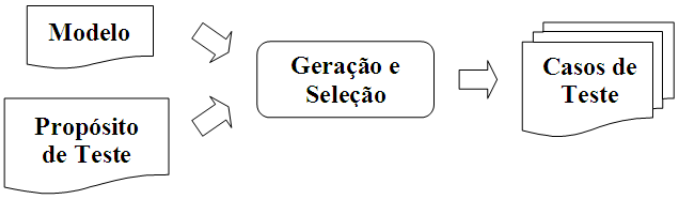


**Figure 8. Fluxo da Seleção de Casos de Teste Baseada em um Propósito**

Considerando o domínio das aplicações para celulares, os únicos casos de teste com interrupções que realmente fazem sentido são aqueles onde há somente uma interrupção por caso de teste. Isso acontece por dois motivos: (1) é considerada a suposição apresentada em [Jorgensen 2001] de que a maioria das falhas[1] ocorrem como resultado da ocorrência de uma única falta[2]; (2) bons casos de teste devem facilitar a localização de faltas [Baudry et al. 2006]. Considerando que cada falha está associada a uma única falta, basta que cada caso de teste verifique as interrupções em um único ponto a cada momento, facilitando assim a detecção de faltas.

Utilizaremos o modelo LTS apresentado na Figura 7 para demonstrar como definir propósitos de teste para verificar as interrupções. Vamos definir um propósito visando testar o cenário onde, ao entrar na central de mensagens e visualizar a opção de acessar a pasta “Mensagens Prediletas”, um aviso chega ao celular. Esse cenário pode ser especificado através do seguinte propósito: “*;A pasta "Mensagens Prediletas"é exibida;Chegada de um aviso;*;Aceitar”. Com a limitação do modelo, segundo o propósito de teste definido, as seguintes arestas do modelo LTS da Figura 7 são excluídas: “Início da Interrupção 1”, “Final da Interrupção 1”, “Início da Interrupção 2”, “Final da Interrupção 2”, “Início da Interrupção 3”, “Final da Interrupção 3”, “Início da Interrupção 4”, “Final da Interrupção 4”, “Início da Interrupção 5”, “Final da Interrupção 5”, “Início da Interrupção 6” e “Final da Interrupção 6”. As Figuras 9 e 10 apresentam os casos de teste obtidos após o processo de limitação do modelo LTS. Note que nos dois casos de teste gerados, a interrupção ocorre após a visualização da pasta “Mensagens Prediletas”, como foi especificado no propósito. Além disso, todos os cenários da *feature* principal são cobertos. No caso de teste da Figura 9, uma interrupção ocorre no cenário onde a mensagem é movida, enquanto que, no caso de teste da Figura 10, uma interrupção ocorre no cenário onde a mensagem não é movida porque não há memória suficiente.

## 6. Estudo de Caso

O objetivo dessa seção é apresentar o estudo de caso realizado com a finalidade de avaliar a aplicação da abordagem proposta neste trabalho. Esse estudo de caso foi realizado com

[1]Falha é a impossibilidade de execução de uma função requisitada. Geralmente é evidenciada através de uma saída incorreta, término anormal, etc.

[2]Falta é a representação de um erro. Sinônimo de defeito ou bug.

| Condição Inicial | A memória não está cheia |
|---|---|

| Passos | Resultados Esperados |
|---|---|
| Ir até a Central de Mensagens | A pasta "Mensagens Prediletas" é exibida |
| Enviar um aviso de um outro celular para o celular sob teste | O aviso é exibido numa caixa de diálogo |
| Selecionar a opção "Ok" | Volta para a aplicação que estava executando anteriormente |
| Ir até a "Caixa de Entrada" | Todas as mensagens da "Caixa de Entrada" são exibidas |
| Ir até uma mensagem | A mensagem é destacada |
| Ir até o "Menu de Contexto" | A opção "Mover para Mensagens Prediletas" é exibida |
| Selecionar a opção "Mover para Mensagens Prediletas" | "Mensagem movida para a pasta Mensagens Prediletas" é exibida |

**Figure 9. Caso de Teste 01**

| Condição Inicial | A memória está cheia |
|---|---|

| Passos | Resultados Esperados |
|---|---|
| Ir até a Central de Mensagens | A pasta "Mensagens Prediletas" é exibida |
| Enviar um aviso de um outro celular para o celular sob teste | O aviso é exibido numa caixa de diálogo |
| Selecionar a opção "Ok" | Volta para a aplicação que estava executando anteriormente |
| Ir até a "Caixa de Entrada" | Todas as mensagens da "Caixa de Entrada" são exibidas |
| Ir até uma mensagem | A mensagem é destacada |
| Ir até o "Menu de Contexto" | A opção "Mover para Mensagens Prediletas" é exibida |
| Selecionar a opção "Mover para Mensagens Prediletas" | Caixa de diálogo "Memória Insuficiente" é exibida |
| Confirmar a caixa de diálogo de informação da memória | O conteúdo da mensagem é exibido |

**Figure 10. Caso de Teste 02**

a cooperação de um time de teste de interação real, que faz parte do Motorola Brazil Test Center (BTC), e as *features* foram escolhidas baseando-se nas *features* que o referido time de teste está trabalhando atualmente.

As *features* escolhidas foram Compositor de Mensagens, Contatos e Chegada de uma Mensagem. A primeira *feature* especifica o comportamento de um editor que permite a criação de mensagens de texto (SMS) e mensagens multimídia (MMS). A partir do editor de mensagens, a *feature* Contatos pode ser usada para escolher um contato da lista de contatos, e assim, enviar a mensagem. A última *feature* escolhida é a *feature* responsável pelas interrupções que ocorrem quando uma nova mensagem é recebida.

O estudo de caso foi focado no cenário onde o usuário cria uma nova mensagem a partir de uma opção de criação de novas mensagens pertencente ao menu “Central de Mensagens”. Como a especificação desse cenário escolhido possui 11 passos, então as interrupções de chegada de uma mensagem poderão ocorrer em 11 pontos diferentes.

Utilizando o modelo LTS com as interrupções especificadas e realizando uma geração exaustiva de casos de teste, 6498 casos de teste foram obtidos. Dentre todos os casos de teste gerados, 6492 verificam cenários onde ocorrem interrupções. Dos 6492 casos de teste com interrupções, apenas 64 contêm somente uma interrupção. Como já foi explicado, os únicos casos de teste com interrupções que realmente interessam ao time de teste de interação são aqueles onde há somente uma interrupção. Por isso, podemos concluir que a geração exaustiva é inadequada ao contexto do teste de interações em aplicações para celulares. Então, o procedimento para a obtenção de casos de teste de interação de forma eficiente deverá ser realizado através da utilização de propósitos de teste. Como a identificação de importantes cenários com interrupções a serem verificados depende da experiência do testador, alguns cenários foram indicados pelo time de teste de interação. Portanto, propósitos de teste serão definidos para testar os seguintes cenários indicados:

1. O usuário abre o editor de mensagens e, antes de começar a digitar a mensagem, uma nova mensagem chega ao celular;
2. O usuário começa a digitar uma mensagem e uma nova mensagem é recebida;
3. O usuário está escolhendo os contatos para enviar a mensagem digitada e uma nova mensagem chega;
4. O usuário está alterando o tipo da mensagem digitada e uma nova mensagem é recebida.

Casos de teste podem ser gerados para verificar o primeiro cenário através do seguinte propósito: “*;A opção direita é "Cancelar";Chegada de uma Mensagem;*;Aceitar”. Com isso, somente seis casos de teste foram selecionados e se observou que os casos de teste contêm somente uma única interrupção e verificam estritamente o cenário desejado. Com relação ao segundo, terceiro e quarto cenários foram gerados, respectivamente, cinco, quatro e seis casos de teste, cumprindo os objetivos.

Através dos exemplos utilizados, pode-se constatar que a geração exaustiva de casos de teste de interrupção é inviável, tornando a utilização de propósitos de teste imprescindível. Com relação a comparação com os casos de teste existentes, obtidos manualmente, foi constatado que os casos de teste obtidos através da estratégia apresentada cobrem todos os casos de teste existentes. Além disso, foi identificado que apenas o segundo cenário estava sendo verificado pelos casos de teste existentes, ou seja, a estratégia apresentada neste trabalho se mostrou ser mais eficiente que o processo manual.

Dada a facilidade e a velocidade com que os propósitos de teste podem ser definidos, a estratégia apresentada neste trabalho é de grande importância para o teste de interrupção em aplicações para celulares, pois permite o foco em interações específicas. Isto facilita o processo de integração e evolução das aplicações: adição de novas interrupções ou *features*. Bem como reduz a suíte de teste, deixando a exaustão das funcionalidades para o teste de *feature*.

## 7. Trabalhos Relacionados

Esta seção apresenta alguns trabalhos relacionados à nossa proposta. Lorentsen et al. [Lorentsen et al. 2001] apresentam uma maneira de categorizar as interações e criar modelos comportamentais que capturem essas interações. Esse trabalho utiliza Redes de Petri Coloridas para, manualmente, gerar os modelos comportamentais do sistema e a partir disso, fazer simulações das interações entre as *features*. Além do modelo resultante desse trabalho não possuir informações inerentes ao processo de teste, não há ferramentas de apoio à geração de casos de teste.

Outro trabalho bastante interessante é o de Jard e Jéron [Jard and Jéron 2005] onde é apresentada a ferramenta chamada TGV, que é uma ferramenta utilizada na geração de casos de teste de conformidade a partir da especificação formal de sistemas reativos. A desvantagem desse trabalho está relacionada ao formato dos modelos de entrada da ferramenta. TGV recebe como entrada um LTS onde os rótulos são particionados entre ações de entrada e ações de saída, não sendo possível representar condições associadas as ações. E também, devido TGV ser uma ferramenta de código fechado, não é possível fazer adaptações no algoritmo de busca em profundidade para que as peculiaridades dos modelos com interrupções sejam consideradas.

Os trabalhos apresentados em [Bryce and Colbourn 2005, Cohen et al. 2003] apresentam uma técnica onde casos de teste são gerados seguindo uma ordem de prioridade e visam garantir que as interações entre cada par de componentes sejam testadas pelo menos uma vez. Dessa forma, mesmo que não seja possível executar toda a suíte de teste, devido às restrições citadas, pode-se ao menos executar os casos de teste mais importantes. Mas note que a informação necessária à priorização é dada pelo testador. Nossa estratégia é similar no sentido de que podemos focar em interações particulares através do uso de propósitos de teste, mas, devido a expressividade do propósito de teste podemos

também adicionar ou excluir funcionalidades particulares associadas às interações.

## 8. Considerações Finais

O trabalho apresentado neste artigo propõe uma estratégia para dar suporte ao teste de interrupção entre *features* em aplicações para celulares. Casos de uso são transformados em um modelo LTS a partir do qual casos de teste podem ser obtidos automaticamente. Após a geração dos casos de teste, técnicas de seleção são aplicadas para reduzir o tamanho da suíte de teste visando testar funcionalidades específicas.

Como suporte à estratégia, a ferramenta LTS-BT foi estendida. O modelo LTS utilizado pela ferramenta foi estendido para capturar os comportamentos das interações. O algoritmo de geração de casos de teste foi adaptado para gerar casos de teste de interação e uma estratégia para a seleção de casos de teste de interação baseada em propósitos de teste foi apresentada.

Através do estudo de caso foi verificada a importância da aplicação de uma estratégia de seleção de casos de teste no contexto do teste de interação, o que permite o foco do teste em interações específicas. Como resultados obtidos, temos que os casos de teste obtidos realmente atendem as expectativas da equipe de teste de interação. E, como trabalhos futuros, mais estudos de caso serão realizados com o intuito de obter métricas que comprovem a viabilidade e a eficácia da estratégia proposta.

Considerando que existem poucos procedimentos sistemáticos para a geração de casos de teste funcional para aplicações de celulares e que o teste de interação é muito pouco explorado, especialmente no contexto do desenvolvimento baseado em *features*, a principal contribuição deste trabalho caracteriza-se pela definição de uma estratégia para geração de casos de teste de interação no referido contexto. Além disso, como a estratégia apresentada está baseada em um modelo comportamental, qualquer tipo de especificação pode ser utilizada, bastando simplesmente representar o comportamento da aplicação através da estrutura do LTS Anotado apresentada.

O procedimento apresentado neste trabalho não adiciona custos extras ao processo de teste, visto que, somente artefatos existentes são utilizados. Muito pelo contrário, custos podem ser reduzidos dado que, quanto mais cedo um erro for encontrado, menos onerosa será a correção. Além disso, este trabalho pode contribuir para a melhoria da qualidade dos produtos vendidos pelas empresas de telecomunicações aos usuários finais, visto que, novos cenários de teste (cenários que causam interações entre as *features*) podem ser incluídos no conjunto de casos de teste.

## References


Andrade, W. L. (2007). Geração de casos de teste de interação para aplicações de celulares. Master's thesis, Universidade Federal de Campina Grande.

Baudry, B., Fleurey, F., and Traon, Y. L. (2006). Improving test suites for efficient fault localization. In *ICSE '06: Proceeding of the 28th international conference on Software engineering*, pages 82–91, New York, NY, USA. ACM Press.

Bryce, R. C. and Colbourn, C. J. (2005). Test prioritization for pairwise interaction coverage. In *A-MOST '05: Proceedings of the first international workshop on Advances in model-based testing*, pages 1–7, New York, NY, USA. ACM Press.

Cabral, G. and Sampaio, A. (2006). Formal specification generation from requirement documents. In *Brazilian Symposium on Formal Methods (SBMF)*, pages 217–232, Natal, Brazil.

Calder, M., Kolberg, M., Magill, E. H., and Reiff-Marganiec, S. (2003). Feature interaction: a critical review and considered forecast. *Comput. Networks*, 41(1):115–141.

Cartaxo, E. G. (2006). Geração de casos de teste funcional para aplicações de celulares. Master's thesis, Universidade Federal de Campina Grande.

Cohen, M. B., Gibbons, P. B., Mugridge, W. B., and Colbourn, C. J. (2003). Constructing test suites for interaction testing. In *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*, pages 38–48, Washington, DC, USA. IEEE Computer Society.

de Vries, R. G. and Tretmans, J. (2000). On-the-fly conformance testing using SPIN. *International Journal on Software Tools for Technology Transfer*, 2(4):382–393.

Fernandez, J.-C., Mounier, L., and Pachon, C. (2003). Property oriented test case generation. In *Proceedings of FATES'03 (Satellite workshop of ASE'03), Montreal, Canada*.

Figueiredo, A. L. L., Andrade, W. L., and Machado, P. D. L. (2006). Generating interaction test cases for mobile phone systems from use case specifications. *SIGSOFT Softw. Eng. Notes*, 31(6):1–10.

Jard, C. and Jéron, T. (2005). Tgv: theory, principles and algorithms: A tool for the automatic synthesis of conformance test cases for non-deterministic reactive systems. *Int. J. Softw. Tools Technol. Transf.*, 7(4):297–315.

Jorgensen, P. (2001). *Software Testing: A Craftman's Approach*. CRC Press, Inc., Boca Raton, FL, USA.

Keck, D. O. and Kuehn, P. J. (1998). The feature and service interaction problem in telecommunications systems: A survey. *IEEE Trans. Softw. Eng.*, 24(10):779–796.

Lorentsen, L., Tuovinen, A.-P., and Xu, J. (2001). Modelling feature interactions in mobile phones. In *Feature Interaction in Composed Systems (ECOOP 2001)*, pages 7–13, Budapest, Hungary.

Tretmans, J. (1996). Test generation with inputs, outputs, and quiescence. In *TACAs '96: Proceedings of the Second International Workshop on Tools and Algorithms for Construction and Analysis of Systems*, pages 127–146, London, UK. Springer-Verlag.

Turner, C. R., Wolf, A. L., Fuggetta, A., and Lavazza, L. (1998). Feature engineering. In *IWSSD '98: Proceedings of the 9th international workshop on Software specification and design*, page 162, Washington, DC, USA. IEEE Computer Society.