Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Linha de Produtos de Software Dinâmica Direcionada por Qualidade: o Caso de Redes de Monitoração do Corpo Humano

Paula Gabriela de Medeiros Fernandes

Brasília
2012

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Linha de Produtos de Software Dinâmica Direcionada por Qualidade: o Caso de Redes de Monitoração do Corpo Humano

Paula Gabriela de Medeiros Fernandes

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Orientador
Prof. Dr. Vander Ramos Alves

Coorientadora
Prof.ª Dr.ª Genaína Rogrigues

Brasília
2012

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Mestrado em Informática

Coordenadora: Prof.ª Dr.ª Mylène Farias

Banca examinadora composta por:

Prof. Dr. Vander Ramos Alves (Orientador) — CIC/UnB
Prof.ª Dr.ª Cecília Mary Fischer Rubira — IC-UNICAMP
Prof. Dr. Ricardo Pezzuol Jacobi — PPGInf-UnB

CIP — Catalogação Internacional na Publicação

Fernandes, Paula Gabriela de Medeiros
.

Linha de Produtos de Software Dinâmica Direcionada por Qualidade: o Caso de Redes de Monitoração do Corpo Humano / Paula Gabriela de Medeiros Fernandes. Brasília : UnB, 2012.
104 p. : il. ; 29,5 cm.

Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2012.

1. Linha de Produtos de Software Dinâmica, 2. Qualidade de Serviço, 3. Redes de Sensores para o Corpo Humano

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Linha de Produtos de Software Dinâmica Direcionada por Qualidade: o Caso de Redes de Monitoração do Corpo Humano

Paula Gabriela de Medeiros Fernandes

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Prof. Dr. Vander Ramos Alves (Orientador)
CIC/UnB

Prof.ª Dr.ª Cecília Mary Fischer Rubira
IC-UNICAMP

Prof. Dr. Ricardo Pezzuol Jacobi
PPGInf-UnB

Prof.ª Dr.ª Mylène Farias
Coordenadora do Mestrado em Informática

Brasília, 17 de dezembro de 2012

Dedicatória

Aos batráquios.

Agradecimentos

Antes de iniciar o meu estudo, gostaria de expor a minha gratidão às pessoas que em maior ou menor escala contribuíram para o meu trabalho: colegas de trabalho, de orientação, professores, familiares e amigos.

Em especial, ao meu orientador Vander pelo tempo, disposição e acreditar no meu trabalho. Agradeço também à minha coorientadora, Genaína, pelas sugestões e críticas ao meu trabalho. Agradeço a Carol por me ajudar no trabalho pesado de programação e de análise dos meus dados.

Agradeço ao Dr. Hervaldo por todo o envolvimento no meu trabalho como meu especialista do domínio. Agradeço ao Sr. Eduardo Borges e aos amigos da INBD que gentilmente cederam os sensores para análise da minha proposta.

Agradeço ao meu namorado, Vinícius, pelo carinho, atenção, compreensão e cumplicidade. Vih, que essa seja mais uma etapa de muitas juntos.

A todos, obrigada.

Abstract

Nowadays, individuals have a more active stance in the investigation of diseases in the sense that they want to monitor their health status continuously. Because it is not sustainable to have dedicated health professional for each individual, more technology support has been applied to assist this monitoring process.

In this context, automatic solutions are being proposed, in particular Body Sensor Network (BSN), in which an individual monitors his vital signs and the system aids him in the prevention and detection of emergency situations.

BSN must manage and balance conflicting requirements, such as availability and reliability, in a way that if the patient is in a normal or low health risk situation, the system can turn off some sensors or disable features to save power and processing. On the other hand, when the individual faints or changes its heartbeats dangerously, the opposite should happen with the sensors and features in order to provide the best service for this high risk situation.

We explore how Dynamic Software Product Line (DSPL) achieves this goal. A DSPL reconfigures itself based on some context changes e.g., the persons' medical situation, to meet a new quality goal for that new situation, as specified by a reliability contract provided by the domain expert (a medical doctor). This contract is modeled as a state machine, whose transitions are medical events (e.g., fall, stroke) and states are target reliability goals, prompting a reconfiguration to meet it. The reliability of any given configuration is measured by a single formula, parametrizing over the features of the DSPL and related quality information. Besides reliability, we also explore other quality parameters such as lifetime, sensor sample rate, quality and amount of information. Strategies for calculating quality such as Simple MultiAttribute Rating Technique (SMART) and goal-oriented are compared in the BSN domain.

We evaluated the proposed approach via simulations with real monitoring data and obtained favorable results with the use of the proposed methodology in the BSN context.

Keywords: Dynamic Software Product Line, Software Quality, Body Sensor Network

Resumo

Na atualidade, os indivíduos passam a ter uma posição mais ativa no processo de investigação de doenças querendo acompanhar o seu estado de saúde continuamente. Por ser é inviável manter um profissional de saúde para cada indivíduo, mais apoio da tecnologia tem sido requerido a fim de auxiliar esse processo de monitoração.

Diante deste quadro, mais soluções automatizadas estão sendo propostas, em particular, Redes de Sensores do Corpo Humano (RSCH), no qual um indivíduo monitora suas atividades diárias e sinais vitais e o sistema o auxilia na prevenção e detecção de situações de emergência.

Este trabalho explora como a metodologia de Linha de Produto de Software Dinâmica (LPSD) no contexto de RSCH gerencia e balanceia requisitos conflitantes, tais como disponibilidade e confiabilidade, de tal forma que quando o indivíduo estiver em uma situação normal de saúde, o sistema possa desativar alguns sensores ou funcionalidades visando economia de bateria e processamento; e por outro lado, quando o indivíduo desmaiar ou alterar seus batimentos cardíacos, o oposto deva acontecer com os sensores a fim de se prover o melhor serviço para o indivíduo em uma situação de alto risco de saúde. Uma LPSD para RSCH se reconfigura baseando-se em mudanças de contexto, no caso, mudança na situação de saúde do indivíduo monitorado, a fim de atingir um novo objetivo de qualidade para esta nova situação de risco.

Neste trabalho, a situação de um indivíduo é especificada como um contrato de qualidade, provido por um especialista no domínio (médico). O contrato é modelado como uma máquina de estados, onde as transições entre estados são causadas por eventos de saúde (queda, desmaio, alteração de pressão) e os estados definem objetivos de qualidade. A verificação de não conformidade com o objetivo de qualidade motiva a reconfiguração do sistema. A confiabilidade de uma determinada configuração é medida como uma única fórmula, parametrizada com a presença e ausência das *features* da LPSD e das qualidades associadas a elas. Além de confiabilidade, exploram-se também parâmetros de qualidade tais como tempo de vida estimado para o sistema, taxa de amostragem, qualidade e quantidade de informação das configurações. As estratégias de cálculo de qualidade *Simple MultiAttribute Rating Technique* (SMART) e orientação a objetivos (GOAL) são comparadas no domínio de RSCH.

Avaliou-se a abordagem proposta via simulações com dados reais de monitoração e obteve-se resultado favorável à utilização da metodologia proposta no contexto.

Palavras-chave: Linha de Produtos de Software Dinâmica, Qualidade de Serviço, Redes de Sensores para o Corpo Humano

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Com o avanço da tecnologia e o barateamento de recursos *hardware*, usuários ao redor do mundo estão insistentemente em busca de soluções de *software* que sejam capazes de auxiliar tarefas diárias tanto no ambiente de trabalho como no ambiente doméstico. Sistemas computacionais já não estão apenas em unidades isoladas de computação, mas rodeiam os indivíduos em dispositivos móveis, multi-usuários [Soylu et al., 2009].

Neste contexto, a indústria e academia investem no desenvolvimento de soluções mais robustas que auxiliem usuários e outros sistemas na execução de tarefas não triviais, tais como controle de uso de energia, tráfego aéreo, análise e investimento financeiro, sistemas de monitoração de saúde, e outros. Em particular, esse trabalho foca no estudo de soluções para monitoração do contexto ou *soluções sensíveis ao contexto* [Alferez and Pelechano, 2011].

Diferentemente de grande parte dos sistemas de *software* onde o ambiente de execução é conhecido e pouco mutável, soluções sensíveis ao contexto sofrem influências do contexto que são de difícil predição [Cetina et al., 2010]. Por esse motivo, elas precisam avaliar constantemente a sua configuração a fim de se decidir se satisfaz os requisitos funcionais e não-funcionais demandados pelos *stakeholders*, domínio e contexto de monitoração. As técnicas tradicionais de desenvolvimento de software se mostram pouco escaláveis, e muitas vezes inviáveis por causa da grande necessidade de adaptação durante o tempo de execução.

Este trabalho está situado no contexto das Redes de Sensores para o Corpo Humano (RSCH), onde um conjunto de sensores é conectado ao indivíduo, e este tem os seus dados vitais capturados e analisados por um sistema. Soluções deste domínio normalmente são muito dinâmicas e sujeitas a erros desde que novas entidades (sensores, atuadores e outros sistemas externos de software) podem ser inseridas ou removidas a qualquer momento [Carvalho, 2005]. Além disso, as soluções de RSCH precisam se adaptar às preferências do indivíduo monitorado e ao contexto e histórico específico de saúde dele [Joonyoung Jung and Kim, 2006]. Ou seja, cada indivíduo utiliza um conjunto específico de funcionalidades que proveem serviços obrigatórios e opcionais para seu estado de saúde. Esse estado de saúde é alterado conforme dados vitais do paciente se alteram.

Em sistemas com muita variabilidade [van Gurp et al., 2001], a engenharia de Linhas de Produto de Software (LPS) pode prover ganhos significativos em qualidade e produtividade através do reuso sistemático de estruturas do software [Clements and Northrop, 2011]. Porém, LPS tradicionais não são capazes de lidar com a mudança em tempo de

execução. Por esse motivo, este trabalho modela a RSCH como uma Linha de Produto de Software Dinâmica (LPSD), um refinamento da LPS tradicional [Hallsteinsen et al., 2008]. A mudança das configurações de uma LPSD é motivada pela mudança de requisitos da RSCH.

1.1 Problema

A mudança de contexto em RSCHs é percebida via eventos [Williams et al., 2006, Chen and Kotz, 2000] tais como febre, desmaio, taquicardia e falta de ar. A análise desses eventos orienta o profissional de saúde no processo de diagnóstico e prevenção de doenças [Baldus et al., 2004]. Entretanto, com a mudança do estado de saúde de um indivíduo, o diagnóstico também se modifica [Dorronzoro et al., 2012]. O profissional da área deve gerenciar as informações do contexto a fim de certificar-se de que as ações tomadas sobre realização de exames, receitas de medicamento e uso aparelhos médicos são apropriadas para o novo estado de saúde do indivíduo.

Entretanto, informações sobre os eventos identificados não são o suficiente para decidir o estado atual de saúde do indivíduo [Carvalho, 2005]. O mesmo evento pode ser interpretado de forma diferente de acordo com o estado de saúde anterior do paciente. A variação da pulsação cardíaca, por exemplo, pode ser interpretada de tal forma que posicione o indivíduo em um estado de alto risco, caso ele já estivesse em um estado de risco moderado. Por outro lado, caso ele estivesse em um estado normal anteriormente, a identificação dessa variação poderia ser interpretada como um erro de leitura do aparelho utilizado ou apenas como um início de atividade física, por exemplo [Carvalho, 2005, Bencomo et al., 2008].

A mudança de estados de saúde do paciente representa uma mudança de requisitos exigidos [Barbosa et al., 2005]. Em situações de alto risco de saúde, por exemplo, é importante que o profissional obtenha mais evidências e fatos sobre os dados vitais que definem o estado de saúde do paciente [Slovic and Weber, 2010]. Por outro lado, quanto mais baixo o risco de saúde, menos recursos são necessários para acompanhar e avaliar a saúde de um indivíduo.

Em virtude da característica dinâmica desse contexto, é preciso constantemente reavaliar o conjunto de ações e recursos utilizados de forma que ele se adeque aos requisitos do estado de risco de saúde do indivíduo monitorado [Fernandes et al., 2011]. Devido ao grande número de variáveis envolvidas na avaliação dessa adequabilidade, não é viável definir todas as configurações possíveis para serem utilizadas em um determinado estado de risco. Além disso, é preciso estabelecer métricas desta avaliação para que o processo de seleção de um conjunto de recursos adequados fique o menos subjetivo possível.

1.2 Solução Proposta

Este trabalho propõe um método para construção de Linha de Produto de Software Dinâmica capaz de selecionar configurações adequadas para um determinado estado de risco a partir da gerência dos múltiplos atributos de qualidade de serviço (QoS). As soluções de *software* que adotam essa arquitetura são capazes de definir os seus eventos, estados, e transições entre eles a partir da gramática sugerida.

A mudança de estado representa uma mudança dos requisitos. Nesse sentido, a arquitetura proposta busca por configurações da LPSD capazes de satisfazer esses requisitos. A busca por configurações da LPSD utiliza algoritmos de satisfabilidade (*SAT-Solvers*) em tempo de execução [Batory, 2005]. Para avaliar a qualidade das configurações com múltiplos atributos de qualidade são usadas as estratégias *Simple Multiattribute Rating Technique* [Edwards, 1977] e orientação a objetivos (*Goal-oriented* - GOAL) [Reuben and Tinetti, 2012].

Os requisitos dos estados são traduzidos em objetivos de qualidade. As configurações que satisfizerem os objetivos de qualidade impostos pelo estado são consideradas adequadas, ou seja, elas podem ser utilizadas pela solução quando esta transitar para o estado associado ao objetivo de qualidade [Calinescu et al., 2011]. Assim, não é necessário especificar todas as possíveis configurações, mas apenas o objetivo a ser atingido.

O método adotado por essa pesquisa trata-se da *pesquisa-ação*. Este método empírico busca resolver um problema real enquanto simultaneamente valida a proposta da solução [Davison et al., 2004]. No processo de diagnóstico, foram levantados os requisitos no domínio no que diz respeito à modelagem de eventos e transições entre os estados de saúde de um indivíduo e a definição da LPSD para o domínio. Na etapa de planejamento, a arquitetura de referência foi refinada para tratar questões de busca por configuração, avaliação de qualidade e mudança de configuração das LPSD. Já no momento da intervenção, construiu-se um *software* para monitoração de sinais vitais modelado como a proposta de solução. Por fim, a avaliação e reflexão sobre a modelagem proposta foram realizadas através da simulação de dados vitais reais sendo tratados pelo sistema modelado. A simulação foi apoiada por duas ferramentas desenvolvidas durante este trabalho. Os resultados obtidos se mostraram favoráveis ao uso da solução no que diz respeito à seleção de configurações da LPS usando as estratégias satisfação de qualidade sugeridas.

Conforme a Modelo de Análise de Linha de Produtos de *Software* (*Product-Line-Analyis Model (PLA Model)*) descrito por von Rhein et al. [2013], no três eixos de avaliação do modelo, amostragem, agrupamento de *features* e formato de variabilidade, esse trabalho está situado da seguinte forma: 1) *Amostragem*: a avaliação de qualidade é feita individualmente para cada produto da LPS. 2) *Agrupamento de features*: a análise de qualidade é feita com a seleção completa das *features* do produto. Entretanto, com a composição através de unidade de vínculo (*Binding Unit*), a proposta se distancia desse eixo em direção ao eixo de avaliação individual das *features*. Por fim, 3) *Formato de Variabilidade*: a proposta utiliza fórmulas genéricas para medição de qualidade de uma instância da LPS, portanto, aproxima-se de uma proposta baseada na LPS (*family-based*).

1.3 Contribuições

A solução proposta tem as seguintes contribuições:

- **Definição de uma arquitetura de uma Linha de Produto de *Software* Dinâmica que é capaz de controlar seus níveis de qualidade avaliando mais de um atributo de qualidade**.
  - *Expressividade das regras de transição entre os estados do sistema.* Mostra como a abordagem proposta pode ser utilizada para aumentar a expressividade

de um sistema dinamicamente adaptável, usando o atual estado do contexto para determinar os novos requisitos de qualidade.

  - *Expressividade da avaliação de qualidade em LPS.* Mostra como calcular qualidade de um produto gerado por uma LPS com qualquer tipo de variabilidade.

- **Avaliação da arquitetura proposta no contexto de Rede de Sensores para o Corpo Humano**.

  - *Monitor de Sinais Vitais.* Implementação do sistema de monitoração de sinais vitais que adota arquitetura proposta e que é capaz de adaptar sua configuração a partir de variações de risco dos dados vitais.
  - *Composição de atributos de qualidade.* Implementação de sistema para criação e avaliação de fórmulas de qualidade baseadas em múltiplos atributos de qualidade.
  - *Avaliação da qualidade.* Compara as técnicas de *Simple Multiattribute Rating Technique* (SMART) com a avaliação orientada a objetivos (GOAL) em uma LPS com os parâmetros de qualidade de confiabilidade, qualidade de informação, quantidade de informação, expectativa de vida do sistema e frequência de amostragem dos dados dos sensores.

1.4 Estrutura

Os principais conceitos de para o entendimento do trabalho são descritos no Capítulo 2. Em especial, são definidos os conceitos de Linha de Produto de Software Dinâmica (LPSD), Avaliação de Qualidade de *Software* e os principais desafios no desenvolvimento de uma solução modelada por LPS capaz de gerenciar seus níveis de qualidade baseando-se em múltiplo atributos.

O problema tratado por este trabalho e seus pontos em aberto são descritos no Capítulo 3. O Capítulo 4 refina a arquitetura de *software* utilizada e aponta como os atributos de qualidade de confiabilidade, qualidade de informação, quantidade de informação, taxa de amostragem dos sensores e expectativa de vida do sistema são utilizados para o cálculo de qualidade de uma configuração de uma LPS.

O Capítulo 5 detalha os *softwares* que foram implementados e descreve como eles foram utilizados para auxiliar o processo de avaliação da proposta deste trabalho. O Capítulo 6 detalha os resultados obtidos na simulação de dados reais no sistema RSCH que aplica as metodologia e arquitetura propostas. Por fim, o Capítulo 7 resume as contribuições, apresenta e discute os trabalhos mais relevantes nas áreas de LPSD e cálculo de QoS em LPS e apresenta as conclusões desta pesquisa propostas de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Este capítulo brevemente revisa alguns conceitos necessários para o melhor entendimento da proposta deste trabalho. Seção 2.1 descreve conceitos relacionados à rede de sensores para o corpo humano. Seção 2.2 descreve os conceitos básicos da estratégia de Linha de Produtos de *Software*. Seção 2.3 detalha os atributos de qualidade tratados por esse trabalho. Por fim, Seção 2.4 define as estratégias para composição de atributos de qualidade utilizados na solução proposta.

2.1 Rede de Sensores para o Corpo Humano

Uma rede de sensores (RS) é um sistema distribuído composto por unidades autônomas capazes de medir algo do contexto (nós sensores), interconectados uma uma conexão sem fio [de A. Barbosa and da Rocha, 2010]. RSs têm sido amplamente aplicadas em sistemas de monitoração, em especial, em sistema de monitoração para o corpo humano, ou Rede de Sensores para o Corpo Humano (RSCH).

Uma RSCH é uma rede de sensores conectados que capturam dados vitais de um indivíduo e os envia para um sistema centralizado. Esse sistema recebe os dados e faz alguma análise com o objetivo de identificar situações de saúde críticas. Aplicações típicas de RSCH incluem o auxílio a pacientes com doenças crônicas tais como doenças cardiovasculares, monitoração de pacientes em unidade de tratamento intensivo, pacientes idosos ou pacientes que se preocupam em acompanhar o estado de saúde [Carvalho, 2005].

A utilização de RSCH em ambientes clínicos reais requer que alguma inteligência seja inserida na solução a fim de manipular mudanças no contexto de monitoração [Baldus et al., 2004]. A falha de um sensor específico não pode, por exemplo, comprometer o funcionamento da RSCH em execução. Além das mudanças autônomas, uma RSCH deve permitir o controle explícito de um profissional de saúde sobre os dados capturados e analisados.

A estrutura básica de uma RSCH é exibida na Figura2.1. Sensores *wireless* são conectados ao indivíduo e capturam os seus dados vitais. Algumas RSCH inserem um outro sensor nó para eliminar dados redundantes [Benny Lo and Yang, 2005], traduzir os protocolos de comunicação entre os sensores ou fazer algumas otimizações. Na Figura 2.1, este nó é representado pelo sensor de controle. Os outros sensores presentes na rede são: (1) acelerômetro (acc), usado para medir a aceleração gravitacional nos três eixos x, y e z; (2) eletrocardiógrafo (ECG), usado para medir batimento cardíaco e a curva do eletrocardio-

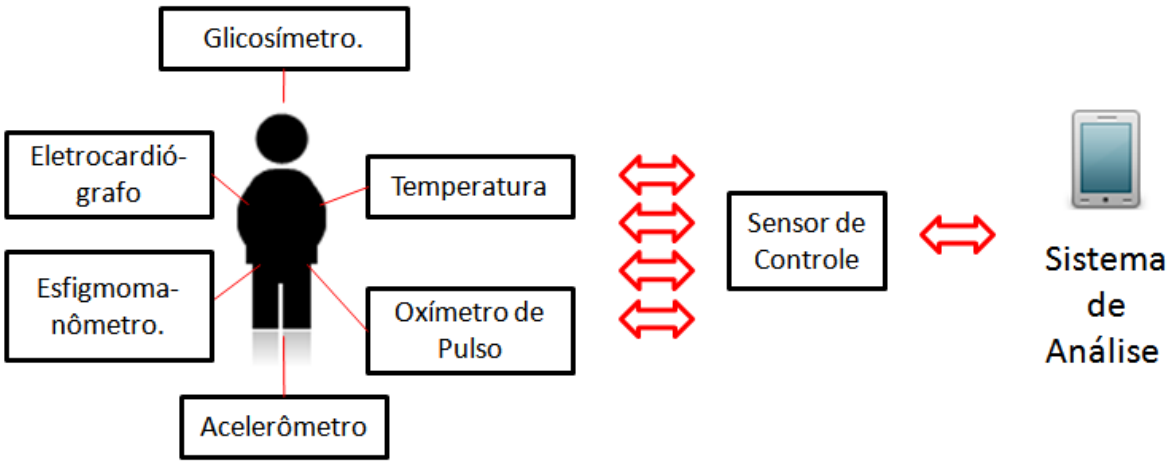


Figura 2.1: Body sensor network design

grama; (3) oxímetro de pulso (Spo2), usado para medir batimento cardíaco, a saturação do oxigênio no sangue e a curva do oxigenação do sangue; (4) o sensor de temperatura, usado para medir a temperatura corpórea, (5) esfigmomanômetro, usado para medir o valor da pressão arterial e (6), o glicosímetro, usado para medir a concentração de glicose no sangue.

2.2 Linha de Produtos de *Software*

O alto grau de reúso e o alto nível de qualidade são fatores fundamentais para os segmentos do mercados de sistemas de alto risco, tais como sistemas de movimentação bancária, sistemas críticos de monitoração [Eriksson et al., 2009]. Além disso, é cada vez mais solicitada a construção de *softwares* especializados, executando para satisfazer requisitos específicos, sejam esses requisitos expressos em termos de confiança, rapidez, performance, custo, funcionalidades e etc.

Neste contexto, a abordagem de Linha de Produtos de *Software* (LPS) vêm ganhando importância por lidar com identificação de variabilidades e comunalidades [Rosenmüller et al., 2011b, Cetina et al., 2010, Benavides et al., 2010]. A **variabilidade de *software*** é definida como a habilidade de um sistema de *software* ou um artefato de ser eficientemente estendido, modificado, customizado ou configurado para um contexto particular [Svahnberg et al., 2005]. Uma **comunalidade**, nesse contexto, é uma semelhança entre *softwares* de um conjunto específico. Em uma LPS, um ponto de variabilidade é um ponto de diferenciação entre produtos [van Gurp et al., 2001]. Linha de Produtos de Software (LPS) é definida como proposto em Clements and Northrop [2011].

Definição 1
Linha de Produtos de *Software* é um conjunto de sistemas compartilhando um conjunto comum e gerenciado de funcionalidades (*features*) que satisfazem necessidades específicas de um segmento, e desenvolvidos a partir de um conjunto comum de artefatos e de uma forma determinada.

Uma LPS é comumente descrita em termos de *features*.

> **Definição 2** Uma *feature* é uma propriedade relevante para um *stakeholder* e é usada para representar comunalidades e diferenças [Czarnecki and Eisenecker, 2000].

Para gerenciar variabilidade é preciso que estas sejam restringidas, ou seja, as possíveis variantes de cada ponto de variabilidade precisam estar especificadas e representadas formalmente [Krueger, 2003]. As restrições de variabilidade de uma LPS podem ser expressas por meio de um **modelo de *features*** (MF), uma estrutura de dados no formato de árvore, que além de implicitamente descrever relações de hierarquia (*features* filhas possuem somente uma *feature* pai), descreve relacionamentos específicos entre *features* irmãs (que possuem uma *feature* pai em comum). O modelo de *features* estabelece os seguintes tipos de restrições: opcionais, obrigatórias, alternativas , ou-features. Adicionalmente, é possível descrever relacionamento entre quaisquer *features* do modelo através da inserção de restrições no modelo, via restrições *cross-tree*. Uma seleção válida de *features* de um MF é aquela que satisfaz todas as regras impostas pela hierarquia e pelas restrições *cross-tree*.

As definições e diferenciações entre as *features* são fornecidas a seguir:

***Features* Opcionais** - Esta restrição aplicada a uma *feature* indica que, se uma *feature* filha é opcional então os produtos gerados podem opcionalmente conter esta *feature*.

***Features* Obrigatória** - Esta restrição aplicada a uma *feature* indica que, todos os produtos que contiverem a *feature* pai devem conter também a *feature* filha.

***Features*-Ou** - Esta restrição aplicada a um conjunto de *features* indica que, os produtos gerados que contiverem a *feature* pai das *features*-ou deve conter obrigatoriamente pelo menos uma das *features* filhas.

Features Alternativa - Esta restrição aplicada a um conjunto de *features* indica que, os produtos gerados que contiverem a *feature* pai das *features* alternativas devem conter obrigatoriamente e exatamente uma das *features* filhas.

Uma restrição *cross-tree* é um comando de inclusão ou exclusão de *features* [Benavides et al., 2006]. A Figura 2.2 exemplifica um modelo de *features* para um sistema de monitoração de sinais vitais. É importante notar que a *feature* raíz é sempre selecionada seja qual for o produto válido da LPS.

No exemplo da Figura 2.2, é possível observar alguns comportamentos sobre as *features*.

1. *Features* `monitoracao`, `informacao_sensor`, `sensor`, `armazenamento` são obrigatórias.
2. *Features* `banco_dados`, `arquivo` e `memoria` são alternativas.
3. *Feature* `exportar_dados` é opcional.
4. *Features* {`oxigenacao`, `pulsacao`, `queda`, `temperatura` e `posicao`} e {`SPO2`, `ECG`, `ACC`, `TEMP`} são *features*-ou.

5. Regras {`oxigenacao` → `SPO2`}, {`pulsacao` → `(SPO2` ∨ `ECG)`}, {`queda` → `acc`}, , {`posicao`→ `acc`} e , {`temperatura` → `TEMP`} são restrições *cross-tree*.

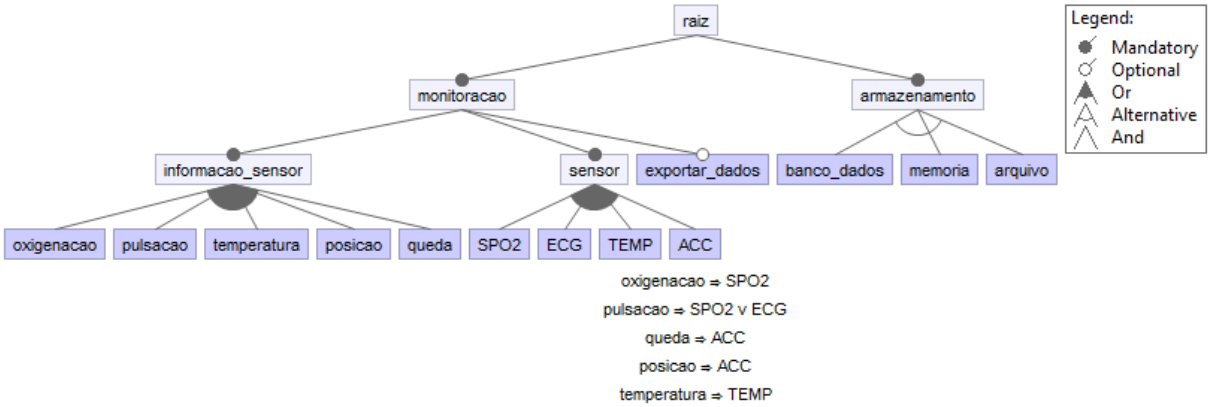


Figura 2.2: Exemplo de modelo de *features*

Uma *feature* de um MF pode representar no produto gerado um conjunto de arquivos, configurações auxiliares, sistemas, componentes de *software*, classes ou linha de código, por exemplo. A seleção de *features* de um MF implica na seleção de um conjunto de **artefatos**, sejam eles artefatos comuns ou variáveis a todos os produtos da LPS. Da perspectiva da implementação, os artefatos são selecionados a fim de gerar produtos de uma LPS.

O mapeamento entre os artefatos e *features* é feito pela estrutura de dados ***Configuration Knowledge*** (CK). Esta identifica quais artefatos (independente do nível de abstração) definem uma *feature* de um MF.

Em resumo, uma LPS pode ser definida como uma tupla desses três elementos, Modelo de *Features*, Artefatos e *Configuration Knowledge*, conforme Equação 2.1.

$$spl = (fm, ck, ats) \tag{2.1}$$

Definição 3 Para um modelo de *features* (MF), um conjunto de artefatos ATS e uma *configuration knowledge* CK, o conjunto (FM, CK, ATS) é dito uma LPS quando o CK diz respeito a *features* no MF, e os artefatos do do CK referem-se somente aos artefatos do ATS , e toda configuração válida do FM leva a uma programa válido [Borba et al., 2010].

Tempo de resolução de variabilidades

A resolução de uma variabilidade é a decisão sobre quais *features* serão selecionadas para resolver cada ponto de variabilidade. As variabilidades em uma LPS podem ser resolvidas em diferentes momentos, quanto mais tarde no ciclo de desenvolvimento for a resolução das variabilidades mais custosa será sua implementação [Svahnberg et al., 2005]. São exemplos de tempo de resolução de variabilidades:

- **Derivação da arquitetura do produto**: a resolução dos pontos de variação de uma arquitetura de uma LPS é o que determina a arquitetura de um produto específico.
- **Compilação**: aspectos estáticos, macros e compilação condicional por exemplo.
- **Ligação**: DLLs, por exemplo.
- **Tempo de execução**: arquivos de configuração, aspectos dinâmicos e mecanismos de *plugin-ins* por exemplo.

Linha de Produtos de Software Dinâmicas

Linhas de produtos de *software* dinâmicas ou (LPSD) produzem *software* capaz de se adaptar às necessidades do usuário ou restrição de recursos. Isso é possível por meio do adiamento da resolução de variabilidades para o tempo de execução. Essa resolução se dá assim que o sistema é inicializado ou durante sua execução. LPSDs podem trocar a resolução de um determinado ponto de variabilidade durante sua execução. Além disso, LPSDs podem apresentar características de sistemas autônomos, auto-adaptáveis, sensíveis ao contexto ou capacidade de tomar decisões automáticas [Hallsteinsen et al., 2008]. A reconfiguração de uma LPSD em tempo de execução pode ocorrer tanto por ação direta do usuário quanto por ação do próprio *software* em reação a algum evento percebido em seu contexto de execução.

2.2.1 Resolução de LPS

Com a expansão do uso de Linha de Produtos de Software, a criação de métodos automáticos para resolver as restrições de uma LPS impostas pelo MF a fim de gerar seus produtos se tornou um objetivo atrativo para a área da engenharia de software, dada a quantidade de configurações existentes em um modelo de *features* e a complexidade de analisar manualmente características de cada um dos seus produtos.

Neste contexto, surgiram várias metodologias de resolução de LPS. Muitas delas voltadas para a utilização do paradigma lógico [Benavides et al., 2010]. Esta seção descreve os principais modelos de análise automatizada sobre uma LPS, detalhando os aspectos mais importantes daquelas que serão utilizadas na metodologia sugerida por este trabalho.

Paradigma Lógico e a automatização da análise de uma LPS

O paradigma lógico está baseado na definição de fórmulas proposicionais.

> **Definição 4** Uma **fórmula proposicional** consiste de um conjunto de símbolos primitivos ou variáveis e um conjunto de concetores restringindo os valores das variávies [Batory, 2005].

As fórmulas proposicionais, quando têm seus símbolos resolvidos, resultam em um valor verdadeiro ou falso. O **problema da satisfabilidade** (Satisfiability Problema - SAT) procura responder se existe valoração para uma fórmula cuja resolução retorne um valor verdadeiro. Embora este problema seja um problema NP-Completo [Gomes

et al., 2008], existem alguns algoritmos que solucionam este problema eficientemente, os algoritmos de **SAT *Solvers***. Esses *solvers* utilizam uma base de dados de conhecimento estabelecido durante a busca para eliminar redundância e buscas desnecessárias [Batory, 2005].

As fórmulas que os *SAT-Solvers* avaliam normalmente são descritas na forma normal conjuntiva (CNF - Conjunctive Normal Form). Esta padronização não tem impacto na resolução, dado que qualquer fórmula proposicional pode ser reescrita na forma normal conjuntiva [Batory, 2005].

O interesse por satisfabilidade no contexto de linha de produtos de software é justificado pela correspondência entre fórmulas proposicionais e a estrutura básica da LPS de modelo de *feature*. É possível estabelecer correspondências lógicas entre as features opcionais, alternativas, ou e obrigatórias, conforme Figura 2.3. Cada *feature* da LPS é transformada em uma variável da fórmula proposicional correspondente.

Feature model relation	**Corresponding formula**
r is the root feature	r
p is parent of optional feature c	$c \rightarrow p$
p is parent of mandatory feature c	$c \leftrightarrow p$
p is parent of grouped features $g_1, \ldots, g_n$, and group cardinality is [1..*] (Inclusive-Or)	$p \leftrightarrow (g_1 \vee \ldots \vee g_n)$
p is parent of grouped features $g_1, \ldots, g_n$, and group cardinality is [1] (Exclusive-Or)	$p \leftrightarrow ((g_1 \wedge \neg g_2 \wedge \ldots \wedge \neg g_n) \vee (g_2 \wedge \neg g_1 \wedge \ldots \wedge \neg g_n) \vee \ldots \vee (g_n \wedge \neg g_1 \wedge \ldots \wedge \neg g_{n-1}))$
Extra constraints	already propositional formulas

Figura 2.3: Correspondência entre relacionamento de features e fórmula proposicional [Batory, 2005]

Neste contexto, os SAT Solvers aplicados à fórmula proposicional correspondente ao modelo de *features* geram configuraçõess completa válida da LPS. As configurações válidas no modelo de *feature* (aquelas que obedecem às restrições impostas pelo modelo de *feature* e as restrições *cross-ree*) são configurações cuja instanciação na fórmula proposicional correspondente resulta em uma valoração verdadeira. O SAT *Solver* auxilia a busca no espaço combinatório do conjunto de valorações possíveis para a seleção ou deseleção de *features* de um modelo de *features*.

Entretanto, o problema SAT restringe a valoração das variáveis aos casos **verdadeiro** ou **falso**. A extensão do problema da satisfabilidade é chamado de *Constraint Satisfaction Problem* (CSP - Problema da satisfação de restrições), que dá suporte à qualquer valoração dos seus termos dentro da fórmula, desde que no momento da resolução a fórmula possa ser reduzida a um valor de verdadeiro ou falso.

Definição 5

Uma ***constraint*** (restrição) é uma relação lógica entre variáveis desconhecidas, cada uma recebndo valores em um determinado domínio. Uma *constraint* restringe

valores que uma varáviel pode ter atribuídos, o que representa uma informação parcial sobre as variáveis de interesse [Laburthe and Jussien, 2012].

O problema de satisfabilidade SAT é um caso específico de um CSP, no sentido de que as atribuições permitidas para as variáveis do SAT são binárias (Verdadeiro ou Falsa), quando que no CSP é permitida também atribuições de números ou intervalos de valores [Tsang, 1993, Batory, 2005]. Uma fórmula dentro do CSP só é resolvida se todas as restrições são satisfeitas. A resolução de CSP está fortemente associada à teoria dos grafos e todo o ferramental proposto por essa área [Tsang, 1993].

2.3 Qualidade do Software

A medição de conformidade de satisfação dos requisitos de um sistema junto a sua construção é feita a partir da análise da **qualidade do *software***. Esta seção realiza o levantamento dos principais conceitos envolvidos na área de qualidade de *software*, em especial aqueles que são tratados para avaliação de qualidade por esse trabalho.

Definição 6 **Qualidade de *software*** refere-se ao grau de atributos de qualidade que um *software* possui [IEEE, 1998].

Definem-se métricas para decidir se um *software* durante o seu ciclo de desenvolvimento possui os atributos de qualidade que se deseja obter. O uso de métricas reduz a subjetividade no controle da qualidade do *software* produzindo uma base quantitativa para tomar decisões sobre a qualidade do software.

Um **atributo de qualidade**, neste contexto, é um requisito não-funcional de um sistema de software, tal como modificabilidade, performance, usabilidade, resiliência, dependabilidade, etc.

Essa seção define as dimensões da qualidade de dados básicas que serão utilizadas neste trabalho, considerando a ordem de importância para os parâmetros de qualidade de dados sugerida por Wand and Wang [1996]. São elas: acurácia e precisão, disponibilidade e atualidade e relevância. Por fim, descreve uma dimensão sobre o comportamento do sistema, a confiabilidade.

Acurácia e Precisão

O parâmetro de **acurácia** diz respeito ao grau com que os dados medidos corretamente representam os objetos reais que eles se propõem a modelar [Loshin, 2006]. Para ser correto, o dado medido deve ser representado em uma forma consistente e não ambígua [Olson, 2002].

Uma forma considerada adequada para representação da **acurácia** elimina ambiguidade.

Disponibilidade e atualidade

O parâmetro de **disponibilidade** do dado diz respeito à expectativa de tempo que a informação está disponível para uso após a sua apuração. Disponibilidade pode ser medida como a diferença de tempo entre o momento no qual a informação é recebida e o momento esperado para a sua leitura ou uso [Olson, 2002].

O parâmetro de **atualidade**, no contexto de qualidade de dados, refere-se ao grau com qual a informação está atualizada com o modelo que ele representa. O parâmetro de atualidade de um dado pode ser medido como uma função entre a taxa de frequência esperada de amostragem dos dados reais e a taxa de atualização dos mesmos [Olson, 2002].

Relevância

Relevância é um parâmetro de qualidade que um dado possui para expressar sua importância sobre os outros dados capturados pelo sistema Wand and Wang [1996].

Confiabilidade

O conceito de confiabilidade utilizado ao longo deste trabalho é definido dentro do contexto de *dependabilidade*[1][Avizienis et al., 2004]. Dependabilidade é habilidade de se entregar serviços em que se possa justificadamente confiar. Esse conceito é composto pelas seguintes propriedades:

- disponibilidade: prontidão para prestação do serviço correto.
- confiabilidade: execução correta do serviço de maneira contínua.
- segurança *(safety)*: ausência de consequências catastróficas.
- integridade: ausência de alterações impróprias do serviço.
- manutenibilidade: relativo à facilidade de se realizar mudanças no serviço.

Esse trabalho utiliza a propriedade de confiabilidade do sistema para determinar o objetivo de qualidade necessário.

2.4 Combinação de atributos de qualidade

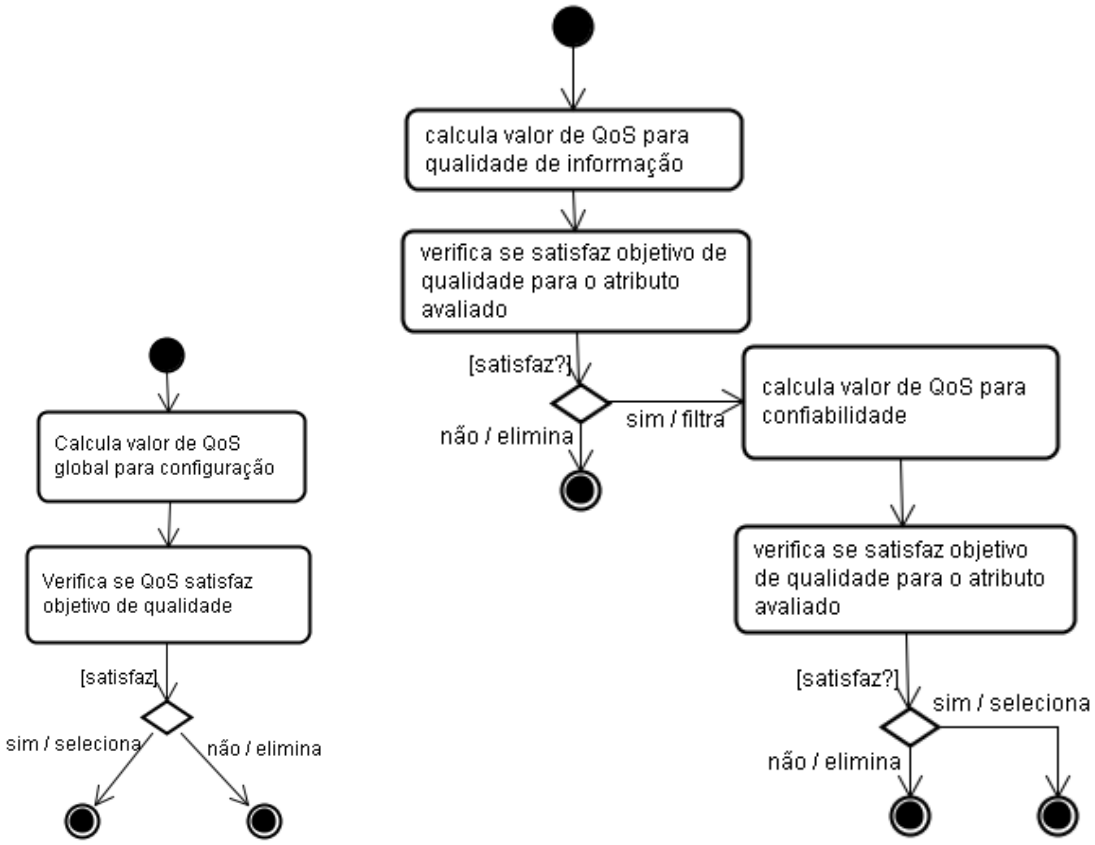
A análise de atributos de qualidade em sistema reais é uma tarefa difícil pois envolve múltiplos parâmetros de qualidade que muitas vezes têm comportamentos contraditórios, no sentido de que o aprimoramento de um atributo deteriora o comportamento do outro [Yoon et al., 1995]. A escolha da melhor solução para um determinado parâmetro pode representar resultados inaceitáveis do ponto de vista de outro atributo [Martens et al., 2010]. Neste contexto, muitos trabalhos têm sido desenvolvidos a fim de especificar como combinar diversos parâmetros de qualidade no auxílio da tomada de decisão, utilizando estimativa de performance [Balsamo et al., 2004], avaliação de confiabilidade em componentes arquiteturais [Gokhale, 2007], criação de fórmulas aritméticas [Tamiz et al., 1998], dentre outros.

[1]Dependabilidade é um neologismo derivado do termo *dependability* do inglês

No domínio específico da medicina, não há muito consenso sobre como agrupar parâmetros de um determinado contexto a fim de tomar a melhor decisão sobre como agir com um paciente em estado de risco [Shannon et al., 2011, Hunink et al., 2001]. A tomada de decisão baseada somente na medição de dados e identificação de doenças não se demonstra adequada para pacientes considerados não saudáveis [Reuben and Tinetti, 2012]. É preciso combinar outros parâmetros a fim de se estabelecer soluções mais apropriadas. No geral, são definidas heurísticas que sugerem o que priorizar em relação aos parâmetros de: expectativa e qualidade de vida do paciente [Shannon et al., 2011], experiência do especialista no domínio, histórico de resultados de pacientes semelhantes e desejo da família e paciente [Winer and Roth, 2006], *software* e *hardware* utilizados no processo de análise ([Chabrol et al., 2006]), custo do tratamento ([Tsevat et al., 1995], [Fakih and Das, 2006]), dentre outros.

Com o intuito de gerar soluções mais genéricas para auxiliar sistemas de monitoração em geral, diversas estratégias para tomadas de decisão têm sido adotadas, tais como algoritmo genérico, decisões orientada a objetivos, análise de multi-parâmetros, inteligência artificial, processo de análise hierárquica ([Chabrol et al., 2006]), análise estatística, dentre outros. Dentre as existentes, este trabalho foca na avaliação da adequação das estratégias de *S*imple *M*ulti*A*ttribute *R*ating *T*echnique (SMART) [Edwards, 1977, Furlong et al., 1998] e orientação a objetivos (GOAL) [Tamiz et al., 1998] em virtude das suas análises positivas em trabalhos na área de monitoração de sinais vitais [Dolan, 2010, Reuben and Tinetti, 2012, Hoeper et al., 2005].

A Figura 2.4 detalha a ideia geral das duas estratégias de qualidade aplicadas nesse trabalho. Somente para exemplificar, considere que o sistema avaliado possui dois parâmetros de qualidade: qualidade de informação e confiabilidade.



(a) Estratégia SMART (b) Estratégia GOAL

Figura 2.4: Diagrama de atividades das estratégias SMART e GOAL

O sistema de decisão baseado na estratégia **Simple MultiAttribute Rating Technique (SMART)** caracteriza-se por estabelecer pesos entre os múltiplos atributos de qualidade. Os valores de qualidade são algebricamente combinados a fim de se gerar um valor único de qualidade em um determinado momento. O número de atributos depende da natureza do problema [Yoon et al., 1995]. A Figura 2.4a detalha as atividades que o sistema realiza quando utiliza fórmulas de qualidade SMART para combinar seus atributos de qualidade. Após o cálculo de qualidade, o sistema seleciona uma configuração baseado nesse valor global de qualidade calculado.

O sistema de decisão baseado na estratégia **Orientada a Objetivos(GOAL)** não almeja definir uma função única de qualidade. Nessa estratégia, cada atributo de qualidade é avaliado independentemente. O sistema seleciona uma configuração se esta configuração satisfaz os objetivos de qualidade individualmente. Esse processo é sequencial, onde é preciso definir prioridades entre os atributos de qualidade. Uma configuração é selecionada se ela satisfaz todos os objetivos de qualidade na ordem de prioridade que eles são definidos [Tamiz et al., 1998]. A Figura 2.4b demonstra o processo de seleção de configuração utilizada pela estratégia GOAL. No caso, o atributo de qualidade de informação tem prioridade sob o atributo de confiabilidade.

Opcionalmente, é possível definir um percentual de satisfação, no sentido de que as fórmulas podem satisfazer percentuais do objetivo de qualidade especificado. Por exemplo, considere que o objetivo de qualidade para o atributo qualidade de informação é 8.0 em

uma escala de 0 até 10. Se defirmos um valor para percentual de satisfação de 80%, a configuração poderia ser selecionada caso seu valor de qualidade para o atributo qualidade de informação fosse 6.4.

Capítulo 3

Modelagem do Problema

O crescente interesse por técnicas de adaptação dinâmica em resposta a mudanças no contexto é uma realidade em sistemas sensíveis ao contexto (*Context-aware systems*) [Alferez and Pelechano, 2011]. Esses sistemas executam em ambientes complexos e heterogêneos, e precisam ser avaliados constantemente a fim de se decidir se satisfazem os requisitos funcionais e não-funcionais demandados pelos *stakeholders* e domínio. Como resultado, definir uma estratégia adequada para adaptação dinâmica é uma necessidade emergente [Alferez and Pelechano, 2011]. Objetivando a economia de recursos, deve-se adotar estratégias que escolham periodicamente um conjunto de funcionalidades que satisfaçam os requisitos do sistema, sempre considerando as restrições de obrigatoriedade ou opcionalidade para o domínio específico.

A ocorrência de eventos motiva a mudança de comportamento das soluções [Williams et al., 2006]. É importante, portanto, mapear a consequência que um evento em particular gera na solução de forma que os requisitos que estão sendo satisfeitos representem a real necessidade do objeto monitorado.

Este capítulo detalha a modelagem do problema tratado por essa dissertação. A Seção 3.1 discute sobre a dependência da mudança de comportamento de uma solução em virtude da ocorrência de eventos e do contexto atual. A Seção 3.2 aponta como uma solução pode ser avaliada a fim de garantir a satisfação dos requisitos impostos por uma necessidade observada do contexto.

3.1 Eventos e estados

A identificação de mudanças de requisitos em um sistema sensível ao contexto é feita via percepção de *eventos* [Williams et al., 2006]. Um evento é qualquer informação que possa ser utilizada para caracterizar uma situação de uma entidade presente no contexto avaliado (pessoa, dispositivo computacional ou um objeto físico qualquer) [Chen and Kotz, 2000].

No domínio médico de diagnóstico e prevenção de doenças, os eventos percebidos estão relacionados aos dados vitais do indivíduo analisado. Os dados observados via instrumentos, soluções de *software* ou percepção do profissional de saúde e paciente são correlacionados de forma a gerar fatos e implicações sobre o estado de saúde do indivíduo. Um evento percebido nesse contexto pode ser, por exemplo, uma identificação de febre, tosse, gripe, pressão baixa, taquicardia, diabetes, depressão, dentre outros. A análise

desses eventos orienta o especialista do domínio no diagnóstico. A partir da avaliação desse profissional, ações tais como receitar medicamento, solicitar mais exames, realizar internação em uma unidade de tratamento intensivo são tomadas para minimizar o risco de saúde do indivíduo.

Entretanto, o diagnóstico pode variar com o decorrer do tempo [Dorronzoro et al., 2012, Baldus et al., 2004]. Novos fatos podem ser percebidos, e um paciente que estava em um estado de baixo risco em um momento pode alterar seu estado de risco de saúde para alto risco em virtude de um infarto, ou atropelamento, por exemplo. Até mesmo a inserção de novas ferramentas e exames podem identificar variáveis que antes não estavam presentes no diagnóstico. Com a mudança de estado de saúde, o profissional de saúde deve ser orientado da melhor forma para tratar corretamente o paciente.

Conforme avaliação junto ao profissional da saúde, os estados de risco de saúde de um indivíduo podem ser modificados por diversos fatores, tais como os econômicos (mudança de situação financeira), sociais (cultura, alimentação, aglomeração de pessoas), biológicos (herança genética, doenças), acidentes, emoções, dentre outros. A identificação de situações de risco requer uma análise de todas essas variáveis.

A fim de convergir a opinião sobre o estado de risco de saúde de um indivíduo, o especialista do domínio pode realizar interpretações indivíduais de cada evento ou uní-los de uma forma organizada, operação chamada de fusão de dados [Tabar, 2006]. Para cada evento, o especialista estabelece um risco associado ao valor medido [Carvalho, 2005]. A identificação do risco associado a um evento é feita baseando-se em valores históricos, valores específicos de normalidade do indivíduo monitorado e experiência e motivação do profissional da saúde [Knaus et al., 1985, of Anesthesiologists, 2002]. De forma genérica, entretanto, as classificações posicionam um evento em risco alto, risco moderado ou risco baixo ou risco irrelevante, conforme aponta Tabela 3.1

Linha	Pulsação Cardíaca (bpm)	Evento	Risco de Saúde
1	x >180	Taquicardia Alta	Risco Alto
2	140 <= x <= 179	Taquicardia Moderada	Risco Moderado
3	110 <= x <= 139	Taquicardia Baixa	Baixo Risco
4	70 <= x <= 109	Normal	Irrelevante
5	55 <= x <= 109	Bradicardia Baixa	Baixo Risco
6	40 <= x <= 54	Bradicardia Moderada	Risco Moderado
7	55 <= x	Bradicardia Alta	Risco Alto

Tabela 3.1: Tabela de interpretação de valores medidos de pulsação cardíaca para um indivíduo genérico [Knaus et al., 1985]

A Tabela 3.1 especifica os eventos relacionados à pulsação cardíaca. Caso a pulsação esteja entre os valores 110 bpm e 139 bpm (l. 3), é identificado um evento de taquicardia baixa. O risco associado a este evento é Baixo Risco.

Um evento como variação moderada na frequência cardíaca de um indivíduo monitorado, por exemplo, pode ter interpretações diversas dependendo do estado atual do indivíduo. Caso ele esteja em um estado de saúde considerado de risco baixo e este evento é identificado, o sistema deve ir para um estado de risco moderado, considerando que essa variação pode ser causada por um início de uma atividade física, ou emoção causada por diversos motivos. Por outro lado, caso o indivíduo esteja em uma situação de risco

moderado, o sistema deve se preparar ou para manter o estado de risco moderado ligando alguns serviços de alerta ou modificar seu estado de risco para o de risco alto.

Por causa desta dependência do estado atual, o especialista no domínio deve mapear eventos que geram transições entre os estados [Bencomo et al., 2008]. Um trecho do mapeamento entre o estado de saúde atual e o novo estado de saúde em virtude da identificação de eventos relacionados à pulsação cardíaca é apresentado na Tabela 3.2.

Linha	Estado Atual	Evento	Novo Estado
1	Risco Alto	Taquicardia Moderada	Risco Moderado
2	Risco Moderado	Bradicardia Baixa	Risco Baixo
3	Risco Baixo	Taquicardia Alta	Risco Alto
4	Risco Irrelevante	Bradicardia Alta	Risco Alto
5	...	...	...

Tabela 3.2: Tabela de mapeamento entre o estado de risco do indivíduo que tem seu valor de pulsação cardíaca moderado [Carvalho, 2005]

Tabela 3.2 especifica um cenário onde o indivíduo em estado de risco de saúde `Baixo` deve passar para o estado de risco de saúde `Alto` no caso da identificação de um evento de `Taquicardia Alta` (l. 3).

3.2 Adequação da solução aos requisitos de um estado de risco

A mudança de estados de risco representa uma mudança de requisitos, e sua ocorrência direciona a escolha de um novo comportamento do sistema que melhor se adeque ao estado atual do sistema [Fernandes et al., 2011]. Em situações de alto risco de saúde, por exemplo, é importante que o profissional obtenha mais evidências e fatos sobre as variáveis que definem o estado de saúde do paciente [Slovic and Weber, 2010]. Neste sentido, o profissional de saúde requer que informações mais detalhadas e precisas, com uma maior frequência sejam fornecidas a fim de se definir ações que minimizem os efeitos do problema de saúde. Por outro lado, quanto menor o risco de saúde menos recursos são necessários para assegurar um controle adequado do estado de saúde do indivíduo [Barbosa et al., 2005].

A avaliação sobre quais recursos utilizar em determinados estados de risco é é fundamentada pelo conceito de *prevalência* de um evento de saúde e o risco de saúde do indivíduo [Woodbury and Houghton, 2004, Humphreys and Smyth, 2006]. Quanto maior é a prevalência, maior é a frequência de ocorrência de um evento [Chipara et al., 2010]. Em um estado normal de saúde, o médico estima que a prevalência de um evento crítico é menor. Desta forma, os recursos disponíveis para a avaliação da saúde podem ser poupados ou distribuídos entre outros indivíduos em estados de alto risco [Dolan, 2010]. De forma contrária, quanto pior é o estado de saúde de um indivíduo, maior é a chance de um evento crítico ocorrer em um espaço de tempo curto [Chipara et al., 2010] e maior é a importância sobre a qualidade dos sinais vitais, uso de instrumentos apropriados.

Baseando-se na experiência adquirida no decorrer dos anos, o profissional de saúde avalia um conjunto de possíveis instrumentos, exames, outros recursos humanos, custo e

disponibilidade de medicamentos para decidir quais deles serão utilizados para auxiliar o diagnóstico de saúde de um indivíduo Olson [1996]. Isto é, a *configuração do ambiente* é avaliada a cada momento para decidir se a solução adotada satisfaz os requisitos do estado de saúde. Essa avaliação, entretanto, pode variar significativamente de acordo com o especialista no domínio.

Neste sentido, é importante que métricas sejam impostas para reduzir a subjetividade nesta avaliação e aumentar o controle da adequação de uma configuração do ambiente sobre os requisitos de um estado de risco de saúde [Fernandes et al., 2011]. Uma possível solução para avaliar essa adequação é a avaliação dos atributos de qualidade de serviço (QoS) providos por essa configuração [Lee and Kang, 2006, Dolan, 2010, Thokala and Duenas, 2012, Milenković et al., 2006, Joonyoung Jung and Kim, 2006, Hunink et al., 2001, O'Connor et al., 2004]. Os requisitos de um estado de risco são interpretados como *objetivos de qualidade*.

Do ponto de vista dos sistemas de *softwares* e *hardwares* presentes em soluções que auxiliam profissionais de saúde no processo de diagnostico, os seguintes atributos de qualidade são normalmente tratados:

Confiança sobre os dados medidos e eventos identificados : Quanto mais instrumentos e exames medirem a mesma informação, maior é a confiança sobre a interpretação desses dados pois menor é a probabilidade de que o desvio das medições desta informação seja alto.

Amostragem da informação : Quanto maior a frequência de amostragem dos dados vitais, maior é a chance de que um dado recebido há pouco tempo represente a situação real do indivíduo monitorado.

Disponibilidade dos recursos : Quanto mais tempo os recursos que avaliam o estado de saúde do indivíduo estiverem disponíveis, maior é o tempo de controle sobre o estado de saúde. A economia destes recursos implica no aumento da disponibilidade deles em um momento futuro.

Quantidade de informação : Quanto maior a quantidade de informação disponível sobre o indivíduo monitorado, mais certeza se tem sobre o estado de saúde dele, no sentido de que mais variáveis podem ser consideradas na tomada de decisão.

Confiabilidade : Parâmetro relacionado à probabilidade de execução correta do serviço. A confiabilidade tem uma relação inversa em relação à probabilidade de falhas.

Quanto mais recursos de *software* ou *hardware* estiverem presentes na solução maior é a probabilidade de falha que algum deles.

A avaliação de cada atributo por vezes leva a comportamentos contraditórios [Fernandes et al., 2011]. A maximização da frequência de amostragem das informações ou da confiança sobre os dados medidos aumentam o gasto dos recursos, o que reduz a disponibilidade destes em um momento futuro [Perillo and Heinzelman, 2003]. Diversos trabalhos utilizam a ideia de maximização dos valores de qualidade a fim de selecionar uma configuração ideal do ambiente para um estado de risco [Alrifai and Risse, 2010]. Porém, eles no geral tratam de maximização de apenas um atributo de qualidade [Ehrgott, 2008,

Martens et al., 2010], priorizando por exemplo, a maximização da disponibilidade dos recursos [Younis et al., 2004].

Nas soluções de *software* sensíveis ao contexto, onde a configuração do ambiente deve satisfazer rapidamente aos novos objetivos de qualidade, os recursos e funcionalidades selecionados não necessariamente precisam atingir os melhores valores de qualidade, mas sim uma faixa de valores aceitáveis [Carvalho, 2005].

Com a verificação de satisfação do objetivo de qualidade, as configurações inadequados do ambiente são eliminadas das possíveis seleções para um estado de risco do sistema. Além de um valor mínimo para o objetivo, o especialista no domínio pode estabelecer um valor máximo para o objetivo de qualidade. Neste caso, é considerada a influência de outros atributos de qualidade e os seus comportamentos contraditórios.

O uso frequente de um aparelho de aferir o valor da pressão arterial, por exemplo, influencia negativamente a acurácia dos valores medidos por este aparelho[1]. Ou seja, a amostragem da informação influencia negativamente na confiança sobre os dados medidos e eventos identificados. Um especialista do domínio, percebendo esse comportamento, pode sugerir que a pressão arterial seja aferida somente uma vez ao dia, ao invés de seis, para indivíduos em estado de saúde de baixo risco [Tabar, 2006].

A Tabela 3.3 exemplifica uma definição de objetivos de qualidade para cada estado de risco de saúde. Nas situações em que a prevalência é menor, a qualidade exigida é menor. Observe que embora sejam definidos valores máximos, o valor de qualidade exigido por um estado de risco é sempre menor quando o risco é menor. Desta forma, configurações que satisfazem à faixa de qualidade para o estado de alto risco também satisfazem o estado de risco moderado.

Estado Atual	**Valor Mínimo**	**Valor máximo desejado**
Risco Alto	0.8	1.0
Risco Moderado	0.6	< 0.8
Risco Baixo	0.3	< 0.6
Risco Irrelevante	0.0	<0.3
...	...	...

Tabela 3.3: Tabela define intervalos de valores de qualidade exigido para cada estado de risco

A escolha por uma configuração adequada deve levar em consideração além dos objetivos de qualidade, a seleção de funcionalidades obrigatórias ou opcionais para o domínio do problema. Uma solução de software para monitorar o estado de saúde de um indivíduo que contem os sensores oxímetro de pulso eletrocardiógrafo, por exemplo, pode obter a informação de frequência cardíaca através desses dois sensores. Em um estado de risco normal, a presença dos dois sensores não é necessária, considerando o objetivo de economia de recursos. Por outro lado, como informações sobre o estado cardíaco e respiratório do indivíduo monitorado são muito importantes em situações de alto risco [Carvalho, 2005], é importante que esses dois sensores simultaneamente estejam sendo utilizados, independente da redundância de informação.

[1]Manual do usuário do *IntelliSense Blood Pressure Monitor* disponível em `http://www.omronhealthcare.com/wp-content/uploads/hem-907xl_im.pdf`

Para a real utilização em seres humanos, é essencial que as soluções de software ponderem a satisfação de alguns atributos de qualidade em virtude do estado de risco do indivíduo e as regras específicas do domínio sobre a presença de funcionalidades.

Capítulo 4

Linha de Produtos de Software Dinâmica Direcionada por Qualidade

Conforme apresentado no Capítulo 3, a mudança de requisitos ou objetivo de qualidade de um sistema sensível ao contexto é motivada pela identificação de eventos do contexto. Esta identificação, entretanto, não é o suficiente para decidir quais os novos requisitos que devem ser satisfeitos. Algumas vezes, o estado atual do sistema é importante para direcionar a escolha dos novos objetivos.

A mudança do estado do sistema representa uma mudança de requisitos. A verificação de satisfação de requisitos requer que alguma análise qualitativa ou quantitativa sobre o sistema seja feita a fim de se estabelecer métricas que informem se os requisitos foram satisfeitos pela solução. Neste sentido, avaliação de atributos de qualidade tais como confiabilidade, disponibilidade, segurança, resiliência se mostra um método objetivo para quantificar a Qualidade de Serviço (QoS) provida pelo sistema. Essa avaliação deve funcionar corretamente para qualquer tipo de funcionalidade, obrigatória ou opcional.

A solução apresentada por este capítulo trata de uma rede de sensores para o corpo humano (RSCH). Neste contexto, um indivíduo possui seus dados vitais capturados por sensores, e estes encaminham esses dados para um sistema monitor que realiza análise dos dados e gerenciamento sobre os estados de risco de saúde deste indivíduo. O objetivo deste capítulo é descrever uma solução proposta para a criação de um sistema dinâmico capaz de gerenciar seus níveis de qualidade no contexto de RSCH. Para tratar as suas características dinâmicas, adota-se a modelagem de Linha de Produtos de Sotware Dinâmica (LPSD).

Este capítulo está dividido como segue: a Seção 4.1 detalha a solução de modelagem para definir o relacionamento entre eventos gerados pelo contexto e mudança de objetivos. A Seção 4.2 descreve como os conceitos de LPSD estão presentes na solução. A Seção 4.4 refina a arquitetura de referência utilizada e demonstra os relacionamentos entre os componentes de *software* e os requisitos de monitoração e avaliação do contexto. Por fim, a Seção 4.5 apresenta as estratégias para a avaliação dos atributos de qualidade do sistema que possua qualquer tipo de funcionalidade.

4.1 Modelo de Qualidade

O diagnóstico sobre o estado de saúde de um indivíduo é feito por meio de identificação e classificação de eventos. Diagnosticar consiste em verificar fatos e suas implicações sobre

a saúde [Clancey and Shortliffe, 1984]. Para tal identificação, o especialista no domínio tem o apoio de instrumentos e exames que quantificam valores e os classificam em relação à discrepância com os valores considerados globalmente normal. Essa discrepância é utilizada para definir os estados de riscos de saúde de um indivíduo [Carvalho, 2005].

A variação de dados vitais geram eventos, e estes podem ocasionar mudanças na interpretação sobre o estado de risco de saúde do indivíduo monitorado. Conforme apontado pela Tabela 3.2, o especialista do domínio conhece os estados de risco de um indivíduo monitorado e como as transições entre esses estados são feitas.

A estratégia de modelagem adotada pela solução adotada neste trabalho está baseada na definição de máquinas de estado ou um autômato determinístico finito (*Deterministic Finite Automaton* - DFA). Formalmente, um DFA é especificado como segue [Hopcroft et al., 2006]:

$A = (Q, \sum, \delta, q_0, F)$

- Q é o conjunto de estados do DFA.
- $\sum$ é o conjunto de símbolos de entrada, ou alfabeto.
- δ é a função de transição ($\delta : Q \times \sum \rightarrow Q$)
- q_0 é o estado inicial.
- F é o conjunto de estados finais.

No contexto de RSCH, a especificação do DFA utiliza as seguintes definições:

- Q é o conjunto de estados de risco de um indivíduo monitorado. $Q = AR, MR, BR$, onde AR representa alto risco de saúde, MR representa médio risco de saúde e BR representa baixo risco de saúde.
- $\sum$ é o conjunto de eventos identificados pelo contexto que modificam o estado de risco de saúde.
- δ é a função de transição ($\delta : Q \times \sum \rightarrow Q$), que mapeia o relacionamento entre os estados de riscos e eventos.
- q_0 é o estado de Alto Risco (AR).
- $F = Q$

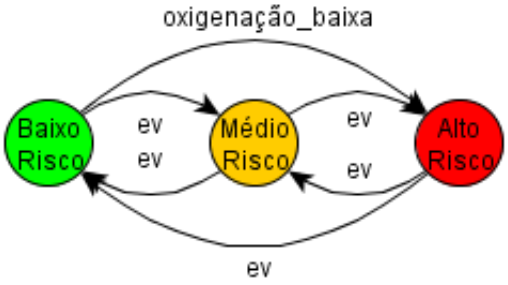


Figura 4.1: DFA representando estados de risco Alto, Médio e Baixo e as transições entre eles

A Figura 4.1 apresenta um exemplo de uma máquina de estado para o contexto de RSCH. Conforme apresentado, caso o indivíduo esteja em um estado de baixo risco e seja identificado um evento de oxigenação baixa, a máquina de estado deve transicionar para o estado de alto risco. Assim como o evento de oxigenação, outros eventos de monitoração podem especificar a transição do estado de baixo risco para o estado de alto risco.

Seres humanos possuem valores diferentes de normalidade para características diferentes de monitoração [Barbosa, 2008]. Por exemplo, assim como várias outras características do corpo humano, o valor normal para uma frequência cardíaca de um indivíduo não depende apenas de um único fator, mas de vários tais como gênero, idade, altura, herança genética, contexto social. Cada indivíduo possui um perfil de normalidade e de aspectos a se monitorar [Carvalho, 2005]. Por isso, a solução de monitoração não deve ser estática, com valores pré-estabelecidos [Barbosa, 2008].

Uma solução de monitoração do corpo humano deve permitir a inserção de regras para identificação do eventos específicos para o indivíduo monitorado e regras para a transição dos estados de risco. A Seção 4.1.1 apresenta a gramática proposta para definição dessas regras.

4.1.1 Linguagem de eventos e transições

A utilização da máquina de estados para modelar os estados de risco de saúde do indivíduo monitorado requer que sejam especificados os estados e a função de transição entre eles. Esta seção apresenta uma proposta de gramática de definição dos eventos, estados de saúde e como os eventos geram transições entre os estados. O correto funcionamento do sistema de monitoração requer que o especialista do domínio especifique pelo menos um estado na máquina de estados.

O Quadro 4.1 detalha as regras de definição dos eventos e transições desta máquina.

Quadro 4.1: Gramática de Definição de Eventos

```
1 <character> := ( "a" | "Z" )
2 <digit>  := ( "0" | "9" )
3 <eq>     := "="
4 <impl>   := "->"
5 <comp>   := "||" | "&&"
6 <lg>     := ">=" | ">" | "<" | "<=" | "==" | "!="
7 <number> := <digit> | <digit> <number>
8 <value>  := <number> "." <number> | <number>
9 <label>  := <character> | <character> <label>
10
11 <states> := "LR" | "MR" | "HR"
12 <SS>     := <states> | "*"
13
14 <source_name> := <label>
15 <prop_name>   := <source_name>"["<label>"]"
16
17 <event> := <label> (<prop_name> <lg> <value>)
18 <fusion_def> := <event> | <event> <comp> <fusion_def>
19 <fusion> := "fusion_"<label> "{" <fusion_def> "}"
20 <transition>    := "(" <SS> "," <fusion> ")" <impl> <SS>
```

Em especial, a gramática possui regras para especificação dos estados de risco de saúde (l. 11), fontes de dados vitais (l. 14), propriedades das fontes que armazenam os dados vitais (l. 15), geração de eventos (l. 17), fusão dos dados (l. 18), lista de fusão de dados (l.19) e função de transição entre os estados de risco de saúde (l. 20). O Quadro 4.2 apresenta uma instanciação da gramática apresentada no Quadro 4.1.

Quadro 4.2: Instanciação da gramática proposta

```
1 //Fonte de dados vitais
2 spo2 (l. 14 da gramática)
3 ecg  (l. 14 da gramática)
4
5 //Propriedades
6 spo2[oxygenation] (l. 15 da gramática)
7 spo2[pulse_rate]  (l. 15 da gramática)
8 ecg[pulse_rate]   (l. 15 da gramática)
9
10 //Eventos
11 spo2_unavailable (spo2[oxigenacao] > 100.0)(l. 17 da gramática)
12 oxygen_n (spo2[oxigenacao] >= 90.0)     (l. 17 da gramática)
13 oxygen_m (spo2[oxigenacao] >= 50.0)     (l. 17 da gramática)
14 oxygen_h (spo2[oxigenacao] >= 0.0)      (l. 17 da gramática)
15 tachycardia (ecg[pulse_rate] >= 140.0)  (l. 17 da gramática)
16
17 //Fusão de dados
18 fusion_stress_moderate {oxygen_n && tachycardia } (l. 18 da gramática)
19 fusion_tachycardia {tachycardia}                  (l. 18 da gramática)
20
21 //Transições
22 (MR, fusion_stress_m {oxygen_n && tachycardia })-> MR (l.20 da gramática)
23 (* , fusion_tachycardia {tachycardia}) -> HR  (l. 20 da gramática)
```

A determinação do estado de saúde do indivíduo é expresso hierarquicamente [Clancey and Shortliffe, 1984]. Os dados são medidos e interpretados a fim de que eventos sejam gerados (l. 17 da gramática). Após a identificação de eventos, eles são agregados de forma sucessiva até a geração de conceitos e relações de alto-nível, expressando a ideia de fusão de dados (l. 18 e 19 da gramática). Por fim, um conjunto de eventos é utilizado para expressar a mudança de estados de risco (l. 20 da gramática).

O DFA exige que as regras de transições ocorram de forma determinística. Ou seja, fixando o estado atual de risco, ao receber um conjunto de eventos no formato de fusão de dados, a máquina deve transitar para o mesmo estado de destino sempre. Entretanto, a fim de flexibilizar a escrita das regras da gramática proposta, é possível descrever o estado de destino com o símbolo "*". O significado imposto por essa regra é de que independente do estado original, sempre que acontecer um conjunto de eventos, o sistema deve transitar para o mesmo estado.

Considere o exemplo do Quadro 4.3. Sem se preocupar com o conjunto de eventos que gera a regra de fusão de dados sobre o evento `diastolic_anormal_moderate_low`, o quadro apresenta um cenário onde é possível transitar para dois estados diferentes caso o evento `diastolic_anormal_moderate_low` ocorra.

Quadro 4.3: Exemplo onde duas regras diferentes para o mesmo evento podem gerar mudanças diferentes na máquina de estado do sistema

```
1 ( NR, diastolic_anormal_moderate_low { ... } ) -> MR    //Regra 1
2 ( * , diastolic_anormal_moderate_low { ... } ) -> HR    //Regra 2
```

No caso exemplificado, para o evento `diastolic_anormal_moderate_low` (o valor da pressão sistólica está anormal moderada para baixo), caso o sistema esteja em um estado de risco normal (`NR`), o evento de transição gerado é `MR`. Enquanto que se o sistema estiver em qualquer outro estado diferente de `NR`, o sistema deve gerar um evento de transição `HR`. O controle sobre qual regra aplicar fica sob responsabilidade da solução. A proposta deste trabalho é que seja priorizada a regra mais específica, no caso a Regra 1 do Quadro 4.3. Dessa forma, caso não exija necessidade de especificar o estado anterior para decidir qual o novo estado o sistema deve transitar, o especialista do domínio não precisará inserir três regras semelhantes, somente modificando o estado do sistema, conforme Quadro 4.4.

Quadro 4.4: Exemplo onde duas regras diferentes para o mesmo evento podem gerar mudanças diferentes na máquina de estado do sistema

```
1 ( NR , diastolic_anormal_moderate_low { ... } ) -> MR   //Regra 1
2 ( MR , diastolic_anormal_moderate_low { ... } ) -> HR   //Regra 2
3 ( HR , diastolic_anormal_moderate_low { ... } ) -> HR   //Regra 3
```

4.2 Linha de Produtos de Software Dinâmica

Os RSCHs devem se adaptar às preferências e necessidades do indivíduo monitorado, que se modificam ao longo do tempo de acordo com os seus problemas de saúde (diabetes, hipertensão, deficiência física), presença/ausência de sensores e a interação direta com usuário. O mesmo sistema de monitoração pode ser utilizado por diversos usuários, mas devido a estas preferências, um conjunto de funcionalidades pode ser variável tanto antes ou durante a execução do sistema. Além disso, para o mesmo usuário, conforme os seus dados vitais variam, é possível que o conjunto de funcionalidades disponível para ele não seja mais o suficiente para avaliar corretamente seu estado de saúde. Um indivíduo que subitamente sofre um infarto, por exemplo, não necessariamente precisa uma solução de monitoração saiba sobre a velocidade de movimentação deste indivíduo, mas sim precisa que funcionalidades relacionadas à notificação e avaliação de variáveis cardíacas sejam levadas em consideração.

Neste contexto, a abordagem de Linha de Produtos de Software (LPS) se mostra promissora para tratar variabilidade do sistema. Entretanto, a modelagem de LPS tradicional apenas trata variabilidade de forma estática, não permitindo, por exemplo, que uma funcionalidade seja ativada ou desativada durante o tempo de execução. Para tal habilidade, este trabalho utiliza Linha de Produto de Software Dinâmica - LPSD [Hallsteinsen et al., 2008].

Uma LPSD pode ser construída a partir da definição de artefatos, modelo de *features* (MF), mapeamento entre essas duas entidades, e regras de transição entre as configurações da LPSD. O modelo de *features* é essencial para capturar e gerenciar as comunalidades e variabilidades entre os sistemas de uma LPS [Czarnecki and Eisenecker, 2000]. A mudança de configuração deve sempre escolher um conjunto de *features* de um modelo de *features*

que satisfaz suas regras de construção, nomeado como *configuração válida*. A Figura 4.2 detalha o modelo de *features* usado por este trabalho.

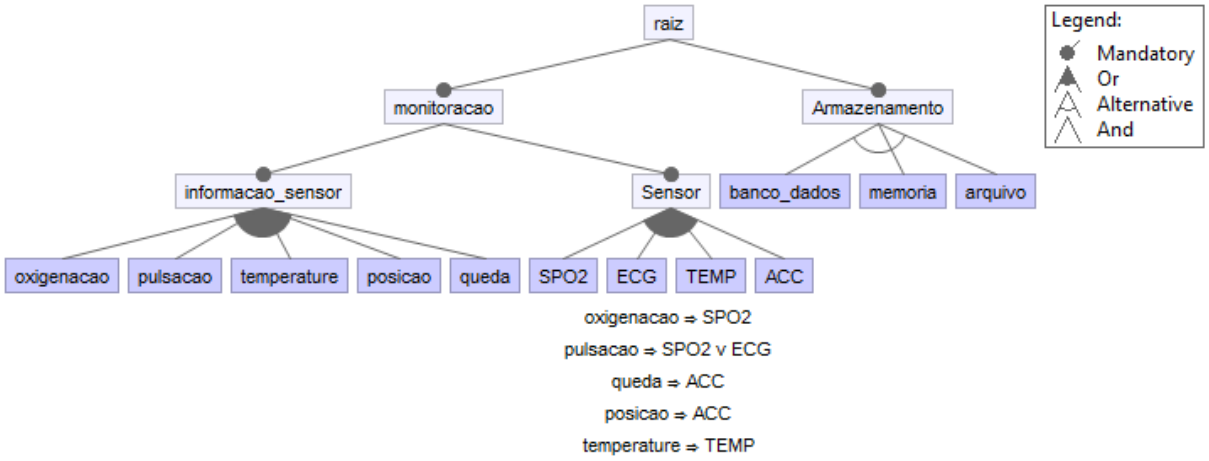


Figura 4.2: Modelo de *features* do Sistema de Monitoração do Corpo Humano

Algumas *features* estão presentes no modelo apenas com o intuito de representar um agrupamento semântico, como por exemplo as *features* de `informacao_sensor`, `sensor` e `armazenamento`. As chamadas *features abstratas* (preenchidas com a cor mais clara no diagrama de features) não estão associadas a nenhum artefato do sistema e por este motivo não interferem nos serviços providos pelo produto [Thüm et al., 2011]. *Features* que estão associadas ao código e alteram a qualidade de serviço provida por um sistema são chamadas de *features* concretas.

As *features* concretas do sistema estão agrupadas por três *features*-pai abstratas: Informação de Sensores (Oxigenação, Pulsação Cardíaca, Temperatura, Posição e Queda), Sensores (Sensor ECG, Sensor SPO2, Sensor Temperatura e Sensor Acelerômetro) e Armazenamento (Banco de Dados, Memória e Arquivo). No decorrer desse capítulo, essa separação será utilizada para diferenciar propriedades que o sistema deve considerar no momento da análise de qualidade de uma configuração da LPS. As *features* concretas afiliadas à *feature* abstrata `Informação de Sensores` serão nomeadas *features de informação*. As *features* concretas afiliadas à *feature* abstrata `Sensores` serão nomeadas *features sensores* e, por fim, as *features* concretas afiliadas à *feature* abstrata `Armazenamento` serão nomeadas *features armazenamento*

A estratégia de busca por configurações válida aplica os seguintes passos:

1. Tradução dos relacionamentos e restrições *cross-tree* de um MF para uma fórmula proposicional.

   As regras de transformação estão representadas na Figura 2.3 [Batory, 2005].

   A Figura 4.3 exibe a fórmula proposicional correspondente ao modelo de *features* da Figura 4.2. Cada parâmetro representa uma *feature* dentro do modelo de *features*.

2. Aplicação da fórmula proposicional resultante em um *SAT-Solver*.

   Qualquer fórmula proposicional pode ser convertida em uma fórmula equivalente na Forma Normal Conjuntiva (Conjunctive Normal Form - CNF) [Cook, 1971]. Este formato é suportado por diversos algoritmos de *SAT-Solvers*. O *SAT-Solver* retorna

$$
\begin{aligned}
&(raiz) \wedge \\
&(raiz \leftrightarrow monitoracao) \wedge \\
&(raiz \leftrightarrow armazenamento) \wedge \\
&(sensor \leftrightarrow monitoracao) \wedge \\
&(informacao\_sensor \leftrightarrow monitoracao) \wedge \\
&(exportar\_dados \rightarrow monitoracao) \wedge \\
&(armazenamento \leftrightarrow ((banco\_dados \wedge \neg memoria \wedge \neg arquivo) \vee (\neg banco\_dados \wedge \\
&memoria \wedge \neg arquivo) \vee (\neg banco\_dados \wedge \neg memoria \wedge arquivo))) \wedge \\
&(sensor \leftrightarrow (SPO2 \vee ECG \vee TEMP \vee ACC)) \wedge \\
&(informacao\_sensor \leftrightarrow (oxigenacao \vee pulsacao \vee temperatura \vee posicao \vee \\
&queda))) \wedge \\
&(oxigenacao \rightarrow SPO2) \wedge \\
&(pulsacao \rightarrow (SPO2 \vee ECG)) \wedge \\
&(posicao \rightarrow ACC) \wedge \\
&(queda \rightarrow ACC) \wedge \\
&(temperatura \rightarrow TEMP)
\end{aligned}
$$

Figura 4.3: Fórmula proposicional correspondente ao modelo de *features* da Figura 4.2

todas as possíveis valorações para as variáveis da fórmula proposicional que quando aplicados à fórmula retorna um valor verdadeiro [Benavides et al., 2010]. A valoração verdadeira para a variável da fórmula representa a seleção da variável no Modelo de *Features*. O contrário acontece quando a variável está valorada como falso.

A estratégia de redução no número de configurações sugerida por este trabalho consiste em agrupar *features* semanticamente semelhantes que são normalmente ligadas ao software em um mesmo momento via construção de unidades de ligação (*Binding Unit* - BU) [Rosenmüller et al., 2011a]. A Figura 4.4 apresenta uma sugestão de construção de *binding units* para uma RSCH. Uma possível binding unit é composta pelas *features spo2*, *ecg*, *pulsacao* e *oxigenacao*, por exemplo.

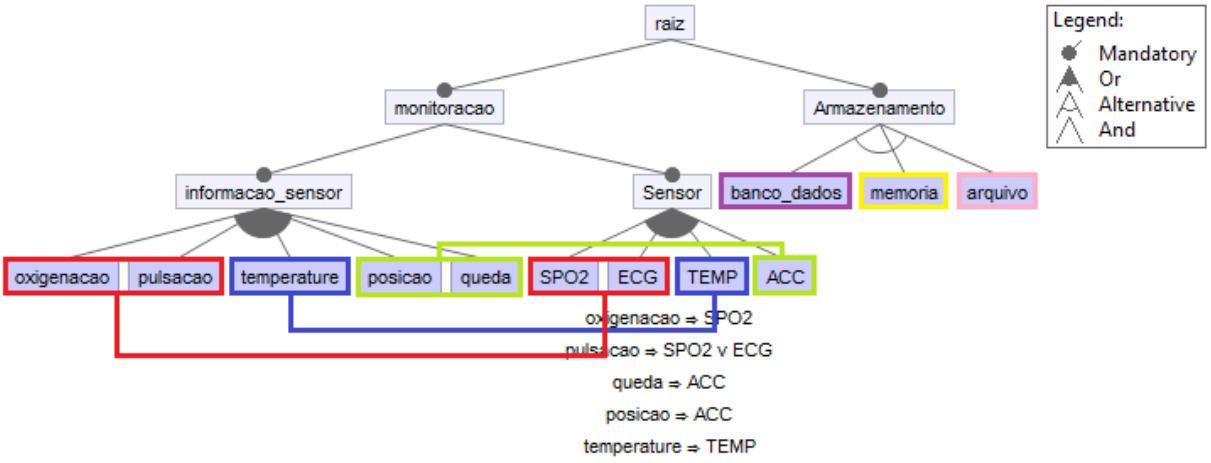


Figura 4.4: Binding Units sugeridos para o modelo de *features* da RSCH

A construção de BUs para a criação da fórmula proposicional modifica a semântica na interpretação do resultado do SAT-Solver, de forma que quando a variável associada

a uma BU tiver o valor verdeiro, todas as *features* da BU são selecionadas. Conforme exemplo apresentado nas Figuras 4.2 e 4.4, o número de variáveis da fórmula proposicional reduz de 12 para 6.

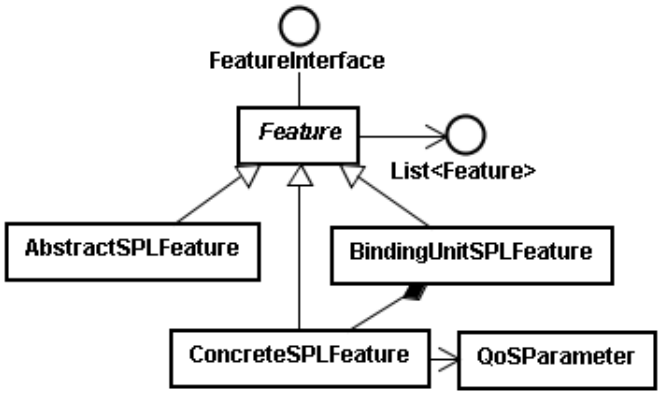


Figura 4.5: Diagrama de classes com as classes de definição de uma *feature*

O diagrama de classes da Figura 4.5 detalha o relacionamento entre as classes que modelam o conceito de *feature* e suas extensões na metodologia proposta. Conforme representação, uma *feature* é modelada como a classe `Feature` do diagrama. Como o MF tem um formato de árvore, cada objeto desta classe possui uma lista de *features* filhas. Uma *feature* abstrata é modelada pela classe `AbstractSPLFeature`. As *features* concretas estão associadas a uma lista de parâmetros de qualidade (Classe `QoSParameter`) e são modeladas pela Classe `ConcreteSPLFeature`. Por fim, as *binding units* agrupam um conjunto de *features* concretas e são modeladas pela classe `BindingUnitSPLFeature`.

4.3 Seleção de configurações adequadas para o estado de risco

A proposta deste trabalho está baseada na hipótese de que é possível estabelecer um valor de Qualidade de Serviço (QoS) para cada configuração do ambiente. Conforme definição apresentada no Capítulo 3 para configuração do ambiente e avaliação da proposta de LPSD para o RSCH realizada na Seção 4.2, percebe-se que todos os elementos que compõem a configuração do ambiente da RSCH estão presentes no modelo de *features* da Figura 4.2. Todos os elementos controlados pelo sistema, variáveis de saúde e instrumentos presentes no RSCH, estão presentes também no modelo de *features*.

A hipótese mais detalhada deste trabalho é de que é possível estabelecer um valor de Qualidade de Serviço (QoS) para cada configuração da LPS, como apresentado na Hipótese 4.1

Hipótese 4.1. *Seja uma lps um conjunto de configurações válidas de uma LPS, c uma configuração em lps. É possível especificar uma função QoS(c) capaz de retornar um valor de qualidade de serviço desta configuração, como apresentado na Fórmula 4.1.*

$$QoS(c) : FM \rightarrow \mathbf{R} \tag{4.1}$$

A mudança de requisitos disparada pela identificação de eventos gera um evento de mudança de configuração do ambiente, e por sua vez, da configuração da LPS. O sistema busca uma nova configuração avaliando os valores de QoS providos por cada configuração. Para configuração ser considerada adequada em um determinado estado de risco, o QoS da configuração deve satisfazer o valor mínimo especificado pelo estado. Ou seja, é possível definir uma função capaz de atribuir um valor mínimo de qualidade requerido e máximo de qualidade desejável por um estado de risco, conforme Fórmula 4.2 e filtrar as configurações de uma LPS usando Hipótese 4.2

$$QoS(s) : States \rightarrow [\mathbf{R}, \mathbf{R}], \text{onde } States \text{ é o conjunto de estados de risco do sistema} \tag{4.2}$$

Hipótese 4.2. *Seja uma lps um conjunto de configurações válidas de uma LPS, c uma configuração em lps. A configuração c é considerada adequada para um estado de risco s, se o valor de QoS provido por ela estiver no intervalo de QoS requerido pelo estado, conforme Fórmula 4.3*

$$Qos(c) \in Qos(s) \rightarrow \text{c é adequada para o estado de risco } s \tag{4.3}$$

O restante deste capítulo descreve como a modelagem proposta avalia as configurações de uma LPS em RSCH observando os seus valores de qualidade e o estado de risco do indivíduo monitorado. A Seção 4.2 descreve a estratégia de modelagem para tratar as diferentes configurações do sistema de acordo com o estado de risco de saúde. A Seção 4.4 refina a arquitetura utilizada por essa abordagem e, finalmente, a Seção 4.5 detalha o processo de cálculo de qualidade de uma configuração.

4.4 Arquitetura de *Software*

Ambientes computacionais com a habilidade de gerenciar sua própria configuração e dinamicamente escolher uma nova em virtude de mudanças de requisitos de negócio, contexto ou objetivos podem dividir suas funcionalidades em monitoração, análise, planejamento e execução de ações [Parashar and Hariri, 2005]. A arquitetura utilizada por este trabalho, entretanto, divide a atividade de análise e separa sua funcionalidades entre as etapas de monitoração, planejamento e execução.

A Figura 4.6 apresenta uma visão dinâmica do relacionamento entre os componentes da arquitetura de referência [Calinescu et al., 2011]. Cada componente é representado por uma caixa e o relacionamento entre eles é expresso via setas. As setas apontam para o componente que recebe os dados de outro componente.

Os componentes Sensor, Percepção e Identificação realizam a função de monitorar o sistema e identificar eventos. O componente de Planejamento é responsável por planejar as ações do sistema e por fim, os componentes de Controle e Ação realizam as mudanças no sistema e nos serviços externos. Um detalhamento sobre a atividade de cada componente é feito abaixo:

Sensor Todos os dados de baixo nível, medidos pelos sensores ou via interação com usuário ou outros sistemas, são recebidos por esta camada de *software*. Dados de

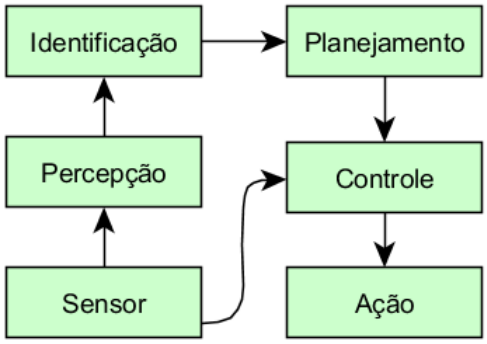


Figura 4.6: Arquitetura de referência

baixo nível que não necessitam de processamento são diretamente encaminhados à camada de controle a fim de reduzir o tempo de processamento.

Percepção Interpreta os dados da camada de sensor. A camada de percepção classifica os dados brutos medidos por serviços externos em eventos. Um evento é uma abstração sobre os dados brutos encaminhados para o sistem.

Identificação Camada responsável por realizar a fusão de dados e a interpretação dos eventos em termos de risco. A identificação é lança eventos que devem ser interpretados pelo sistema para que este verifique se é necessário modificar seu estado atual.

Planejamento Camada recebe evento da camada de identificação e verifica se é necessário modificar o seu estado atual. Caso haja necessidade, define qual a nova configuração a ser adotada e que tipo de ação deve ser executada a fim de satisfazer os novos requisitos de qualidade.

Controle Camada realiza a modificação do sistema conforme instruções da camada de planejamento. A partir de comandos recebidos diretamente de outros serviços que são recebidos pela camada de *sensing*, o controle altera a configuração dos sistemas.

Ação Camada influencia o exterior encaminhando dados de baixo nível.

4.4.1 Refinamento da arquitetura

Esta seção descreve a arquitetura concreta utilizada por este trabalho. A arquitetura de referência descrita na Seção agrupa seus componentes em três abstrações: gerenciador de contexto, gerenciador de adaptação e configurador [IESE, 2008, Gat, 1998, Kramer and Magee, 2007]. A Figura 4.7 apresenta uma visão dinâmica entre os componentes. As anotações nas setas indicam ações geradas pelo componente de origem no componente de destino.

De acordo com a Figura 4.7, o Gerenciador de Contexto (`GCxt`) é responsável por se comunicar com serviços externos ou usuário. Além disso, inclui as funcionalidades de interpretação de dados brutos, transformação destes dados em eventos e fusão dos eventos. Cabe a este componente ainda a identificação da presença de um evento que motive a mudança do estado de saúde do indivíduo. Quando este evento é identificado, o

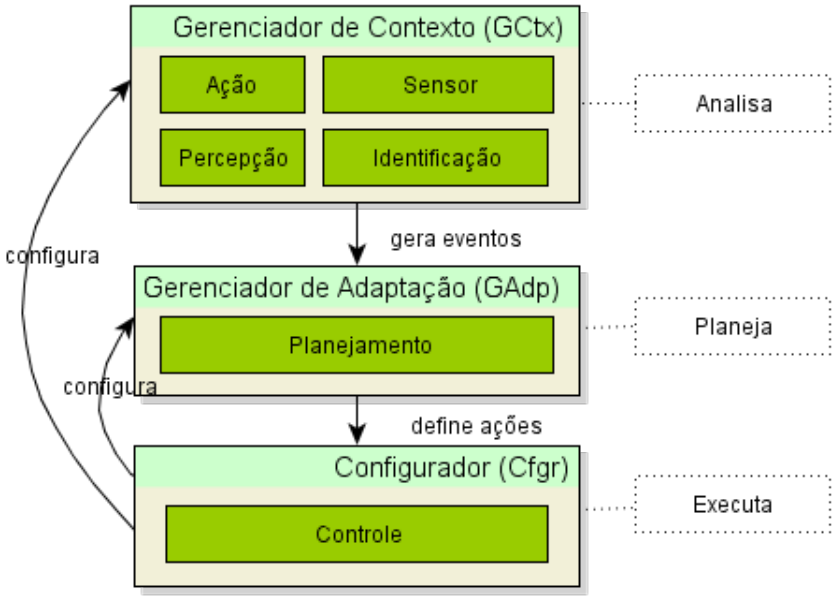


Figura 4.7: Refinamento da arquitetura de referência para a arquitetura concreta

GCxt repassa para o gerenciador de adaptação uma notificação de reavaliação do estado atual de risco.

O Gerenciador de Adaptação (`GAdp`) é responsável por realizar a avaliação sobre uma necessidade de mudança do estado de risco. Ele avalia se a configuração atual do sistema é capaz de prover os serviços necessários para o atual estado de saúde. Caso não seja, o GAdp busca uma nova configuração e repassa para o Configurador uma notificação de reconfiguração e a nova configuração a ser adotada.

Por fim, o Configurador (`Cfgr`) é responsável por modificar componentes das camadas anterior a fim de adaptar o sistema às novas exigências do contexto. O `Cfgr` ativa ou desativa funcionalidades do sistema conforme nova configuração.

O refinamento da arquitetura de referência se faz necessário a fim de descrever uma arquitetura real, obtendo assim uma estrutura mais complexa e um detalhamento maior sobre o comportamento dos componentes [Bass et al., 1998]. A estrutura do sistema deve refletir as funcionalidades de cada camada proposta na Figura 4.6.

Gerenciador de Contexto

Este componente é responsável por: (1) receber dados brutos enviados por serviços externos ou diretamente pelo usuário via interface gráfica; (2) interpretar dados brutos e gerar eventos; (3) agregar eventos de forma a aumentar o nível de abstração sobre os dados; e (4) perceber a presença de um evento que pode alterar o estado de risco de saúde do indivíduo monitorado. A Figura 4.8 apresenta uma visão do comportamento dinâmico dos componentes presentes no Gerenciador de Contexto.

Os canais de comunicação com a RSCH e serviços externos normalmente são abertos via conexão que permite envio em duas direções. Ou seja, pela mesma conexão é possível enviar e receber dados sem que haja um controle específico para a saída ou entrada de dados. Por esse motivo, as camadas de Sensor e Ação da arquitetura de referência são

tratadas pelo mesmo componente quando se trata de comunicação com serviços externos (Componente de Comunicação com Serviço Externo), ou com o usuário (Componente de Interface Gráfica).

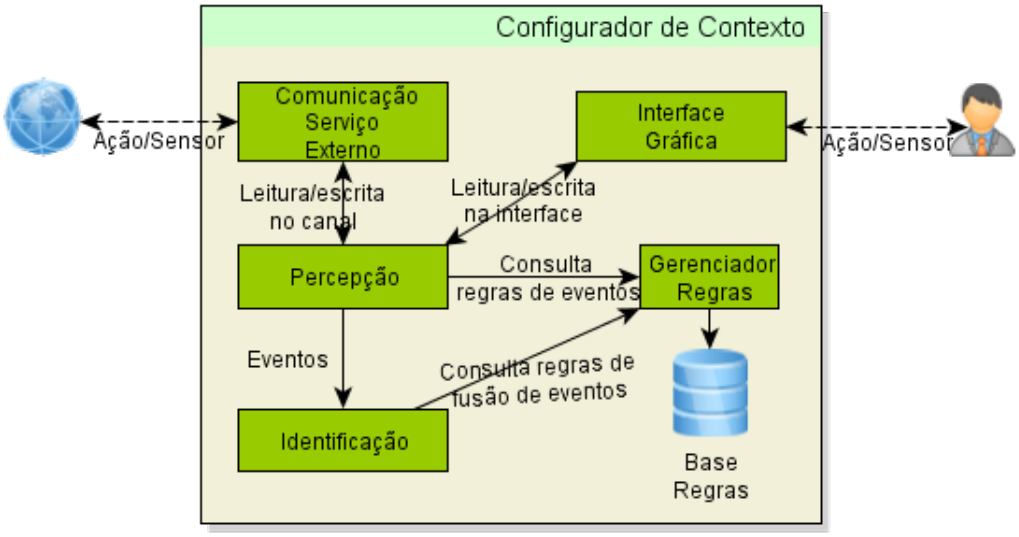


Figura 4.8: Refinamento do Gerenciador de Contexto

Como o sistema deve se adaptar a qualquer indivíduo monitorado, e indivíduos diferentes podem ter seus dados interpretados de forma diferentes (Ver Seção 3), o sistema deve permitir que os dados recebidos sejam interpretados conforme avaliação do especialista no domínio. Para tal, regras de geração de eventos e transições no DFA devem ser inseridas no sistema e avaliadas sempre que um novo dado é recebido. Essas regras são armazenadas na base de regras e acessíveis via o Gerenciador de Regras. O componente de percepção apenas gera eventos, enquanto que o componente de Identificação é capaz de unir esses eventos.

Cada sensor (dispositivo externo) possui uma parte do software específica para o tratamento dos seus dados brutos enviados, que chamaremos de interpretador de dados de sensores (`IDS`). Esse `IDSs` é representado pelo componente de `Percepção`. Quando o `IDS` de um sensor recebe um dado bruto, ele consulta as regras de geração de eventos via Gerenciador de Regras. Caso algum evento de saúde seja identificado, o `IDS` gera um evento e encaminha para o Identificador. O Identificador também consulta as regras de fusão de eventos via Gerenciador de Regras. Caso haja alguma regra de fusão de eventos que pode modificar a interpretação sobre o estado de saúde do indivíduo monitorado, o Identificador repassa essa notificação de provável mudança para o Gerenciador de Adaptação.

Gerenciador de Adaptação

O gerenciador de adaptação contém a lógica que a RSCH deve usar para satisfazer os requisitos de qualidade do estado do sistema. Seu comportamento dinâmico é detalhado na Figura 4.9.

Após a recepção do evento que pode modificar a interpretação sobre o estado de saúde enviado pelo Identificador (Ver Seção 4.4.1), o Controlador de Adaptação (`CTRL_ADP`) con-

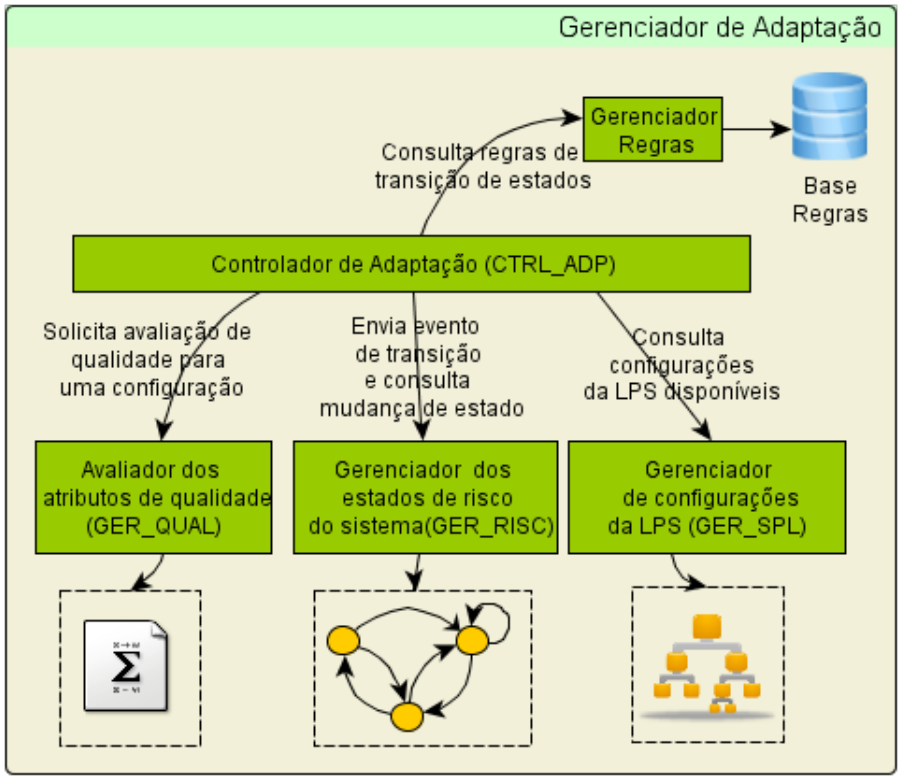


Figura 4.9: Refinamento do Gerenciador de Adaptação

sulta o Gerenciador dos estados de risco do sistema (`GER_RISC`) do sistema. O `GER_RISC` utiliza o evento junto ao estado atual do sistema e verifica se este risco gera a mudança de estado na máquina de estado. Caso exista, retorna para o `CTRL_ADP` o novo estado do sistema, `ESTADO`. Os estados de risco do sistema estão modelados conforme definição da Seção 4.1.

Com o novo estado, o `CTRL_ADP` consulta qual estratégia de cálculo de qualidade deve ser utilizada para este estado específico. Esta consulta é feita junto ao Gerenciador de Qualidade do Sistema (`GER_QUAL`). Após obter a estratégia de cálculo, o `CTRL_ADP` busca por uma configurações válidas da LPS via consulta ao Gerenciador de Configuração da LPS `GER_SPL`.

O `GER_SPL` retorna uma configuração válida da LPS utilizando a transformação de suas restrições de *features* (ou-*feature*, *features* alternativas, obrigatórias ou opcionais) e as restrições *cross-tree* em expressões lógicas. Por meio de um SAT-Solver, resultados que satisfazem essa expressão lógica são traduzidos em seleção/deseleção de *features* que geram uma configuração válida para um modelo de *features*, chamada de `CONF`. Após escolher uma possível configuração para ser utilizada pelo sistema, o `CTRL_ADP` repassa ao `GER_QUAL` a tarefa de calcular o valor de qualidade da `CONF`. Após receber o valor de qualidade para a `CONF`, o `CTRL_ADP` consulta o `ESTADO` e verifica qual a faixa de valor de qualidade que o estado define. Se o valor de qualidade calculado para `CONF` estiver na faixa de valores ou for maior do que a faixa, esta configuração é inserida em uma estrutura temporária que armazena possíveis configurações a serem escolhidas.

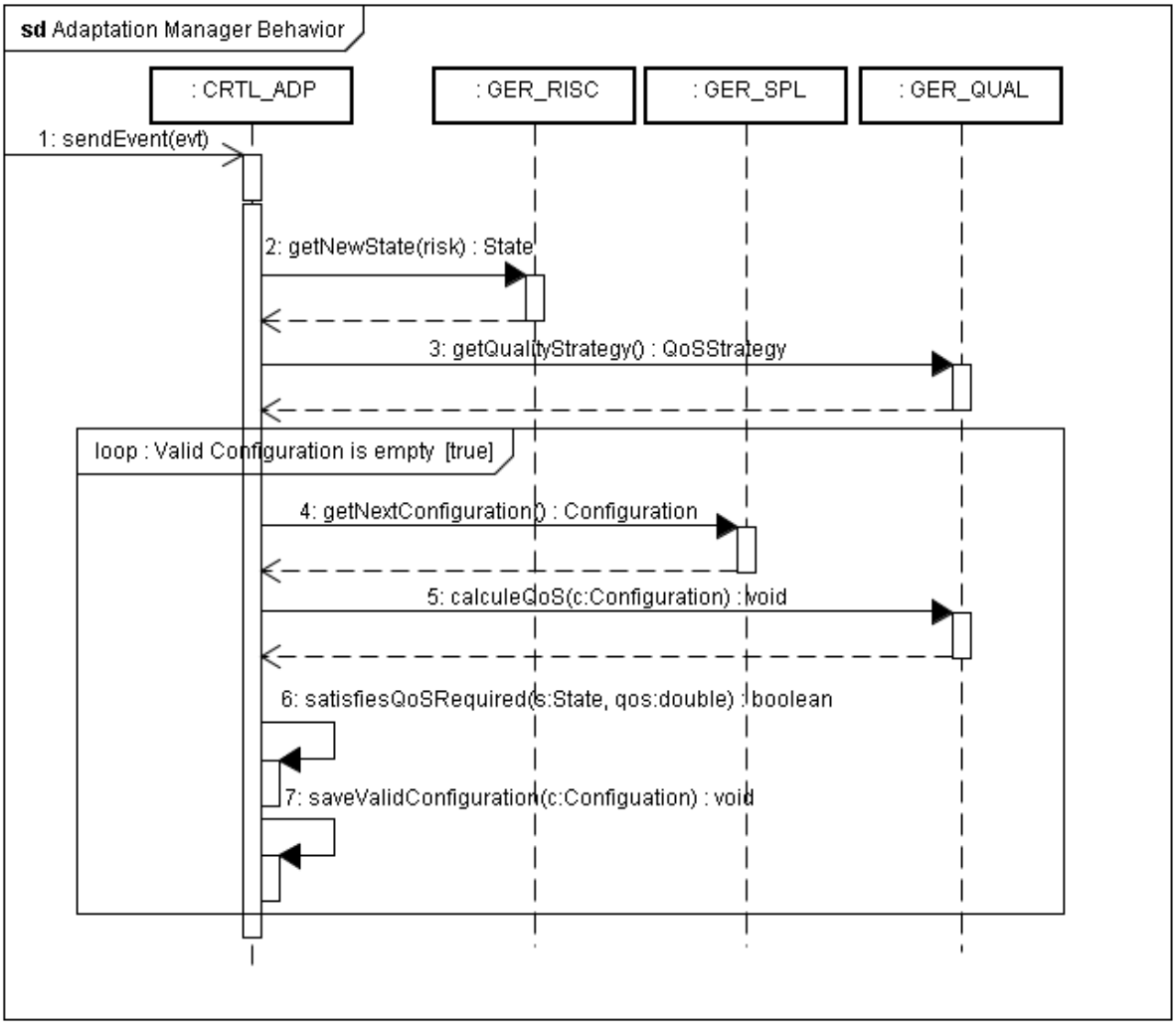


Figura 4.10: Diagrama de Sequência - Identificação de um evento e busca por uma configuração

O processo de identificação de um evento que motiva uma mudança de estado e a busca por uma nova configuração é detalhado na Figura 4.10.

Configurador

De posse da configuração escolhida, o Gerenciador de Adaptação faz o repasse da nova configuração a ser executada para o Configurador (`Cfgr`). Os componentes do Configurador estão representados na Figura 4.11.

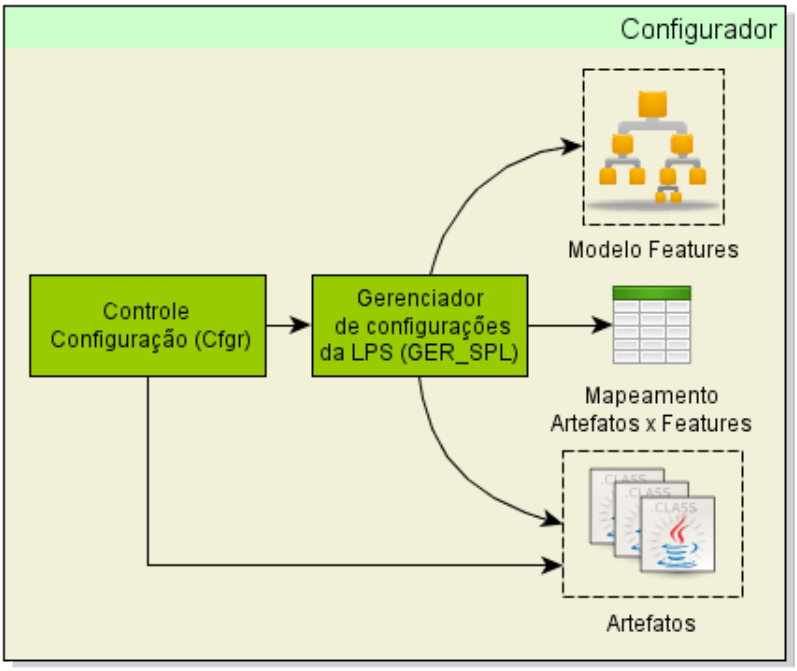


Figura 4.11: Componentes do Configurador

O `Cfgr` percorre a configuração atual e examina quais componentes presentes na nova configuração já estão ativos. Esta consulta é feita com o Gerenciador de Configuração da LPS (`GER_SPL`). Os componentes presentes na configuração atual que não estão presentes na nova configuração são desativados. O `GAdp` pode definir algumas ações para modificações de parâmetros de qualidade de sensores. Cabe ao `Cfgr` traduzir as ações definidas pelo `GAdp` em comandos de baixo nível que deverão sem encaminhados ao `GCtx`, e dele ser encaminhado aos serviços externos.

O mapeamento entre as *features* e o trecho de código responsável pela sua execução é consultada via `GER_SPL`.

4.5 Gerenciamento de Qualidade

O uso de métricas de *software* reduz a subjetividade na avaliação e controle sobre a qualidade do software, fornecendo uma base quantitativa para a tomada de decisão [IEEE, 1998]. A avaliação típica de atributos de qualidade em sistemas sensíveis ao contexto inclui o levantamento de custo de operação e valores probabilísticos de qualidade tais como disponibilidade, confiabilidade e reputação [Calinescu et al., 2011].

Esta seção descreve como são calculados os valores de qualidade considerados neste trabalho. Ao todo, cinco parâmetros de qualidade são avaliados: confiabilidade, qualidade e quantidade de informação, tempo de vida estimado do sistema e frequência da amostra dos dados medidos por sensores. Por ser uma solução de monitoração de saúde e se tratar de um domínio crítico, o sistema deve controlar a probabilidade de falha dos seus componentes [Rodrigues et al., 2012]. Esta avaliação é feita via análise do valor de confiabilidade. Os parâmetros de qualidade de informação e taxa de amostragem são

necessários para decidir se os eventos identificados representam a real situação do indivíduo monitorado [Olson, 2002]. O parâmetro de quantidade de informação controla o número de informações avaliadas para cada risco de saúde. Quanto maior a quantidade de informações, mais preciso é o diagnóstico sobre o estado do risco de saúde do indivíduo [Carvalho, 2005]. O controle sobre o tempo de vida estimado permite que em situações de baixo risco, a solução economize energia dos sensores a fim de prolongar a monitoração do estado de saúde do indivíduo (Using Probabilistic Model Checking for Dynamic Power Management).

A Seção 4.5.1 apresenta como são calculados os atributos de qualidade individualmente. A Seção 4.5.2 detalha como as duas estratégias de cálculo de qualidade com múltiplo atributos são considerados pela arquitetura proposta.

4.5.1 Parâmetros de Qualidade de uma RSCH

Após a identificação de mudança de estado de risco e percepção de que a configuração do sistema atual não fornece serviços ou fornece além do necessário serviços para este novo estado, o componente de Gerenciador de Adaptação (`GAdp` - Seção 4.4.1) inicia processo de busca por uma configuração mais adequada. Este processo consiste de uma verificação de algumas configurações válidas da LPS e a avaliação dos seus valores de qualidades.

A avaliação de qualidade de uma configuração consiste na medição de cinco parâmetros de qualidade e da combinação destes dados. Esta seção detalha o processo de avaliação destes parâmetros de forma individual.

Qualidade de Informação

Com o objetivo de aumentar a confiança sobre os valores medidos por um sensor, é comum que uma rede de sensores utilize diversos tipos de sensores para mensurar a mesma informação. Além da vantagem de aumentar a precisão da informação, é possível manter a execução correta do sistema mesmo que algum desses sensores apresente uma falha.

O parâmetro de *Qualidade de Informação* (QoI) está relacionado ao valor de acurácia da medição feita pelos sensores e outros métodos de entrada de dados do RSCH. Através da análise de domínio e da acurácia dos sensores, é possível estabelecer valores de confiança entre os sensores e serviços e o valor medido por eles [Olson, 2002]. Trataremos este nível de confiança como a avaliação do atributo de qualidade de informação. Em uma RSCH, por exemplo, o valor de pulsação cardíaca pode ser mensurado via oxímetro de pulso (SPO2) ou via eletrocardiógrafo (ECG). A qualidade da informação de um ECG sobre este dado vital é maior do que a qualidade da informação medida por um SPO2 [Carvalho, 2005].

A Figura 4.12 apresenta o relacionamento entre diversos sensores dentro de uma RSCH e suas qualidades de informação. Os dados medidos por sensores são apresentados como retângulos, enquanto que os sensores do sistema são apresentados como elipses.

De acordo com a Figura 4.12, a pressão arterial pode ser aferida por três sensores diferentes: sensor de oxímetro (QoI 0.7), sensor de pressão arterial (QoI 1.0) e sensor de fluxo sanguíneo (QoI 0.8). Além dos sensores, outras fontes de informação podem ser utilizadas para medir dados vitais de um indivíduo. É possível, por exemplo, que o sistema forneça uma tela onde o usuário pode manualmente inserir uma informação sobre

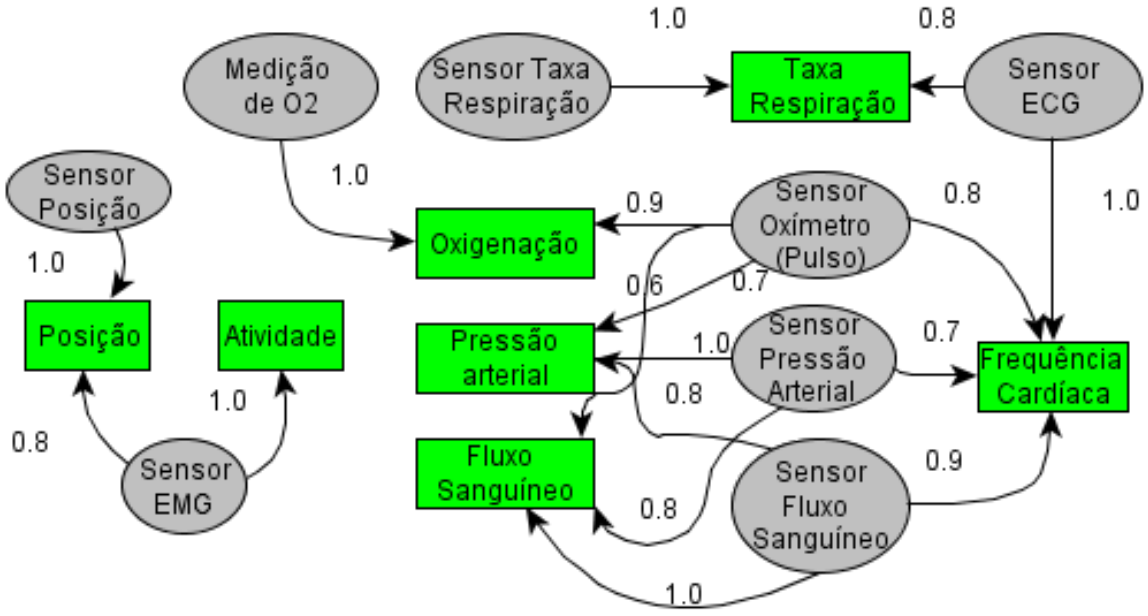


Figura 4.12: Qualidade de Informação dos valores medidos por sensor em um RSCH [Carvalho, 2005]

a sua pressão arterial. Para esses casos, devido à falta de informação sobre a acurácia dos valores inseridos, o valor de QoI atribuído é 1.0.

Este trabalho avalia em tempo de execução a QoI das seguintes informações medidas via sensores: pulsação cardíaca, oxigenação, temperatura, movimento e queda.

Conforme apresentado na Seção 4.2, as *features* do sistema que estão relacionadas aos eventos gerados por sensores são chamadas de *features* de informação(*finf*). O valor de uma (*finf*) deve ser recebido por alguma fonte de informação, seja ele formulário de entrada, serviços externos tais como *web-services* ou sensores. A Definição 7 apresenta a especificação do conjunto de *features* fontes para uma *feature* de informação `finf`.

Definição 7 Seja uma *feature* de informação $finf$ e lps um conjunto de configurações válidas de uma LPS. O conjunto de fontes $SRC(finf, lps)$ é o conjunto de *features* em lps que são fontes da informação relacionada à $finf$.

O valor de QoI é medido via comparação entre a fonte utilizada por uma `finf` e a melhor fonte possível para ela. A função especificada na Definição 8 seleciona a melhor fonte de informação para todas aquelas possíveis dentro da LPS.

Definição 8 Seja $finf$ uma *feature* de informação e lps um conjunto de configurações válidas de uma LPS. A função $fQMax(finf, lps)$ retorna o maior valor de qualidade que uma fonte de informação em uma lps oferece à feature $finf$ conforme Fórmula 4.4.

$$fQMax(finf, lps) = \max\{QoI(s, finf), \text{onde } s \in SRC(finf, lps)\} \quad (4.4)$$

O cálculo do valor de QoI de uma *finf* é feito individualmente para os valores de qualidade associados às fontes desta informação. Sua valoração representa uma proporção sobre o máximo valor de qualidade associado à *finf*, conforme Definição 9.

Definição 9 Seja $finf$ uma *feature* de informação, *lps* um conjunto de configurações válidas de uma LPS e c uma configuração em *lps*. O valor de *qualidade de informação (QoI)* de uma c é a relação de proporção entre a soma de todas os valores de QoI das *features* de informação $finf$ presentes em c e a soma das máximas QoI destas $finf$, conforme Fórmula 4.5.

$$QoI(c) = \frac{\sum_{finf \in c} fQMax(finf, \{c\})}{\sum_{finf \in c} fQMax(finf, spl)}, \text{onde } finf \in c, c \in spl \quad (4.5)$$

Quantidade de Informação

Conforme apresentado na Seção 4.5.1, a inserção de novas *features* na LPS diminui o valor de confiabilidade do sistema. Para compensar esse comportamento, junto ao especialista de domínio, decidiu-se inserir um novo parâmetro de qualidade relacionado à quantidade de informação presente no sistema. A quantidade de informação diz respeito ao número de diferentes informações manipuladas pelo sistema [Ganzach and Schul, 1995].

O número máximo de *features* selecionadas simultaneamente em uma LPS não necessariamente representa o número máximo de *features* no modelo de features. Este comportamento é observado em virtude dos relacionamentos de exclusão e alternatividade entre as features. Considerando que *features* abstratas do sistema não acrescentam ou alteram parâmetros de qualidade de uma LPS, o número máximo de *features* de uma LPS é sempre menor ou igual ao número máximo de *features* concretas do sistema. A Definição 10 especifica uma função auxiliar para a seleção das *features* concretas de uma configuração válida de uma LPS.

Definição 10 Seja uma *lps* um conjunto de configurações válidas de uma LPS, c uma configuração em *lps*. A função $features\_concrete$ seleciona o conjunto de *features* concretas em c, conforme Fórmula 4.6

$$features\_concrete(c) = \{f\}, onde f \in c, c \in lps \text{ e } f \text{ é uma feature concreta} \quad (4.6)$$

O valor de quantidade de informação medido por este trabalho é calculado a partir da proporção de *features* selecionadas de uma configuração em relação ao número máximo de *features* selecionadas simultaneamente no sistema, conforme Definição 11.

Definição 11 Seja uma *lps* um conjunto de configurações válidas de uma LPS, c uma configuração em *lps*, e o MAX_CONF o tamanho da configuração c_1 válida em LPS com o maior número de *features* concretas. O valor de quantidade de informação

de uma configuração, $QtdI(c)$ é a proporção entre a quantidade de *features* concretas em c em relação ao máximo número de *features* selecionadas simultaneamente em *lps*, conforme expresso na Fórmula 4.7.

$$QtdI(c) = \frac{|features\_concrete(c)|}{MAX\_CONF}, features \in c, c \in lps \quad (4.7)$$

Tempo de funcionamento (Tempo de vida) dos sensores da rede de sensores

O consumo de energia é crítico para rede de sensores que operam em dispositivos com fontes de energias limitadas tais como bateria ou módulos fotovoltaicos. O controle de consumo de energia se torna mais importante quando os sistemas alimentados por essa fontes fontes de energias precisam manter parâmetros de qualidade relacionados à acurácia e taxa de amostragens altas [Kirousis et al., 2000].

Em uma RSCH, os requisitos de qualidade de um indivíduo em uma situação de baixo risco estão relacionados à maximização do tempo de vida e redução de confiabilidade ou qualidade dos dados. É possível, do ponto de vista médico, reduzir o número de sensores ativos ou reduzir a taxa de amostragens destes baseado na justificativa de prevalência dos eventos de risco [Carvalho, 2005].O valor de tempo de vida estimado para cada sensor da rede de sensores utilizada neste trabalho é de no máximo 20h.

Conforme apresentado na Seção 4.2, as *features* do sistema que estão relacionadas aos sensores são chamadas de *features* sensores ($fsensor$). A Definição 12 especifica uma função auxiliar de seleção das *features* sensores de uma configuração válida de uma LPS.

Definição 12 Seja uma *lps* um conjunto de configurações válidas de uma LPS, c uma configuração em *lps*. A função $features\_sensor$ seleciona o conjunto de *features* sensores em c, conforme Fórmula 4.8.

$$features\_sensor(c) = \{f\}, ondef \in c, c \in lps \text{ e } f \text{ é uma feature sensor} \quad (4.8)$$

É conhecido que a curva de descarga do *hardware* da bateria não segue um comportamento linear [Kirousis et al., 2000]. O tempo de vida de uma bateria depende do tipo de material do seu *hardware*, valores de ambiente a qual a bateria está exposta (temperatura do ambiente, por exemplo), e perfil de uso. Este trabalho, por sua vez, não se propõe a calcular esse valor em tempo de execução. Considera-se que o valor encaminhado pelo sensor sobre o status da sua bateria reflete o valor linear de seu tempo de vida, e que fica sob responsabilidade do *driver* do sensor a transformação linear sobre este valor, conforme Definição 13. O valor 100% recebido por um sensor, por exemplo, significa que o tempo de vida estimado é 20h.

Definição 13 Seja *lps* um conjunto de configurações válidas de uma LPS, $fsensors$ o conjunto de *features* de sensores em *lps* e $fsen$ uma *feature* sensor. A função $bat(fsen)$ retorna o valor em porcentagem do tempo de vida estimado pelo sensor associado a $fsen$.

$$bat : fsensors \rightarrow [0, 100] \tag{4.9}$$

Para uma determinada configuração de uma LPS, o tempo de vida estimado do sistema é calculado de acordo com Definição 14.

Definição 14 Seja uma *lps* um conjunto de configurações válidas de uma LPS, c uma configuração em *lps*. O parâmetro de qualidade relacionado ao tempo de vida de uma configuração c, $bat(c)$, é a proporção entre a soma dos valores de bateria dos sensores e o valor máximo de bateria fornecido por uma configuração c, conforme expresso na Fórmula 4.10.

$$bat(c) = \frac{\sum_{fsen \in c} bat(fsen)}{|features\_sensor(c)| * 100}, \text{onde } c \in lps, fsen \in features\_sensor(c) \tag{4.10}$$

Taxa de amostragem dos sensores da rede de sensores

Um outro importante fator para a avaliação de qualidade sobre a configuração de uma RSCH trata-se de quão recente é o dado recebido e tratado pelo sistema em relação ao momento em que foi medido. Quanto mais recente for o dado, maior é a chance de que ele represente o valor atual do indivíduo monitorado [Olson, 2002]. O valor de atualidade é tratado por este trabalho através da análise do parâmetro de taxa de amostragem dos sensores presentes na rede. Quanto maior é a taxa de amostragem, mais recente é o dado e mais rápido o sistema é capaz de perceber a mudança do estado de saúde do indivíduo monitorado. Embora traga benefícios ao diagnóstico, a maximização da taxa de amostragem tem um comportamento contrário à maximização do tempo de vida do sistema [Lanzisera et al., 2009]. Apesar disto, neste trabalho não foi desenvolvido nenhum estudo mais detalhado sobre a relação entre esses dois parâmetros.

Assim como o valor máximo de bateria, o valor da taxa de amostragem de um sensor depende dos *hardwares* de cada sensor presente na RSCH. Com o intúito de generalizar a solução, este trabalho considera que os valores encaminhados para a taxa de amostragem são normalizados antes de serem encaminhados ao sistema. Ou seja, fica sob responsabilidade do *driver* do sensor a normalização sobre este valor. A taxa de amostragem de qualquer sensor está entre 0 e o valor MAX_AMS, como na Fórmula 4.11.

$$Amostragem : feature \rightarrow [0, MAX\_AMS] \tag{4.11}$$

O valor de atualidade dos dados é medido conforme Definição 15.

Definição 15 Seja *lps* um conjunto de configurações válidas de uma LPS, c uma configuração em *lps*. O valor de qualidade de atualidade de uma configuração c, $QAtld(c)$, é a relação de proporção entre a soma de todos os valores de amostragem das *features* de sensores $fsen$ presentes em c e a soma das máximas amostragens destas $fsen$, conforme Fórmula 4.12.

$$QAtld(c) = \frac{\sum_{fsen \in c} Amostragem(fsen)}{MAX\_AMS * |features\_sensor(c)|}, fsen \in features\_sensor(c), c \in lps \tag{4.12}$$

Confiabilidade

A construção de um sistema cujos serviços providos sejam corretos e confiáveis tem sido uma preocupação até os dias atuais [Avizienis et al., 2004]. Nesse contexto, garantir a *dependabilidade* de um *software*, isto é, garantir que o *software* apresente níveis adequados de disponibilidade, confiabilidade, segurança (*safety*), integridade e manutenibilidade é um problema relevante. Em particular a confiabilidade, funcionamento continuamente correto, deve ser medida quantitativamente.

Medir a confiabilidade de um *software* não é uma tarefa trivial. Por meio da confiabilidade estimada dos diversos componentes é possível estimar a confiabilidade do software como um todo [Rodrigues et al., 2012]. Quando se trata de LPS, o desafio é ainda maior uma vez número de possíveis produtos cresce de maneira exponencial com a variabilidade. Assim, é necessário a utilização de técnicas que tornem viável o esforço de verificação de confiabilidade dos diversos produtos de uma LPS. Adicionalmente, o problema tratado nesse trabalho requer que a confiabilidade de determinados componentes seja estimada em tempo de execução uma vez seus correspondentes valores estimados de confiabilidade são dependentes do contexto de monitoração. Assim, a confiabilidade de produto da SPL que contenham tais componentes somente poderá ser verificada em tempo de execução.

Este trabalho utiliza uma técnica de *model checking* paramétrico para avaliar o valor de confiabilidade das diferentes configurações da LPSD [Nunes et al., 2012]. A estratégia de verificação de modelos paramétricos tem se mostrado uma abordagem adequada para avaliação de confiabilidade de LPSs, onde é gerada uma única fórmula para avaliar a LPS e não várias para avaliação das configurações individuais [Nunes et al., 2012, Ghezzi and Sharifloo, 2011]. Por meio de parâmetros definidos no modelo é possível gerar uma fórmula aritmética de tal forma que diferentes valorações de seus parâmetros representem diferentes configurações. Entretanto, a parametrização pode ser utilizada para outras finalidades que não apenas o tratamento de variabilidade. Assim, é possível definir parâmetros representando valores de confiabilidade de componentes cujo valor estimado não é conhecido inicialmente ou é variável em tempo de execução. O processo de avaliação de confiabilidade adotado por esse trabalho executa conforme sequência do fluxo apresentado na Figura 4.13.

A técnica utiliza como entrada o modelo de *features* de uma LPS e os diagramas de atividades e sequência da LPS. Cada atividade dentro de um diagrama de atividades (DA) é refinada por um diagrama de sequência (DS) de tal forma que o diagrama de atividades representa uma visão geral do fluxo de execução dos produtos da LPS e os diagramas de sequência detalham cada uma dessas atividades. As transições entre os componentes do DS são anotadas com o valor de confiabilidade estimada do componente.

Utilizando-se desses diagramas são construídos modelos probabilísticos baseados em cadeia de Markov [Kay, 2006]. Por meio da verificação desses modelos é obtida uma fórmula paramétrica capaz de calcular os diferentes valores de confiabilidade das configurações da LPS.

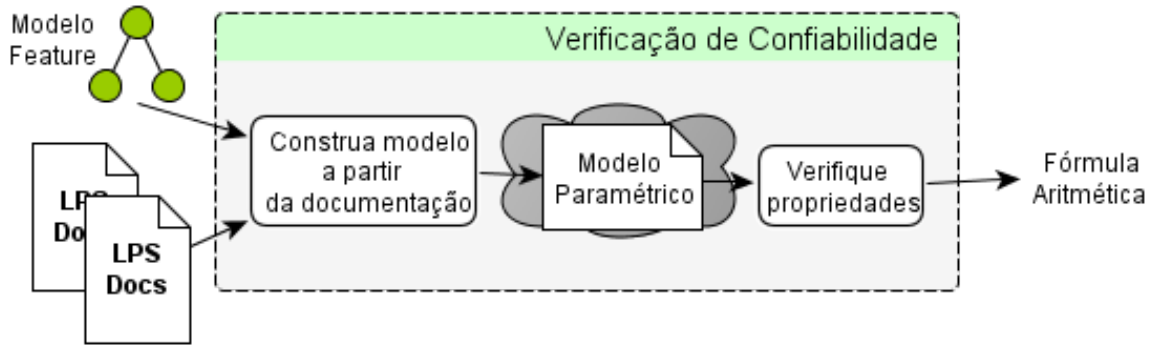


Figura 4.13: Processo de verificação de propriedades para a LPS [Nunes et al., 2012]

Toda *feature* variável dentro da LPS possui um parâmetro correspondente no modelo (e consequetemente na fórmula) para representar a presença/ausência da *feature*, valorado com 1 ou 0 (x tal que $0 \leq x \leq 1, x \in \mathbb{I}$). Todo componente cuja confiabilidade estimada não é conhecida inicialmente ou é variável em tempo de execução possui um parâmetro correspondente no modelo para ser utilizado como valor de confiabilidade (qualquer valor x tal que $0 < x \leq 1, x \in \mathbb{R}$). O valor de confiabilidade expressa a probabilidade de execução correta do componente mapeado pela feature. Apesar da fórmula final poder avaliar qualquer combinação desses dois parâmetros para todas as features, o processo de construção da fórmula impõe algumas restrições semânticas à valoração das features uma vez que a fórmula não provê uma forma de determinar se uma valoração de seus parâmetros é valida ou não. No caso de duas *features* alternativas, por exemplo, a seleção de uma feature implica na não seleção da outra. O cálculo da fórmula de qualidade em uma configuração não válida da LPS (as duas *features* selecionadas simultaneamente) gera um valor cuja interpretação é indefinida.

Conforme observado na Seção 3.2, a busca por configurações adequadas para um estado de risco do indivíduo monitorado deve avaliar parâmetros de qualidade de uma configuração considerando a existência de funcionalidades obrigatórias ou opcionais. O cálculo de confiabilidade deve oferecer suporte a esses tipos de *features*.

A estratégia de cálculo de confiabilidade adotada não impõe restrição sobre o tipo de *feature* existente na LPS, já que é capaz de inserir no modelo probabilístico qualquer tipo de variabilidade [Nunes et al., 2012]. Por meio de um mecanismo de tratamento de variabilidade capaz de representar *features* opcionais a técnica utilizada é capaz de representar qualquer outro tipo de variabilidade, uma vez que os demais tipos (Obrigatória, alternativa ou feature-ou) podem ser refinadas para *features* opcionais acrescidas de restrições *cross-tree* [Gheyi et al., 2008]. O tratamento adequado de *features* opcionais favorece a adoção dessa estratégia em qualquer LPS real.

O valor de confiabilidade obtido por meio dessa técnica é combinado com os outros parâmetros apresentados na Seção 4.5.1 a fim de definir um valor de qualidade único para uma configuração da LPS.

O diagrama de classes da Figura 4.14 representa a classe que modela a fórmula de confiabilidade do sistema. Note que a classe que modela a fórmula de confiabilidade de uma configuração possui dois parâmetros para cada *feature* presente no sistema, cujo modelo de *features* está representado na Figura 4.2. Todos os campos classe com o prefixo

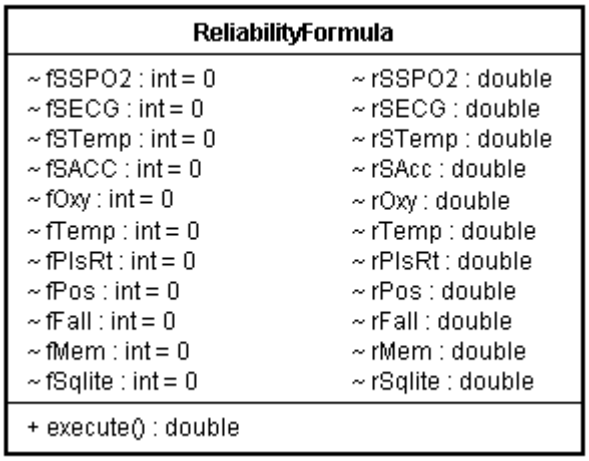


Figura 4.14: Diagrama de classes detalha modelagem da fórmula de confiabilidade avaliada pelo sistema

`f` são utilizados para determinar se a *feature* está presente ou não na fórmula. Já os atributos da classe com o prefixo `r` são utilizados para determinar o valor de confiabilidade da *feature* relacionada ao campo.

4.5.2 Estratégias de tomada de decisão

Diferentes abordagens têm sido propostas para a auxiliar o processo de diagnóstico, reduzir os conflitos e estimular o paciente a ser mais ativo no processo de decisão sem aumentar sua ansiedade sobre o tratamento. Mesmo assim, os sistemas de tomada de decisão ainda não são totalmente aceitos no ambiente médico Dolan [2010], D et al. [2011].

Uma dificuldade na análise de qualidade de uma determinada configuração de um software diz respeito à avaliação de parâmetros conflitantes [Thokala and Duenas, 2012]. Embora diversos trabalhos já tenham definido estratégias para a escolha de uma configuração que maximize apenas um parâmetro de qualidade, a definição de estratégias que trabalhem com mais de um parâmetro de qualidade ainda é um trabalho em aberto.

A arquitetura de *software* proposta na Seção 4.4 avalia se a configuração da LPS utilizada é adequada para um estado de risco do paciente via análise do valor de qualidade provido pela configuração. Caso o valor de qualidade provido pela configuração satisfaça o objetivo de qualidade do estado de risco, esta configuração é inserida na lista de configurações adequadas ao objetivo de qualidade. Por outro lado, caso a configuração não satisfaça o objetivo de qualidade do estado de risco, essa é descartada da lista de adequadas ao objetivo de qualidade. Essa análise é repetida para todas as configurações possíveis até que se ache pelo menos uma que satisfaça o objetivo de qualidade.

Note que o valor de qualidade não somente depende das *features* presentes na configuração, mas também do contexto no qual o valor de qualidade é avaliado. O valor de estimativa do tempo de vida do sistema, por exemplo, depende do valor de bateria dos sensores envolvidos. Por esse motivo, a cada mudança de estado de risco, é necessário avaliar novamente as configurações da LPS. A avaliação manual destas configurações é uma tarefa inviável dado o número de configurações possíveis em uma LPS, o que requer

que a solução utilizada em aplicada no contexto de sistemas dinâmicos automatizem o processo de análise de qualidade [Martens et al., 2010].

Este trabalho adota duas estratégias de cálculo de qualidade: uma utilizando o processo de avaliação de multi-critérios baseado em uma fórmula única para o valor de qualidade (Simple MultiAttribute Rating Technique - SMART) [Dolan, 2010, Thokala and Duenas, 2012] e outra baseada na satisfação de múltiplos objetivos de qualidade [Reuben and Tinetti, 2012, Paluska et al., 2006], a estratégia orientada a objetivos - GOAL.

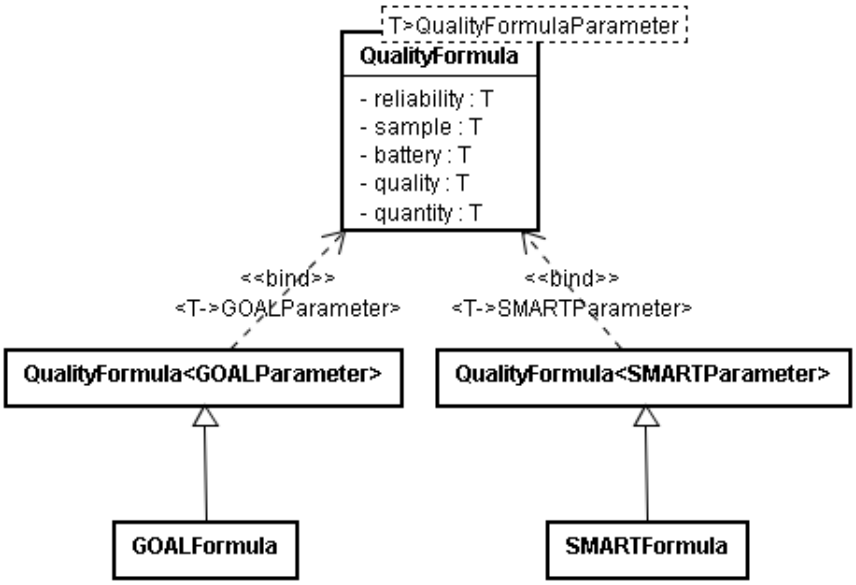


Figura 4.15: Diagrama de classes para definição de estratégias de cálculo tratadas por essa metodologia

A Figura 4.15 detalha o relacionamento entre as classes que modelam essas estratégias. A classe `QualityFormula` define uma *super* classe para qualquer estratégia de qualidade. Nela são definidos cinco campos que modelam os cinco os cinco atributos de qualidade mencionados no início desta seção. Esses campos são implementações da classe `QualityFormulaParameter`, e esta, por sua vez, parametriza a classe `QualityFormula`. A implementação da estratégia de cálculo SMART é feita pela classe `SMARTFormula` enquanto que a implementação da estratégia de cálculo orientada a objetivos é feita pela classe `GOALFormula`.

Cabe ressaltar que embora o Capítulo 6 faça uma breve discussão sobre o comportamento dessas duas estratégias no domínio de monitoração de sinais vitais do corpo humano, não faz parte do escopo deste trabalho propor uma estratégia de cálculo, mas sim comparar e sugerir utilização de técnicas existentes neste domínio. A escolha por essas duas estratégias foi feita por meio do levantamento dos requisitos do especialista do domínio e sua experiência sobre como compor os atributos de qualidade. Esses requisitos estão relacionados à priorização de atributos de qualidade de acordo com os riscos de saúde dos pacientes (associado à estratégia SMART) e o filtro recursivo dos atributos de qualidade (associado à estratégia GOAL). Acreditamos que a utilização de outras estratégias, tais como as de predição do valor de qualidade [Kolesnikov et al., 2013, Siegmund et al.,

2011] não tenha impacto sobre o funcionamento da arquitetura proposta. Porém, para confirmar essa hipótese é necessário uma avaliação mais detalhada da nossa proposta.

Estratégia SMART

A avaliação SMART é um método simplificado para atribuir um valor de qualidade a uma solução que contém múltiplas variáveis [Edwards, 1977]. A ideia básica de funcionamento é que todo sistema tem dimensões diferentes de qualidade e que essas dimensões podem ser combinadas por meio de uma média ponderada linear [Olson, 1996]. O valor gerado por essa média representa o valor de qualidade do sistema. Embora a técnica aplique uma função matemática simplificada para avaliar atributos de qualidade, alguns experimentos sugerem que os valores de qualidade obtidos pela estratégia SMART atingem valores próximos ao valor medido por funções mais complexas não-linear[Edwards and Barron, 1994].

A estratégia SMART (Representada pela classe `SMARTFormula` na Figura 4.15) aplicada ao RSCH associa a cada atributo de qualidade do sistema um peso. Este peso é utilizado na fórmula da média ponderada que definirá o valor de qualidade da configuração que está sendo avaliada. Não existem restrições na estratégia SMART sobre os valores atribuídos aos pesos, mas a fim de definir um parâmetro de comparação entre as estratégias de cálculo, nesta solução os valores estão em uma faixa de 0 até 10.

Conforme apresentado na Seção 4.5, cada atributo de qualidade possui uma fórmula que calcula a qualidade da configuração para esse atributo específico. O parâmetro de qualidade de informação, por exemplo, é calculado como apresentado na Fórmula 4.4. Já o parâmetro de tempo de vida estimado do sistema é calculado como apresentado na Fórmula 4.10.

A estratégia SMART associa para cada atributo de qualidade, uma função peso e uma funçao de cálculo de qualidade, como detalha Definição 16.

Definição 16 Sejam *attribute* um atributo de qualidade avaliado pela estratégia SMART e *attrs* o conjunto destes atributos. Então $attribute_i$ está associado uma função de qualidade conforme apresentado na Fórmula 4.13 e a uma peso conforme apresentado na Fórmula 4.14.

$$QoSAttribute_i : MF \rightarrow \mathbf{R} \tag{4.13}$$

$$Weight_i : \mathbf{R} \tag{4.14}$$

O valor de qualdiade de uma configuração de uma LPS calculado a partir da estratégia SMART é obtido conforme Definição 17.

Definição 17 Seja *lps* um conjunto de configurações válidas de uma LPS, c uma configuração em *lps*. O valor de qualidade de c utilizando a estratégia SMART ($Smart(c)$) é a média ponderada dos pesos e funções de qualidade dos atributos de qualidade analisados, conforme Fórmula 4.15.

$$Smart(c) = \frac{\sum_{i=1}^{n} QosAttribute_i(c) * weight_i}{\sum_{i=1}^{n} weight_i}, \text{ onde } n = |attris| \quad (4.15)$$

O diagrama de classes da Figura 4.16 detalha a modelagem adotada por esse trabalho para definir uma fórmula SMART e como essa fórmula se relaciona com os seus atributos de qualidade. Observe que além dos atributos de qualidade, a fórmula também contém um valor de objetivo associado a ela. A classe *ContextGoal* modela o valor de objetivo para cada estado do sistema (Risco normal, moderado, alto). Esses objetivos de qualidade para cada estado de são avaliados dinamicamente pelo Gerenciador de Adaptação apresentado na Seção 4.4.1. Os atributos de qualidade da *SMARTFormula* são herdados da classe *QualityFormula*.

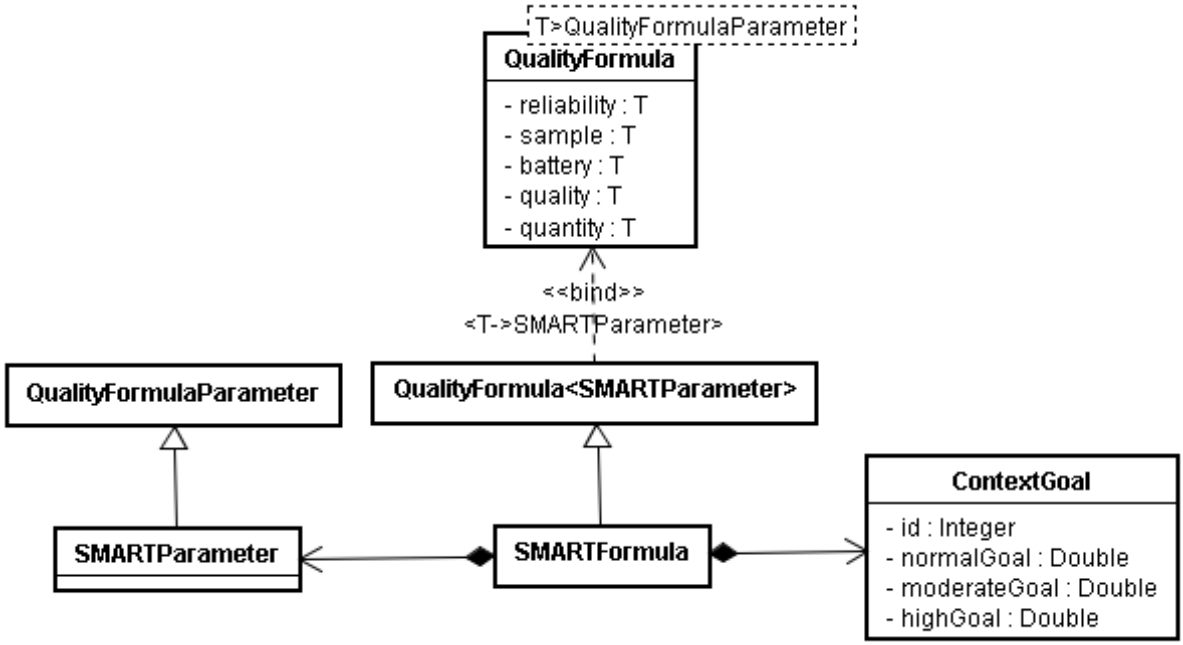


Figura 4.16: Diagrama de classes de uma Fórmula SMART

Estratégia Orientada a Objetivos - GOAL

De acordo com a estratégia cálculo de qualidade orientada a objetivos (GOAL), não existe uma relação clara entre os múltiplos parâmetros de qualidade avaliados de forma que seja possível estabelecer uma fórmula única capaz de uní-los. Desse ponto de vista, é mais interessante avaliar cada um deles individualmente e verificar a qualidade da configuração para cada dimensão de qualidade avaliada [Paluska et al., 2006]. A estratégia GOAL já vem sendo amplamente utilizada no domínio da saúde [Hoeper et al., 2005, Reuben and Tinetti, 2012, Winer and Roth, 2006]

A estratégia (Representada pela classe `GoalFormula` na Figura 4.15) estabelece um algoritmo que realiza um filtro sucessivo da configuração para os parâmetros de qualidade.

O algoritmo consiste em verificar, para cada atributo de qualidade, se uma configuração satisfaz o objetivo de qualidade determinado pelo parâmetro em um estado de risco do sistema, conforme algoritmo apresentado no Algoritmo 1. Opcionalmente, o parâmetro de qualidade pode definir uma porcentagem de satisfação de qualidade. Em termos práticos, o especialista do domínio pode exigir que apenas 50% do objetivo de qualidade seja satisfeito pela configuração. Essa flexibilização permite que caso não exista nenhuma configuração que satisfaça 100% dos objetivos de qualidade para o estado de risco, configurações alternativas que satisfaçam parcialmente o objetivo de qualidaade possam ser utilizadas.

```
Input: Configuração c cuja qualidade será avaliada
Input: Definição da fórmula GOAL f
Input: Estado s de risco do sistema
Input: Parâmetro satisf que estabelece se o cálculo deve considerar a
       porcentagem de satisfação dos objetivos de qualidade
Output: Verdadeiro caso c satisfaça os objetivos de qualidade de f, falso caso
        contrário
while todos os parâmetros de f não tiverem sido avaliados do
    p = parâmetro de f ;
    v = valor de qualidade de p para a configuração c ;
    if satisf then
        if não v satisfaz o objetivo de qualidade de parâmetro para o estado s
        considerando porcentagem de satisfação then
            return falso;
        end
    end
    if não v satisfaz o objetivo de qualidade de parâmetro para o estado s then
        return falso;
    end
end
return verdadeiro;
```

Algoritmo 1: Algoritmo da estratégia orientada a objetivos seleciona uma configuração se ele satisfaz os objetivos de qualidade estabelecidos individualmente por cada parâmetro de qualidade

O digrama de classes apresentado pela Figura 4.17 detalha a modelagem para essa estratégia nesta proposta. Cada atributo de qualidade está associado a um objeto da classe *ContextGoal*. Por meio deste objeto, são estabelecidos os objetivos de qualidade para os estados de alto, médio e baixo risco. Além disso, cada atributo de qualidade também está associado a um objeto da classe *PriorityPercentage*, que define a prioridade e o valor de satisfação requerida deste atributo.

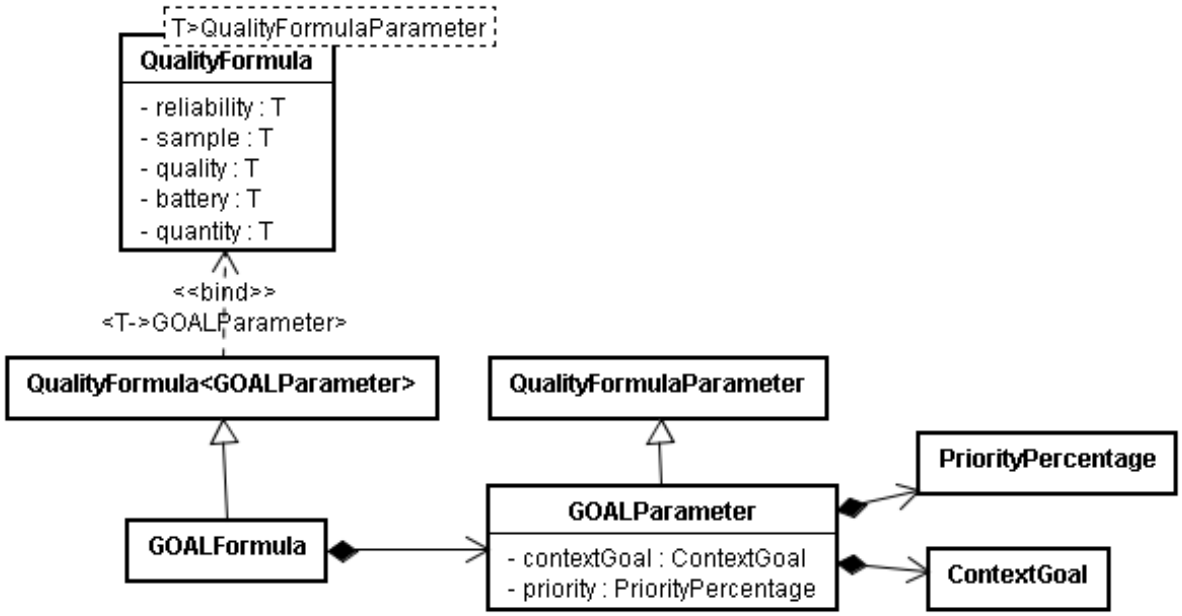


Figura 4.17: Diagrama de classes para fórmulas definidas pela estratégia GOAL

Diferentemente da estratégia SMART, a estratégia orientada a objetivos (GOAL) define um objetivo de qualidade para cada parâmetro de qualidade. Não existe uma influência direta de um parâmetro de qualidade no objetivo de qualidade do outro. A configuração avaliada não combina os valores de qualidade para cada parâmetro, isto é, não existe um único valor de qualidade para a configuração. A qualidade de uma configuração avaliada por essa estratégia é uma quíntupla de valores, conforme Fórmula 4.16. Uma configuração válida será utilizada em um estado de risco caso os valores de qualidade dos atributos de qualidade satisfaçam os objetivos de qualidade de cada atributo de qualidade para o estado de risco.

$$Goal(c) = (qualidade, quantidade, tempodevida, amostragem, confiabilidade) \quad (4.16)$$

Capítulo 5

Implementação

Neste capítulo nós descrevemos as principais ferramentas construídas para auxiliar a realização das simulações de monitoração com o intuito de avaliar a modelagem proposta.

A avaliação das arquitetura e metodologia propostas por esse trabalho foi feita via simulação. O processo de simulação consiste em enviar dados reais de monitoração para um sistema que realiza um tratamento nesses dados a fim de identificar situações de alerta. O sistema de monitoração de sinais vitais (MDV), descrito na Seção 5.1, desempenha essa função. Os dados reais foram obtidos por meio de uma consulta à base do *Beth Israel Hospital* (MIT-BIH) [1]. O envio desses dados é feito por outra ferramenta desenvolvida nesse estudo, o simulador de dados vitais (SDV), descrito na Seção 5.2.

O sistema MDV adota a arquitetura proposta no Capítulo 4. Para o correto funcionamento dessa arquitetura proposta, é preciso estabelecer os parâmetros que serão utilizados nas fórmulas de qualidade. A Seção 5.3 descreve uma ferramenta de apoio para construção das fórmulas de qualidade do MDV. Por meio dessa ferramenta, é possível simular alguns cenários de monitoração e verificar o processo de escolha das configurações da LPS.

5.1 Sistema de Monitoração de Dados Vitais (MDV)

O Sistema de Monitoração de Dados Vitais (MDV) foi construído com o objetivo de avaliar a arquitetura proposta no Capítulo 4. Na estrutura básica de uma RSCH, o MDV é o componente responsável por estabelecer conexão com os sensores, capturar e enviar dados, identificar eventos, notificar situações de alerta e gerenciar os recursos de forma a oferecer os serviços demandados pelo estado do indivíduo monitorado.

A Figura 5.1a apresenta a tela principal do MDV. Para monitorar seus dados vitais, o usuário deve explicitamente estabelecer uma conexão com os sensores de dados via *menu* `Configuração` ***`Bluetooth`*** apresentado da Figura 5.1a. Após o estabelecimento da conexão com sucesso, o usuário solicita a inicialização da monitoração. Ao receber esses comandos, os sensores enviam os dados vitais capturados para o MDV. O usuário, durante a monitoração, pode fazer um acompanhamento sobre seu estado de saúde verificando as telas de sobre os eventos dos sensores da monitoração. É possível também acompanhar as configurações selecionadas pela LPS em tempo de execução, conforme apresenta Figura 5.1c.

[1] `http://ecg.mit.edu/`

(a) Tela principal do MDV. (b) Configuração conexão *Bluetooth* (c) Visualização monitoração

Figura 5.1: Telas capturadas do sistema MDV

A Figura 5.2 apresenta os principais casos de uso do sistema. Via aplicação, o usuário monitorado é capaz de configurar a comunicação *bluetooth* entre os sensores; cadastrar o indivíduo a ser monitorado, familiares e profissionais da saúde que serão notificados em caso de emergência; definir estratégias de qualidade e heurísticas que poderão ser aplicadas à configuração no momento da mudança no estado de risco; iniciar e parar a monitoração dos sinais vitais e observar os estados de risco para cada informação individualmente e, por fim, consultar dados medidos previamente e armazenados pelo sistema.

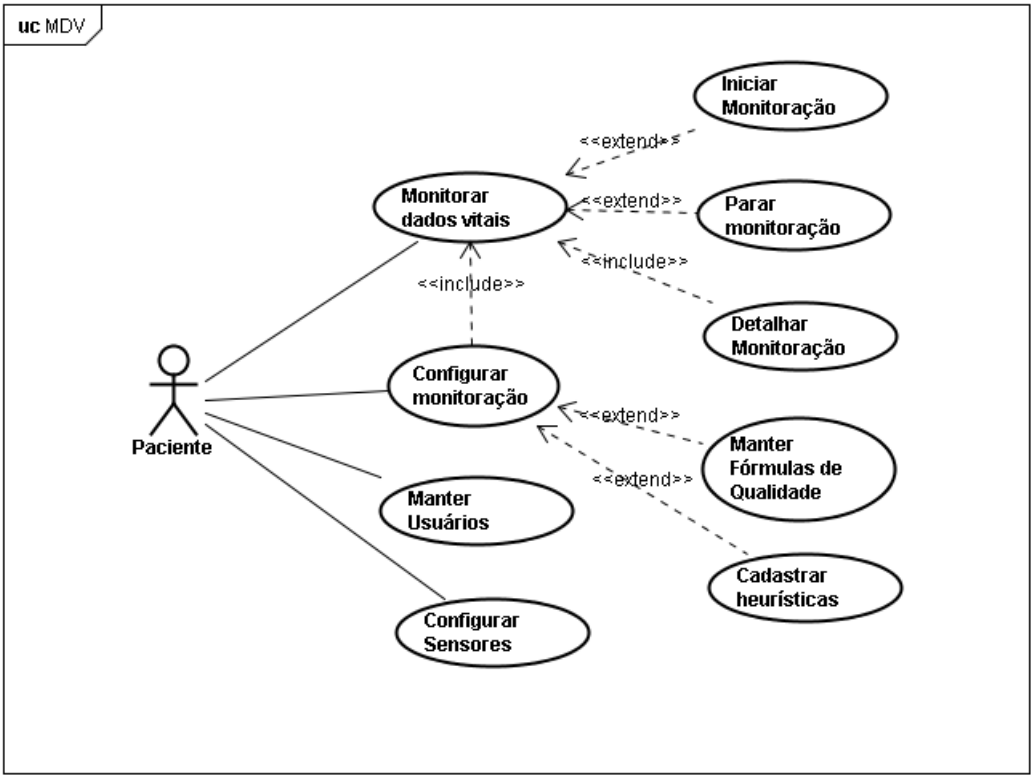


Figura 5.2: Casos de uso do MDV

A estrutura básica da arquitetura contém os seguintes pacotes: `cm`, `adaptation`, `configurator`, `persistence`, `activity`, `control` e `commom`. Os pacotes `cm` , `adaptation` e `configurator` representam, respectivamente, os seguintes refinamentos da arquitetura proposta no Capítulo 4: Gerenciador de Contexto, Gerenciador de Adaptação e Configurador do Sistema.

O pacote de `persistence` contém as classes responsáveis pela manipulação dos registros em banco de dados, tais como registro de eventos, usuários e configurações da estratégia para cálculo de qualidade. O pacote `activity` contém as classes que gerenciam as telas do sistema, bem como os serviços de notificação ao usuário. O pacote `control` contém interfaces para os serviços acessados via interface gráfica. Por fim, o pacote `common` contém as classes de modelo, *builder*[2] e funções utilitárias que são utilizadas no sistema.

No momento da inicialização da monitoração, o MVD supõe que o estado de saúde do indivíduo é de alto risco. Esse comportamento foi definido pelo especialista do domínio com a justificativa de que é mais adequado prover mais serviço do que o necessário do que ter um custo inicial para se adequar a uma situação de emergência, que exige baixo tempo de resposta.

O sistema Monitoração de Dados Vitais (MDV) foi construído para executar na plataforma Android[3] versão 2.1 ou superior. O sistema é composto por 144 classes Java e 27 telas de interação com o usuário.

O MDV segue a arquitetura refinada no Capítulo 4. O restante dessa seção descreve como os componentes Gerenciador de Contexto, Gerenciador de Adaptação e Configurador implementam os requisitos impostos pelo sistema. Devido a importância da identificação correta dos eventos, necessidade de composição entre eles e a influência dos estados na mudança de configuração, a Seção 5.1.1 detalha como as regras da gramática foram implementadas para representar corretamente essas abstrações. A Seção 5.1.2 descreve como o Gerenciador de Adaptação implementa o processo de busca dinâmica por configurações da LPS. Por fim, a Seção 5.1.3 aponta as restrições de implementação relacionadas ao Configurador do sistema e qual a semântica relacionada à ativação e desativação de uma *feature*.

5.1.1 Gerenciador de Contexto

O gerenciador de contexto é responsável por fazer a leitura das regras definidas pela gramática descrita na Seção 4.1. Cada regra da gramática para definir eventos é interpretada como um objeto da classe `GrammarRule`.

Cada regra da gramática possui um campo que identifica o evento que será gerado por essa regra. A regra da Linha 2 do Quadro 5.1 define um evento de nome *OXYGENATION_NORMAL* (Oxigenação normal), por exemplo. De acordo com a especificação dessa linha, um evento *OXYGENATION_NORMAL* é gerado caso o valor de oxigenação seja maior ou igual ao valor 94,0. Observe que a ordem com o qual as regras são definidas é relevante para identificação correta do evento. A regra da linha 1 impõe algumas restrições sobre os valores que serão analisados pela regra da linha 2, de forma que esta só deva ser aplicada caso o valor de oxigenação for menor ou igual a 100. De forma semelhante,

[2]Padrão de Projeto *Builder*

[3]Sistema operacional baseado em Linux desenvolvido para plataformas móveis tais como *smartphones* e *tablets*. Mais detalhes disponíveis em `http://www.android.com/`

a regra da linha 3 só é aplicada se as regras da linha 1 e 2 não forem satisfeitas e assim sucessivamente.

Quadro 5.1: Trecho de código define regras para interpretar dados de oxigenação.

```
1 new GrammarRule(GREATER      , OXYGENATION, (Float) 100.0f, SP02_SENSOR_UNAVAILABLE));
2 new GrammarRule(GREATER_THAN, OXYGENATION, (Float) 94f, OXYGENATION_NORMAL));
3 new GrammarRule(GREATER_THAN, OXYGENATION, (Float) 90f, OXYGENATION_MODERATE));
4 new GrammarRule(GREATER_THAN, OXYGENATION, (Float) 0.0f, OXYGENATION_LOW));
```

As regras da gramática também podem gerar eventos a partir da combinação de outros eventos conforme Quadro 5.2. De acordo com a definição do evento das linhas 1-2 da Quadro 5.2, é gerado um evento *FAINT* (Desmaio) quando os eventos de hipotermia e pulsação cardíaca altos são identificados.

Como os eventos são identificados de forma independente entre si, o sistema deve tratar alguns aspectos de sincronização entre os eventos. Os eventos que são especificados pela regra de composição de eventos devem acontecer em um intervalo de tempo máximo de 1 minuto[4]. Ou seja, quando um evento é identificado e existe uma regra de composição de eventos para este evento, o sistema verifica se os outros eventos aconteceram. Se sim, ele verifica se a diferença entre o instante de identificação do primeiro evento é de 1 minuto. Se for, o sistema gera um novo evento identificado pela regra. Se não, ele armazena o evento mais recente e descarta aqueles eventos que aconteceram em um período maior do que 1 minuto.

Quadro 5.2: Trecho de código define regras para composição de eventos.

```
1 new GrammarRule(AND,
2     new ComposedObject(new Integer[]{HYPOTHERMIA_MODERATE,PULSE_HIGH}), FAINT));
3 new GrammarRule(AND,
4     new ComposedObject(new Integer[]{HYPOTHERMIA_LOW,PULSE_LOW}, FAINT)));
```

Por fim, a gramática proposta permite a especificação de eventos de transição entre os estados. O Quadro 5.3 detalha a interpretação de algumas regras para transição dos estados do sistema. A regra definida nas linhas 1-2, por exemplo, especifica que se for identificado um evento *FAINT* (Desmaio) e o sistema estiver em um estado de risco moderado, o sistema deve transitar para o estado de risco alto. A regra das linhas 5-6 especifica que se for identificado um evento de *OXYGENATION_LOW* (oxigenação baixa), independente do estado atual, o sistema deve transitar para o estado de alto risco.

Quadro 5.3: Trecho de código define regras que podem gerar transições nos estados.

```
1 new GrammarRule(AND,
2   new ComposedObject(new Integer[] {FAINT,MODERATE_RISK}),State.H));
3 new GrammarRule(AND,
4   new ComposedObject(new Integer[] {PULSE_LOW,HIGH_RISK}), State.H));
5 new GrammarRule(AND,
6   new ComposedObject(new Integer[] {OXYGENATION_LOW,ANY_RISK}), State.H));
7 new GrammarRule(AND,
8   new ComposedObject(new Integer[] {TEMPERATURE_HIGH,NORMAL_RISK}), State.M));
9 new GrammarRule(AND,
10  new ComposedObject(new Integer[] {TEMPERATURE_NORMAL,MODERATE_RISK}), State.N));
```

[4]Este intervalo foi definido a partir da identificação de que o período máximo dos sensores presentes na RSCH avaliada é de 20 segundos. Com o período de 1 minuto, é possível encadear 3 eventos de forma que eles sejam considerados simultâneos para a interpretação das regras

5.1.2 Gerenciador de Adaptação

O processo de busca pela configuração é feito conforme Seção 4.4.1.

No momento da identificação de necessidade de mudança, o Gerenciador de Adaptação quantifica o valor de qualidade de toda configuração válida de uma LPS para um determinado contexto. Toda configuração que satisfizer o objetivo de qualidade é inserida na lista de configurações válidas para o estado. Esta lista, entretanto, pode conter mais de um elemento. Neste caso, o Gerenciador de Adaptação prioriza a escolha de alguma delas baseando-se nos critérios determinados pelo especialista no domínio, conforme definido na listagem a seguir.

Estado de Alto Risco Prioridade dos atributos de qualidade : (1) Confiabilidade. (2) Qualidade de Informação. (3) Quantidade de Informação. (4) Taxa de Amostragem dos sensores. (5) Tempo de vida estimado dos sensores.

Estado de Médio Risco Prioridade dos atributos de qualidade : (1) Quantidade de Informação. (2) Qualidade de Informação. (3) Taxa de Amostragem. (4) Confiabilidade. (5) Tempo de vida estimado dos sensores.

Estado de Baixo Risco Prioridade dos atributos de qualidade : (1) Tempo de vida estimado dos sensores. (2) Quantidade de Informação (3) Taxa de Amostragem. (4) Confiabilidade. (5) Qualidade de Informação.

Caso uma ou mais configurações possuam o mesmo valor de qualidade para o sistema, a configuração é escolhida de forma aleatória. Foi definida essa estratégia pois acredita-se que essas fórmulas gerarão resultado semelhantes na execução do sistema.

Gerenciador de Configurações da LPS

A busca por configurações de uma LPS é feita através da transformação do modelo de *features* em fórmulas proposicionais. A seleção de um conjunto de *features* de um modelo de *features* é considerada válida caso o resultado da fórmula proposicional correspondente ao MF seja satisfazível para uma valoração de *features* onde uma *feature* falsa indica que ela não está presente na configuração, e uma feature verdadeira, sim [Benavides et al., 2010].

O modelo de *features* da RSCH deve ser é traduzido para um conjunto de proposições lógicas, usando as estratégias da Figura 2.3. As fórmulas geradas por cada restrição do modelo de *features* são codificadas na solução MDV.

A modelagem das fórmulas é possível via utilização de uma biblioteca Java para resolução de Problemas de Satisfação de Restrições (Constraint Satisfaction Problems - CSP), Choco [5]. Esta API está baseada no mecanismo de propagação de eventos com estruturas de rastreamento [Laburthe and Jussien, 2012]. Para a fórmula proposicional avaliada é gerado um objeto `model` da biblioteca. As restrições do modelo de *features* são conectadas ao modelo. A resolução é possível via instanciação de um objeto `solver` da biblioteca.

De um modo geral, os *Sat-Solvers* esperam que as entradas da fórmula proposicional a ser resolvida estejam na Forma Conjuntivo Normal (conjunctive normal form - CNF).

[5] Disponível em `http://choco.sourceforge.net`

Porém, a biblioteca Choco permite que a fórmula seja montada com os diversos tipos de restrições tais como *e*, *ou*, se e *se somente se*. O processo manual de reescrita foi realizado apenas com o objetivo de exercício sobre as transformações realizadas.

O Quadro 5.4 apresenta uma ideia geral dos comandos necessários para a utilização da biblioteca Choco. Inicialmente, constrói-se o objeto choco que representa o modelo a ser satisfeito (l.1-3). Para cada *feature*, é definida uma variável cujo valor pode ser 0 (falso) ou 1 (verdadeiro) (l.4-9). Após isso, são criadas restrições entre as *features* (l.10-15). Essas restrições são adicionadas ao modelo. (l.16-20). Por fim, é criado um objeto *solver* que faz a leitura das restrições no modelo e resolve a fórmula proposicional gerada (l. 21-26).

Quadro 5.4: Trechos código exemplifica uso da biblioteca Choco para busca de configurações válidas da LPS.

```
1  // Define o modelo a ser satisfeito
2  Model m = new CPModel();
3
4  //Define variáveis presentes no modelo. Uma para cada feature.
5  this.varInfoTemperature = Choco.makeIntVar("iT", 0, 1 );
6  this.varInfoOxygenation = Choco.makeIntVar("iO", 0, 1 );
7  this.varSensorTemp = Choco.makeIntVar("sT", 0, 1 );
8  this.varSensorSPO2 = Choco.makeIntVar("sS", 0, 1 );
9  //...
10
11 //Define algumas restrições entre as features
12 Constraint cSensor = Choco.or(varSensorTemp, varSensorECG, varSensorSPO2, varSensorAcc);
13 Constraint temp_IMPLIES_stemp =
14   Choco.ifOnlyIf(Choco.eq(varInfoTemperature, 1), Choco.eq(varSensorTemp, 1));
15 //...
16
17 //Adiciona restrições ao modelo
18 m.addConstraint(cSensor);
19 m.addConstraint(temp_IMPLIES_stemp);
20 //...
21
22 CPSolver s = new CPSolver();
23 s.read(m);
24 if(s.solve()){
25   //busca valoração de features que satisfaz expressão lógica.
26 }
```

A busca por configurações válidas foi utilizada no sistema de monitoração do indivíduo e no projeto de avaliação de parâmetros de qualidade (Seção 5.3).

5.1.3 Configurador

Após a escolha pela configuração, o Gerenciador de Adaptação repassa a nova configuração para o Configurador. Uma configuração representa uma lista de *features* do sistema. A lista de todas as *features* que estão em execução antes do repasse da configuração é mantida pelo Configurador. No momento que ele recebe a nova configuração, todas as *features* que estão na nova configuração e não estão em execução são ativadas. Já as *features* que estão em execução mas não estão presentes na nova configuração são desativadas.

O *Configuration Knowledge* (CK) associado à essa LPS trata os artefatos associados às features com o alto grau de granularidade. Todas as features que estão associadas a um artefato de código, ou seja, features concretas, estão mapeadas para *threads* Java. O CK do sistema é descrito na Tabela 5.1.

Feature	Classe Java
Sensor SPO2	Spo2SensorFeatureImpl.java
Sensor ECG	EcgSensorFeatureImpl.java
Sensor ACC	AccSensorFeatureImpl.java
Sensor TEMP	TemperatureSensorFeatureImpl.java
Posição	PositionFeatureImpl.java
Queda	FallFeatureImpl.java.java
Temperatura	TemperatureFeatureImpl.java
Pulsação	PulseRateFeatureImpl.java
Oxigenação	OxygenationFeatureImpl.java
Memória	MemoryFeatureImpl.java
Banco de Dados	DatabaseFeatureImpl.java
Arquivo	FileFeatureImpl.java

Tabela 5.1: Configuration Knowledge da RSCH

Ativar uma *feature* significa iniciar a execução da *thread* java mapeada. Desativar, por sua vez, significa interromper a execução desta *thread*. O Quadro 5.5 apresenta um trecho de código utilizado para ativar as features de oxigenação, temperatura, posição e queda. Observe que o método *startOxygenationFeature*, chamado caso a *feature* a ser configuração se chame *INFO_OXYGENATION_FEATURE*, inicializa a *thread oxygenationThread*, responsável por tratar eventos de oxigenação e mapeada pela *feature* Oxigenação.

Quadro 5.5: Trechos código ativação das *features*.

```
1  //..
2  if(feature.getName().equals(FeatureNameConstant.INFO_OXYGENATION_FEATURE)){
3      startOxygenationFeature();
4  }else if(feature.getName().equals(FeatureNameConstant.INFO_TEMPERATURE_FEATURE)){
5      startTemperatureFeature();
6  }else if(feature.getName().equals(FeatureNameConstant.INFO_POSITION_FEATURE)){
7      startPositionFeature();
8  }else if(feature.getName().equals(FeatureNameConstant.INFO_FALL_FEATURE)){
9      startFallFeature();
10 }
11 //..
12 //Método de início da feature de oxigenação.
13 private void startOxygenationFeature() {
14     if(oxygenationThread!=null && !oxygenationThread.isAlive()){
15         if(oxygenationThread.isInterrupted()){
16             oxygenationThread.onDestroy();
17         }
18         this.oxygenationThread = null;
19     }
20     if(oxygenationThread==null){
21         this.oxygenationThread = new OxygenationFeatureImpl(grammarManager, contextManager);
22         this.oxygenationThread.start();
23     }
24 }
```

5.2 Simulador de Dados Vitais - SDV

A ferramenta de *S*imulação dos *D*ados *V*itais - SDV foi construída para auxiliar o processo de simulação dos dados vitais. Por meio dessa ferramenta, é possível simular os sensores físicos e os dados vitais que são obtidos por eles. A Figura 5.3 detalha a interface gráfica da ferramenta. Cada sensor pode enviar dados de bateria, taxa de amostragem e os dados vitais monitorados por ele. A Figura 5.3 apresenta um exemplo onde o sistema possui um sensor acelerômetro (Aba Acc 1), um sensor oxímetro de pulso (Aba Spo2 1), um sensor eletrocardiógrafo (Aba Ecg 1) e um sensor temperatura (Temp 1).

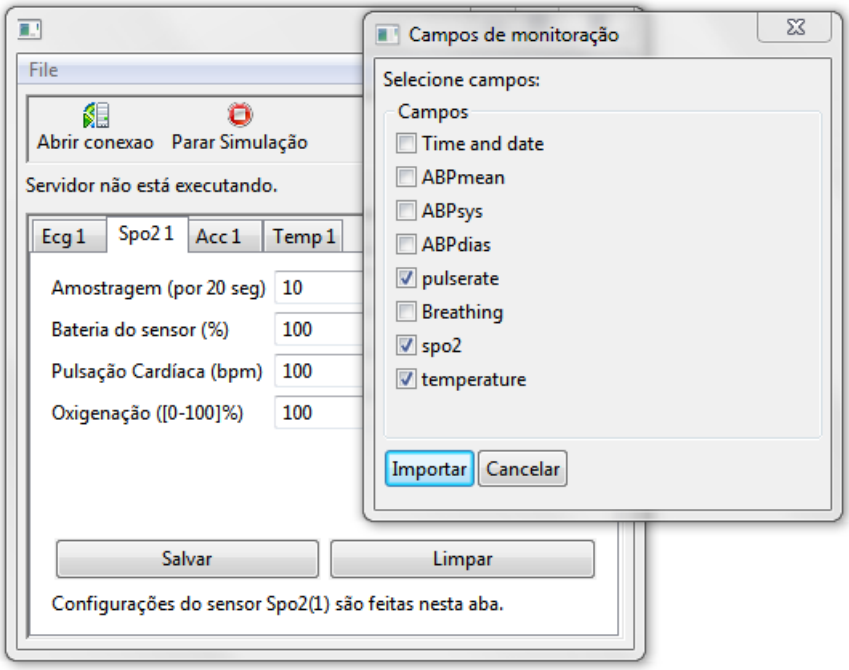


Figura 5.3: Leitura de dados vitais via importação de arquivo no SDV

O SDV permite interação direta do usuário com os dados enviados, mas oferece suporte à leitura de dados via arquivo com extensão *csv*[6] com um padrão definido de construção. Cada coluna desse arquivo deve representar um dado vital, onde a primeira linha identifica o dado vital pelo nome. Atualmente, o sistema oferece suporte aos seguintes dados vitais: pulsação, respiração, oxigenação, temperatura, aceleração gravitacional nos eixos x, y e z.

5.3 Avaliador das Estratégias de Qualidade para Rede de Sensores para o Corpo Humano

A ferramenta de avaliação das estratégias para cálculo de qualidade em Redes de Sensores para o Corpo Humano (QSE-BSN [7]) foi desenvolvida para realizar simulações

[6]CSV - Comma-Separated Values

[7]Disponível em https://code.google.com/p/quality-strategy-evaluation-spl-bsn/, SimulatorQoSSWT project

automáticas sobre as melhores escolhas de configuração no contexto de RSCH usando as estratégias GOAL e SMART.

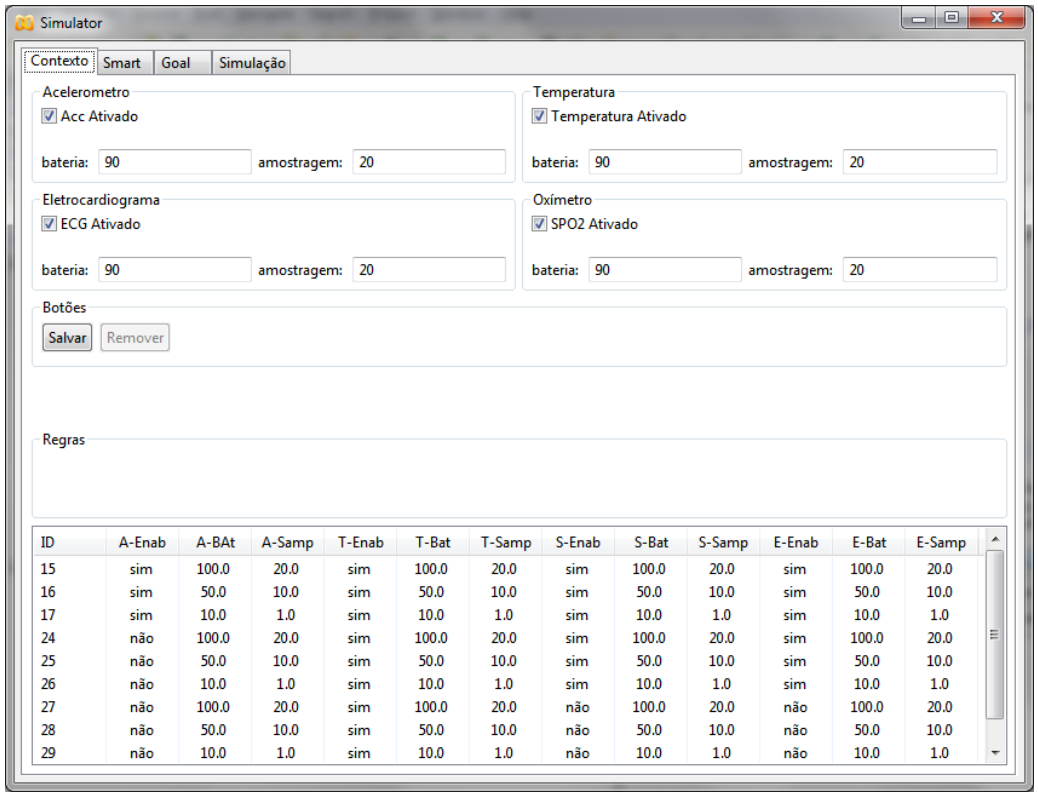
Através da ferramenta, o usuário é capaz de customizar definições de fórmulas que utilizam as duas estratégias detalhadas por esse trabalho, SMART e GOAL. As Figuras 5.4a ,5.4b,5.4c e 5.4d apresentam um exemplo de manipulação da ferramenta. É possível criar cenários de configuração do contexto de execução na aba `Contexto` representada pela Figura 5.4a. Para cada sensor (dispositivo externo), o usuário pode especificar se ele está presente/ausente, qual o valor de bateria e qual o valor de taxa de amostragem.

As abas `Smart` e `Goal`, das Figura 5.4b e Figura 5.4c, apresentam as opções de criação para as fórmulas que utilizam as estratégias SMART e GOAL respectivamente. Em ambas as abas, o usuário pode trabalhar com cinco atributos de qualidade: confiabilidade, qualidade de informação, quantidade de informação, taxa de amostragem e expectativa de tempo de vida. Entretanto, cada estratégia avalia os objetivos de qualidades de forma diferente. Enquanto que a aba SMART exige que o usuário forneça um objetivo de qualidade para os estados de risco baixo, médio e alto, a aba GOAL exige que o usuário forneça um objetivo de qualidade para todos esses três estados de risco em cada atributo de qualidade analisado.

A avaliação das estratégias de qualidade é feita na aba `Simulação` exibida na Figura 5.4d. O usuário deve selecionar o contexto de simulação na lista de opções disponibilizadas na aba e previamente cadastradas na aba `Contexto`. A simulação é iniciada ao se clicar no botão `Iniciar` da aba. A lista de melhores configurações é apresentada na área de texto Configurações, conforme Figura 5.4d. A LPS avaliada na ferramenta é a apresentada na Figura 4.2. É possível modificar a configuração avaliada via alteração do código do gerenciador de configurações de LPS.

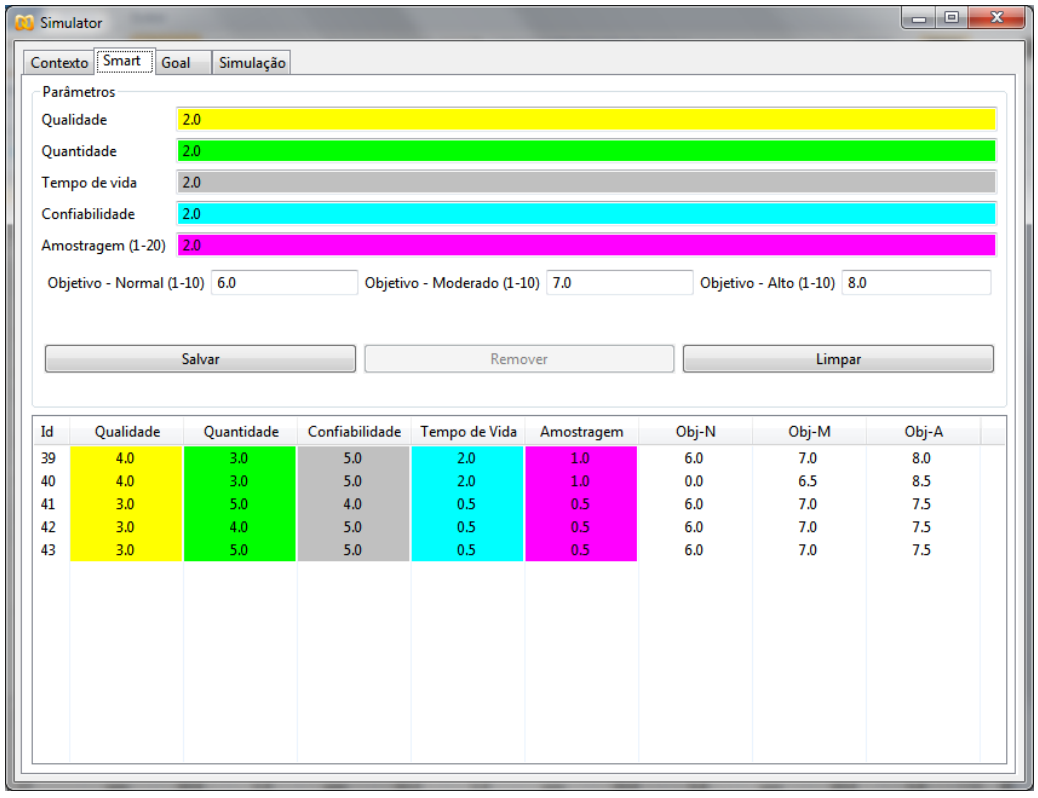
O QSE-BSN foi desenvolvido na linguagem Java para executar na plataforma J2SE. As interfaces gráficas são desenvolvidas por meio de extensão das classes da *API SWT* [8]. Os resultados das simulações são armazenados em um gerenciador local de banco de dados HSQLDB [9]. O sistema é composto por 76 classes.

[8] `http://www.eclipse.org/swt/snippets/`
[9] `http://hsqldb.org/`



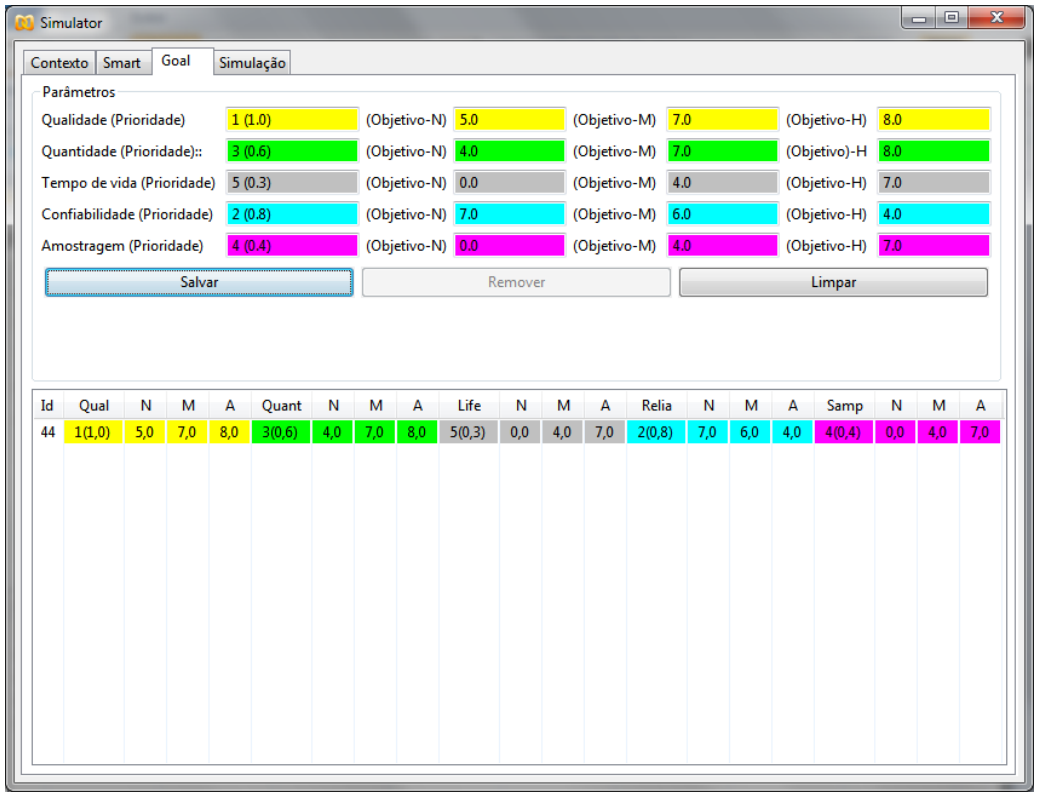
ID	A-Enab	A-BAt	A-Samp	T-Enab	T-Bat	T-Samp	S-Enab	S-Bat	S-Samp	E-Enab	E-Bat	E-Samp
15	sim	100.0	20.0	sim	100.0	20.0	sim	100.0	20.0	sim	100.0	20.0
16	sim	50.0	10.0	sim	50.0	10.0	sim	50.0	10.0	sim	50.0	10.0
17	sim	10.0	1.0	sim	10.0	1.0	sim	10.0	1.0	sim	10.0	1.0
24	não	100.0	20.0	sim	100.0	20.0	sim	100.0	20.0	sim	100.0	20.0
25	não	50.0	10.0	sim	50.0	10.0	sim	50.0	10.0	sim	50.0	10.0
26	não	10.0	1.0	sim	10.0	1.0	sim	10.0	1.0	sim	10.0	1.0
27	não	100.0	20.0	sim	100.0	20.0	não	100.0	20.0	não	100.0	20.0
28	não	50.0	10.0	sim	50.0	10.0	não	50.0	10.0	não	50.0	10.0
29	não	10.0	1.0	sim	10.0	1.0	não	10.0	1.0	não	10.0	1.0

(a) Aba de definição dos parâmetros de contexto

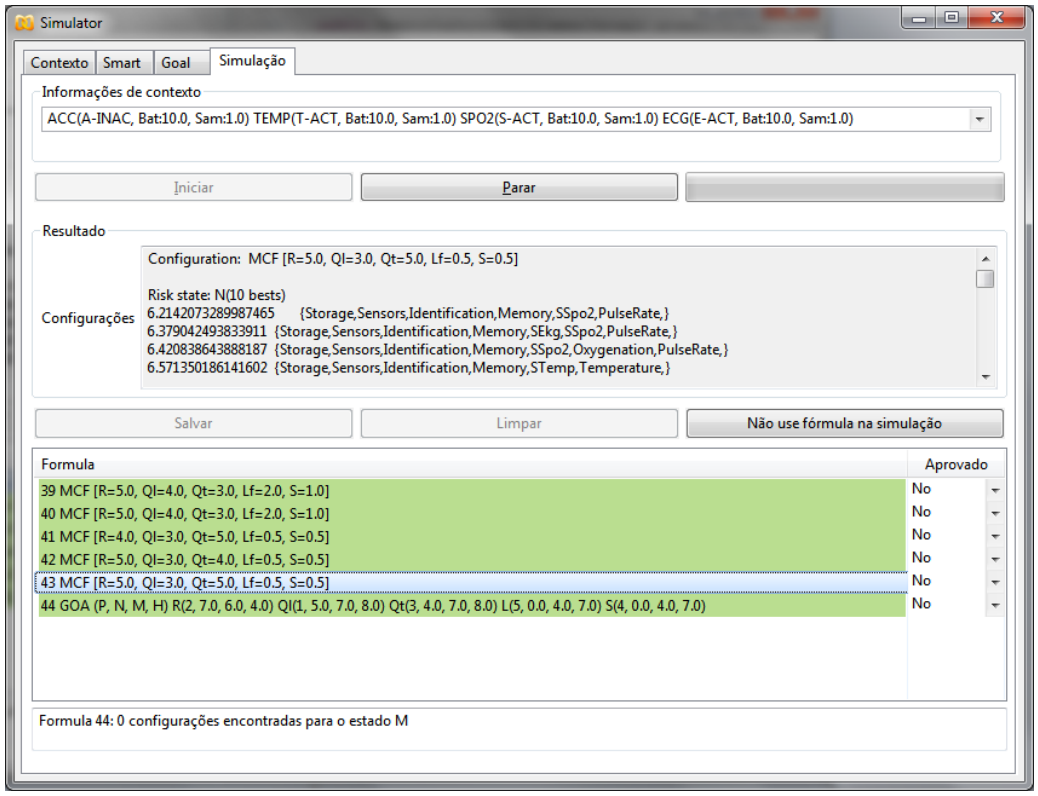


Id	Qualidade	Quantidade	Confiabilidade	Tempo de Vida	Amostragem	Obj-N	Obj-M	Obj-A
39	4.0	3.0	5.0	2.0	1.0	6.0	7.0	8.0
40	4.0	3.0	5.0	2.0	1.0	0.0	6.5	8.5
41	3.0	5.0	4.0	0.5	0.5	6.0	7.0	7.5
42	3.0	4.0	5.0	0.5	0.5	6.0	7.0	7.5
43	3.0	5.0	5.0	0.5	0.5	6.0	7.0	7.5

(b) Aba de definição das fórmulas SMART



(c) Aba de definição das fórmulas GOAL



(d) Aba de simulação de resultados

Capítulo 6

Avaliação dos Resultados

Esta capítulo apresenta a avaliação do método proposto por este trabalho no contexto RSCH. Esta avaliação é feita através de *simulações*. O processo de simulação consiste em enviar dados reais de monitoração para o sistema gerado a partir da arquitetura proposta, o MDV descrito na Seção 5.1, e observar o seu comportamento em resposta às mudanças no estado de saúde do indivíduo. A mudança de comportamento deve satisfazer aos requisitos de qualidade do estado de saúde do indivíduo, de forma que quanto mais baixo for o risco para a saúde, menor deve ser o valor de qualidade exigido pelo sistema. De forma análoga, quanto maior o risco, maior deve ser a qualidade exigida pelo sistema e, por sua vez, maior deve ser a qualidade provida pela configuração ativa da LPS no sistema.

As simulações realizadas por este trabalho seguem a metodologia descrita em Law [2003], que consiste em: (1) formular o problema, (2) coletar informações e dados para a construção do modelo conceitual, (3) validar o modelo conceitual, (4) programar o modelo, (5) validar a programação, (6) conduzir análises e experimentos e (7) executar simulação.

Em particular, as simulações tratadas neste capítulo envolvem:

- **Avaliar fórmulas para cálculo de qualidade de configurações (Seção 6.1)**. Simulação para definir valores de pesos, prioridade e objetivos de qualidade das fórmulas SMART e GOAL no domínio do RSCH.
- **Avaliar mudança de configurações com a mudança de estados de saúde (Seção 6.2)**. Observar o comportamento do sistema MDV quando o estado de risco de saúde do indivíduo real se altera.
- **Investigar utilização de *binding units* (Seção 6.3)**. Observar quais benefícios e malefícios o uso de *binding unit* traz ao MDV.

A Seção 6.4 resume os principais resultados dessas simulações e as limitações do processo de avaliação.

6.1 Estratégia de cálculo de qualidade

A avaliação do comportamento do sistema consiste em verificar se as configurações selecionadas por ele satisfazem os objetivos de qualidade do estado de risco do paciente cujos valores estão sendo enviados via simulação. O processo de seleção de configurações

depende, dentre outros fatores, do tipo de fórmula utilizada (SMART ou GOAL) e da especificação das fórmulas para cálculo de qualidade. Isto é, a satisfação de uma configuração da LPS em relação aos objetivos de qualidade de um estado depende da estratégia de cálculo de qualidade (SMART ou GOAL) e dos pesos e prioridades dos atributos de qualidade considerados por essas fórmulas. Por isso, é preciso escolher alguma fórmula de qualidade para avaliar o comportamento do sistema.

Tanto a estratégia SMART como a GOAL estabelecem pesos diferentes para os parâmetros de qualidade avaliados. A determinação numérica desses pesos, por sua vez, é uma tarefa que exige experiência do especialista do domínio e uma análise qualitativa dos resultados gerados pela aplicação da fórmula. Esta seção descreve o procedimento de criação, avaliação e ajuste das fórmulas para cálculo de qualidade SMART e GOAL que são utilizados pelo MDV.

A maior dificuldade nessa análise consiste em definir os valores para os objetivos de qualidade. No que diz respeito à estratégia SMART, em especial, a especificação numérica dos objetivos de qualidade é uma tarefa mais difícil dado que múltiplas dimensões de qualidade são agrupadas e o valor final gerado pela fórmula não possui uma semântica clara.

Nas duas estratégia, algumas etapas devem ser realizadas antes da determinação dos limites de qualidade. São elas:

1. **Especificação dos estados de risco existentes no sistema.** Os estados de Alto, Médio e Baixo risco foram utilizados.
2. **Definição de uma escala de qualidade.** Adotou-se uma escala de 0 até 10.
3. **Normalização dos valores de qualidade individual.** Todos os atributos de qualidade variam seus valores em uma escala de 0 até 1. Esta normalização foi feita apenas para facilitar o entendimento do valor atribuído a cada parâmetro de qualidade, no sentido de que 1 representa a maior qualidade possível para este parâmetro, enquanto que 0 representa a pior, ou inexistência de qualidade. A normalização é compensada pelos pesos de cada atributo na fórmula.
4. **Priorização dos atributos de qualidade**. Conforme apontado na Seção 5.1.2, os atributos de qualidade possuem prioridades diferentes de acordo com o risco de saúde do paciente. Essa priorização foi utilizada para definição dos pesos na estratégia SMART e ordem dos filtros de configurações na estratégia GOAL.

As fórmulas GOAL, em especial, exigem que o valor de porcentagem de satisfação dos objetivos de qualidade seja descrito.

1. **Porcentagem de satisfação dos objetivos de qualidade.** Para os atributos de tempo de vida estimado e taxa de amostragem, os pesos de porcentagem satisfação dos objetivos de qualidade da estratégia GOAL variaram de [0-10%] uma vez que esses parâmetros transitam entre os seus valores mínimos e máximos com maior frequência do que os valores de quantidade, qualidade e confiabilidade. Para esses últimos, a porcentagem de satisfação dos objetivos de qualidade foi de 60%.
2. **Definição dos valores limites de objetivo** Na estratégia de cálculo de qualidade GOAL, a avaliação individual de cada atributo de qualidade para as configurações

selecionadas pelo especialista do domínio definiu o objetivo de qualidade no estado de risco para este atributo.

Os valores de objetivo de qualidade das fórmulas SMART são gerados a partir de comparações entre as configurações. O procedimento de comparação é descrito a seguir:

1. **Configuração da LPS como critério de aceitação da fórmula.** Para cada estado de risco, o especialista no domínio fixou uma ou mais configurações que continham minimamente um conjunto de *features* aceitáveis para o estado de risco. O especialista no domínio apontou, por exemplo, que no estado de risco moderado, ou o sensor Spo2 ou Ecg deveria ser selecionado.
2. **Cálculo da qualidade para as configurações de comparação.** Para cada configuração fixa, foi calculado o valor de QoS obtido pela fórmula SMART.
3. **Especificação dos valores dos objetivos de qualidade**. Os valores de QoS calculados para cada configuração fixa foram utilizados como limites de qualidade.

Em ambas as estratégias, o valor mínimo do objetivo de qualidade para o estado de risco moderado define o valor máximo do objetivo de qualidade aceito para o estado de risco baixo. O valor mínimo do objetivo de qualidade para o estado de risco alto define o valor máximo do objetivo de qualidade do risco moderado. Por fim, o valor máximo do objetivo de qualidade aceito pelo risco alto é 10.

Os passos descritos nessa seção até então descrevem em alto nível a estratégia para construção e aceitação das fórmulas SMART e GOAL. A fim de se estabelecer valores numéricos para cada parâmetro dessas fórmulas, foi feita uma simulação sobre o comportamento das fórmulas no contexto da RSCH avaliada. A Seção 6.1.1 detalha a execução dessa simulação e os principais resultados obtidos.

6.1.1 Simulação e Avaliação das Estratégias de Cálculo no contexto da RSCH

O objetivo de realizar uma simulação sobre o comportamento das fórmulas SMART e GOAL é definir valores numéricos para cada parâmetro que as compõem a fim de viabilizar o uso delas no MDV. Simular e avaliar o comportamento das fórmulas significa verificar se a fórmula, com os seus pesos, prioridade e objetivos de qualidade, seleciona configurações da LPS adequadas para o domínio. Para isso, antes de mais nada, é preciso definir quais configurações podem ser geradas, ou seja, qual o modelo de *features* do cenário de simulação.

Cenário e premissas de simulação O modelo de *features* da Figura 6.1 apresenta as *features* no MDV e o relacionamento entre elas. Este modelo é composto por 12 *features* concretas: 4 *features* de sensores, 5 *features* de informação de sensores e 3 *features* de armazenamento.

O critério escolhido para definir se uma fórmula deve ou não ser aceita no domínio é a condição de que o sistema utilizando a fórmula, em um estado de risco, escolha algumas configurações específicas da LPS. Essas configurações são chamadas de *configurações de critério*. Com auxílio do especialista do domínio, essas configurações foram definidas.

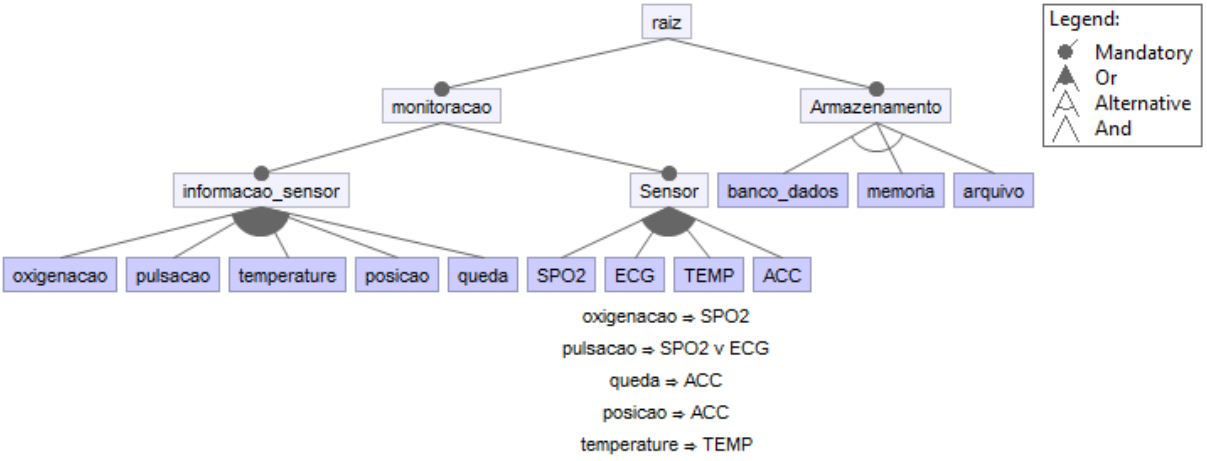


Figura 6.1: Modelo de *features* do MVD

A Tabela 6.1 detalha as *features* presentes nessas *configurações critério*. Conforme apresentado, as fórmulas utilizadas no estado de alto risco devem selecionar pelo menos uma configuração dentre as listadas na tabela para o estado de alto risco. Isto é, pelo menos uma das seguintes configurações:

- Configuração com as *features* spo2, pulsacao , ecg, oxigenacao, banco de dados;
- Configuração com as *features* spo2, pulsacao , temp, oxigenacao, temperatura, banco de dados;
- Configuração com as *features* ecg, pulsacao , temp , temperatura, banco de dados.

Se o sistema, utilizando a fórmula, não consegue selecionar pelo menos esta configuração, então a fórmula é considerada inadequada para o estado de alto risco. Entretanto, se o sistema utilizando a fórmula consegue selecionar alguma das configurações apresentadas na Tabela 6.1 para o estado de risco moderado, então a fórmula é considerada válida para o estado de risco moderado. Esse critério de adequação por estado de risco é necessário caso não seja possível gerar uma fórmula que selecione as configurações adequadas nos três estados de risco. Preferencialmente, utilizam-se as fórmulas que quando utilizadas pelo sistema são capazes de selecionar configurações requeridas nos três estados.

Risco de Saúde	**Configurações**
Baixo	{ spo2, oxigenacao, memoria } { ecg, pulsacao, memoria } { temp, temperatura, file }
Moderada	{ spo2, oxigenacao, arquivo } { ecg, pulsacao, banco_de_dados } { spo2, pulsacao, ecg, oxigenacao, file }
Alta	{ spo2, pulsacao, ecg, oxigenacao, banco_de_dados } { spo2, pulsacao, temp, oxigenacao, temperatura, banco_de_dados } { ecg, pulsacao, temp , temperatura, banco_de_dados }

Tabela 6.1: Configurações obrigatórias para cada estado de risco

O comportamento das fórmulas de cálculo de qualidade é influenciado pelo contexto da RSCH de tal forma que não faz sentido a fórmula selecionar configurações que utilizem sensores que não estão presentes no contexto. A simulação observa o comportamento das fórmulas avaliadas em nove cenários diferentes de monitoração. Cada cenário diferencia-se em relação à presença/ausência de sensores, e variação nos valores de bateria e amostragem desses. Os cenários são descritos na Tabela 6.2.

	Ecg			**Acc**			**Spo2**			**Temp**		
Id	Presente	Bateria	Amostragem	Presente	Bateria	Amostragem	Presente	Bateria	Amostragem	Presente	Bateria	Amostragem
1	Sim	100	20	Sim	100	20	Sim	100	20	Sim	100	20
2	Sim	50	10	Sim	50	10	Sim	50	10	Sim	50	10
3	Sim	10	1	Sim	10	1	Sim	10	1	Sim	10	1
4	Não			Sim	100	20	Sim	100	20	Sim	100	20
5	Não			Sim	50	10	Sim	50	10	Sim	50	10
6	Não			Sim	10	1	Sim	10	1	Sim	10	1
7	Não			Não			Não			Sim	100	20
8	Não			Não			Não			Sim	50	10
9	Não			Não			Não			Sim	10	1

Tabela 6.2: Cenários do contexto para a avaliação das fórmulas de qualidade

Após a definição dos objetivos de qualidade, por meio da ferramenta QSE-BSN foram simulados alguns cenários de contexto de monitoração a fim de se observar o comportamento das fórmulas. A partir de uma análise qualitativa, foram feitos alguns ajustes nos pesos, prioridade e objetivos de qualidade dessas fórmulas até que elas fossem consideradas adequadas pelo especialista do domínio. Os atributos de qualidade utilizados foram: qualidade de informação, quantidade de informação, confiabilidade, taxa de amostragem dos sensores ou atualidade dos dados, e tempo de vida estimado do sistema.

Definição de Pesos, Prioridades e Objetivos de Qualidade: A simulação sobre o comportamento das fórmulas de cálculo de qualidade é auxiliada pela ferramenta QSE-BSN, descrita na Seção 5.3. Via QSE-BSN, definimos os parâmetros das fórmulas e o contexto de monitoração. A ferramenta lista as configurações que o sistema seleciona ao utilizar a fórmula avaliada no momento da simulação.

Devido a diferença de comportamento entre as fórmulas SMART e GOAL, os passos para se estabelecer os limites de qualidade são específicos da estratégia.

No que diz respeito a definição de valores iniciais dos objetivos de qualidade para a estratégia GOAL, o especialista do domínio estabeleceu alguns valores limites iniciais. Além disso, conforme descrito na Seção 5.1.2, estabeleceu alguns valores de prioridade. Para cada risco, os valores iniciais de objetivo de qualidade dos atributos são:

- Nos estados de risco baixo, os atributos de qualidade devem estar no intervalo [0.0-4.0].

- Nos estados de risco moderado, os atributos de qualidade devem estar no intervalo [4.1-6.0].
- Nos estados de risco moderado, os atributos de qualidade devem estar no intervalo [6.1-10.0].

Observe que esses valores iniciais estão nos intervalos definidos no início da Seção 6.1.

Para representar as mudanças de prioridades dos parâmetros de qualidade em cada estado de risco, conforme apresentado na Seção 5.1.2, foram geradas fórmulas GOAL diferentes em relação aos pesos, porcentagens de satisfação e objetivos de qualidade. Por meio de um processo de refinamento e ajuste desses parâmetros, e validação das configurações selecionadas pelo sistema, algumas fórmulas GOAL foram aceitas no domínio.

As Tabelas 6.3 e 6.4 detalham os valores de prioridades e objetivo de qualidade de cada fórmula.

	Qualidade				**Confiabilidade**				**Tempo Vida**			
Fórmula	Prioridade	Baixo	Moderado	Alto	Prioridade	Baixo	Moderado	Alto	Prioridade	Baixo	Moderado	Alto
GOAL_F1	1	5	7	8	2	5,6	4,8	3,2	5	0	0,4	0,7
GOAL_F2	2	4	5,6	6,4	1	7	6	4	5	0	0,4	0,7
GOAL_F3	1	2,5	3,5	4	2	3,5	3	2	5	0	2	3,5
GOAL_F4	1	5	7	8	2	7	6	4	5	0	4	7

Tabela 6.3: Fórmulas GOAL aceitas

	Quantidade				**Amostragem**			
Fórmula	Prioridade	Baixo	Moderado	Alto	Prioridade	Baixo	Moderado	Alto
GOAL_F1	3	2,4	4,2	4,8	4	0	0,04	0,07
GOAL_F2	3	2,4	4,2	4,8	4	0	0,4	0,7
GOAL_F3	3	2	3,5	4	4	0	2	3,5
GOAL_F4	3	4	7	8	4	0	4	7

Tabela 6.4: Fórmulas GOAL aceitas (cont.)

É possível observar que os objetivos de qualidade envolvendo os parâmetros taxa de amostragem e tempo de vida são bem inferiores aos objetivos de qualidade dos outros atributos. Caso o parâmetro de tempo de vida tivesse um objetivo de qualidade elevado, por exemplo 5.0, o sistema não conseguiria selecionar configurações para contexto onde os

sensores estivessem com um valor abaixo de 50%, mesmo que os valores de confiabilidade e qualidade de informação satisfizessem os objetivos de qualidade nos estados de risco.

A definição dos objetivos de qualidade das fórmulas SMART utiliza as configurações de critério estabelecidas na Tabela 6.1. O valor de qualidade das configurações de critério de um estado de risco é utilizado para estabelecer os valores dos objetivos de qualidade desse estado. Considere, por exemplo, que se quer estabelecer o objetivo de qualidade de uma fórmula SMART para o estado de alto risco. Para tal avaliação, é calculado o valor de qualidade das configurações de critério para o estado de alto risco da Tabela 6.1. A média desses valores define o valor inicial de objetivo para o estado de risco alto utilizando essa fórmula. Os pesos dos atributos são definidos inicialmente a partir das prioridades definidas para os estados na Seção 5.1.2.

Para representar as mudanças de prioridades dos parâmetros de qualidade em cada estado de risco, conforme apresentado na Seção 5.1.2, foram geradas fórmulas SMART diferentes em relação aos pesos atribuídos a cada parâmetro de qualidade e objetivos de qualidade. Por meio de um processo de refinamento e ajuste desses objetivos de qualidade, pesos dos atributos de qualidade, e validação das configurações selecionadas pelo sistema, algumas fórmulas SMART foram aceitas no domínio. A Tabela 6.5 detalha os pesos dos atributos e os objetivos de qualidade dessas fórmulas.

	Parâmetros					**Objetivos**		
ID	Qualidade	Quantidade	Confiabilidade	Tempo Vida	Amostragem	Baixo	Moderado	Alto
SMART_F1	4.0	3.0	5.0	2.0	1.0	6.0	7.0	8.0
SMART_F2	4.0	3.0	5.0	2.0	1.0	0.0	6.5	8.5
SMART_F3	4.0	5.0	3.0	1.0	2.0	0.0	5.5	8.0
SMART_F4	3.0	5.0	4.0	0.5	0.5	6.0	7.0	7.5
SMART_F5	3.0	4.0	5.0	0.5	0.5	6.0	7.0	7.5
SMART_F6	3.0	5.0	5.0	0.5	0.5	6.0	7.0	7.5

Tabela 6.5: Fórmulas SMART aceitas no domínio

Pelo mesmo motivo explicado na estratégia GOAL, a estratégia SMART atribui peso pequeno para os atributos de qualidade tempo de vida e amostragem dos sensores.

Avaliação dos Resultados: A fim de comparar os comportamentos das duas estratégias de cálculo de qualidade no contexto da RSCH, foi feito um estudo qualitativo sobre a seleção das *features* pelo sistema utilizando as fórmulas aceitas do ponto de vista do domínio e critérios estabelecidos na Seção 6.1.

Ao todo, foram analisados os comportamentos de quatro fórmulas GOAL aceitas e seis fórmulas SMART aceitas. No geral, observou-se um comportamento semelhante sobre a escolha das configurações da LPS. Entretanto, para um estado de saúde, o sistema seleciona mais configurações utilizando as fórmulas GOAL do que utilizando as fórmulas

SMART, conforme Figura 6.2. Observe, por exemplo, que no estado de alto risco, o sistema utilizando uma das fórmulas GOAL aceitas é capaz de selecionar 129 configurações, enquanto que o número máximo de configurações selecionadas pelo sistema utilizando as fórmulas SMART aceitas é 64. Esse comportamento é percebido também para os estados de risco moderado e baixo.

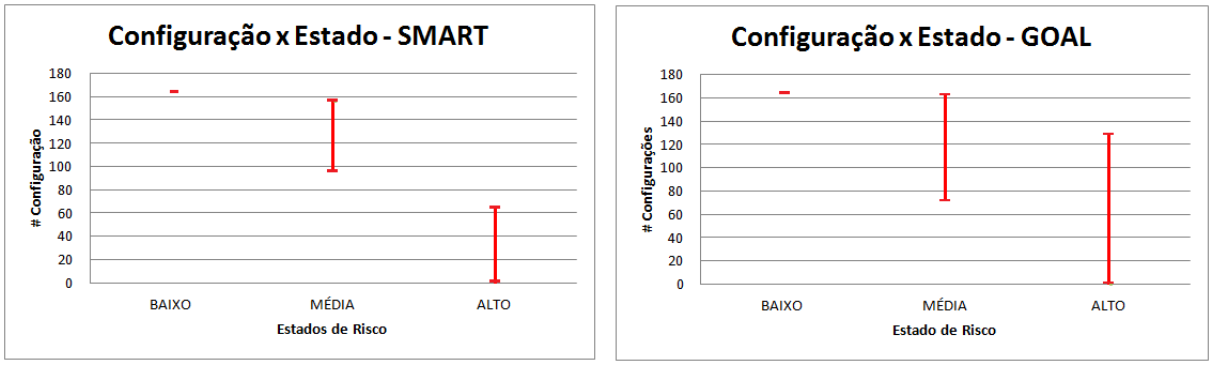


Figura 6.2: Comportamento das Estratégias GOAL e SMART em relação ao número de configurações por estado.

No processo de simulação, foi possível perceber que o ajuste das fórmulas GOAL é mais complexo do que o ajuste das fórmulas SMART. Um dos possíveis motivos é o próprio número de parâmetros de configuração. Para todo atributo de qualidade considerado, são definidos quatro parâmetros, o que representa o ajuste de 20 parâmetros (Prioridade, percentual de satisfação, objetivos de qualidade para os estados de risco baixo, moderado e alto para os cinco atributos de qualidade considerados na simulação). Por outro lado, as fórmulas SMART só exigem que sejam especificados 8 parâmetros (5 para os pesos dos atributos de qualidade considerados, e 3 para os objetivos nos riscos de saúde baixo, moderado e alto).

Algumas das fórmulas geradas pela estratégia GOAL demonstraram um comportamento contraditório ao esperado pelo sistema em relação à linearidade do número de configurações por estado. Ou seja, algumas fórmulas da estratégia GOAL selecionaram mais configurações para o estado de risco moderado do que para o estado de risco alto, por exemplo, mesmo satisfazendo os critérios estabelecidos na Seção 6.1. Esse comportamento acontece quando os parâmetros de percentual de satisfação e os objetivos de qualidade de atributos específicos são mal ajustados. Essas fórmulas foram eliminadas da lista de fórmulas aceitas.

Por fim, mesmo selecionando mais configurações do que o necessário, ao se utilizar as fórmulas GOAL e os critérios estabelecidos na Seção 5.1.2 para priorizar algumas configurações e critérios estabelecidos como premissas na simulação, o sistema selecionou configurações mais adequadas utilizando a estratégia GOAL do que utilizando a estratégia SMART. Porém, consideramos que o número de simulações realizadas não é o suficiente para decidirmos qual a melhor estratégia dentre as duas no contexto RSCH. As duas se mostraram adequadas quando tiveram seus parâmetros qualitativamente bem ajustados.

6.2 Monitoração dos Sinais Vitais

Esta seção descreve a simulação realizada para avaliar o comportamento do MDV em resposta a mudanças no estado de saúde de um indivíduo com dados vitais reais.

6.2.1 Reconfiguração do sistema em virtude da mudança dos estados do indivíduo monitorado

O objetivo de realizar a simulação dos dados vitais é observar o comportamento do sistema em resposta à mudanças no estado de saúde do indivíduo. Espera-se que com o aumento da gravidade do estado de saúde do indivíduo monitorado, o sistema modifique sua configuração de forma a satisfazer os requisitos desse estado de maior gravidade. Da mesma forma, com a diminuição da gravidade do estado de saúde do indivíduo monitorado, o sistema modifique sua configuração de forma a satisfazer os requisitos do estado de menor gravidade. Nesse caso, os requisitos são os objetivos de qualidade impostos pelos estados do sistema.

Como uma situação de emergência é de difícil controle, dada a imprevisibilidade de um evento de saúde com riscos moderados ou altos, a simulação realizada utiliza dados reais de uma base de dados do *Beth Israel Hospital* (MIT-BIH) [1]. Uma ferramenta desenvolvida para auxiliar o processo de simulação, o Simulador de Dados Vitais (SDV) descrito na Seção 5.2, lê os dados da base MIT-BIH e encaminha para o sistema de monitoração de dados vitais (MDV) descrito na Seção 5.1. Em geral, os dados fornecidos dizem respeito à pulsação cardíaca, oxigenação, temperatura, pressão arterial, respiração de indivíduos com mais de 60 anos de idade, e com histórico de problemas cardíacos tais como infarto ou hipertensão.

O processo de simulação deste trabalho simula os sensores físicos de uma RSCH.

6.2.2 Cenários e premissas de simulação

Um dos cenários de avaliação do sistema foi construído utilizando os dados vitais do paciente identificador por $MIMIC-055N-ALL-ANN$ da base MIT-BIH. Este paciente representa uma senhora com 65 anos com histórico de hipertensão [2]. Foram monitorados os dados vitais de oxigenação, pulsação cardíaca, temperatura e respiração. O sistema RSCH implementado não trata a informação de respiração, por isso essa informação foi retirada dessa simulação.

A LPS do MDV possui o modelo de *features* apresentado na Figura 6.1. O modelo de *features* foi traduzido em expressões lógicas conforme apresentado na Fórmula 6.1 e codificado no sistema MDV.

[1] `http://ecg.mit.edu/`

[2] Descrição sucinta do estado de saúde do paciente disponível em `http://www.physionet.org/physiobank/database/mghdb/patient-guide.shtml\#mgh055`

$$
\begin{aligned}
mf = &((oxigenacao \vee pulsacao\_cardiaca \vee temperatura \vee posicao \vee queda) \wedge \\
&(ecg \vee temp \vee spo2 \vee acc) \wedge \\
&((\neg memoria \wedge arquivo \wedge \neg temperatura) \vee \\
&(memoria \wedge \neg arquivo \wedge \neg temperatura) \vee \\
&(\neg memoria \wedge \neg arquivo \wedge temperatura)) \wedge \\
&(oxigenacao \vee \neg spo2) \wedge \\
&(pulsacao\_cardiaca \vee \neg(spo2 \vee ecg)) \wedge \\
&(queda \vee \neg acc) \wedge \\
&(posicao \vee \neg acc) \wedge \\
&(temperatura \wedge \neg temp))
\end{aligned}
\tag{6.1}
$$

Com o auxílio de um especialista do domínio, a gramática proposta foi instanciada com os valores detalhados no Quadro 6.1.

Quadro 6.1: Gramática especificada para a realização da simulação.

```
1 //Fonte de dados vitais
2 spo2
3
4 //Propriedades
5 spo2[oxygenation]
6
7 //Eventos
8 spo2_unavailable (spo2[oxigenacao] > 100.0)
9 oxygen_n (spo2[oxigenacao] >= 94.0)
10 oxygen_m (spo2[oxigenacao] >= 90.0)
11 oxygen_h (spo2[oxigenacao] >= 0.0)
12
13 //Fusão de dados
14 fusion_oxygen_n {oxygen_n}
15 fusion_oxygen_m {oxygen_m}
16 fusion_oxygen_h {oxygen_h}
17
18 //Transições
19 (NR, fusion_oxygen_m {oxygen_m})-> MR
20 (MR, fusion_oxygen_h {oxygen_h})-> HR
```

Os dados vitais desse paciente fornecidos pelo MIT-BIH correspondem a 1h de monitoração. Entretanto, a simulação descrita aqui foi feita com duração de 10 minutos, no instante 00:06:00 até 00:16:00. Este intervalo específico foi escolhido porque entre os instantes 00:08:35 e 00:08:55, esta paciente apresenta uma diminuição leve de oxigenação, conforme Figura 6.3. Entre os instantes 00:00:00 até 00:08:35 todos os dados da paciente estavam em um estado normal de acordo com a gramática do Quadro 6.1.

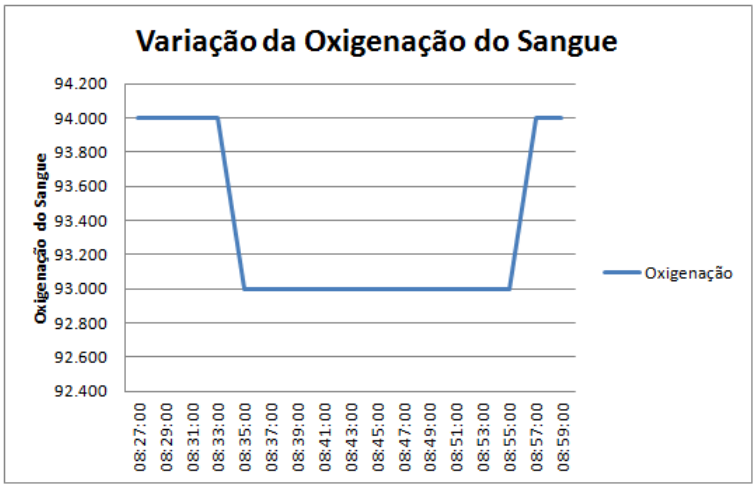


Figura 6.3: Variação de oxigenação no intervalo de 00:08:35 até 00:08:59 segundos de monitoração.

A avaliação do comportamento do sistema consiste em verificar se as configurações selecionadas por ele satisfazem os objetivos de qualidade do estado de risco do paciente cujos valores estão sendo enviados via simulação. O processo de seleção de configurações depende, dentre outros fatores, do tipo de fórmula utilizada (SMART ou GOAL) e da especificação das fórmulas para cálculo de qualidade. Isto é, a satisfação de uma configuração da LPS em relação aos objetivos de qualidade de um estado depende da estratégia de cálculo de qualidade (SMART ou GOAL) e dos pesos e prioridades dos atributos de qualidade considerados por essas fórmulas. Por isso, é preciso escolher alguma fórmula de qualidade para avaliar o comportamento do sistema.

Esta simulação descreve o comportamento do sistema utilizando duas fórmulas de qualidade, uma para a estratégia GOAL outra para a estratégia SMART. A Tabela 6.6 apresenta os atributos de qualidade e os seus valores respectivos de prioridade, percentual de satisfação e os objetivos de qualidade para os estados de risco baixo, moderado e alto da fórmula GOAL utilizada. A Tabela 6.7 apresenta os atributos de qualidade, os seus pesos e os objetivos de qualidade para os estados de risco baixo, moderado e alto da fórmula SMART utilizada.

Fizemos uma amostragem das fórmulas aceitas pelo processo de simulação descrito na Seção 6.1.1. Acreditamos que com a escolha das outras fórmulas aceitas pelo processo descrito, o sistema se comportaria de forma semelhante em relação às escolhas de configuração da LPS para um estado de risco de saúde.

6.2.3 Execução da Monitoração

Esta seção descreve os acontecimentos durante a execução da simulação para as premissas e contexto estabelecidos na Seção 6.2.2. A execução da monitoração é auxiliada pela ferramenta SDV, descrita na Seção 5.2. O SDV lê os dados vitais do indivíduo $MIMIC-055N-ALL-ANN$ e os encaminha para o MDV. A simulação descrita aqui acontece entre os instantes 00:06:00 e 00:16:00 da base de dados original.

Atributos de Qualidade	***Parâmetro***	**Valor**
Qualidade de Informação	Prioridade	2
	Percentual Satisfação	0.8
	Obj. Risco Baixo	0.5
	Obj. Risco Moderado	7.0
	Obj. Risco Alto	8.0
Confiabilidade	Prioridade	1
	Percentual Satisfação	1.0
	Obj. Risco Baixo	7.0
	Obj. Risco Moderado	6.0
	Obj. Risco Alto	4.0
Tempo de Vida	Prioridade	5
	Percentual Satisfação	0.1
	Obj. Risco Baixo	0.0
	Obj. Risco Moderado	4.0
	Obj. Risco Alto	7.0
Quantidade de Informação	Prioridade	3
	Percentual Satisfação	0.6
	Obj. Risco Baixo	4.0
	Obj. Risco Moderado	7.0
	Obj. Risco Alto	8.0
Taxa de Amostragem	Prioridade	4
	Percentual Satisfação	0.1
	Obj. Risco Baixo	0.0
	Obj. Risco Moderado	4.0
	Obj. Risco Alto	7.0

Tabela 6.6: Fórmula GOAL utilizada na simulação

Observe que no instante de 00:08:35 da Figura 6.3, a oxigenação do sangue do paciente passa do valor de 94% para 93%. Conforme regra da linha 10 na gramática do Quadro 6.1, o evento *oxygen_ m* de é gerado. A regra de fusão de evento da linha 15 gera um novo evento e este por sua vez, altera o estado de risco de normal (baixo) para o risco moderado, conforme regra da linha 19.

O cenário antes do instante 00:08:35 atua com uma configuração com as *features* apresentadas no Quadro 6.2[3]

Quadro 6.2: *Features* do sistema antes do instante 00:08:35 da simulação.

```
1 memória, spo2, oxigenacao.
```

A mudança de estado motiva a mudança de configuração da LPS utilizada. O sistema, então, busca por configurações mais adequadas ao objetivo de qualidade definido pelo novo estado de risco. As fórmulas utilizadas para avaliar a qualidade das configurações normalmente selecionam mais configurações do que o necessário. Assim, o sistema só

[3]São omitidas do quadro as *features* abstratas Armazenamento, Sensores, Informação de Sensores

Parâmetro da Fórmula	**Valor**
Qualidade de Informação (Peso)	4.0
Quantidade de Informação (Peso)	5.0
Confiabilidade (Peso)	3.0
Tempo de Vida (Peso)	1.0
Taxa de Amostragem (Peso)	2.0
Obj. Risco Baixo	0.0
Obj. Risco Moderado	5.5
Obj. Risco Alto	8.0

Tabela 6.7: Fórmula SMART utilizada na simulação

pode utilizar uma única configuração por vez, enquanto que a lista de configurações selecionada a partir da aplicação da fórmula de cálculo de qualidade seleciona mais de uma configuração. Para cada estado, os critérios de priorização das configurações estão estabelecidos na Seção 5.1.2.

A mudança de estado de saúde de normal para risco moderado utilizando a fórmula GOAL detalhada na Tabela 6.6 motiva o sistema a buscar novas configurações. A listagem de configurações possíveis selecionadas pelo sistema utilizando essa fórmula é informada no Quadro 6.3.

Quadro 6.3: *Features* do sistema antes do instante 00:08:35 da simulação pela estratégia GOAL

```
1 Banco de dados,Temp.,Spo2,Acc,Oxigenacao,Temperatura,Queda
2 Banco de dados,Temp.,Spo2,Acc,Oxigenacao,Temperatura,Posição
3 Banco de dados, Temp., Spo2, Acc,Oxigenacao,Temperatura,Posição,Queda
4 Memoria, Temp., Spo2, Acc,Oxigenacao,Temperatura,Queda
5 Memoria, Temp., Spo2, Acc,Oxigenacao,Temperatura,Posição
6 Memoria, Temp., Spo2, Acc,Oxigenacao,Temperatura,Posição,Queda
7 Arquivo, Temp., Spo2, Acc,Oxigenacao,Temperatura,Queda
8 Arquivo, Temp., Spo2, Acc,Oxigenacao,Temperatura,Posição
9 Arquivo, Temp., Spo2, Acc,Oxigenacao,Temperatura,Posição,Queda
```

O mesmo cenário de monitoração foi simulado para avaliar a escolha da estratégia SMART. A mudança de estado de saúde de normal para risco moderado utilizando a fórmula SMART da Tabela 6.7 motiva o sistema a buscar por novas configurações. A listagem de configurações possíveis selecionadas pelo sistema utilizando essa fórmula é informada no Quadro 6.4.

Quadro 6.4: *Features* do sistema antes do instante 00:08:35 da simulação pela estratégia SMART.

```
1 Banco de dados, Temp, Spo2, Acc,Oxigenacao,Temperatura,Queda
2 Banco de dados, Temp, Spo2, Acc,Oxigenacao,Temperatura,Posição
3 Banco de dados, Temp, Spo2, Acc,Oxigenacao,Temperatura,Posição,Queda
4 Memoria, Temp., Spo2,Oxigenacao, Pulsação, Temperatura, Ecg
5 Memoria, Temp., Spo2,Oxigenacao, Pulsação, Temperatura, Ecg
6 Memoria, Temp., Spo2,Oxigenacao, Pulsação, Temperatura, Ecg
7 Arquivo, Temp., Spo2, Acc,Oxigenacao,Temperatura,Queda
8 Arquivo, Temp., Spo2, Acc,Oxigenacao,Temperatura,Posição
```

```
9 Arquivo , Temp. , Spo2 , Acc , Oxigenacao , Temperatura , Posição , Queda
```

Utilizando os critérios de priorização de configuração estabelecidos no início desta Seção, as configurações selecionadas pelo sistema utilizando as fórmula SMART e GOAL são diferentes. A configuração escolhida para a estratégia GOAL foi o da linha 1 do Quadro 6.3, e a escolhida pela estratégia SMART foi a da linha 3 do Quadro 6.4. Essas configurações são igualmente possíveis em um cenário real, no sentido de que se adequam ao domínio. A diferença entre as configurações se dá em virtude da diferença de pesos e objetivos de qualidade entre as fórmulas.

Devido à falta de dados reais para simular situações de alto risco nos arquivos do MIT-BIH, a simulação de um cenário de alto risco foi feita manualmente via SDV. No cenário em questão, a oxigenação do sangue do paciente simulado foi de 94% para 50%. Conforme regra da linha 11 da gramática do Quadro 6.1, um evento *oxygen_h* é gerado. A regra de fusão de evento da linha 16 gera um novo evento e este por sua vez, altera o estado de risco de moderado para o risco alto, conforme regra da linha 20.

As configurações selecionadas pela estratégia SMART e GOAL estão descritas nos Quadros 6.5 e 6.6, respectivamente. Aplicando critérios de seleção definidos na Seção 5.1.2, as configurações da primeira linhas de ambos os quadros foram escolhidas. Observe que a estratégia GOAL é mais restritiva sobre as *features* selecionadas, de forma que a *feature Memoria* foi eliminada das configurações de alto risco devido a sua característica de baixa qualidade sobre os dados manipulados.

Quadro 6.5: *Features* selecionadas pela estratégia SMART em um cenário de alto risco.

```
1 Banco Dados , Temp , Spo2 , Ecg , Acc , Oxigenacao , Temperatura , Posicao , Queda , Pulsacao
2 Memoria , Temp , Spo2 , Ecg , Acc , Oxigenacao , Temperatura , Posicao , Queda , Pulsacao
3 Arquivo , Temp , Spo2 , Ecg , Acc , Oxigenacao , Temperatura , Posicao , Queda , Pulsacao
```

Quadro 6.6: *Features* selecionadas pela estratégia GOAL em um cenário de alto risco.

```
1 Banco Dados , Temp , Spo2 , Ecg , Acc , Oxigenacao , Temperatura , Posicao , Queda , Pulsacao
2 Arquivo , Temp , Spo2 , Ecg , Acc , Oxigenacao , Temperatura , Posicao , Queda , Pulsacao
```

Ambas as seleções são possíveis em ambiente um real. adequadas.

6.2.4 Avaliação dos Resultados

A simulação feita com os dados da base de dados vitais MIT-BIH demonstrou o funcionamento esperado para o sistema de MDV. Isto é, com o aumento do risco de saúde, menos configurações foram selecionadas mas com mais qualidade. Da mesma forma, com a diminuição do risco de saúde, mais configurações foram permitidas para a seleção porém com menor qualidade.

Para observar a seleção das configurações, foi necessário definir um contexto de execução do sistema e qual fórmula de cálculo de qualidade seria utilizada. Por amostragem, foram escolhidas uma fórmula de cada estratégia sugerida por esse trabalho: uma fórmula SMART e outra GOAL. As fórmulas foram escolhidas a partir de uma lista de possíveis fórmulas construída por meio de um processo qualitativo semi-automático descrito na Seção 6.1.1.

Apesar da pequena amostra, acreditamos que o comportamento dessa simulação com outras fórmulas seria semelhante ao descrito aqui. Entretanto, uma avaliação mais rígida é necessária para confirmar essa hipótese.

Do ponto de vista do domínio, a estratégia SMART selecionou configurações mais fracas do que a estratégia GOAL em cenários de maior risco. Algumas fórmulas SMART, em situação de alto risco, consideram a possibilidade de utilizar a feature memória para armazenar os dados recebidos dos sensores, o que impossibilita a recuperação desses valores em um período posterior. Do ponto de vista médico, o diagnóstico é comprometido. Apesar disso, consideramos que o número de simulações realizadas não é o suficiente para decidirmos qual a melhor estratégia dentre as duas no contexto RSCH onde dados reais são submetidos ao sistema. Mesmo com configurações menos apropriadas para o estado de risco, os critérios de seleção definidos na Seção 5.1.2 foram o suficiente para orientar o sistema na melhor seleção do ponto de vista do domínio.

6.3 Inserção de *Binding Units*

Na identificação de mudanças de estado de saúde do indivíduo, o sistema deve buscar por configurações da LPS que melhor se adequem aos requisitos de qualidade, isto é, objetivos de qualidade do novo estado. O processo de busca, conforme descrito na Seção 5.1.2, é implementado via utilização da API de uma biblioteca Java para satisfabilidade.

Com o intuito de observar o comportamento do sistema sobre o processo de busca por configurações da LPS, realizamos alguns experimentos para decidir como as estruturas de *binding units* poderiam ser utilizadas para reduzir o espaço de busca de configurações possíveis. Em uma avaliação mais detalhada no processo de seleção de configurações da LPS, identificamos que com as 12 *features* concretas do sistema era possível construir 165 configurações diferentes. Esse valor não considera o contexto de monitoração, ausência ou presença de features, ou aplicação de fórmulas de cálculo de qualidade. Ou seja, é possível combinar de 165 formas diferentes as *features* no modelo de *features* e gerar configurações válidas da LPS.

Como já explicado, uma forma de reduzir o número de configurações possíveis de serem avaliadas é através do uso de *binding units*. Objetivando observar o comportamento da busca por configurações válidas da LPS, construímos algumas *binding units* e observamos o número máximo de configurações possíveis na LPS.

Em um modelo de *features* sem nenhuma *binding unit*, é necessário construir um modelo com 12 variáveis lógicas: 4 para representar sensores (Sensor Oxímetro, Sensor Eletrocardiógrafo, Sensor Acelerômetro, Sensor Temperatura), 5 para representar informações fornecidas pelos sensores (Temperatura, Oxigenação, Pulsação Cardíaca, Queda, Posicionamento) e 3 para representar as *features* de armazenamento (Armazenamento na memória, banco de dados e arquivo). Com a inserção das restrições lógicas relacionadas à alternatividade, opcionalidade, condição *feature-ou* e restrições *cross-tree*, é preciso gerar mais 6 variáveis no modelo.

Para avaliar a redução no número de configurações possíveis, foram construídas três *binding units* no modelo :[4].

[4]Construção de *binding units* considera, além do tempo de conexão da *feature* com o sistema, o relacionamento semântico entre *features*

BU1 - Composta por {acc, posição, queda}

BU2 - Composta por {spo2, pulsação, oxigenação}

BU3 - Composta por {spo2, ecg, pulsação, oxigenação},

Cada *binding unit* foi inserida no modelo, e os seus números máximos de configurações anotados. A Tabela 6.8 apresenta os resultados sobre a redução no número de configurações da LPS com a inserção de *binding untis*.

BU	# de Variáveis Modelo Choco	# de Soluções
{}	63	165
{BU1}	65	84
{BU2}	66	48
{BU1, BU3}	68	24
{BU3}	67	24
{BU1, BU2}	69	12

Tabela 6.8: Quantificação das soluções para a RSCH

Observe que com a inserção de uma dessas *binding units* (BU1, BU2 ou BU3), o número de configurações reduz quase que pela metade. Em contrapartida, o número de variáveis que devem ser adicionadas ao modelo Choco chega a uma média aproximada de 3 por *binding unit*. Apesar disso, por causa da baixa quantidade de *features* no RSCH avaliado, optamos por não inserir *binding units* no MDV. Consideramos que 165 configurações é um número baixo para influenciar a performance do sistema no momento da busca. Além disso, a inserção de *binding units* reduz a variabilidade do sistema, o que no nosso caso representa que um estado de risco de saúde escolhe quase sempre a mesma configuração.

6.4 Discussões gerais sobre os resultados

O objetivo principal das simulações descritas neste capítulo foi avaliar o comportamento do sistema de monitoração de dados vitais (MDV) quando dados reais vitais fossem enviados para esse sistema. Por avaliar o comportamento, nos referimos, verificar se com o aumento do risco de saúde do indivíduo, o sistema utiliza configurações da LPS com maior qualidade. Da mesma forma, se com a diminuição do risco de saúde do indivíduo, o sistema utiliza configurações da LPS com menor qualidade.

Para tal simulação, foi necessário definir algumas fórmulas de cálculo de qualidade utilizando as estratégias sugeridas SMART e GOAL. Apesar do processo qualitativo na avaliação das fórmulas, acreditamos que as restrições impostas pelo especialista do domínio e critérios de seleção utilizados geram fórmulas com comportamentos semelhantes e adequados ao domínio. Por esse motivo, embora tenhamos selecionado por amostragem as fórmulas de cálculo de qualidade que foram utilizados no MDV durante a simulação, as configurações selecionadas devem ser as mesmas ou semelhantes às configurações selecionadas durante esse processo. É necessário desenvolver uma avaliação quantitativa mais precisa para confirmar essa hipótese.

Esta simulação foi realizada em uma LPSD com uma quantidade pequena de *features*. Esse cenário limitado traz implicações sobre os resultados principalmente no que diz respeito às atividades de busca por configurações válidas e configuração dinâmica da LPS. Como o problema de satisfabilidade (SAT) é NP-Completo, o desempenho da busca por uma configuração adequada pode vir a ficar comprometido com o aumento do número de *features* na LPS. Como forma de tratar esse problema, sugerimos a utilização de *binding units* para reduzir o espaço de busca.

6.4.1 Ameaças à validade

As ameaças à validade do nosso trabalho podem ser apresentadas da seguinte forma:

- *Validade de construção* refere-se ao estabelecimento de medidas operacionais corretas para os conceitos que estão sendo estudados. As principais construções utilizadas no nosso trabalho estão relacionadas ao estado de risco e a transição entre eles e o modelo de objetivo de qualidade associado a cadas estado. Esse trabalho considera que quanto maior o risco maior deve ser a qualidade exigida pelo sistema.

  Todas as simulações realizadas consideram a existência de três estados de risco de saúde: baixo, moderado e alto. A arquitetura proposta não se restringe a esse número e deve funcionar para mais estados, porém não foi feita nenhuma avaliação sobre esse aspecto. Acreditamos que os 9 cenários construídos para a avaliação da adequação das fórmulas de cálculo de qualidade são o suficiente para filtrar fórmulas não adequadas para o domínio, mas é necessária uma investigação quantitativa para confirmar essa hipótese.

- *Validade Interna* trata em estabelecer uma relação de causalidade em que certas condições são mostrados para levar a outras condições. Embora o processo de criação das fórmulas nas duas estratégias de qualidade tenham sido orientadas por um especialista do domínio, alguns valores foram ajustados sem a intervenção direta deste. Mesmo com a intervenção deste profissional, apenas um especialista do domínio foi envolvido na criação dessas fórmulas.

  Embora os valores para a avaliação do comportamento das fórmulas de qualidade tenham sido geradas por um processo automático, a avaliação de adequação das fórmulas para um estado de risco foi feita de forma qualitativa e não quantitativa.

- *Validade Externa* diz respeito em estabelecer o domínio para que os resultados de um estudo possam ser generalizados. Embora, o nosso estudo concentre-se em um domínio específico de monitoração de dados vitais e uma solução de RSCH, acreditamos que a arquitetura e conceitos apresentados como solução neste estudo possam ser utilizados em outros domínios. A dificuldade de utilizar a estratégia proposta está em definir objetivos de qualidade para o domínio. Para tal definição, é preciso avaliar extensivamente os resultados gerados pelas fórmulas em relação à seleção de configurações da LPS.

  Apenas as estratégias de cálculo SMART e GOAL são utilizadas para avaliar qualidade das configurações. Mesmo assim, não é possível estabelecer quantitativamente uma comparação entre as seleções realizadas por elas.

Os dados simulados foram obtidos a partir de uma única base, onde os dados de monitoração eram de pessoas idosas com histórico de problema cardíaco. Uma validação mais apropriada deve ser realizada para avaliar o comportamento da modelagem proposta e do sistema de monitoração em indivíduos com outros perfis.

- *Confiabilidade* se preocupa em demonstrar que as operações de um estudo podem ser repetidas com os mesmos resultados. Dados a solução, a implementação e descrições dos processos de construção das fórmulas de qualidade, esperamos que repetições do nosso estudo ofereçam resultados similares aos nossos. Em particular, aplicando as fórmulas de qualidade para cada atributo de qualidade conforme explicado, além da atribuição dos mesmos pesos e objetivo de qualidade na LPS. As ferramentas disponibilizadas auxiliam esse processo de reprodução dos resultados.

Capítulo 7

Conclusões e Trabalhos Futuros

Linha de Produtos de Software Dinâmica (LPSD) trata de sistemas que são capazes de modificar o seu próprio comportamento em resposta a mudanças em seu ambiente operacional em tempo de execução. Entretanto, a decisão sobre qual configuração utilizar considerando objetivos de qualidade ainda é um problema em aberto.

Este trabalho detalha uma arquitetura para sistemas dinâmicos cuja tomada de decisão sobre a adequabilidade de uma configuração aos requisitos do contexto pode ser feita via análise dos parâmetros de qualidade de serviço (QoS) dessa configuração. Em especial, a avaliação de qualidade de uma configuração de LPSD deve levar em consideração a possibilidade de inserção e remoção de qualquer tipo de *feature* e ainda tratar a variação dos valores de qualidade das *features* com a mudança do contexto. Por esse motivo, os valores de qualidade de todos os parâmetros observados neste trabalho são calculados por meio de funções que observam a seleção de *features* de uma configuração e o contexto de execução no momento do cálculo. A adaptação do sistema leva em conta os eventos gerados pelo contexto e os possíveis estados do sistema. Essas informações são especificada de forma declarativa na gramática proposta por esse trabalho.

O método proposto por esse trabalho foi aplicada ao contexto de Redes de Sensores para o Corpo Humano (RSCH). Neste contexto, um indivíduo modifica seus dados vitais via identificação de eventos. A ocorrência desses eventos em um certo estado do sistema pode vir a alterar o estado do sistema e o conjunto de requisitos demandados pelo estado de risco. Na RSCH avaliada, nós identificamos e tratamos com sucesso os atributos de qualidade e quantidade de informação, confiabilidade, tempo de vida estimado do sistema e taxa de amostragem dos sensores. Todos esses atributos avaliaram a qualidade da configuração de uma LPS levando em consideração a presença e ausência de *features* e os tipos diferenciados de variabilidade existente no modelo de *features*.

Para compor e avaliar simultaneamente todos os atributos de qualidade de uma configuração, utilizamos as estratégias de cálculo de qualidade S*imple* M*ulti*A*ttribute* R*ating* T*echnique* (SMART) e orientação a objetivos de qualidade (GOAL). A ferramenta de avaliação das estratégias de cálculo desenvolvida por esse trabalho permite que o especialista do domínio ajuste os parâmetros para o correto funcionamento dessas estratégias. No contexto de monitoração dos estados de saúde, essas duas abordagens foram consideradas adequadas pelo especialista do domínio.

A avaliação da arquitetura e modelagem propostas por esse trabalho foi feita a partir da simulação de dados vitais de um indivíduo idoso onde informações tais como pulsação

cardíaca, oxigenação do sangue e temperatura foram coletadas. Dados reais da base do *Boston's Beth Israel Hospital* (BIT-MIT) foram encaminhados ao sistema de monitoração de sinais vitais (MSV) desenvolvido utilizando a modelagem proposta. Observou-se que o sistema altera suas configurações adequadamente do ponto de vista do domínio, onde quanto maior o risco de saúde, maior é a qualidade provida pela configuração selecionada, conforme se esperava.

Realizamos outra simulação com o objetivo de comparar os comportamentos das estratégias de cálculo de multi-atributos de qualidade SMART e GOAL em relação ao número de configurações no contexto de RSCH. Essa simulação foi feita de forma semi-automática com o auxílio de uma ferramenta desenvolvida neste trabalho, QSE-BSN. Por meio da análise dos resultados, observamos que as fórmulas GOAL geraram configurações mais adequadas para o domínio do que as fórmulas SMART. Entretanto, devido à baixa amostragem não é possível garantir esse comportamento para qualquer cenário de monitoração.

Embora tenham executado conforme esperado, as simulações para avaliação da proposta foram feitas em cenários limitados com uma quantidade baixa de amostras. As avaliações foram feita de forma qualitativa e podem ter sido enviesadas pelos pesquisadores e especialista do domínio inseridos no processo de simulação. É necessário realizar um estudo quantitativo para avaliar se o comportamento descrito nas simulações de aplica a qualquer cenário de monitoração. Entretanto, acreditamos que os resultados gerados serão semelhantes aos obtidos por essa pesquisa.

7.1 Trabalhos relacionados

Esta seção discute os trabalhos relevantes relacionados às áres de sistemas sensíveis ao contexto, reconfiguração dinâmica de Linha de Produtos de Software, e análise de qualidade das configurações. Em particular, a Seção 7.1.1 apresenta as principais limitações dos trabalhos atuais para tratar identificação de evento e transição entre os estados do sistema. A Seção 7.1.2 discute sobre outras estratégias de seleção de configurações adequadas para os requisitos de um estado do sistema, e por fim, a Seção 7.1.3 aponta os trabalhos relacionados às estratégias de cálculo de QoS de sistemas com múltiplas dimensões de qualidade.

7.1.1 Adaptação Dinâmica

A reconfiguração dinâmica de produtos vêm sendo amplamente utilizada em vários domínio tais como sistema de monitoração de auto-cura [Garlan and Schmerl, 2002], sensíveis ao contexto [Yau et al., 2002], implantação de software[van der Hoek and Wolf, 2003] e computação ubíqua[Banavar and Bernstein, 2002]. Em sistemas dinâmicos, eventos detectados em virtude de mudanças no contexto operacional podem demandar a reconfiguração de um produto a fim de que ele forneça serviços relevantes para o domínio ou satisfaça requisitos de qualidade, tais como performance [Lee and Kang, 2006].

A especificação de uma gramática para definir os eventos que serão tratados pelo sistema é uma prática comum em sistemas dinamicamente reconfiguráveis [Fernandes et al., 2011, Bencomo et al., 2008, Alferez and Pelechano, 2011, Rosenmüller et al., 2011b]. Entretanto, os eventos definidos por essas gramáticas influenciam a mudança de comportamento do sistema de uma mesma forma durante sua execução. Ou seja, a mudança de

um estado de saúde motivada pela identificação de um evento não é influenciada pelo estado atual do sistema. A adoção dessa estratégia no contexto de RSCH levaria ao mesmo comportamento do sistema sempre que fosse identificado que o indivíduo monitorado está com o batimento cardíaco acima do normal, por exemplo, o que não representa de forma precisa o procedimento de diagnóstico do estado de saúde realizado pelos profissionais da área [Carvalho, 2005].

Para tal representação, este trabalho especifica uma gramática capaz de definir como os eventos são gerados a partir da avaliação do contexto, quais estados que o sistema pode ter e como os eventos associados ao estado atual do sistema influenciam na escolha do novo estado de risco. Apoiado pela estrutura dos autômatos determinísticos finitos (*Deterministic Finite Automaton* - DFA), a instanciação da gramática define como o sistema deve gerenciar a mudança entre os estados do sistema a partir de eventos. Através dessa estratégia, o mesmo evento identificado pelo sistema pode levar a mudanças de comportamento diferentes.

Assim como no contexto de RSCH, outros domínio também utilizam DFAs para descrever os estados de risco do sistema [Bencomo et al., 2008] e como esses estados são influenciados por ocorrências de eventos. Diferentemente do nosso trabalho, os estados determinam qual a nova configuração o sistema deve adotar. Nós, por outro lado, associamos o estado a um objetivo de qualidade. Uma configuração é selecionada no estado do sistema se ela satisfaz o objetivo de qualidade.

7.1.2 Reconfiguração guiada por qualidade

A identificação de eventos pode gerar uma mudança de requisitos no sistema de tal forma que seja necessário modificar a configuração atual para outra que melhor satisfaça às novas exigências. Nesse contexto, diversos trabalhos foram propostos para especificar como selecionar configurações adequadas ao novo estado do sistema [Fernandes et al., 2011, Nascimento et al., 2011, Bencomo et al., 2008, Alferez and Pelechano, 2011, Calinescu et al., 2011, Zeng et al., 2004].

Alguns trabalhos propõem o mapeamento de um conjunto de ações e regras que devem ser executadas e aplicadas à configuração atual a fim de que seja gerada uma nova configuração [Fernandes et al., 2011, Nascimento et al., 2011, Bencomo et al., 2008, Alferez and Pelechano, 2011]. Utilizando essa estratégia, o sistema exige que algum *stakeholder*, no geral o especialista do domínio, especifique todas as mudanças necessárias possíveis na ocorrência de um evento. Carvalho et al. [2010] sugere que a escolha de uma nova configuração seja guiada pelas especificações de contrato, de forma que a nova configuração satisfaça algumas condições sobre serviços e ajustes de funcionamento das *features*.

Outra estratégia adotada para avaliar como as configurações são selecionadas na mudança de requisitos é a avaliação dos parâmetros de qualidade de serviço (QoS) da solução [Brandt et al., 1998, Hu, 2003, Perillo and Heinzelman, 2003, Ko et al., 2008, Loshin, 2006, Martens et al., 2010, Alrifai and Risse, 2010]. Kuusela and Savolainen [2000] propõem uma extensão no modelo de *features* para anotar características de qualidade de uma feature. Siegmund et al. [2011] estima o peso da inserção de cada *feature* na configuração em relação ao parâmetro de qualidade tratado. Assim, uma configuração é escolhida de acordo com a soma dos pesos estimados para as *features* presentes nela e alguns ajustes neste valor com o intúito de eliminar sobreposição das características de qualidade

das *features*. Essas abordagens, entretanto, consideram que o valor de qualidade de uma configuração da LPS não varia com o contexto. Ou seja, independente do estado do contexto, uma configuração prover o mesmo valor para confiabilidade, acurácia, qualidade de informação, e outros parâmetros de qualidade.

A estratégia adotada por esse trabalho para seleção das configurações avalia dinamicamente o valor de qualidade de uma configuração, e não de uma *feature* individualmente. O valor de qualidade é obtido a partir da aplicação de um conjunto de funções matemáticas que compõem os múltiplos parâmetros de qualidade. Uma configuração é considerada adequada se ela satisfaz um conjunto de objetivos de qualidade impostos por um estado [Calinescu et al., 2011, Zeng et al., 2004, Cardellini et al., 2009, Kolesnikov et al., 2013]. No modelo de análise de uma LPS proposto por von Rhein et al. [2013], esse trabalho foca em avaliação individual dos produtos, embora o uso de *binding-unit* favoreça características de avaliação baseada em LPS.

Além de aplicar essa técnica de seleção em uma LPSD, as configurações selecionadas por nossa abordagem não necessariamente são maximais em relação às funções de objetivo de qualidade, diferentemente de Calinescu et al. [2011], Zeng et al. [2004], Cardellini et al. [2009].

No que diz respeito à busca por configurações válidas de uma LPS em tempo de execução, utiliza-se a estratégia de reescrita do Modelo de *Features* em fórmula proposicional [Batory, 2005].

De fato, uma das dificuldades associadas à avaliação de qualidade em LPSs é o grande número de configurações possíveis. Assim como apresentado neste trabalho, o uso de *features* abstratas e *binding units* reduz esse número e aumenta a facilidade de gerenciamento de *features* durante a reconfiguração [Thüm et al., 2011, Lee and Kang, 2006, Siegmund et al., 2012].

Outras formas de agrupamento de features foram propostas para reduzir o espaço de busca [Kolesnikov et al., 2013]. Por meio de um levantamento estatístico, a solução proposta em Kolesnikov et al. [2013] prevê os valores de qualidade de interesse para um conjunto de produtos da LPS. Nosso trabalho não aplica nenhuma estratégia estatística para avaliação de qualidade do produto.

7.1.3 Múltiplos atributos de qualidade

Uma característica importante na avaliação de qualidade de sistemas reais é a necessidade de tratar várias dimensões de qualidade de serviço (QoS) para o mesmo sistema [Alrifai and Risse, 2010]. Em especial, porque essas dimensões muitas vezes apresentam comportamentos contraditórios no sentido de que a maximização de uma implica na minimização de outra [Fernandes et al., 2011].

Diversos trabalhos utilizam a ideia de maximização dos valores de qualidade a fim de selecionar uma configuração ideal da LPS [Alrifai and Risse, 2010, Krishna et al., 2006, Martens et al., 2010]. Porém, eles no geral tratam de maximização de apenas um atributo de qualidade [Ehrgott, 2008, Martens et al., 2010], priorizando por exemplo, a maximização da disponibilidade dos recursos [Younis et al., 2004].

Ao contrário de trabalhos que procuram por configurações do sistema que maximizem o valor de qualidade [Calinescu et al., 2011, Tamiz et al., 1998], o presente trabalho sugere que não é necessário avaliar configurações maximais. Nossa proposta é de que qualquer

configuração que satisfaça os objetivos de qualidade determinados pelo estado do sistema é adequada. As estratégias Simple MultiAttribute Rating Techine (SMART) [Edwards, 1977] e e cálculo orientada a objetivos (Goal-Oriented) [Tamiz et al., 1998] são comparadas por esse trabalho para avaliar satisfação de qualidade.

Alguns trabalhos propõem a avaliação de qualidade baseada em dados estatísticos e análise do impacto no comportamento dos componentes do *software* em relação à seleção de *features* [Kolesnikov et al., 2013, Siegmund et al., 2011]. Entretanto, eles tratam de poucos atributos de qualidade simultaneamente, cenário não aplicável no contexto desse trabalho.

7.2 Trabalhos Futuros

A abordagem proposta deixa em abertos alguns pontos que poderão ser explorados em trabalhos futuros. São eles:

- **Avaliação da modelagem em outros domínios e sistemas sensíveis ao contexto.** A avaliação desse trabalho foi feita em um domínio específico de monitoração dos sinais vitais de um indivíduo. Entretanto, acreditamos que é possível aplicar a gramática, arquitetura e fluxos em outros domínios *sensíveis ao contexto*.
- **Utilizar outras estratégias para cálculo de qualidade.** As duas estratégias para cálculo de qualidade aplicadas nesse trabalho funcionam de forma simplificada, e não são capazes avaliar padrões de comportamento do sistema. Sugerimos que outras técnicas baseada em dados estatísticos/aprendizado sejam avaliadas no processo de cálculo de qualidade com múltiplos atributos, tais como técnicas relacionadas à predição do comportamento do sistema baseado na seleção de *features*.
- **Avaliação da adequação da proposta ao contexto** 1) *Modelos formais*: Modelar formalmente o funcionamento da estratégia de adaptação de forma a garantir que o indivíduo monitorado sempre esteja com uma configuração adequada para o seu estado de risco. 2) *Testes*: aplicar soluções baseadas em teste para verificar se comportamento do sistema satisfaz os requisitos exigidos pelo contexto.
- **Análise quantitativa.** Avaliar quantitativamente o comportamento do sistema MDV em virtude de mudanças no contexto. Nesse sentido, utilizar outras bases de dados reais, com indivíduos em outros contexto. Realizar simulações com múltiplos dados vitais variando simultaneamente, por exemplo, variar a pulsação cardíaca e a oxigenação simultaneamente. Aumentar o número de atributos de qualidade avaliados simultaneamente. Por fim, avaliar o comportamento do sistema utilizando mais fórmulas SMART e GOAL.
- **Busca por configurações que satisfazem os requisitos de qualidade.** Aumentar o modelo de *features* e observar o desempenho do sistema no processo de busca. Sugerir otimizações para reduzir o espaço de busca.
- **Implementação**
  - Automatizar a construção das fórmulas proposicionais. A atual implementação do sistema de monitoração de dados vitais escreve em código o modelo de

features do sistema avaliado. Sugere-se que sejam feitas extensões no sistema para flexibilizar a definição do modelo de *features* utilizado.

– Aprimorar o Configurador de forma que seja possível tratar corretamente *features* espalhadas nos artefatos, ou um único artefato com mais de uma *feature*, utilizando por exemplo, técnicas de orientação a aspectos.

Referências

Germán H. Alferez and Vicente Pelechano. Context-aware autonomous web services in software product lines. In *Proceedings of the 2011 15th International Software Product Line Conference*, SPLC '11, pages 100–109, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-0-7695-4487-8. doi: 10.1109/SPLC.2011.21. URL `http://dx.doi.org/10.1109/SPLC.2011.21`.

Mohammad Alrifai and Thomas Risse. Efficient qos-aware service composition. In Walter Binder and Schahram Dustdar, editors, *Emerging Web Services Technology Volume III*, Whitestein Series in Software Agent Technologies and Autonomic Computing, pages 75–87. Birkhauser Basel, 2010.

A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, Jan 2004. URL `http://doi.ieeecomputersociety.org/10.1109/TDSC.2004.2`.

Heribert Baldus, Karin Klabunde, and Guido Muesch. Reliable set-up of medical body-sensor networks. In *Proc. Wireless Sensor Networks, First European Workshop (EWSN 2004)*, Berlin, Germany, January 2004.

S. Balsamo, A. di Marco, P. Inverardi, and M. Simeoni. Model-based performance prediction in software development: a survey. *Software Engineering, IEEE Transactions on*, 30(5):295 – 310, may 2004.

Guruduth Banavar and Abraham Bernstein. Software infrastructure and design challenges for ubiquitous computing applications. *Commun. ACM*, 45(12):92–96, December 2002. ISSN 0001-0782.

T. M. G. de A. Barbosa, I. G. Sene JR., H. S. Carvalho, A. F. da Rocha, and Nascimento F. A. Arquitetura de software para redes de sensores sem fios: a proeminência do middleware. *Anais do XXV Congresso da Sociedade Brasileira de Computação*, 2005.

Talles Marcelo Gonçalves de Andrade Barbosa. *Uma Arquitetura de Redes de Sensores do Corpo Humano*. PhD thesis, Universidade de Brasília. Faculdade de Tecnologia. Departamento de Engenharia Elétrica, 2008.

Len Bass, Paul Clements, and Rick Kazman. *Software architecture in practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998. ISBN 0-201-19930-0.

Don Batory. Feature Models, Grammars, and Propositional Formulas. In Henk Obbink and Klaus Pohl, editors, *Software Product Lines*, volume 3714 of *Lecture Notes in Computer Science*, chapter 3, pages 7–20. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-28936-4. doi: 10.1007/11554844_3. URL http://dx.doi.org/10.1007/11554844_3.

David Benavides, Sergio Segura, Pablo Trinidad, and Antonio Ruiz-Cortés. A first step towards a framework for the automated analysis of feature models. Technical report, 2006.

David Benavides, Sergio Segura, and Antonio Ruiz-Cortés. Automated analysis of feature models 20 years later: A literature review. *Inf. Syst.*, 35(6):615–636, September 2010. ISSN 0306-4379. doi: 10.1016/j.is.2010.01.001. URL http://dx.doi.org/10.1016/j.is.2010.01.001.

N. Bencomo, P. Sawyer, G. Blair, and P. Grace. Dynamically adaptive systems are product lines too - using model-driven techniques to capture dynamic variability of adaptive systems. In *Proceedings of the 2nd International Workshop on Dynamic Software Product Lines (DSPL 2008)*, 2008.

Rachel King Benny Lo, Surapa Thiemjarus and Guang Zhong Yang. Body sensor network - a wireless sensor platform for pervasive healthcare monitoring. In *3rd International Conference on Pervasive Computing (PERVASIVE 2005)*, pages 77–80, May 2005.

Paulo Borba, Leopoldo Teixeira, and Rohit Gheyi. A theory of software product line refinement. In *Proceedings of the 7th International colloquium conference on Theoretical aspects of computing*, ICTAC'10, pages 15–43, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-14807-7, 978-3-642-14807-1.

S. Brandt, G. Nutt, T. Berk, and J. Mankovich. A dynamic quality of service middleware agent for mediating application resource usage. In *Proceedings of the IEEE Real-Time Systems Symposium*, RTSS '98, pages 307–, Washington, DC, USA, 1998. IEEE Computer Society. ISBN 0-8186-9212-X. URL http://dl.acm.org/citation.cfm?id=827270.829014.

Radu Calinescu, Lars Grunske, Marta Z. Kwiatkowska, Raffaela Mirandola, and Giordano Tamburrelli. Dynamic qos management and optimization in service-based systems. *IEEE Trans. Software Eng.*, 37(3):387–409, 2011.

Valeria Cardellini, Emiliano Casalicchio, Vincenzo Grassi, Francesco Lo Presti, and Raffaela Mirandola. Qos-driven runtime adaptation of service oriented architectures. In *Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, ESEC/FSE '09, pages 131–140, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-001-2.

Hervaldo Sampaio Carvalho. *Data Fusion Implementation in Sensor Networks Applied to Health Monitoring*. PhD thesis, Department of Computer Science Federal University of Minas Gerais, 2005.

Sergio T. Carvalho, Orlando Loques, and Leonardo Murta. Dynamic variability management in product lines: An approach based on architectural contracts. *Software Components, Architectures and Reuse, Brazilian Symposium on*, 0:61–69, 2010. doi: http://doi.ieeecomputersociety.org/10.1109/SBCARS.2010.16.

Carlos Cetina, Pau Giner, Joan Fons, and Vicente Pelechano. Designing and prototyping dynamic software product lines: techniques and guidelines. In *Proceedings of the 14th international conference on Software product lines: going beyond*, SPLC'10, pages 331–345, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-15578-2, 978-3-642-15578-9. URL `http://dl.acm.org/citation.cfm?id=1885639.1885670`.

Michelle Chabrol, Julie Chauvet, Pierre Féniès, and Michel Gourgand. A methodology for process evaluation and activity based costing in health care supply chain. In *Proceedings of the Third international conference on Business Process Management*, BPM'05, pages 375–384, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-32595-6, 978-3-540-32595-6.

Guanling Chen and David Kotz. A survey of context-aware mobile computing research. Technical report, Hanover, NH, USA, 2000.

Octav Chipara, Chenyang Lu, Thomas C. Bailey, and Gruia-Catalin Roman. Reliable clinical monitoring using wireless sensor networks: experiences in a step-down hospital unit. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 155–168, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0344-6. doi: 10.1145/1869983.1869999. URL `http://doi.acm.org/10.1145/1869983.1869999`.

William J. Clancey and Edward H. Shortliffe, editors. *Readings in medical artificial intelligence: the first decade*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984. ISBN 0-201-10854-2.

Paul Clements and Linda Northrop. *Software Product Lines Practices and Patterns*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2011.

Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM. doi: 10.1145/800157.805047. URL `http://doi.acm.org/10.1145/800157.805047`.

Krzysztof Czarnecki and Ulrich W. Eisenecker. *Generative programming: methods, tools, and applications*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000. ISBN 0-201-30977-7.

Stacey D, Bennett CL, Barry MJ, Col NF, Eden KB, Holmes-Rovner M, Llewellyn-Thomas H, Lyddiatt A, Légaré F, and Thomson R. Decision aids for people facing health treatment or screening decisions. In *Cochrane Database Syst Rev*, 2011.

Robert M. Davison, Maris G. Martinsons, and Ned Kock. Principles of canonical action research. *Information Systems Journal*, 14:65–86, 2004.

Talles Marcelo G. de A. Barbosa and Adson Ferreira da Rocha. A smart system to program body sensor networks. In *IEEE Conf. of Intelligent Systems*, pages 168–172. IEEE, 2010.

James G. Dolan. Multi-criteria clinical decision support: A primer on the use of multiple-criteria decision-making methods to promote evidence-based, patient-centered healthcare. In *The Patient: Patient-Centered Outcomes Research*, 2010.

Enrique Dorronzoro, Ana Verónica Medina, Isabel Gómez, José Antonio Gómez, and Manuel Merino Monge. A standard-based body sensor network system proposal. In *IT Revolutions*, volume 82 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 106–115. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-32304-1.

W. Edwards. How to use multiattribute utility measurement for social decision making. *IEEE Transactions on Systems, Man and Cybernetics*, 7(5):326–340, 1977.

Ward Edwards and F.Hutton Barron. Smarts and smarter: Improved simple methods for multiattribute utility measurement. *Organizational Behavior and Human Decision Processes*, 60(3):306 – 325, 1994.

Matthias Ehrgott. Multiobjective optimization. *AI Magazine*, 29(4):47–57, 2008.

Magnus Eriksson, Jürgen Börstler, and Kjell Borg. Managing requirements specifications for product lines Ű An approach and industry case study. *Journal of Systems and Software*, 82(3):435–447, March 2009. ISSN 01641212. doi: 10.1016/j.jss.2008.07.046. URL `http://dx.doi.org/10.1016/j.jss.2008.07.046`.

S. J. Fakih and T. K. Das. Lead: a methodology for learning efficient approaches to medical diagnosis. *Trans. Info. Tech. Biomed.*, 10(2):220–228, April 2006. ISSN 1089-7771. doi: 10.1109/TITB.2005.855538. URL `http://dx.doi.org/10.1109/TITB.2005.855538`.

Paula Fernandes, Cláudia Werner, and Eldínae Teixeira. An approach for feature modeling of context-aware software product line. *j-jucs*, 17(5):807–829, mar 2011.

W Furlong, D Feeny, G Torrance, C Goldsmith, S DePauw, Z Zhu, M Denton, and M Boyle. Multiplicative multi-attribute utility function for the health utilities index mark 3 (hui3) system: A technical report. Centre for Health Economics and Policy Analysis Working Paper Series 1998-11, Centre for Health Economics and Policy Analysis (CHEPA), McMaster University, Hamilton, Canada, 1998.

Yoav Ganzach and Yaacov Schul. The influence of quantity of information and goal framing on decision. *Acta Psychologica*, 89:23–36, 1995.

David Garlan and Bradley Schmerl. Model-based adaptation for self-healing systems. In *Proceedings of the first workshop on Self-healing systems*, WOSS '02, pages 27–32, New York, NY, USA, 2002. ACM. ISBN 1-58113-609-9.

Erann Gat. On three-layer architectures. In *Artificial Intelligence and Mobile Robots*. MIT Press, 1998.

Rohit Gheyi, Tiago Massoni, and Paulo Borba. Algebraic laws for feature models. *J. UCS*, 14(21):3573–3591, 2008.

Carlo Ghezzi and Amir Molzam Sharifloo. Verifying non-functional properties of software product lines: Towards an efficient approach using parametric model checking. In *Proceedings of the 2011 15th International Software Product Line Conference*, SPLC '11, pages 170–174, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-0-7695-4487-8. doi: 10.1109/SPLC.2011.33. URL `http://dx.doi.org/10.1109/SPLC.2011.33`.

Swapna S. Gokhale. Architecture-based software reliability analysis: Overview and limitations. *Dependable and Secure Computing, IEEE Transactions on*, 4(1):32 –40, jan.-march 2007.

Carla P. Gomes, Henry Kautz, Ashish Sabharwal, and Bart Selman. Satisfiability solvers, 2008.

Svein Hallsteinsen, Mike Hinchey, Sooyong Park, and Klaus Schmid. Dynamic software product lines. *IEEE Computer*, pages 93–95, 2008.

M M Hoeper, I Markevych, E Spiekerkoetter, T Welte, and J Niedermeyer. Goal-oriented treatment and combination therapy for pulmonary arterial hypertension. *Eur Respir J*, 26(5):858–63, 2005.

John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006. ISBN 0321455363.

Michael Hu. Web services composition, partition, and quality of service. In *In Distributed System Integration and Reengineering - presented at XML Conference 2003*, 2003.

H. Humphreys and E. T. M. Smyth. Prevalence surveys of healthcare-associated infections: what do they tell us, if anything? *Clinical Microbiology and Infection*, 12(1):2–4, 2006. ISSN 1469-0691. doi: 10.1111/j.1469-0691.2005.01273.x. URL `http://dx.doi.org/10.1111/j.1469-0691.2005.01273.x`.

Myriam M. G. Hunink, Paul P. Glasziou, Joanna E. Siegel, Jane C. Weeks, Joseph S. Pliskin, Arthur S. Elstein, and Milton C. Weinstein. *Decision Making in Health and Medicine: Integrating Evidence and Values*. Cambridge University Press, December 2001. ISBN 0521770297. URL `http://www.worldcat.org/isbn/0521770297`.

IEEE. Ieee standard for a software quality metrics methodology. *IEEE Std 1061-1998*, 1998.

Fraunhofer IESE. Emergency monitoring and prevention - emerge, 2008.

Jeonwoo Lee Youngsung Kim Joonyoung Jung, Kiryong Ha1 and Daeyoung Kim. Wireless body area network in a ubiquitous healthcare system for physiological signal monitoring and health consulting. In *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 2006.

S. M. Kay. *Intuitive Probability and Random Processes Using MATLAB*. Springer-Verlag, New York, first edition, 2006.

Lefteris M. Kirousis, Evangelos Kranakis, Danny Krizanc, and Andrzej Pelc. Power consumption in packet radio networks. *Theor. Comput. Sci.*, 243(1-2):289–305, July 2000. ISSN 0304-3975.

W. A. Knaus, E. A. Draper, D. P. Wagner, and J. E. Zimmerman. APACHE II: a severity of disease classification system. *Critical care medicine*, 13(10):818–829, October 1985. ISSN 0090-3493. URL `http://view.ncbi.nlm.nih.gov/pubmed/3928249`.

Jong Myoung Ko, Chang Ouk Kim, and Ick-Hyun Kwon. Quality-of-service oriented web service composition algorithm and planning architecture. *Journal of Systems and Software*, 81(11):2079–2090, 2008.

Sergiy Kolesnikov, Sven Apel, Norbert Siegmund, Stefan Sobernig, Christian Kästner, and Semah Senkaya. Predicting quality attributes of software product lines using software and network measures and sampling. In *International Workshop on Variability Modelling of Software-intensive Systems (VaMoS)*, 2013. URL `http://wwwiti.cs.uni-magdeburg.de/iti_db/publikationen/ps/auto/KAS+13.pdf`.

Jeff Kramer and Jeff Magee. Self-managed systems: an architectural challenge. In *2007 Future of Software Engineering*, FOSE '07, pages 259–268, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2829-5. doi: 10.1109/FOSE.2007.19. URL `http://dx.doi.org/10.1109/FOSE.2007.19`.

Arvind S. Krishna, Aniruddha S. Gokhale, and Douglas C. Schmidt. Context-specific middleware specialization techniques for optimizing software product-line architectures. *SIGOPS Oper. Syst. Rev.*, 40(4):205–218, April 2006. ISSN 0163-5980.

Charles W. Krueger. Towards a taxonomy for software product lines. In *Software Product Family Engineering*, pages 323–331, 2003.

J. Kuusela and J. Savolainen. Requirements engineering for product families. In *Proceedings of the 22nd international conference on Software engineering*, pages 61–69, New York - NY - USA, 2000. ACM Press.

F Laburthe and N. Jussien. Choco solver, 2012. URL `http://choco.mines-nantes.fr`.

Steven Lanzisera, Ankur M. Mehta, and Kristofer S. J. Pister. Reducing average power in wireless sensor networks through data rate adaptation. In *Proceedings of the 2009 IEEE international conference on Communications*, ICC'09, pages 480–485, Piscataway, NJ, USA, 2009. IEEE Press. ISBN 978-1-4244-3434-3. URL `http://dl.acm.org/citation.cfm?id=1817271.1817361`.

A. M. Law. How to conduct a successful simulation study. In *Winter Simulation Conference*, volume 1, pages 66–70. IEEE, December 2003.

Jaejoon Lee and Kyo C. Kang. A feature-oriented approach to developing dynamically reconfigurable products in product line engineering. In *Proceedings of the 10th International on Software Product Line Conference*, SPLC '06, pages 131–140, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2599-7. URL `http://dl.acm.org/citation.cfm?id=1158337.1158687`.

David Loshin. Monitoring data quality performance using data quality metrics, November 2006. URL `http://www.it.ojp.gov/documents/Informatica_Whitepaper_Monitoring_DQ_Using_Metrics.pdf`.

Anne Martens, Danilo Ardagna, Heiko Koziolek, Raffaela Mirandola, and Ralf Reussner. A hybrid approach for multi-attribute qos optimisation in component based software systems. In *QoSA*, pages 84–101, 2010.

Ar Milenković, Chris Otto, and Emil Jovanov. Wireless sensor networks for personal health monitoring: Issues and an implementation. *Computer Communications (Special issue: Wireless Sensor Networks: Performance, Reliability, Security, and Beyond*, 29: 2521–2533, 2006.

Amanda Sávio Nascimento, Cecília Mary Fischer Rubira, and Jaejoon Lee. An spl approach for adaptive fault tolerance in soa. In *Proceedings of the 15th International Software Product Line Conference, Volume 2*, SPLC '11, pages 15:1–15:8, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0789-5.

Vinicius Nunes, Paula Fernandes, Vander Alves, and Genaina Rodrigues. Variability management of reliability models in software product lines: an expressiveness and scalability analysis. *SBCARS -2012*, 2012.

A.M. O'Connor, H.A. Llewellyn-Thomas, and A.B. Flood. Modifying unwarranted variations in health care: shared decision making using patient decision aids. *Health Aff (Millwood)*, Suppl Variation, 2004.

American Society of Anesthesiologists. *Relative Value Guide 2012: A Guide for Anesthesia Values*. Cambridge University Press, 2002.

D.L. Olson. *Decision Aids for Selection Problems*. Springer Series in Operations Research. Springer Verlag, New York, 1996.

Jack Olson. *Data Quality: The Accuracy Dimension*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002. ISBN 1558608915.

Justin Mazzola Paluska, Hubert Pham, Umar Saif, Chris Terman, and Steve Ward. Reducing configuration overhead with goal-oriented programming. In *Proceedings of the 4th annual IEEE international conference on Pervasive Computing and Communications Workshops*, PERCOMW '06, pages 596–, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2520-2. doi: 10.1109/PERCOMW.2006.116. URL `http://dx.doi.org/10.1109/PERCOMW.2006.116`.

Manish Parashar and Salim Hariri. Autonomic computing: An overview. In *Unconventional Programming Paradigms*, pages 247–259. Springer Verlag, 2005.

Mark Perillo and Wendi Heinzelman. Providing application qos through intelligent sensor management. In *First IEEE International Workshop on Sensor Network Protocols and Applications (SNPA*, 2003.

David B Reuben and Mary E Tinetti. Goal-oriented patient care - an alternative health outcomes paradigm. *N Engl J Med*, 366(9):777–9, 2012. ISSN 1533-4406.

Genaína Nunes Rodrigues, Vander Alves, Renato Silveira, and Luiz A. Laranjeira. Dependability analysis in the ambient assisted living domain: An exploratory case study. *J. Syst. Softw.*, 85(1):112–131, January 2012. ISSN 0164-1212. doi: 10.1016/j.jss.2011.07.037. URL `http://dx.doi.org/10.1016/j.jss.2011.07.037`.

Marko Rosenmüller, Norbert Siegmund, Sven Apel, and Gunter Saake. Flexible feature binding in software product lines. *Automated Software Engg.*, 18(2):163–197, June 2011a. ISSN 0928-8910. doi: 10.1007/s10515-011-0080-5. URL `http://dx.doi.org/10.1007/s10515-011-0080-5`.

Marko Rosenmüller, Norbert Siegmund, Mario Pukall, and Sven Apel. Tailoring dynamic software product lines. pages 3–12, 2011b. doi: 10.1145/2047862.2047866. URL `http://doi.acm.org/10.1145/2047862.2047866`.

Scott Shannon, Andrew Weil, and Bonnie J. Kaplan. Medical Decision Making in Integrative Medicine: Safety, Efficacy, and Patient Preference. *Alternative and Complementary Therapies*, 17(2):84–91, April 2011. ISSN 1076-2809. URL `http://dx.doi.org/10.1089/act.2011.17210`.

Norbert Siegmund, Marko Rosenmuller, Christian Kastner, Paolo G. Giarrusso, Sven Apel, and Sergiy S. Kolesnikov. Scalable prediction of non-functional properties in software product lines. In *Proceedings of the 2011 15th International Software Product Line Conference*, SPLC '11, pages 160–169, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-0-7695-4487-8.

Norbert Siegmund, Marko Rosenmüller, Martin Kuhlemann, Christian Kästner, Sven Apel, and Gunter Saake. Spl conqueror: Toward optimization of non-functional properties in software product lines. *Software Quality Control*, 20(3-4):487–517, September 2012. ISSN 0963-9314.

Paul. Slovic and E. Weber. Perception of risk posed by extreme events. In *Regulation of Toxic Substances and Hazardous Waste (2nd edition)*, 2 edition. Ed. J. S. Applegate, J. G. Laitos, J. M. Gaba, and N. M. Sachs, 2010.

Ahmet Soylu, Patrick De Causmaecker, and Piet Desmet. Context and adaptivity in context-aware pervasive computing environments. In *Proceedings of the 2009 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing*, UIC-ATC '09, pages 94–101, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3737-5. doi: 10.1109/UIC-ATC.2009.11.

Mikael Svahnberg, Jilles van Gurp, and Jan Bosch. A taxonomy of variability realization techniques. *Software: Practice and Experience*, 35(8):705–754, 2005. doi: 10.1002/spe.652. URL `http://dx.doi.org/10.1002/spe.652`.

Ali Maleki Tabar. Smart home care network using sensor fusion and distributed vision-based reasoning. In *In Proc. of VSSN 2006*, pages 145–154. ACM Press, 2006.

Mehrdad Tamiz, Dylan Jones, and Carlos Romero. Goal programming for decision making: An overview of the current state-of-the-art. *European Journal of Operational Research*, 111(3):569 – 581, 1998. ISSN 0377-2217. doi: 10.1016/S0377-2217(97)00317-2. URL `http://www.sciencedirect.com/science/article/pii/S0377221797003172`.

Praveen Thokala and Alejandra Duenas. Multiple criteria decision analysis for health technology assessment. In *Journal of the International Society for Pharmacoeconomics and Outcomes Research*, October 2012.

Thomas Thüm, Christian Kästner, Sebastian Erdweg, and Norbert Siegmund. Abstract features in feature modeling. In *SPLC*, pages 191–200, 2011.

Edward Tsang. Foundations of constraint satisfaction, 1993.

Joel Tsevat, Daniella Duke, Lee Goldman, Marc A. Pfeffer, Gervasio A. Lamas, Jane R. Soukup, Karen M. Kuntz, and Thomas H. Lee. Cost-effectiveness of captopril therapy after myocardial infarction. *Journal of the American College of Cardiology*, 26(4):914–919, 1995. doi: 10.1016/0735-1097(95)00284-1. URL `+http://dx.doi.org/10.1016/0735-1097(95)00284-1`.

André van der Hoek and Alexander L. Wolf. Software release management for component-based software. *Softw. Pract. Exper.*, 33(1):77–98, January 2003. ISSN 0038-0644.

Jilles van Gurp, Jan Bosch, and Mikael Svahnberg. On the notion of variability in software product lines. In *In Proceedings of the Working IEEE/IFIP Conference on Software Architecture (WICSA'01*, pages 45–54. IEEE Computer Society, 2001.

Alexander von Rhein, Sven Apel, Christian Kästner, Thomas Thüm, and Ina Schaefer. The pla model: On the combination of product-line analyses. In *International Workshop on Variability Modelling of Software-intensive Systems (VaMoS)*, New York, NY, USA, January 2013. ACM. To appear.

Yair Wand and Richard Y. Wang. Anchoring data quality dimensions in ontological foundations. *Commun. ACM*, 39(11):86–95, November 1996. ISSN 0001-0782. doi: 10.1145/240455.240479. URL `http://doi.acm.org/10.1145/240455.240479`.

M. Howard Williams, Nick K. Taylor, Ioanna Roussaki, Babak Farshchian, and Kevin Doolin. Developing a pervasive system for a mobile environment, 2006.

J. S. Winer and J. Roth. Avoiding iatrogenic harm to patient and family while discussing goals of care near the end of life. *JOURNAL OF PALLIATIVE MEDICINE - 2006*, 9 (2), 2006.

M.G. Woodbury and P.E. Houghton. Prevalence of pressure ulcers in canadian healthcare settings. *Ostomy Wound Manage*, 50(10):22–4, 26, 28, 30, 32, 34, 36–8, 2004.

Stephen S. Yau, Fariaz Karim, Yu Wang, Bin Wang, and Sandeep K. S. Gupta. Reconfigurable context-sensitive middleware for pervasive computing. *IEEE Pervasive Computing*, 1(3):33–40, July 2002. ISSN 1536-1268.

Paul K. Yoon, Ching-Lai Hwang, and Kwangsun Yoon. *Multiple Attribute Decision Making: An Introduction (Quantitative Applications in the Social Sciences)*. Sage Pubn Inc, March 1995. ISBN 0803954867.

Mohamed Younis, Kemal Akkaya, Mohamed Eltoweissy, and Ashraf Wadaa. On handling qos traffic in wireless sensor networks. In *Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 9 - Volume 9*, HICSS '04, pages 90292.1–, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2056-1. URL `http://dl.acm.org/citation.cfm?id=962757.963320`.

Liangzhao Zeng, Boualem Benatallah, Anne H.H. Ngu, Marlon Dumas, Jayant Kalagnanam, and Henry Chang. Qos-aware middleware for web services composition. *IEEE Trans. Softw. Eng.*, 30(5):311–327, May 2004. ISSN 0098-5589.