UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

BACHARELADO EM ENGENHARIA DE SOFTWARE

LUAN PEREIRA LIMA

VISUALIZANDO A MANUTENIBILIDADE DOS MODELOS DE *FEATURES* EM LINHA DE PRODUTO DE SOFTWARE DINÂMICAS

QUIXADÁ – CEARÁ

2016

LUAN PEREIRA LIMA

VISUALIZANDO A MANUTENIBILIDADE DOS MODELOS DE *FEATURES* EM LINHA DE PRODUTO DE SOFTWARE DINÂMICAS

Monografia apresentada no curso de Engenharia de Software da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Engenharia de Software. Área de concentração: Computação.

Orientadora: Profa. Dra. Carla Ilane Moreira Bezerra

QUIXADÁ – CEARÁ

2016

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

L698v Lima, Luan Pereira.
Visualizando a manutenibilidade dos modelos de features em linha de produto de software dinâmicas / Luan Pereira Lima. – 2016.
117 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Engenharia de Software, Quixadá, 2016.
Orientação: Prof. Dr. Carla Ilane Moreira Bezerra.

1. Engenharia de Software. 2. Engenharia de Linha de Produto de Software. 3. Feature Model. 4. Software-Controle de Qualidade. I. Título.

CDD 005.1

LUAN PEREIRA LIMA

VISUALIZANDO A MANUTENIBILIDADE DOS MODELOS DE *FEATURES* EM LINHA DE PRODUTO DE SOFTWARE DINÂMICAS

Monografia apresentada no curso de Engenharia de Software da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Engenharia de Software. Área de concentração: Computação.

Aprovada em:

BANCA EXAMINADORA

Profa. Dra. Carla Ilane Moreira Bezerra (Orientadora)
Campus Quixadá
Universidade Federal do Ceará – UFC

Prof. M.e. Victor Aguiar Evangelista de Farias
Campus Quixadá
Universidade Federal do Ceará - UFC

Prof. Dr. Emanuel Ferreira Coutinho
Instituto UFC Virtual
Universidade Federal do Ceará - UFC

A minha família.

Principalmente ao meu pai, João Moacir de Lima, que possibilitou meu manter em Quixadá para a realização da graduação.

AGRADECIMENTOS

Ao Prof. Dr. Carla Ilane Moreira Bezerra, pela excelente orientação.

Aos professores participantes da banca examinadora Victor Aguiar Evangelista de Farias e Emanuel Ferreira Coutinho pelo tempo, pelas valiosas colaborações e sugestões.

Aos entrevistados, pelo tempo concedido no questionário.

A minha namorada, pelo carinho e companheirismo.

Aos colegas da turma da graduação, pela amizade e ajuda nas horas difíceis.

“Não é que eu sou tão esperto, é que apenas eu fico com os problemas por mais tempo”

(Albert Einstein)

RESUMO

Linha de Produto de Software (LPS) consiste em uma estratégia para realizar o reuso de forma sistemática para a construção de sistemas com menos esforço desde que estes pertençam a um mesmo domínio de mercado. O modelo de *features* é um importante artefato de uma LPS, pois descreve as características de possíveis sistemas a serem gerados e as relações existentes entre os mesmos, representando todas as similaridades e variabilidades em uma LPS. As LPSs representam variabilidades do produto gerado de forma estática, sua adaptação ocorre em tempo de desenvolvimento, diferente das Linhas de Produtos de Software Dinâmicas (LPSDs), que permitem o desenvolvimento de softwares capazes de se adaptar a diferentes requisitos e restrições diante das mudanças no ambiente. A aplicação de medidas de qualidade é um importante mecanismo para verificar se um modelo de *features* é adequado para gerar um produto consistente, uma vez que um erro ou inconsistência neste artefato pode ser propagado para todos os seus produtos e outros artefatos de uma LPS ou LPSD. A visualização de medidas de qualidade aplicadas em modelos de *features* é um ponto importante para melhorar o entendimento da qualidade de produtos derivados de LPSs e LPSDs. Dessa forma, o trabalho se propõe a estender a ferramenta DyMMer para adicionar um módulo de visualização de dados. A DyMMer possibilita a extração de medidas de manutenibilidade do modelo de *features* de LPSDs. O módulo desenvolvido nesse trabalho é capaz de usar as medidas de qualidade que a ferramenta dispõe e ilustrá-las em visualizações. Além disso, houve adição de uma nova visualização para a ilustração dos modelos de *features*, antes apresentada como uma lista hierarquizada. Para validação das visualizações foram aplicados dois questionários, o primeiro utilizado para classificar as subcaracterísticas e medidas relevantes para a visualização da manutenibilidade do modelo de *features* de LPSDs, o segundo para validar as visualizações desenvolvidas. Os resultados obtidos foram satisfatórios, houve um alto nível de aceitação das visualizações propostas. Como resultados deste trabalho, foram geradas cinco tipos de visualizações, uma nova visualização para a ilustração da árvore de modelos de *features* e a criação de um repositório para armazenamento de modelos de *features*.

Palavras-chave: Engenharia de Linha de Produto de Software. Feature Model. Engenharia de Software. Software-Controle de Qualidade

ABSTRACT

Software Product Line (SPL) consists of a strategy to systematically perform reuse for the construction of systems with less effort as long as they belong to the same market domain. The features model is an important artifact of a LPS, because it describes the characteristics of possible systems to be generated and the relationships existing between them, representing all similarities and variabilities in a SPL. SPLs represent variabilities of the statically generated product, their adaptation occurs at development time, different from Dynamic Software Product Lines (DSPLs), which allow the development of software capable of adapting to different requirements and constraints in the face of changes in environment. The application of quality measures is an important mechanism to verify if a features model is suitable to generate a consistent product, since an error or inconsistency in this artifact can be propagated for all its products and other artifacts of an SPL or DSPL . The visualization of quality measures applied in models of features is an important point to improve the understanding of the quality of products derived from PSLs and DSPLs. In this way, the work proposes to extend the DyMMer tool, to add a data visualization module. DyMMer enables the extraction of maintainability measures from the LPSD features model. The module developed in this work is able to use the quality measures that the tool has and illustrates them in visualizations. In addition, a new visualization was added for the illustration of feature models, previously presented as a hierarchical list. To validate the visualizations, two questionnaires were applied, the first used to classify the subcharacteristics and measures relevant to the visualization of the maintainability of the LPSDs features model, the second to validate developed visualizations. The results were satisfactory, there was a high level of acceptance of the proposed visualizations. As results of this work, five types of visualizations were generated, a new visualization for the illustration of the tree of features models and the creation of a repository for storage of features models.

Keywords: Software Product Line Engineering. Feature Model. Software Engineering. Software-Quality Control

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

LPS	Linha de Produto de Software
LPSDs	Linhas de Produto de Software Dinâmicas
DyMMer	*Dynamic Feature Model Tool Based on Measures*

SUMÁRIO

1 INTRODUÇÃO

Linha de Produto de Software (LPS) tem como objetivo identificar *features* (características) comuns e variáveis entre os possíveis produtos gerados. Ela é uma referência a linhas de produção das indústrias de manufatura de software as quais introduziram uma revolução no processo produtivo, que satisfazem as necessidades de um segmento de mercado particular e são desenvolvidas a partir de um conjunto comum de artefatos de uma maneira predeterminada (CLEMENTS; NORTHROP, 2001).

Segundo Durscki et al. (2004), existem diversas vantagens atribuídas a LPS: redução da participação do software no custo total dos sistemas, redução da mão de obra de desenvolvimento de software, tempo de entrega (*time-to-market*) reduzido, aumento na qualidade dos sistemas desenvolvidos e na satisfação dos clientes.

Um artefato importante na LPS é o modelo de *features*. Este modelo é estruturado em um diagrama em formato de árvore, em que cada nó é a representação de uma possível *feature* que será desenvolvida no sistema. *Feature* é uma característica comum e variável entre sistemas de um determinado domínio de aplicação (BEUCHE; DALGARNO, 2007). Este modelo descreve as características de possíveis sistemas a serem gerados e as relações existentes entre os mesmos, representando todas as similaridades e variabilidades em uma LPS.

Os softwares têm se tornado cada vez mais complexos, exigindo um alto grau de adaptação, dadas as mudanças de requisitos e restrições conforme o ambiente a qual está inserido. Apesar de suas vantagens, a LPS não ajuda muito nesse quesito, pois elas representam as variabilidades do produto gerado de forma estática, enquanto a adaptação ocorre em tempo de desenvolvimento, demandando configurações diferentes para as necessidades dos *Stakeholders* (HALLSTEINSEN et al., 2008).

Dessa forma, surgiram as Linhas de Produto de Software Dinâmicas (LPSDs), permitindo o desenvolvimento de softwares capazes de se adaptar aos diferentes requisitos e restrições diante das mudanças no ambiente. Segundo Hallsteinsen et al. (2008), LPSDs possuem as seguintes propriedades: i) variabilidade dinâmica; ii) capacidade de adaptar-se diversas vezes durante seu tempo de vida; iii) mudança de características variáveis durante sua execução; iv) tratar-se com mudanças inesperadas ou com mudanças de requisitos de usuário, como requisitos funcionais ou de qualidade; v) ser sensível ao contexto (opcional); e vi) ter propriedades autoadaptativas (opcional).

Pode-se avaliar a qualidade dos produtos gerados pelas LPSs e LPSDs a partir de

medidas aplicadas aos modelos de *features*, que são os principais artefatos para a linha de produtos (BEZERRA; ANDRADE; MONTEIRO, 2015). Bezerra, Andrade e Monteiro (2015) apresentam um estudo que relata o emprego bem sucedido de medidas na avaliação desses modelos.

A visualização das medidas dos modelos de *features* é um ponto importante para melhorar o entendimento da qualidade do modelo de *features*. Formas de representar esses modelos foram discutidas, como apresentado em Urli et al. (2015), uma forma de visualização para decomposição de modelo de *features* complexas que representa restrições internas e externas do modelo. A visualização das medidas, definidas por Bezerra, Andrade e Monteiro (2015) de forma integrada á visualização dos modelos de *features*, é crucial para se obter uma melhor visão dos resultados das medidas de um conjunto de modelos de *features*.

Algumas das medidas que podem ser utilizadas para apoiar a avaliação da qualidade dos modelos de *features* são implementadas na ferramenta DyMMer (BEZERRA et al., 2016b). DyMMer é uma ferramenta *desktop* desenvolvida com o intuito de extrair a coleta de medidas relacionadas aos modelos de *features* de LPSDs. Ela disponibiliza funcionalidades de visualização de um modelo com ou sem contexto, edição e exportação de todos os resultados das medidas aplicadas em um modelo de *features* (BEZERRA et al., 2016b).

Este trabalho tem como objetivo definir visualizações de medidas para modelos de *features* em Linha de Produto de Software Dinâmicas. Com a execução deste trabalho espera-se alcançar três objetivos específicos, que são: (i) analisar e adaptar a ferramenta DyMMer, para adicionar a visualização das medidas dos modelos de *features* e alterar a visualização dos modelos de *features* apresentados pela ferramenta para um novo modelo; (ii) criar um repositório para armazenar as informações dos modelos de *features*; e (iii) validar o módulo de visualização implementado. Como público alvo deste trabalho, identifica-se engenheiros de domínio, apoiando a avaliação da qualidade do modelo de *features* de forma eficiente.

Este trabalho está organizado em outras seis seções, além desta seção de introdução. A Seção 2 especifica os conceitos que fundamentam este trabalho. A Seção 3 apresenta os trabalhos que tratam sobre visualização de dados e a ferramenta estendida. Na Seção 4 são apresentados os passos executados para a realização deste trabalho. A Seção 5 apresenta o processo de visualização das medidas e as validações realizadas no trabalho desenvolvido, focando a forma que a validação foi realizada, os resultados obtidos e a discussão realizada sobre os mesmos. A Seção 6 apresenta o desenvolvimento realizado no trabalho desenvolvido, focando

a lista de requisitos que serão supridos, a arquitetura da ferramenta estendida, o desenvolvimento do repositório de modelos de *features* da DyMMer, e o desenvolvimento de funcionalidades adicionadas na ferramenta estendida. Finalmente, a Seção 7 apresenta algumas considerações, incluindo as contribuições e as limitações deste trabalho, além das possibilidades de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção serão apresentados os conceitos que fundamentam o trabalho proposto: (i) Linha de Produto de Software Dinâmica; (ii) Gerenciamento de Variabilidade; (iii) Medidas para a Avaliação do Modelo de *Features* de LPSDs; (iv) Visualização do Modelo de *Features*; e (v) a Ferramenta DyMMer.

2.1 Linha de Produto de Software Dinâmica

Com o aumento da capacidade da produção das indústrias, os softwares se tornaram importantes para automatizar grande parte da produção dos mesmos. Com uma produção cada vez maior, essas indústrias tiveram a necessidade de se ter softwares com alta qualidade, e dessa forma os softwares se tornaram cada vez mais complexos (BORBA, 2009). Com a LPS isso tem se tornado possível, com a construção de famílias de softwares em vez do desenvolvimento de produtos separados (BORBA, 2009).

De acordo com Clements e Northrop (2001), LPS é um conjunto de sistemas que usam software intensivamente, compartilhando um conjunto de características comuns e gerenciadas, que satisfazem as necessidades de um seguimento em particular de mercado ou missão. LPSs são desenvolvidas a partir de um conjunto comum de ativos principais e de uma forma preestabelecida.

Os benefícios das LPSs para o negócio são bem amplos: alta qualidade, baixo *time-to-market*, uso efetivo de arquivos limitados, produtos alinhados, baixo custo de produção, baixo custo de manutenção e customização em massa são exemplos desses benefícios, fazendo com que a eficiência e produtividade sejam aumentadas (NORTHROP; CLEMENTS, 2001).

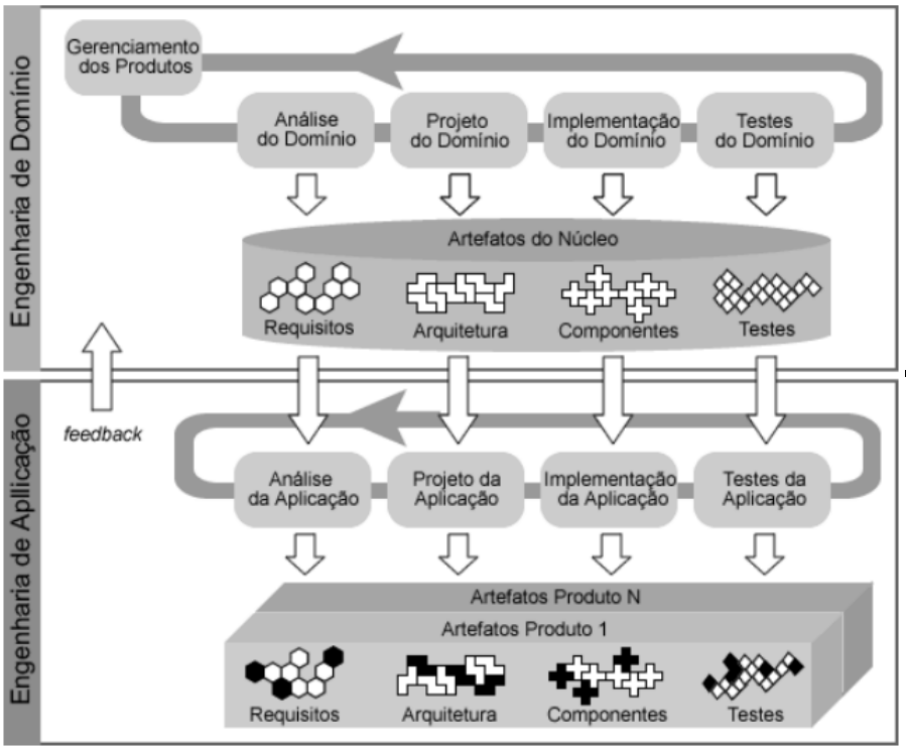
Dois processos são definidos para a engenharia da LPS: a Engenharia de Domínio e de Aplicação. A primeira envolve a fase de desenvolvimento, onde são estabelecidos as similaridades e variabilidades da LPS; a segunda é responsável por derivar aplicações de uma linha de produtos. A gestão técnica e organizacional faz a supervisão destes dois processos (BORBA, 2009).

Da mesma forma como a atividade de gestão técnica citada anteriormente, também existem as atividades de desenvolvimento de *core assets* e desenvolvimento de produtos. A primeira pertence à Engenharia de Domínio e a segunda,0 a Engenharia da Aplicação.

A Figura 1 ilustra o ciclo de vida da LPS, onde são mostrados os subprocessos dos

processos de Engenharia de Domínio e da Aplicação e suas atividades relacionadas.

Figura 1 – Ciclo de vida da LPS.



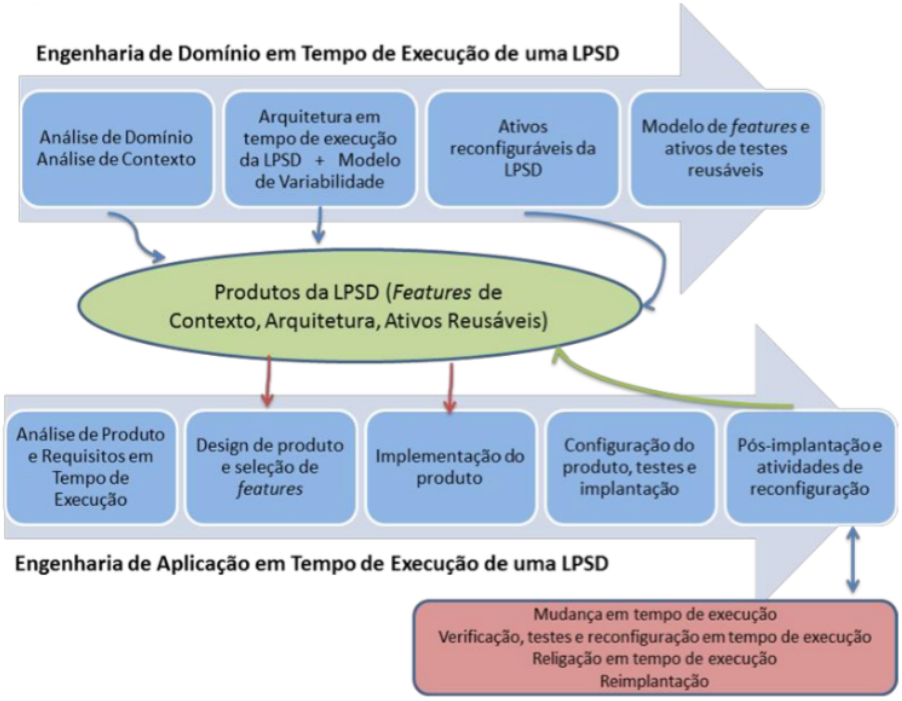
Fonte: Adaptado de Pohl, Böckle e Linden (2005).

Ainda na Figura 1, temos a Engenharia de Domínio, que é o processo onde os pontos comuns e variáveis da LPS são definidos; dentro deste processo temos os subprocessos: (i) Gerenciamento de Produto; (ii) Engenharia de Requisitos de Domínio; (iii) Projeto de Domínio; Realização de Domínio; e (iv) Testes de Domínio. Temos também a engenharia da aplicação, que é o processo em que a partir da reutilização dos artefatos de domínio e variabilidades da LPS pode-se construir sistemas. Dentro deste segundo processo temos os subprocessos: (i) Engenharia de Requisitos de Aplicação; (ii) Projeto de Aplicação; (iii) Realização de Aplicação e (iv) Testes de Aplicação.

Com o passar do tempo, os softwares tem se tornado cada vez mais adaptativos ao ambiente ao qual estão inseridos, isso se tornou um problema para as LPS, pois elas são estáticas e a sua adaptação ocorre apenas em tempo de desenvolvimento, fazendo com que se necessite de configurações diferentes para as necessidades do *stakeholders*. Foi por esse motivo que surgiram as Linhas de Produtos de Software Dinâmicas, elas se adaptam ao ambiente de acordo com os diferentes requisitos e restrições diante das mudanças que ocorrem no mesmo (HALLSTEINSEN et al., 2008).

A Figura 2 ilustra o ciclo de vida da LPSD, onde são apresentados os subprocessos da Engenharia de Domínio em Tempo de Execução de uma LPSD e Engenharia de Aplicação em tempo de execução de uma LPSD.

Figura 2 – Ciclo de vida da LPSD.



Fonte: Adaptado de Capilla et al. (2014).

Ainda na Figura 2, na Engenharia de Domínio em Tempo de Execução de uma LPSD, temos os subprocessos: (i) Análise de Domínio e Análise de contexto; (ii) Arquitetura em tempo de execução da LPSD + Modelo de Variabilidade; (iii) Ativos reconfiguráveis da LPSD; e (iv) Modelo de *features* e ativos de testes reusáveis. Na Engenharia de Aplicação em Tempo de Execução de uma LPSD, temos os subprocessos: (i) Análise de Produto e Requisitos em Tempo de Execução; (ii) Design de produto e seleção de *features*; (iii) implementação do produto; (iv) Configuração do produto, testes e implementação; (v) Pós-implementação e atividades de reconfiguração; e (vi) Mudança em tempo de execução, Verificação, testes e reconfiguração em tempo de execução, Religação em tempo de execução e Reimplantação. Essas duas engenharias geram os produtos da LPSD, tais como, *Features* de Contexto, Arquitetura e Artigos Reusáveis.

Na fase da Engenharia de Domínio em tempo de execução, a atividade de análise de domínio é realizada com a inclusão da análise de contexto, esta atividade é responsável por identificar as *features* de contexto que são relevantes para os produtos de LPSDs, fazendo com que seja possível identificar quais *features* devem ser adicionada à nova configuração, quando

um produto altera seu comportamento (CAPILLA et al., 2014). É dessa arquitetura que temos a implementação dos ativos reconfiguráveis.

Na fase de engenharia de aplicação em tempo de execução, temos o desenvolvimento dos produtos de uma LPSD a partir dos requisitos estáticos e em tempo de execução. Os produtos são testados, configurados e implantados e, se uma nova configuração é necessária em tempo de execução, o ciclo de vida de LPSD suporta as tarefas de pós-implantação e de reconfiguração para lidar com as novas mudanças (CAPILLA et al., 2014).

Pode-se citar domínios de aplicação específicos onde as LPSD podem servir bem, tais como: Sistemas Orientados a Serviços; Sistemas Mobile; Ecossistemas; e Sistemas Autônomos e Auto-adaptativos (CAPILLA et al., 2014).

As relações existentes entre LPS e LPSD são encontradas no Quadro 1.

Quadro 1 – Relações entre LPS e LPSD.

Linhas de Produtos Clássicas	**Linhas de Produto de Software Dinâmicas**
Gerenciamento de variabilidade descreve possíveis formas diferentes de sistemas.	Gerenciamento de variabilidade descreve diferentes adaptações do sistema.
A arquitetura da LPS provê um framework comum para um conjunto induvidual de produtos.	A arquitetura LPSD é uma única arquitetura de sistema, que fornece uma base para todas as adaptações possíveis do sistema.
O escopo de negócio identifica o mercado comum para o conjunto de produtos.	A adaptabilidade do escopo identifica a faixa de adaptações suportadas pelo LPSD.
A abordagem de ciclo de vida descreve dois ciclos de vida de engenharia, a de domínio e a de aplicação.	A LPSD conta com o ciclo de vida da engenharia, que visa o desenvolvimento sistemático do produto, e o ciclo de vida de uso, que visa explorar a adaptabilidade em uso.

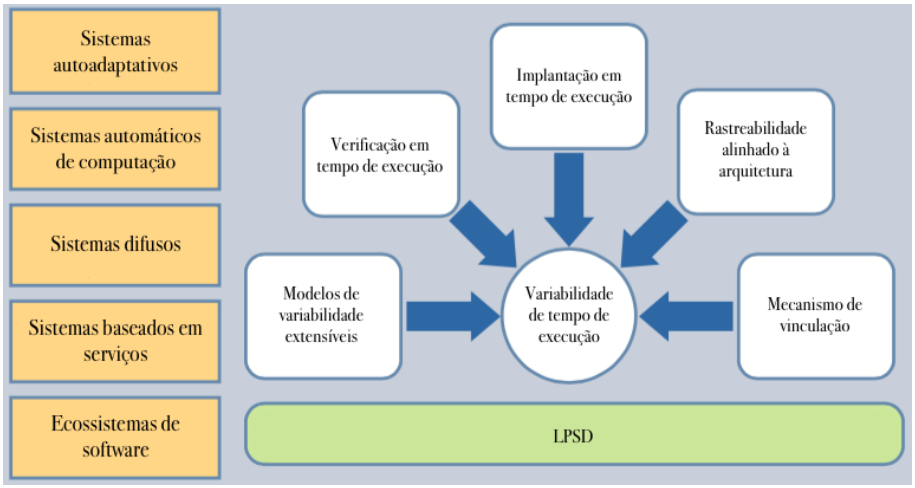
Fonte: Adaptado de Hinchey, Park e Schmid (2012).

2.2 Gerenciamento da Variabilidade

Variabilidade é a capacidade de mudança ou customização de um sistema (HINCHEY; PARK; SCHMID, 2012). Existem diferenças entre a variabilidade das LPSs e LPSDs, porque a LPS trabalha de forma estática e a LPSD de forma dinâmica. Os modelos de *features* estáticos não podem modificar as características do sistema, nem adaptar-se automaticamente em tempo de execução. Nas LPSDs, por outro lado, pode-se adicionar, remover, ou modificar *features* em tempo de execução e ,assim, o sistema pode sofrer modificações (BENCOMO; HALLSTEINSEN; ALMEIDA, 2012). Na Figura 3 é ilustrado os passos necessários para a transição de uma LPS para uma LPSD, visto que existem as diferenças entre as duas. As atividades apresentadas do lado direito da imagem, acima do símbolo que contém "LPSD", precisam ser implantadas para uma correta transição. Do lado esquerdo da

imagem, é ilustrado os tipos de sistemas e mecanismos dinâmicos que fazem parte de uma LPSD.

Figura 3 – Mecanismos de variabilidade em tempo de execução que são necessários para transitar de uma LPS para LPSD.



Fonte: Adaptado de Capilla e Bosch (2011).

Para que as LPSDs possam garantir a vinculação do modelo de *features* em tempo de execução, é necessário um mecanismo de verificação em tempo de execução, garantindo que mudanças de adição ou remoção de *features* não gerem reconfigurações inconsistentes do modelo de *features*. Uma LPSD tem como objetivo: (i) dar suporte a ativação e desativação de *features*; (ii) realizar a vinculação das *features* e (iii) controlar as adaptações dos produtos (CAPILLA; BOSCH, 2011).

Enquanto as LPSs possuem um novo gerenciamento da variabilidade para cada diferentes formas que um sistema pode adotar, a LPSDs tem um único gerenciamento da variabilidade contemplando as diferentes adaptações que um sistema pode ter. Isso se deve ao fato da LPS ser estática, com as modificações realizadas em tempo de desenvolvimento, e a LPSD ser dinâmica, com modificações realizadas em tempo de execução.

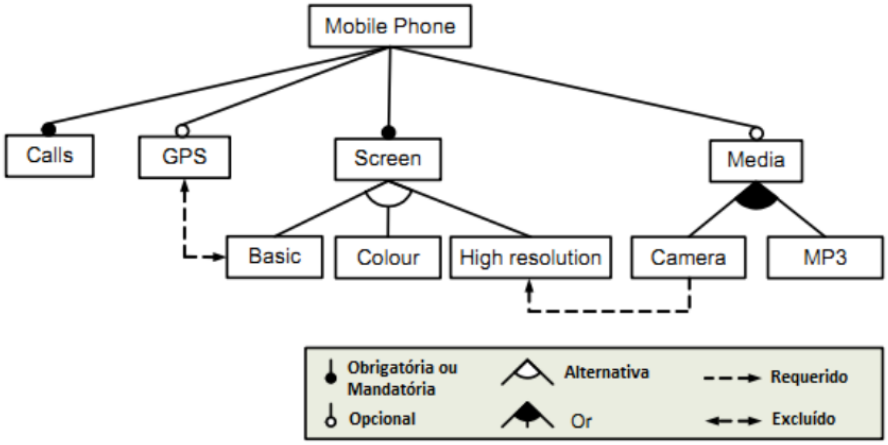
O modelo de *features* é um dos principais artefatos utilizados para representar variabilidades, sendo uma abordagem visual comumente utilizada (PLEUSS; BOTTERWECK, 2012).

2.2.1 Modelo de Features

Como citado na Seção 1, o modelo de *features* é um artefato importante em LPS, pois este modelo descreve as características de possíveis produtos a serem gerados e as relações

existentes entre as mesmas, representando todas as similaridades e variabilidades em uma LPS (BEUCHE; DALGARNO, 2007). Modelos de *features* apresentam uma excelente forma de representar as variabilidades de um sistema. O conceito de *feature* simplifica o trabalho com os requisitos, porque ele pode ser usado para agrupar um conjunto de requisitos relacionados, as *features* são uma forma de abstraí-los, dessa forma existe uma relação direta com os mesmos (SILVA et al., 2011). Um exemplo de modelo de *features* pode ser visto na Figura 4.

Figura 4 – Exemplo de Modelo de *Features* simples.



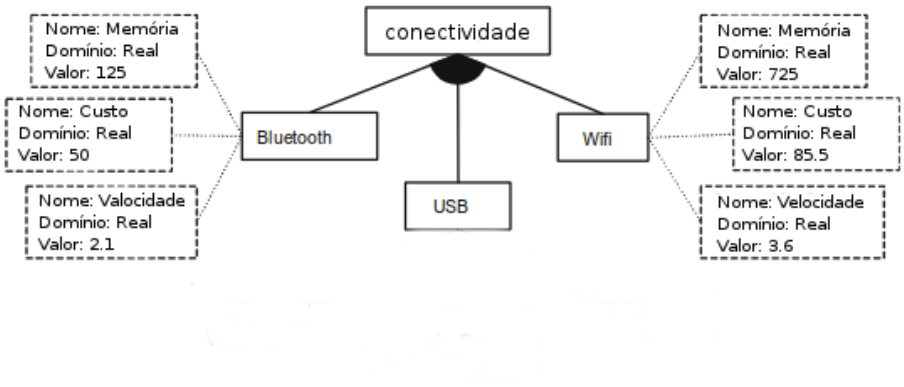
Fonte: Adaptado de Benavides, Segura e Ruiz-Cortés (2010).

No modelo de *features* há diferentes representações para mostrar a relação entre as *features* do modelo, dentre elas: Modelo de *features* básica (*Basic features models*), Modelo de *features* baseada em cardinalidade (*Cardinality-based feature models*) e Modelo de *features* estendido (*Extended feature models*). Para a primeira, temos as seguintes relações entre as *features*: (i) obrigatórias, as *features* que são obrigatórias para o modelo em questão; (ii) opcionais, as *features* que podem ou não estar selecionadas no modelo em questão; (iii) alternativas, as *features* que se encontram em conjunto com outras, e que apenas uma dessas pode ser selecionada; (iv) ou, as *features* que se encontram em conjunto com outras, e que uma ou mais dessas podem ser selecionadas; (v) requerimento, a relação de requerimento entre uma *feature* e outra(s); e (vi) exclusão, a relação de exclusão entre uma *feature* e outra(s).

Para a segunda, baseada em cardinalidade, temos relações adicionais, tais como: (i) cardinalidade da *feature*, é usado para indicar o número de ocorrências da *feature* em um produto, isso através de um intervalo [i..n]; e (ii) cardinalidade de grupo de *features*, é usado para limitar o número de *features* filhas, caso um pai seja selecionado, isso também através de um intervalo $[i..n]$. Para a última, estendida, temos a adição de informação nas *features*, chamadas de

atributos, que especificam informações adicionais sobre essas *features*, como visto na Figura 5.

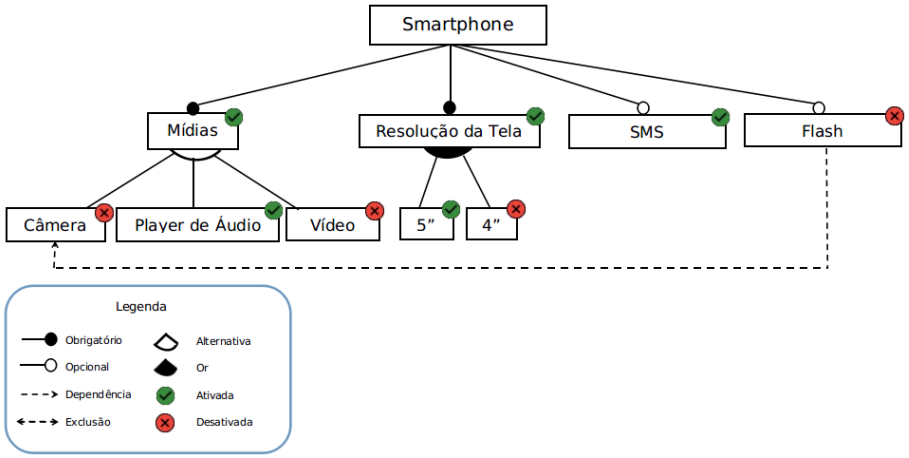
Figura 5 – Exemplo de Modelo de *Features* estendido.



Fonte: Adaptado de Benavides, Segura e Ruiz-Cortés (2010).

Diferente do modelo de *features* de uma LPS tradicional, temos a LPSD, representado na Figura 6, que ilustra um modelo de *features* da LPSD de um sistema de casa inteligente. Além das diferentes representações mostradas anteriormente do modelo de *features* da LPS, temos no modelo de *features* da LPSD as *features* desativadas (*Inactive*) e ativadas (*Active*), que formam as variações de contextos.

Figura 6 – Exemplo de Modelo de *Features* da LPSD.



Fonte: Adaptado de Capilla et al. (2014).

A Figura 6 ilustra um exemplo de modelo de *features* de uma LPSD, são representadas: (i) *features* obrigatórias, "Mídias"e "Resolução de Tela"; (ii) *features* opcionais, "SMS"e "Flash"; (iii) *features* alternativas, "Câmera", "Player de Áudio"e "Vídeo"; (iv) *features* Or, "5 e "4; (v) *features* ativadas, todas as *features* marcadas com o símbolo Ativada; (vi) *features* desativadas, todas as *features* marcadas com o símbolo Desativada; (vii) restrições entre

as *features*, todas as *features* que apresentam ligações com os símbolos de Dependência ou Exclusão. No modelo de *features* de uma LPSD, os contextos são representados por diferentes seleções que são realizadas nas *features* de todo o modelo, a forma como as *features* da Figura 6 estão selecionadas representa um contexto. De acordo com modificações no ambiente, o contexto se modificaria em tempo de execução, modificando as seleções das *features* da árvore.

2.3 Medidas para Avaliação do Modelo de *Features* de LPSs

As medidas para a avaliação dos modelos de *features* são importantes, uma vez que provêm uma forma de se verificar a qualidade desses modelos. A avaliação da qualidade é essencial para verificar se um modelo de *features* é adequado para gerar um produto conveniente para uso, pois um pequeno erro ou inconsistência em um artefato LPS pode ser propagada para todos os seus produtos (BEZERRA; ANDRADE; MONTEIRO, 2015). É mais complexo avaliar a qualidade em uma LPS do que em um software tradicional, pois: (i) diferentes produtos podem ser derivados da LPS; e (ii) diferentes produtos na LPS podem requerer diferentes níveis de qualidade (BEZERRA; ANDRADE; MONTEIRO, 2015).

No Quadro 2, a coluna "Subcaracterísticas", contém os atributos que se referem à qualidade do modelo observado e a coluna "Medidas"relaciona as medidas utilizadas para medir essas Subcaracterísticas. Todas essas medidas são realizadas usando fórmulas, elas podem ser observadas no Quadro 3.

Em Bezerra, Andrade e Monteiro (2015) foram realizadas medidas em alguns modelos de *features*, os autores definem cada um deles: (i) *Strategy Mobile Game*, LPS dinâmico de um jogo de estratégia *multiplayer* para dispositivos móveis; (ii) *Mobile Guide* (*Mobiline*), LPS para dispositivos móveis e aplicações de *context-aware* em que o modelo de *features* é específico para o domínio guia de visitante para dispositivos móveis; e (iii) *Mobile Media 2*, LPS para aplicações que manipulam fotos, músicas e vídeos em dispositivos móveis, como telefones celulares.

Um estudo realizado em Bezerra et al. (2016a) apresentam a evolução de versões de modelos de *features*, e como uma mudança de versão pode modificar as medidas. Isso mostra que a qualidade deve ser verificada a cada modificação de modelo e não apenas no início. Nesse mesmo estudo é mostrado o repositório de modelo de *features*, S.P.L.O.T. (MENDONCA; BRANCO; COWAN, 2009), que foi utilizado para a busca dos modelos, ele é o mesmo repositório que será usado para a busca de modelos para o trabalho que será desenvolvido.

No trabalho desenvolvido os resultados destas medidas geradas de forma automática pela ferramenta DyMMer, representadas no Quadro 2, são apresentadas por meio de visualizações, na tentativa de facilitar sua leitura e entendimento.

Quadro 2 – Medidas de qualidade para suportar a avaliação da manutenibilidade do modelo de *features*.

Característica	**Subcaracterísticas**	**Medidas**
Manutenibilidade	**Analisabilidade**	Número de *features* folhas (NLeaf)
	Complexidade Cognitiva / Entendibilidade	Complexidade Cognitiva do Modelo de *Features* (CogC)
	Extensibilidade	Extensibilidade da Feature (FEX)
	Flexibilidade	Flexibilidade da Configuração (FoC)
	Modularidade	*Features* Dependentes Cíclicas Únicas (SCDF); *Features* Dependentes Cíclicas Múltiplas (MCDF)
	Complexidade Estrutural	Número de *Features* (NF); Número de *Features* Mandatórias (NM); Número de *Features* Top (NTop); Profundidade da Árvore (DT Max, DT Mean and DT Median); Complexidade Ciclomática (CyC); Complexidade Composta (ComC); Restrições *Cross-tree* (CTC); Restrições Variáveis *Cross-tree* (CTCV); Taxa de Conectividade do Grafo (RCon); Densidade do Grafo (RDen); Coeficiente de Densidade da Conectividade (CoC); Número de *Features* Agrupadas (NGF); Número de Filhos por *Feature* (BF Max)
	Variabilidade Estática	Número de *Features* Opcionais (NO); *Single Hotspot Features* (SHoF); *Multiple Hotspot Features* (MHoF); *Rigid Nohotspot Features* (RNoF); Número de *Features* Variáveis (NVF); Número de Configurações Válidas (NVC); Taxa de Variabilidade (RoV); Número de Grupos Or (NGOr); Número de Grupos XOr (NGXOr); Taxa de *Features* Or (ROr); Taxa de *Features* XOr (RXOr)
	Variabilidade Dinâmica	Número de Contextos (NC); Número de *Features* Desativadas (NDF); Número de *Features* Ativadas (NAF); *Features* de Contextos em Restrições (CFC); Número de Restrições de Contexto (NCC); Número de *Features* de Contextos (CF); Número de *Features* Ativadas por Contexto (AFCA); Número de *Features* Desativadas por Contexto (DFCA)

Fonte: Elaborada pelo autor

Quadro 3 – Medidas e suas propriedades

Acrônimo	**Nome da Medida**	**Função da Medida**
NF	Número de *Features*	Número de *features* no modelo
NO	*Features* Opcionais	Número de *features* opcionais no modelo
NM	*Features* Obrigatórias	Número de *features* obrigatórias no modelo
NTop	Número de *Features* Top	Número de descendentes do root
NLeaf	*Features* Leaf	Número de *features* com nenhuma *feature* filha ou especialização
DT Max	Profundidade máxima da árvore	Número de *features* até a profundidade máxima da árvore
DT Mean	Profundidade da metade da árvore	Número de *features* até a metade da profundidade da árvore
DT Median	Profundidade mediana da árvore	Número de *features* até a mediana da profundidade da árvore
CogC	Complexidade cognitiva	Número de pontos variantes
FEX	Extensibilidade da *feature*	NLeaf + SCDF + MCDF
FoC	Flexibilidade de configuração	NO/NF
SCDF	Ciclos individuais de *features* dependentes	∑(#Participantes das restrições das *features* e participantes filhas de variantes de pontos com cardinalidade [1..1])
MCDF	Ciclos multiplos de *features* dependentes	∑(#Participantes das restrições das *features* e participantes filhas de variantes de pontos com cardinalidade [1..*])
CyC	Complexidade ciclomática	Número de restrições de integridade
ComC	Complexidade composta	NF^2 + (NM^2 + 2*NOr^2 + 3*$NXOr^2$ + 3*NGF^2 + 3*R^2)/9 R = NGF + CyC
NGF	*Features* agrupadas	Número de *features* com pelo menos um filho
CTCV	Restrições variáveis na árvore	Número de variáveis distintas na árvore de restrições
CTCR	Taxa de restrições variáveis na árvore	NFRI*/NF *Número de *features* envolvidas na integridade das restrições do modelo
RCon	Conectividade da taxa de dependência do grafo	Taxa de *features* que referenciam outras *features* (exceto os pais) nas restrições
CoC	Coeficiente da densidade de conectividade	(Número de arestas)/NF
NVF	*Features* variáveis	NA + NO
SHoF	Ponto de acesso de *feature* único	Número de *features* filhas de pontos variantes com cardinalidade [1..1]
MHoF	Ponto de acesso de *feature* múltiplos	Número de *features* filhas de pontos variantes com cardinalidade [1..*]
RNoF	Sem ponto de acesso rígido de *feature*	Número de *features* não filhas de pontos variantes
RoV	Taxa de variabilidade	∑(Média do número de filhas de nós)
NVC	Configurações válidas	Número de possíveis e válidas configurações do modelo de *features*
BF Max	Fator de ramificação máximo	Número máximo de filhas por *features*
NGOr	Número de grupos Or	Número de pontos variantes com relacionamentos Or
NGXOr	Número de grupos XOr	Número de pontos variantes com relacionamentos XOr
ROr	Taxa de *features* do tipo Or	Taxa de *features* filhas de um relacionamento Or
RXOr	Taxa de *features* XOr	Taxa de *features* filhas de um relacionamento XOr

Fonte: Adaptado de Bezerra, Andrade e Monteiro (2015).

2.4 Visualizações de Dados

As medidas representadas na seção anterior precisam ser representadas por meios de visualizações, dessa forma a visualização de dados torna-se um campo importante. As técnicas de visualização de informações procuram representar graficamente dados de um determinado domínio. A representação visual gerada explora a capacidade de percepção do homem, a partir das relações espaciais exibidas, fazendo com que o mesmo interprete e compreenda as informações apresentadas e deduza novos conhecimentos a partir dela (FREITAS et al., 2001).

Para o processo de visualização é necessário a determinação de qual técnica deve ser empregada em uma determinada situação. Para auxiliar na escolha da técnica existem as classificações de técnicas de visualizações por tipo de dados e por tarefas. Existem técnicas unidimensionais (1D), temporais, bidimensionais (2D), tridimensionais (3D), e multidimensionais (nD), com suporte a tarefas de obtenção de visão geral, visão detalhada, filtragem, identificação de relacionamentos e extração de informações diversas (FREITAS et al., 2001).

Em Freitas et al. (2001) são relatadas três virtudes que são base para mostrar que a complexidade da criação de sistemas para visualização é alta: (i) Necessidade de criação de uma metáfora visual que permita codificar visualmente o conjunto de informações com o grau de fidelidade necessário à aplicação; (ii) Mecanismos de interação necessários para manipular os frequentemente volumosos e/ou complexos conjuntos de dados; e (iii) Frequente necessidade de implementar algoritmos geométricos complexos tanto para a criação da representação visual como para sua manipulação.

Os dados podem ser caracterizados com critérios, apresentados no Quadro 4, e dessa forma entrar em tipos de classes de representações, apresentadas no Quadro 5.

Quadro 4 – Caracterização de dados.

Critério	Classes	Exemplos
Classe de informação	Categoria, Escalar, Vetorial, Tensorial e Relacionamento	Gênero, Temperatura, Grandezas físicas associadas, Dinâmica de fluidos e Link em um hiperdocumento
Tipos dos valores	Alfanumérico, Número (inteiro, real), Simbólico	Gênero, Temperatura, Link em um hiperdocumento
Natureza do domínio	Discreto, Contínuo e Contínuo-discretizado	Marcas de automóveis, Superfície de um terreno e Anos (tempo discretizado)
Dimensão do domínio	1D, 2D, 3D e nD	Fenômeno ocorrendo no tempo, Superfície de um terreno, Volume de dados médicos e Dados de uma população

Fonte: Freitas et al. (2001)

Quadro 5 – Classes de representações visuais.

Classe	Tipo	Utilização
Gráficos, 2D e 3D	Pontos, Circulares, Linhas, Barras e Superfícies (para 3D)	Representação da distribuição dos elementos no espaço domínio, representação da dependência/correlação entre atributos
Ícones, Glifos, Objetos geométricos	Elementos geométricos 2D ou 3D diversos	Representação de entidades num contexto, representação de grupos de atributos de diversos tipos.
Mapas	Pseudo cores, Linhas, Superfícies, Ícones, Símbolos diversos	Representação de campos escalares ou de categorias. Representação de linhas de contorno de regiões. Idem, no espaço 3D. Representação de grupos de atributos (categoriais, escalares, vetoriais, tensoriais).
Diagramas	Nodos e Arestas	Representação de relacionamentos diversos: É-um, É-parte-de, Comunicação, Sequência, Referência, etc.

Fonte: Freitas et al. (2001)

2.5 Visualizações do Modelo de *Features*

A modelagem do modelo de *features* para visualização não é uma tarefa trivial. Na indústria, o número de *features* de um modelo pode ser bastante extenso, o que faz com que o modelo gerado para ser visualizado seja de difícil entendimento. Além de representar as *features* do modelo, a visualização também precisa ilustrar atributos, dependências, relações, alternativas e a hierarquia do modelo de *features* (SAIZ, 2009).

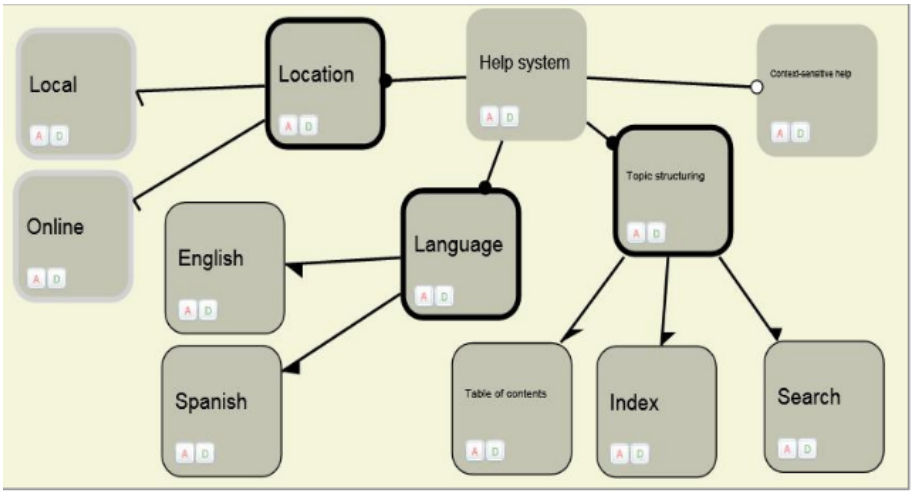
Alguns tipos de visualizações adequados para a visualização dos modelos de *features* serão mostrados, na sequência: (i) O modelo gráfico (*The graph model*); (ii) Lista hierarquizada (*Indented list*); e (iii) Visão em árvore (*Tree-view*) como visualizado em (SAIZ, 2009). Eles são mostrados com mais detalhes abaixo nas Seções 2.5.1, 2.5.2 e 2.5.3.

2.5.1 Modelo Gráfico

De acordo com Saiz (2009), "um gráfico é um diagrama que mostra um conjunto de relações, muitas vezes funcionais, entre um grupo de pontos ou números. Cada um destes pontos ou números tem coordenadas determinadas pelas suas relações".

Para um melhor entendimento dessa relação, a Figura 7 mostra um exemplo de modelo gráfico apresentado na ferramenta proposta no trabalho de Saiz (2009).

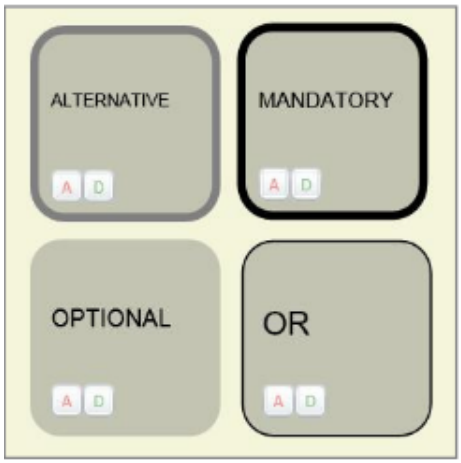
Figura 7 – Exemplo de modelo gráfico.



Fonte: Saiz (2009)

A Figura 8 mostra como as *features* ficam representadas quando elas se diferenciam por serem alternativas (*alternative*), obrigatórias (*mandatory*), opcionais (*optional*) e ou (*or*). Cada tipo é identificado visualmente por uma diferente borda de *feature*.

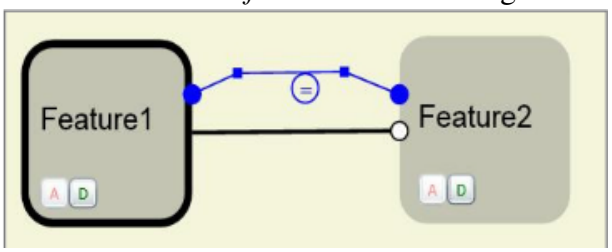
Figura 8 – Tipos das *features* do modelo gráfico.



Fonte: Saiz (2009)

Como se pode observar na Figura 9, as dependências são representadas com uma ligação, em cor específica e um simbolo. De acordo com variações de cores e símbolos, temos a um tipo de dependência diferente, as variações podem ser vistas na Figura 10.

Figura 9 – Representação das dependências entre as *features* do modelo gráfico.



Fonte: Saiz (2009)

Figura 10 – Representação das dependências entre as *features* do modelo gráfico.

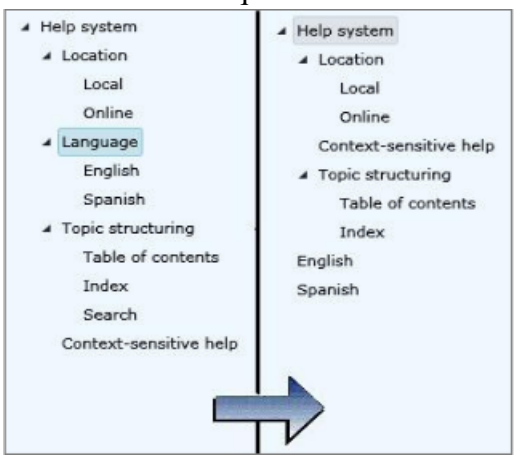
Feature dependency name	Symbol	Color
Excludes		
Extends	Ext	
Includes	Inc	
Incompatible		
Requires	Req	
Uses	Use	
Same		

Fonte: Saiz (2009)

2.5.2 Lista Hierarquizada

Lista hierarquizada é uma série de objetos dispostos em uma ordem lógica (SAIZ, 2009). As *features* são disposta hierarquicamente, como se pode observar na Figura 11.

Figura 11 – Exemplo de lista hierarquizada.



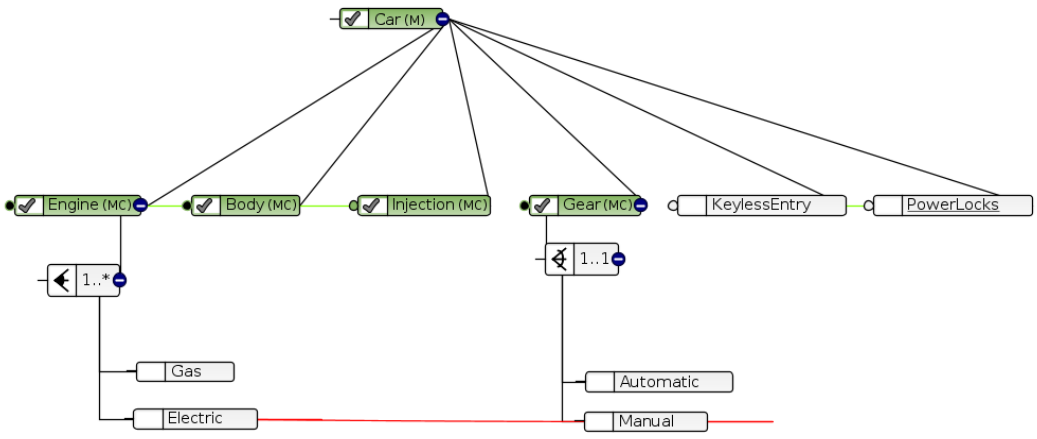
Fonte: Saiz (2009)

A ferramenta DyMMer, que faz a utilização dessa visualização, representada no lado direito da Figura 16, utiliza símbolos para representar a relação entre as *features* do modelo, mais detalhes podem ser vistos na Seção 2.6. Ela utiliza também o modelo de *features* baseada em cardinalidade, tratada melhor na Seção 2.2.1.

2.5.3 Visão em Árvore

Na Visão em árvore as *features* são conectadas por nódulos, formando uma hierarquia. De acordo com Saiz (2009) essa forma de visualizar modelo de *features* é a que se mostra mais interessante para os visualizadores, pois a visualização se torna de fácil compreensão e entendimento.

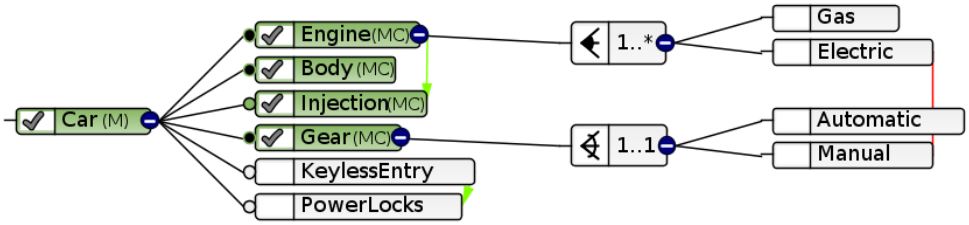
Figura 12 – Exemplo de modelo de *features* em formato de árvore.



Fonte: Saiz (2009)

A ferramenta *S2T2 Configurator* (BOTTERWECK; JANOTA; SCHNEEWEISS, 2009) trabalha muito bem com esse tipo de visualização de modelos de *features*. Seus autores utilizaram os símbolos com inspiração na notação FODA (ROBAK, 2003) e notações de cardinalidade, visto na Seção 2.2.1, para representar as dependências das *features* do modelo representado. Pode-se observar na Figura 13 que as linhas vermelhas representam a relação de exclusão, e a verde, relação de dependência; existe um campos de marcação em cada *feature* para ilustrar se ela está selecionada ou não; os ciclos escuro e branco, representam respectivamente, *features* obrigatórias e opcionais. A árvore pode ter perspectivas diferentes, como mostrado na Figura 13.

Figura 13 – Exemplo de modelo de *features* em formato de árvore com mudança de perspectiva.



Fonte: Botterweck, Janota e Schneeweiss (2009)

2.6 Ferramenta DyMMer

A ferramenta *desktop* DyMMer, como citado na Seção 1, foi desenvolvida com o intuito de implementar medidas para apoiar na avaliação da qualidade de modelos de *features* em LPSDs. Ela foi desenvolvida usando a tecnologia Java, na plataforma J2SE, com algoritmos para a leitura dos arquivos do repositório do S.P.L.O.T. A ferramenta DyMMer também oferece a visualização dos modelos do repositório, e se pode utilizar a sua área de edição de modelos para a adição de contexto nos modelos de *features* importados (BEZERRA et al., 2016b).

Baseados nas necessidades de avaliação e edição da ferramenta para o modelo de *features* das LPSDs a ferramenta conta com os seguintes requisitos (BEZERRA et al., 2016b):

- Importação de modelo de *feature* com base em estrutura de arquivos gerados pelo S.P.L.O.T.;
- Visualização do modelo de *feature* conforme o contexto selecionado, também exibido pela aplicação;

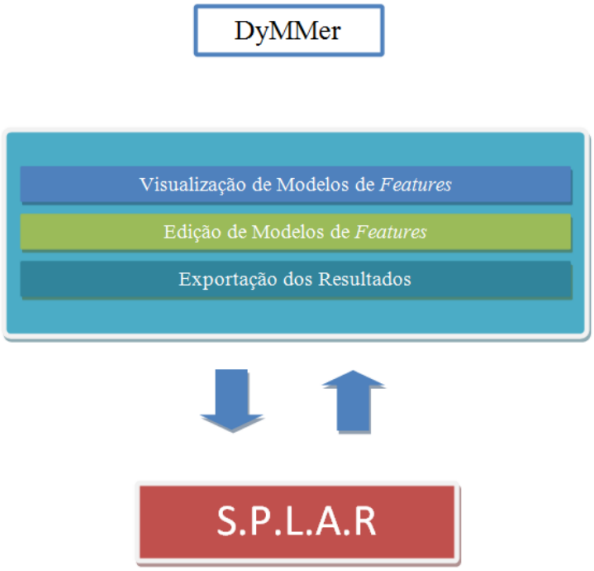
- Aplicação de diferentes medidas de qualidade sobre o modelo de *feature* e seus contextos;
- Área de edição do modelo, onde é possível criar novos contextos e definir *features* ativas ou desativadas; e
- Exportação de resultados das medidas de qualidade, baseados em diversos modelos de *features* sem contexto ou resultado das medidas de um modelo e seus contextos.

A arquitetura da DyMMer é composta por três camadas, como mostrado em (BEZERRA et al., 2016b).

1. Camada para exportação dos dados: permite a exportação dos resultados das medidas aplicadas aos modelos e seus contextos;
2. Camada de visualização de modelos: possibilita a visualização do modelo de *feature*, bem como os contextos compreendidos;
3. Camada de edição dos modelos: habilita criar novos contextos e regras de restrições conforme edita-se o modelo.

Essas camadas utilizam o componente S.P.L.A.R (MENDONCA; BRANCO; COWAN, 2009), que é uma biblioteca responsável pela análise automática dos modelos de *features*. A arquitetura é representada na Figura 14.

Figura 14 – Arquitetura da Dymmer.

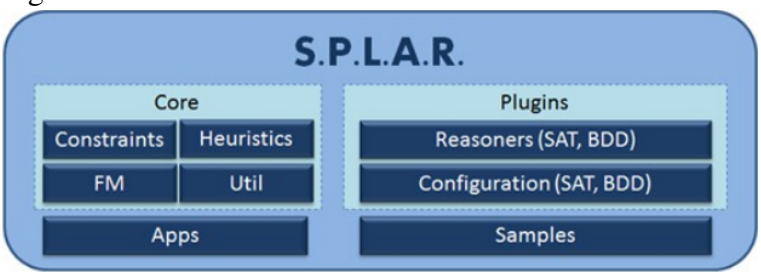


Fonte: Bezerra et al. (2016b)

A Figura 15 ilustra as partes que fazem parte do S.P.L.A.R. No *core*, existem artefatos

que representam e implementam as restrições de integridade do modelo de *feature*, o próprio modelo, heurísticas e utilidades que tratam do modelo. Em *plugins*, encontramos implementações e facilitadores para aplicação do SAT e BDD (BEZERRA et al., 2016b).

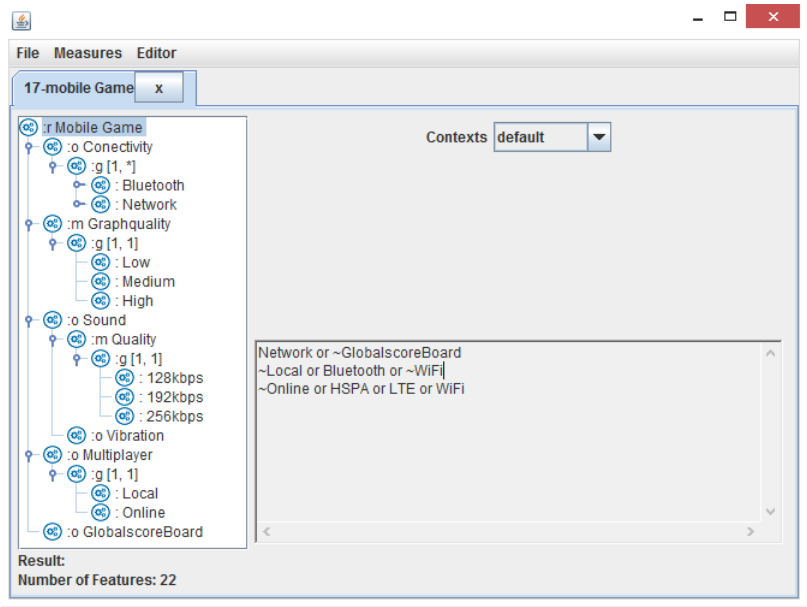
Figura 15 – S.P.L.A.R.



Fonte: Bezerra et al. (2016b)

Como pode ser observado na Figura 16, a DyMMer é capaz de abrir modelos de *features* para serem representados, como ilustrado na lista hierarquizada no lado esquerdo da figura. Contextos podem ser selecionados, e de acordo com esse contexto a visualização será modificada, os contextos podem ser vistos no lado direito e acima da Figura e logo abaixo pode são ilustradas as restrições que as *features* do modelo possuem.

Figura 16 – Visualização da DyMMer.

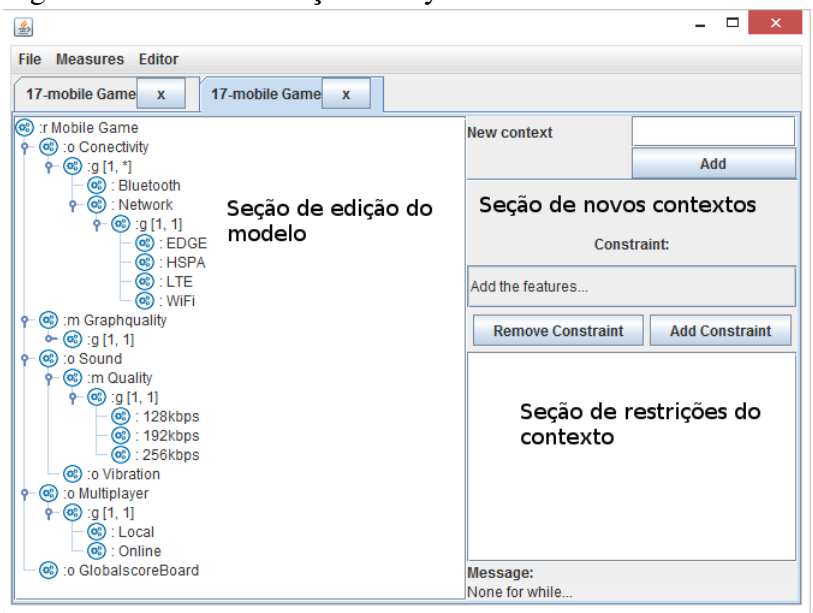


Fonte: Bezerra et al. (2016b)

Como ilustrado na Figura 17, a DyMMer é capaz também de editar os modelos de *features* existentes. No lado esquerdo destacado como "Seção de edição do modelo", temos a representação do modelo, em que cada característica pode ser modificada; no lado direito acima

destacado como "Seção de novos contextos", podemos adicionar novos contextos ao modelo representado; logo abaixo temos o destaque "Seção de restrições do contexto", onde temos as restrições do contexto selecionado, pode-se também adicionar mais restrições a esse contexto selecionado.

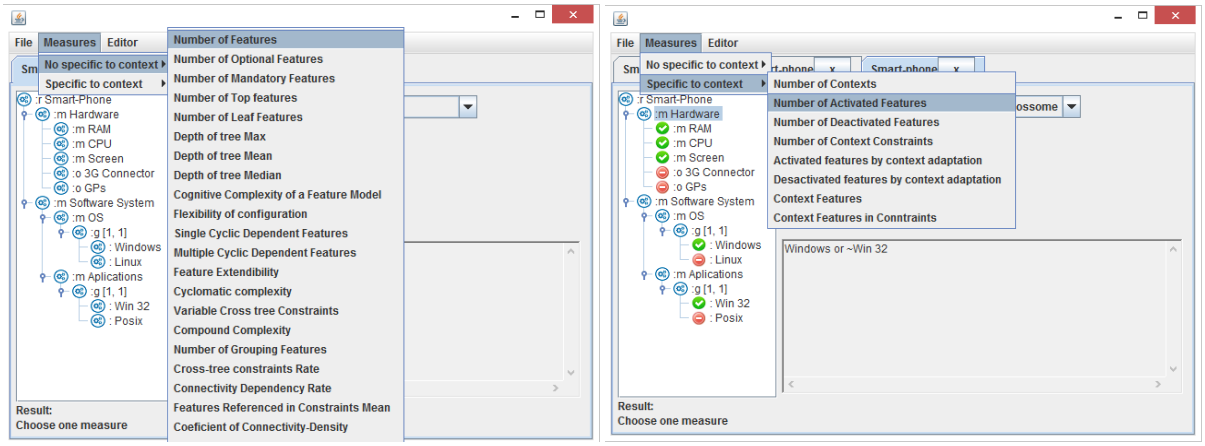
Figura 17 – Área de Edição da DyMMer.



Fonte: Bezerra et al. (2016b)

A DyMMer é capaz de realizar exportação de resultados das medidas de qualidade, baseados em diversos modelos de *features* sem contexto ou com contexto (ver Figura 18). No trabalho a ser desenvolvido os resultados destas medidas serão apresentadas por meio de visualizações, que facilitem a sua leitura e entendimento.

Figura 18 – Medidas que a ferramenta DyMMer pode ilustrar.



Fonte: Elaborada pelo autor

3 TRABALHOS RELACIONADOS

Esta seção visa apresentar estudos que se relacionam com este trabalho e são discutidas as principais contribuições dos trabalhos relacionados e um comparação com o trabalho desenvolvido.

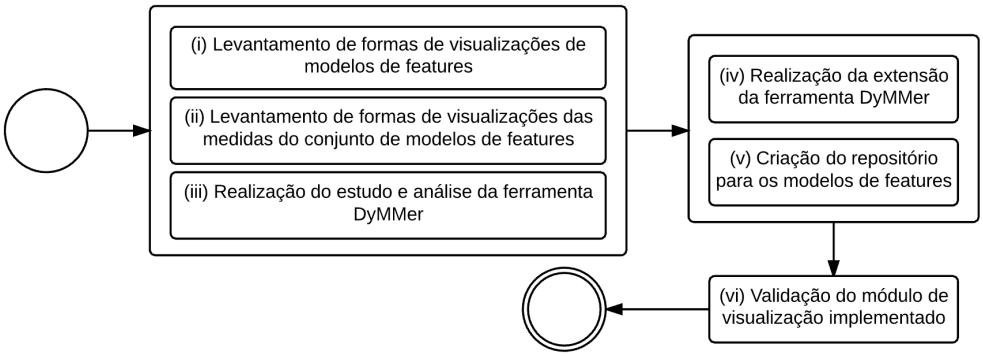
Bezerra et al. (2016b) desenvolveram a ferramenta DyMMer, que tem como objetivo apoiar a avaliação da qualidade de modelos de *features* em LPSDs por meio de medidas. Detalhes sobre a DyMMer são descritas na Seção 2.6. Esta ferramenta foi a base para o desenvolvimento deste trabalho e foi estendida com novas funcionalidades para visualização de medidas relacionados ao modelo de *features*. Na ferramenta, foi adicionada um modelo de visualização para as medidas dos o modelo de *features*, um novo modelo de visualização para árvore do modelo de *features* e um repositório para gerenciamentos de modelos de *features*; essas adições são explicadas com detalhes na Seção 6.

Urli et al. (2015) apresentam uma abordagem visual para decomposição de modelos de *features* complexos, para o desenvolvimento da visualização das restrições internas e externas das *features* em uma árvore do modelo de *features*, a ferramenta *Roassal* é usada para o desenvolvimento da abordagem visual apresentada. Portanto, o trabalho proposto visa gerar visualizações para as medidas dos modelos de *features* que são apresentadas pela ferramenta DyMMer, e criar uma nova visualização em formato de árvore para ilustrar os modelos de *features*.

4 PROCEDIMENTOS METODOLÓGICOS

Esta Seção apresenta os procedimentos executados do trabalho proposto. Como representado na Figura 19, os seguintes passos foram executados: (i) Realização do estudo e análise da ferramenta DyMMer; (ii) Levantamento de formas de visualizações das medidas do conjunto de modelos de *features*; (iii) Levantamento de formas de visualizações dos modelos de *features*; (iv) Realização da extensão da ferramenta; (v) Criação do repositório para os modelos de *features*; e (vi) Validação do módulo de visualização implementado. As atividades i, ii e iii ocorrem em paralelo. Já as atividades iv e v também ocorreram em paralelo.

Figura 19 – Procedimentos para a execução do trabalho.



Fonte: Elaborada pelo Autor

4.1 Realização do estudo e análise da ferramenta DyMMer

Neste passo é realizado o estudo da arquitetura da ferramenta DyMMer estendida, focando nas propriedades alteradas para a extensão da ferramenta. O estudo da ferramenta é realizado analisando o passo a passo de criação apresentado em Bezerra et al. (2016b).

4.2 Levantamento de formas de visualizações das medidas do conjunto de modelos de *features*

Neste passo é realizada o levantamento de visualizações para a representação das medidas dos modelos de *features*, que serão visualizados na ferramenta a ser estendida. São realizadas pesquisas de forma não sistemática à bibliografias que mostrem formas de visualizações para estas medidas. É realizado um questionário, apresentado no Apêndice A, com cinco participantes, três desses são alunos de graduação e dois alunos de pós-graduação, para classificar as subcaracterísticas e medidas de qualidade relevantes para a visualização da

manutenibilidade do modelo de *features* de LPSDs.

4.3 Levantamento de formas de visualizações de modelos de *features*

São investigadas as formas de visualizações para a representação dos modelos de *features* que são visualizados na ferramenta estendida. Também são realizadas pesquisas de forma não sistemática à bibliografias que mostrem formas de representações que melhor se encaixem para os modelos de *features*, para que a visualização dos modelos seja facilitada.

4.4 Realização da extensão da ferramenta DyMMer

Neste passo é implementado a extensão da ferramenta DyMMer, com base nas informações recolhidas na análise do procedimento 4.1. Juntamente com essa extensão, são adicionados os novos modelos de visualização selecionados no procedimento 4.2 e 4.3.

4.5 Criação do repositório para os modelos de *features*

Neste passo é criado o repositório de dados para o armazenamento dos modelos de *features* que serão usados na ferramenta DyMMer estendida. É realizada a criação de um repositório para o armazenamento dos modelos de *features*. Os modelos são salvos em um banco de dados NoSQL hospedado em um servidor disponibilizado pela UFC Campus Quixadá.

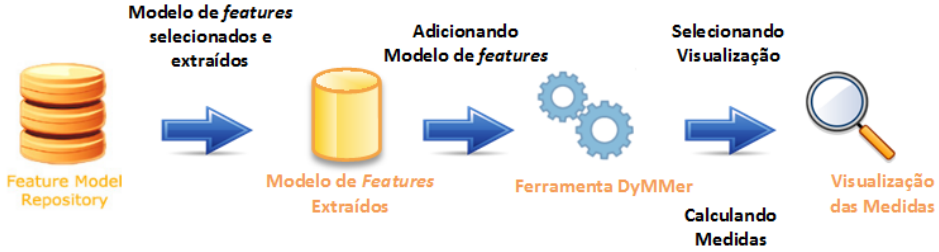
4.6 Validação do módulo de visualização implementado

Neste passo são validadas as visualizações implementadas na ferramenta DyMMer. É verificado a facilidade e o entendimento da nova árvore do modelo de *features* e as ilustrações dos resultados das medidas. Para isso, é realizada uma validação com especialistas de engenharia de domínio a partir de um questionário, apresentado no Apêndice A, onde são ilustradas as visualizações com breve explicações, e logo após perguntas sobre e eficácia das mesmas.

5 PROCESSO DE VISUALIZAÇÃO DE MEDIDAS

Nesta Seção é descrito o processo de visualização de medidas adicionado a ferramenta DyMMer, permitindo a interpretação do subconjunto de medidas de qualidades apresentados na Tabela 2. O processo utilizado para gerar uma visualização é ilustrado na Figura 20. Os modelos podem ser extraídos do *MAcchiaTO Repository* que contém 218 modelos de *features*, do repositório S.P.L.O.T, ou criados diretamente na ferramenta. Desse modo, o engenheiro de domínio seleciona um ou mais modelos de *features*, em seguida selecionada a visualização desejada, as medidas são calculadas, e por fim apresentadas ao engenheiro de domínio por meio de visualizações de dados.

Figura 20 – Visão geral do processo de visualização de medidas.



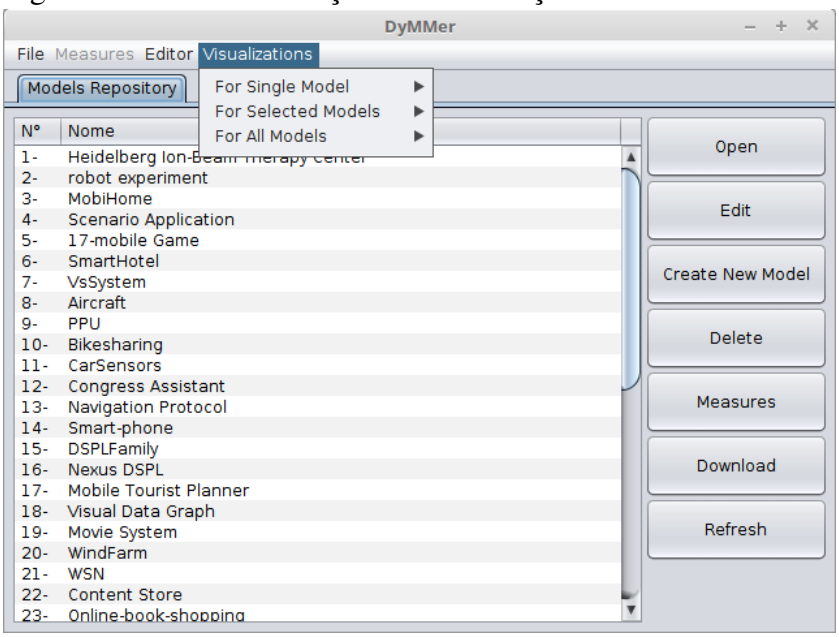
Fonte: Elaborada pelo autor

5.1 Selecionando os modelos de *features* para as visualizações de suas medidas

A ferramenta DyMMer estendida dispõe de uma lista de modelos de *features* obtidos do seu próprio repositório, onde é possível selecionar um ou mais modelos para visualização de suas medidas, como apresentado na Subseção 6.4.1. Para toda visualização ilustrada na ferramenta, é necessário a seleção de um ou mais modelos, sendo, em alguns casos, versões de um modelo de *features*.

A Figura 21 ilustra o menu de visualizações selecionado, na sua listagem é possível observar que existem visualizações de medidas para um modelo selecionado (*For Single Model*), para vários modelos selecionados (*For Selected Models*) e para todos os modelos do repositório (*For All Models*).

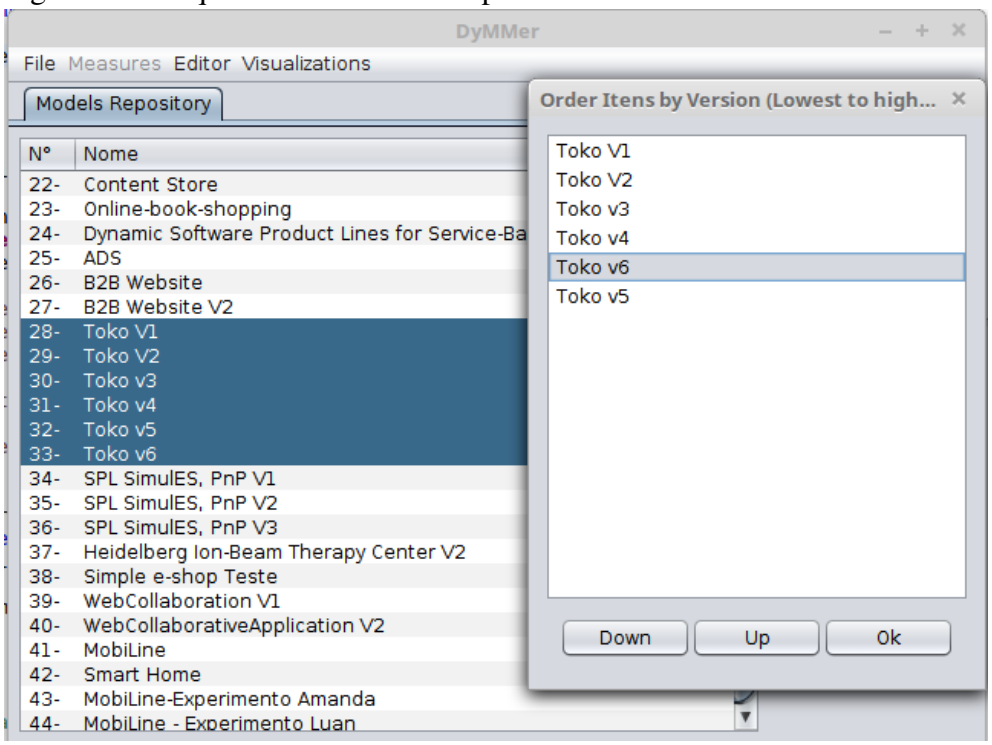
Figura 21 – Menu de seleção de visualização.



Fonte: Elaborada pelo autor

A Figura 22 apresenta um caso em que é necessário uma ordenação das versões de um modelo. A ferramenta estendida não conta com um controle de versão, os modelos com diferentes versões devem ser adicionados no repositório da mesma forma que se estivesse adicionando um novo modelo. Se os modelos que foram selecionados não estiverem em uma ordem correta de versão, os cálculos realizados para a visualização não saem da forma que deveriam. Para suprir essa necessidade de ordenação, foi criado um modo de ordenar os modelos, em que se pode selecionar um modelo na janela a direita da Figura 23, e move-lo para cima ou para baixo, ordenando dessa forma os modelos.

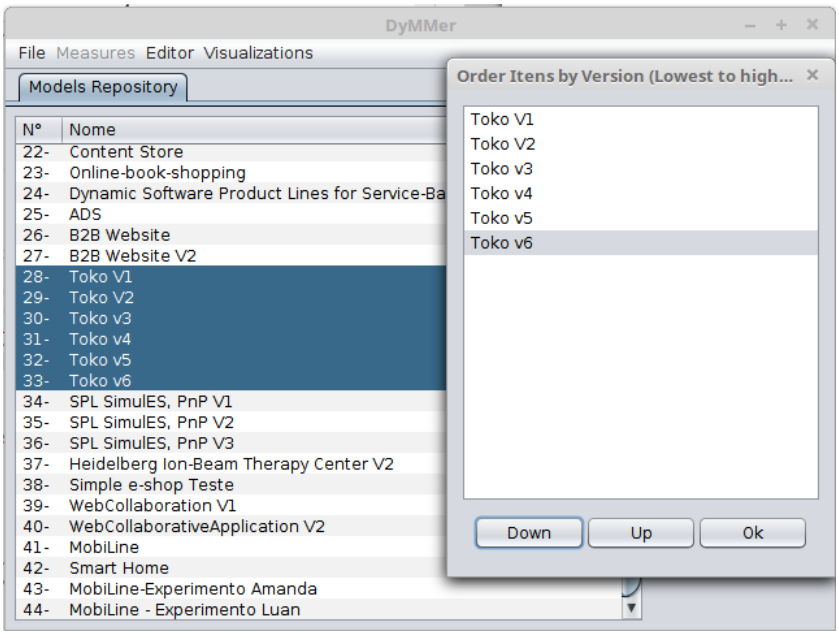
Figura 22 – Sequência de versão comprometida.



Fonte: Elaborada pelo autor

A figura 22 ilustra também uma ordenação errada, como pode ser observada na janela do lado direito da Figura. Para resolver esse problema, é necessário mover o item “Toko v6” para baixo (usar o botão “*Down*”) ou mover o item “Toko v5” para cima (usar o botão “Up”), realizando uma dessas operações a lista estará ordenada, como apresentada na Figura 23.

Figura 23 – Versão de modelo ordenada corretamente.



Fonte: Elaborada pelo autor

5.2 Visualizações das Medidas

Para identificação das visualizações das medidas, foi aplicado um questionário com cinco pessoas com conhecimento em LPSs e LPSDs, três delas do grupo *STRATEGy* (Grupo de Reuso e Qualidade de Software) e dois alunos de pós-graduação, para classificar as subcaracterísticas e medidas de qualidade relevantes para a visualização da manutenibilidade do modelo de *features* de LPSDs. O nível de formação dos envolvidos pode ser observado na Figura 24 (ver Apêndice A).

Figura 24 – Nível de formação dos envolvidos.

Fonte: Elaborada pelo autor.

No questionário, as Subcaracterísticas e Medidas, ilustradas na Tabela 2, foram classificadas com o nível de importância: menos importante, pouco importante, indiferente, importante e muito importante. Após os participantes do questionário responde-lo com essas possíveis classificações, foi possível obter os dados que terão maior ênfase para a criação das visualizações.

Após a análise realizada no questionário, foram obtidas as Subcaracterísticas e Medidas com mais votos no nível de importância: "muito importante"as mesmas estão ilustradas no Quadro 6.

O questionário também realizava perguntas aos participantes sobre as subcaracterísticas e suas medidas relacionadas, ilustradas na Tabela 2, como a seguinte: “Quais perguntas você gostaria de responder a partir de uma visualização, para a Subcaracterística Extensibilidade?”. Todas as perguntas e respostas podem ser visualizadas no apêndice A.

Dezoito perguntas foram obtidas das respostas dos participantes, todas relacionadas as Subcaracterísticas e suas Medidas de qualidade, encontradas no Apêndice A. Dessas 18 perguntas, foram selecionadas apenas as que tinham relação direta com as Subcaracterísticas votadas como “muito importante” pelos participantes, as mesmas Subcaracterísticas ilustradas

Quadro 6 – Subcaracterísticas e Medidas consideradas muito importantes

Subcaracterísticas	**Medidas**
Extensibilidade	Extensibilidade da Feature (FEX)
Complexidade Estrutural	Número de *Features* (NF); Número de *Features* Mandatórias (NM); Número de *Features* Top (NTop);Restrições *Cross-tree* (CTC)
Variabilidade Estática	Número de *Features* Opcionais (NO); Número de Configurações Válidas (NVC); Taxa de Variabilidade (RoV)
Variabilidade Dinâmica	Número de Contextos (NC); Número de *Features* Desativadas (NDF); Número de *Features* Ativadas (NAF); *Features* de Contextos em Restrições (CFC); Número de Restrições de Contexto (NCC); Número de *Features* de Contextos (CF); Número de *Features* Ativadas por Contexto (AFCA); Número de *Features* Desativadas por Contexto (DFCA)

Fonte: Elaborada pelo autor

no Quadro 6. As perguntas foram as seguintes:

1. Qual o impacto a linha de produtos sofrerá quando uma mudança ocorre no modelo de *features*?
2. Qual o impacto que a linha sofreria ao estender uma determinada *feature*?
3. Qual o aumento no número de configurações a partir da inclusão de novas *features*?
4. Qual a evolução da complexidade do modelo de *features* quando uma *feature* é adicionada e removida?
5. Qual dos contextos presentes em um modelo de *features*, possui mais dinamicidade em termos de ativação e desativação ?
6. Qual a complexidade do modelo de *features*?
7. Qual a variabilidade dinâmica do modelo de *features*?
8. Qual modelo de *features* é de fácil extensibilidade?

Com essas perguntas obtidas, foram desenvolvidas visualizações que poderiam respondê-las, ilustradas nas próximas subseções.

5.2.1 Bibliotecas Utilizadas Para Gerar as Visualizações das Medidas

Para apoiar o desenvolvimento das medidas, foi necessário a utilização de bibliotecas que facilitem a criação de visualizações de dados. Algumas bibliotecas em java foram utilizadas para a criação de visualizações, como a *JFreeChart* [1], mas as visualizações não eram agradáveis. Por falta de bibliotecas gratuitas, de retorno visual agradável e com uma alta gama de tipos de

[1] http://www.jfree.org/jfreechart/

gráficos, foi necessário realizar buscas de bibliotecas que utilizem uma linguagem diferente da Java. A linguagem *JavaScript* mostrou uma gama de possibilidades que poderiam ser utilizadas, como *Chart.js*[2], *Raw*[3], *Dygraphs*[4], *ZingChart*[5], *FusionCharts*[6], *D3.js*[7]. As utilizadas para gerar as visualizações para a ferramenta estendida foram a *Chart.js e D3.js*.

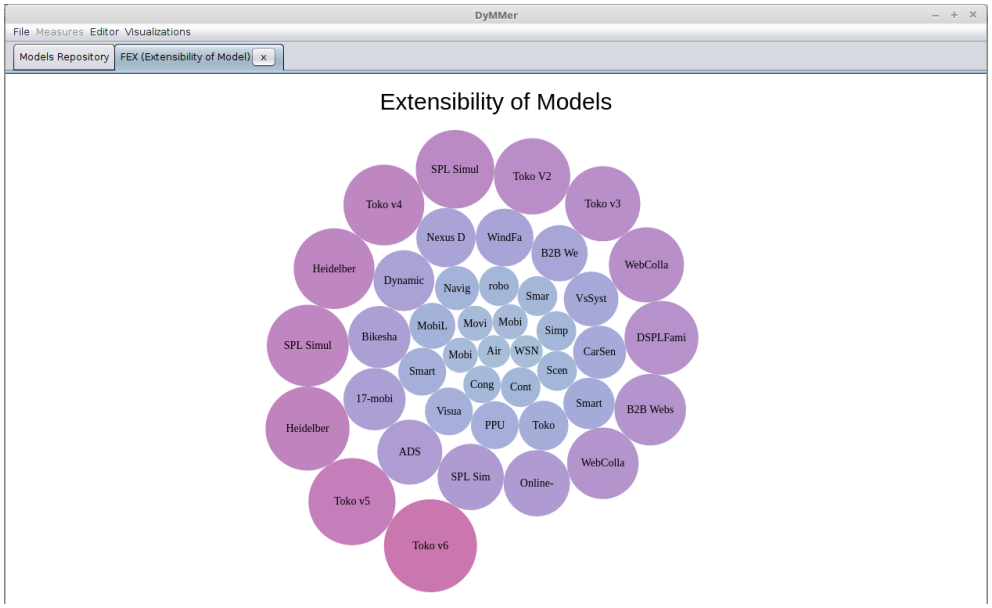
A utilização de *JavaScript* em aplicação java utilizando *Swing* é bem complicado, pois é necessário uma biblioteca *web browser component* que consiga mostrar páginas web na aplicação. Para suprir essa necessidade a biblioteca *JxBrowser*[8] foi utilizada, mesmo sendo uma biblioteca paga, a mesma disponibiliza uma licença gratuita para sistemas *open sources*.

5.2.2 Extensibilidade dos Modelos

A Figura 25 apresenta uma visualização de classe 2D, e do tipo Circular que ilustra a extensibilidade de todos os modelos que existem no repositório da DyMMer. Os Modelos com extensibilidade menores são encontrados mais centralizados, com cor mais clara e com tamanho menores, enquanto os com extensibilidade maiores são encontrados mais distantes do centro, com cor mais escura e tamanho maiores. A medida utilizada foi: Extensibilidade da *Feature* (FEX), que tem ligação com a Subcaraterística Extensibilidade, ilustrada na Tabela 2.

[2] http://www.chartjs.org/
[3] http://raw.densitydesign.org/
[4] http://dygraphs.com/
[5] https://www.zingchart.com/
[6] http://www.fusioncharts.com/
[7] https://d3js.org/
[8] https://www.teamdev.com/jxbrowser

Figura 25 – Extensibilidade dos Modelos.



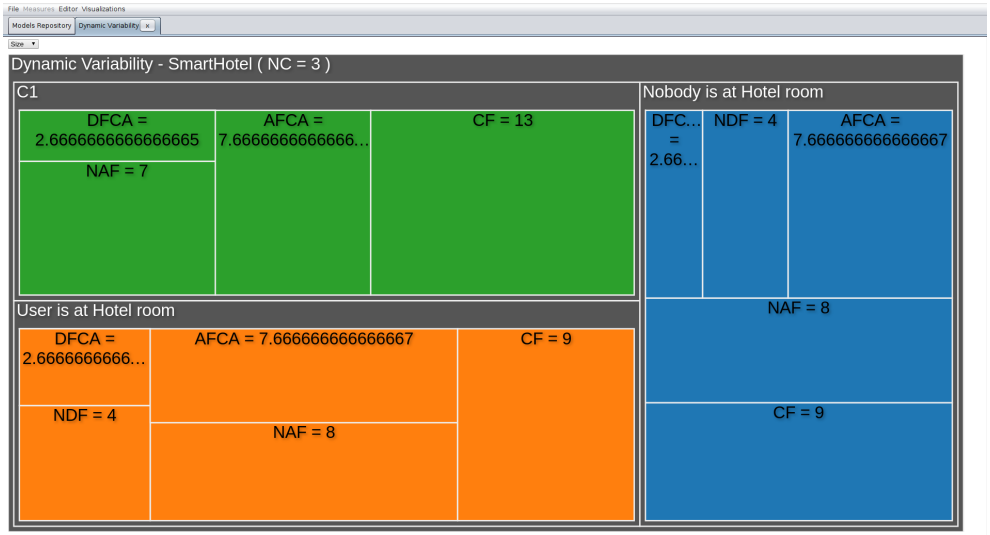
Fonte: Elaborada pelo autor

A visualização foi proposta para responder a seguinte pergunta: (8) “Qual modelo de *features* é de fácil extensibilidade?”.

5.2.3 Variabilidade Dinâmica

A Figura 26 apresenta uma visualização de classe 2D, e do tipo *TreeMap* que ilustra a variabilidade dinâmica de um modelo de *features*. Os contextos do modelo são recuperados e adicionados na visualização, que são os seguintes: “C1”, “*Nobody is at Hotel room*” e “*User is at Hotel room*”; para cada contexto do modelo, são ilustrados as medidas: Número de Contextos (NC), Número de *Features* Ativadas, (NaF), Número de *Features* Desativada (NdF), Número de *Features* de Contextos (CF), *Features* de Contextos em Restrições (CFC), Número de *Features* Ativadas por Contexto (AFCA), Número de *Features* Desativadas por Contexto (DFCA); as medidas que não são ilustradas em algum contexto do modelo estão com valor zero. Para realizar a leitura da visualização, é necessário observar as medidas de cada contexto ilustrado e realizar uma comparação dos seus valores ou tamanhos, dessa forma é possível obter uma conclusão em relação a variabilidade dinâmica do modelo apresentado.

Figura 26 – Variabilidade Dinâmica.



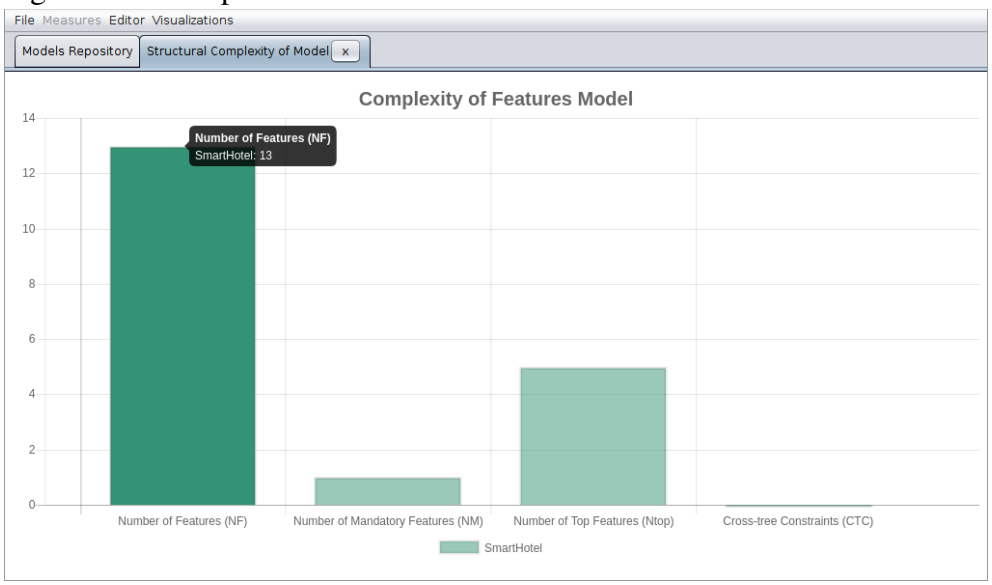
Fonte: Elaborada pelo Autor.

A visualização foi proposta para responder as seguintes perguntas: (7) “Qual a variabilidade dinâmica do modelo de *features*?”; e (5) “Qual dos contextos presentes em um modelo de *features*, possui mais dinamicidade em termos de ativação e desativação?”.

5.2.4 Complexidade Estrutural de um modelo de features

A Figura 27 apresenta uma visualização de classe 2D, e do tipo Barras que ilustra a Complexidade Estrutural de um modelo. Um modelo é selecionado e dele é obtido: O Número de *Features* (NF), O Número de *Features* Mandatórias (NM), O Número de *Features* Top (Ntop) e as Restrições *Cross-tree* (CTC). Para realizar a leitura da visualização, é necessário observar as medidas ilustradas a partir de sua altura e valor, dessa forma é possível obter uma conclusão em relação a complexidade estrutural do modelo apresentado.

Figura 27 – Complexidade Estrutural.



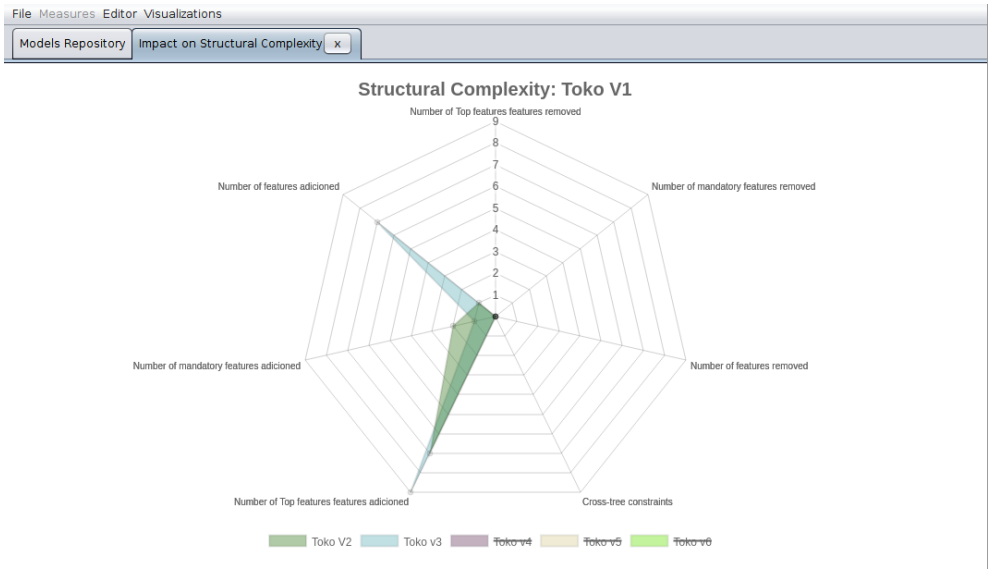
Fonte: Elaborada pelo Autor.

A visualização foi proposta para responder a seguinte pergunta: (6) “Qual a complexidade do modelo de *features*?”.

5.2.5 Complexidade Estrutural de um conjunto de versões de um modelo

A Figura 28 apresenta uma visualização de classe 2D, e do tipo Radar que ilustra a complexidade estrutural de um modelo de acordo com suas versões. Versões de um modelo são selecionadas, e desse conjunto é obtido: O Número de *Features* adicionadas e removidas, o Número de *Features* Mandatórias adicionadas e removidas, o Número de *Features* Top adicionadas e removidas, e as Restrições *Cross-tree* das versões. Na visualização pode-se observar que o modelo Toko V4, Toko v5 e Toko V6 não estão selecionados, isso pode ser realizado dinamicamente na ferramenta, para assim ser possível observar melhor a relação entre Toko V2 e Toko V3; com apenas esses dois ilustrados na visualização, pode-se observar que da Toko V2 a Toko V3 ouve um aumento de 7 *features*, 1 *features* obrigatória, 9 *features* top, e nenhuma remoção.

Figura 28 – Complexidade Estrutural.



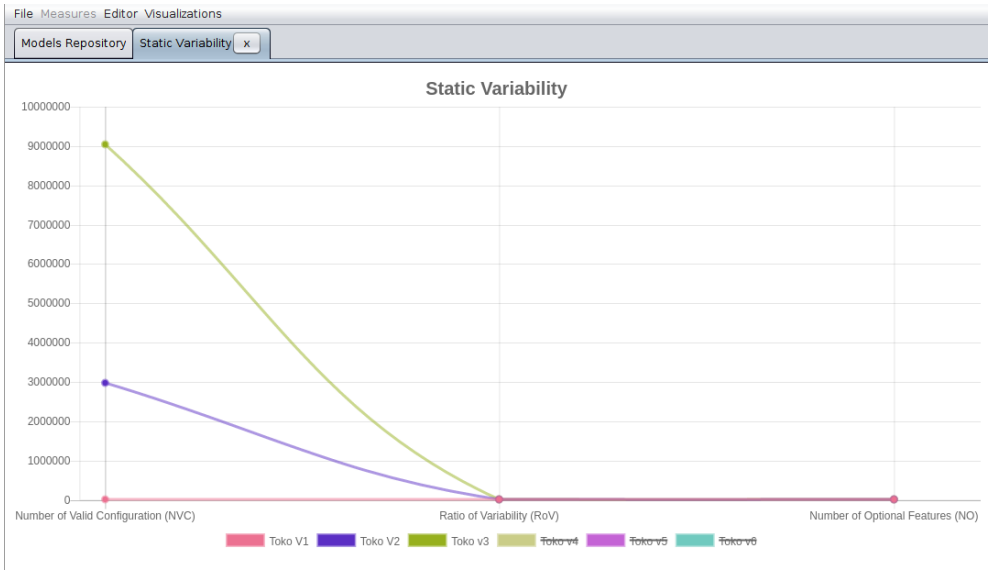
Fonte: Elaborada pelo Autor.

A visualização foi proposta para responder as seguintes perguntas: (2) “Qual o impacto que a linha sofreria ao estender uma determinada *feature*?”; (1) “Qual o impacto a linha de produtos sofrerá quando uma mudança ocorre no modelo de *features*?”; e (4) “Qual a evolução da complexidade do modelo de *features* quando uma *feature* é adicionada e removida?”.

5.2.6 Variabilidade Estática

A Figura 28 representa uma visualização de classe 2D, e do tipo Linhas que ilustra a variabilidade estática do modelo de *features*. Versões de um modelo são selecionadas, e desse conjunto é obtido: O Número de configurações válidas (NVC), A Taxa de variabilidade (RoV) e O Número de *features* opcionais (NO); é ilustrado na visualização a relação entre Toko V1, Toko v2 e Toko v3, onde Toko v3 tem um número de configurações maior que Toko v2 e Toko v3, mas as outras medidas são pareadas. Para realizar a leitura da visualização, é necessário observar as medidas ilustradas a partir de sua altura e valor, dessa forma é possível obter uma conclusão em relação a variabilidade estática do modelo apresentado.

Figura 29 – Variabilidade Estática.



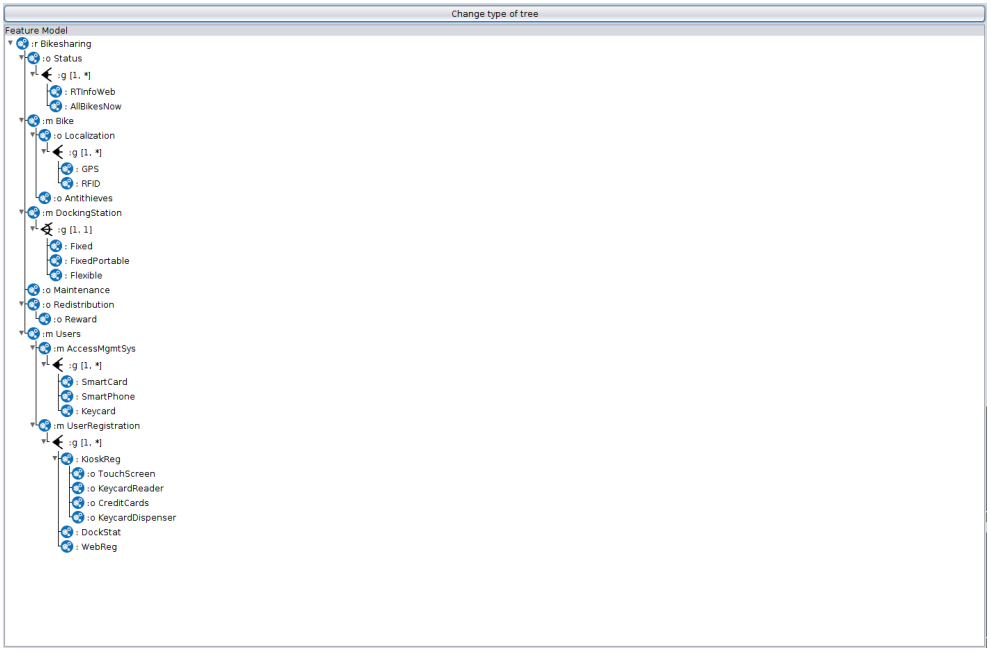
Fonte: Elaborada pelo Autor.

A visualização foi proposta para responder a seguinte pergunta: (3) “Qual o aumento no número de configurações a partir da inclusão de novas *features*?”.

5.3 Visualização do modelo de *features*

Logo após a análise das formas de visualizações dos modelos de *fatures*, como descrito na Seção 2.5, foi possível concluir que a visão em árvore é de mais fácil compreensão e entendimento, por esse motivo a visualização foi focada em ser desenvolvida nesse formato. A visão em lista hierarquizada antes apresentada pela DyMMer ainda permanece na ferramenta como opção para o usuário, apresentada ao lado esquerdo da Figura 30, podendo ser ilustrada acessando o botão “*Change type of Tree*", localizado na parte superior da Figura 30.

Figura 30 – Visualização em Lista Hierárquica do modelo de *features*.

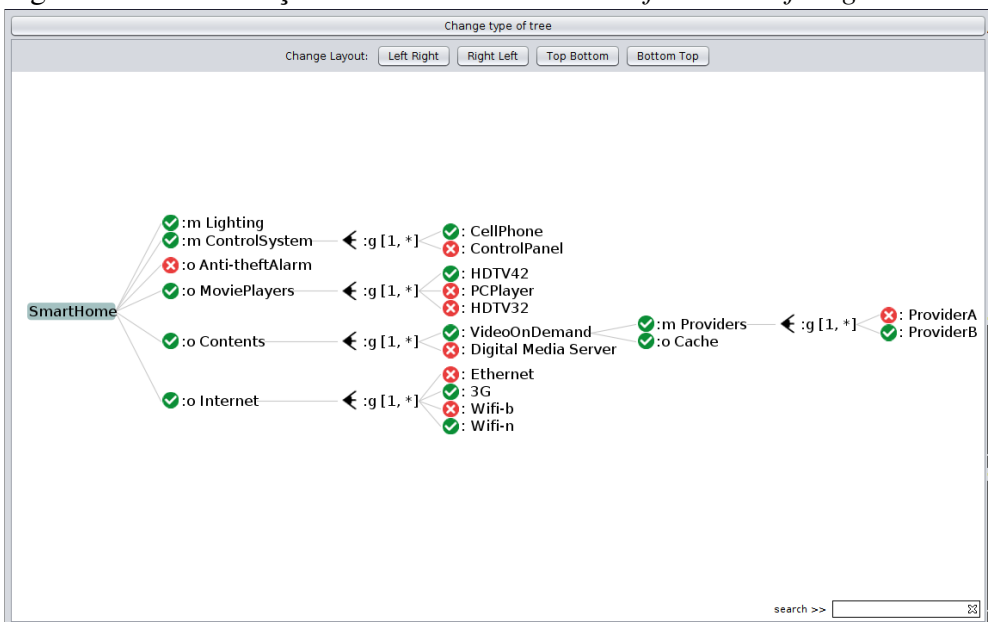


Fonte: Elaborada pelo autor

A nova árvore de modelos de *features* foi desenvolvida utilizando a biblioteca *Prefuse*[9], a mesma que a ferramenta *S2T2 Configurator*, mostrada na Seção 2.5.3, utiliza. Com essa biblioteca pode-se criar visualizações de dados ricas em interações. As Figuras 31 e 32 ilustram exemplos da nova árvore. Pode-se observar que a visualização tem várias orientações, são elas: (i) *Left Right*, direciona-se da esquerda para direita; (ii) *Right Left*, direciona-se da direita para esquerda; (iii) *Top Bottom*, direciona-se de cima para baixo; e (iv) *Bottom Top*, direciona-se de baixo para cima.

[9] http://prefuse.org/

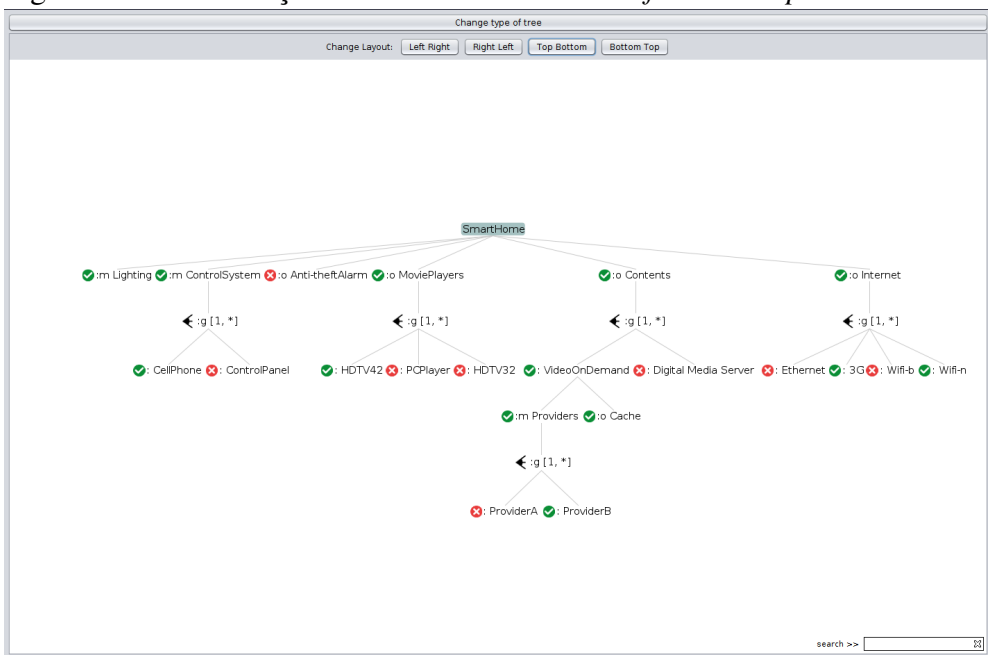
Figura 31 – Visualização em árvore do modelo de *features Left Right*.



Fonte: Elaborada pelo autor

As várias orientações que a árvore pode tomar, fazem com que haja uma melhor visualização dos itens que um modelo de *features* apresenta, de acordo com a preferência do usuário da ferramenta.

Figura 32 – Visualização em árvore do modelo de *features Top Bottom*.



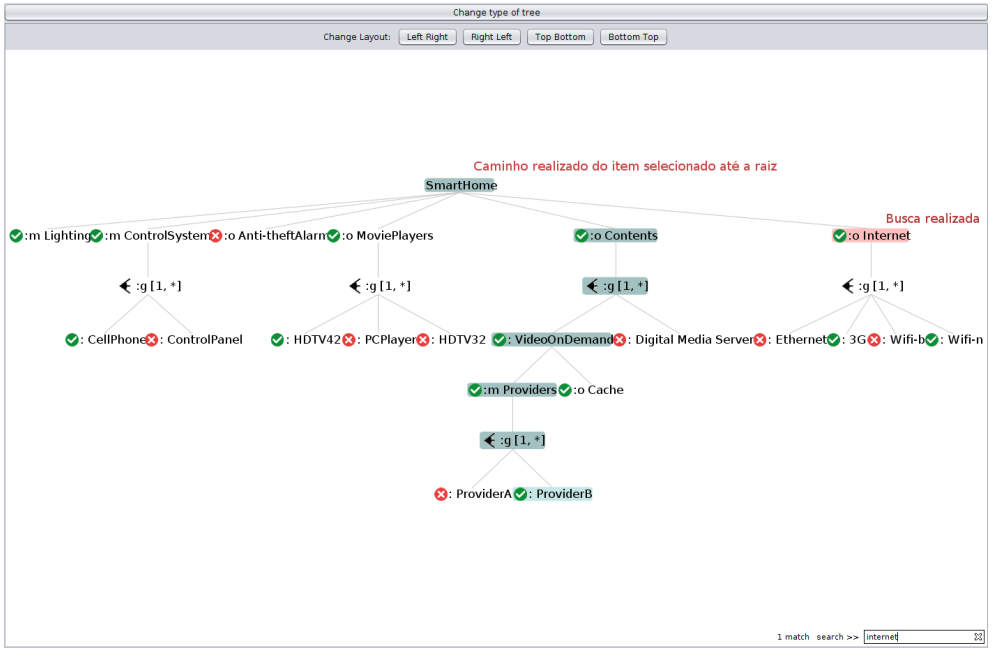
Fonte: Elaborada pelo autor

Na visualização da árvore criada, também é possível: (i) realizar buscas de palavras em todo o modelo de *features* representado, como ilustrado na Figura 33, a palavra ”internet” é

buscada, e o nó da árvore que contém a palavra é marcada com a cor rosa. Para isso ser realizado é necessário selecionar o campo de busca, encontrado na parte inferior à direita da árvore, e logo após digitar a palavra que deseja ser buscada; (ii) verificar o caminho de um nó até a raiz da árvore, como ilustrado na Figura 33 um item da árvore é selecionado, com isso um percurso é traçado do nó selecionado até a raiz da árvore; e (iii) aplicar *zoom* na árvore, a diferença pode ser percebida entre a Figura 31 e 32, em que a primeira está mais aproximada que a segunda.

As funcionalidades que a nova árvore proporciona ajudam na melhora da usabilidade da ferramenta, pois o usuário passa a encontrar itens perdidos em uma árvore muito extensa com a função de busca, por exemplo, e também visualizar quais *features* influenciam na ativação de um determinada das *features* até a raiz da árvore.

Figura 33 – Funcionalidades da nova árvore do modelo de *features*.



Fonte: Elaborada pelo autor

5.4 Validação das Visualizações Propostas

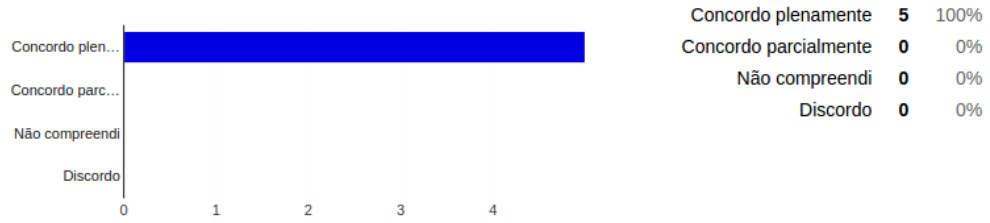
Um questionário foi aplicado para os mesmos engenheiros de domínio que responderam ao questionário apresentado na Subseção 5.2, que teve como objetivo validar as visualizações desenvolvidas, ilustradas na Subseção 5.2. Essa validação buscou assegurar se as visualizações foram adequadas e se atendem às necessidades dos engenheiros de domínio, ou seja, a confirmação de que estes consigam responder as suas perguntas, já apresentadas na Subseção 5.2.

As visualizações foram dispostas para os participantes e foram realizadas perguntas, como: “Você concorda que a visualização acima responde a pergunta: “Qual a variabilidade dinâmica do modelo de *features*?””, cada pergunta poderia ser respondida com as seguintes alternativas: concordo plenamente, concordo parcialmente, não compreendi e discordo. Para cada pergunta, também existe a possibilidade de realizar observações, as mesmas podem ser usadas para obter opiniões dos participantes. (ver Apêndice B).

A seguir são representadas as perguntas para cada visualização e seu nível de aceitação.

1. **Extensibilidade dos Modelos**

Figura 34 – Você concorda que a visualização mostra qual modelo de *features* é de fácil extensibilidade?

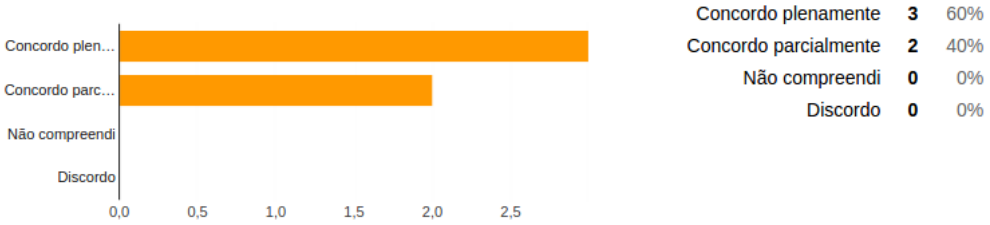


Fonte: Elaborada pelo Autor.

a) Observação 1: *Fácil visualização*

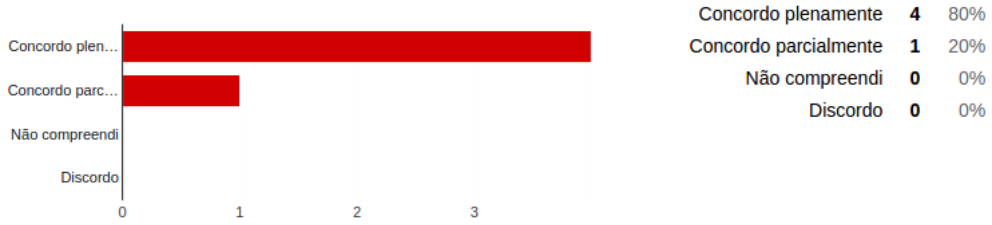
2. **Variabilidade Dinâmica**

Figura 35 – Você concorda que a visualização reponde a pergunta: “Qual a variabilidade dinâmica do modelo de *features*?”



Fonte: Elaborada pelo Autor.

Figura 36 – Você concorda que a visualização responde a pergunta: "Qual dos contextos presentes em um modelo de *features*, possui mais dinamicidade em termos de ativação e desativação?"



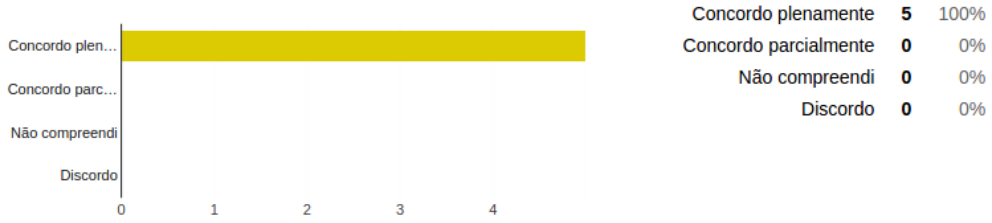
Fonte: Elaborada pelo Autor.

a) Observação 1: *Se variabilidade corresponde apenas ao número de features ativadas, a primeira pergunta responde totalmente isso. Entretanto, se há o desejo de saber "quais" features foram ativadas, então não.*

b) Observação 2: *No exemplo dado, temos apenas 3 contextos, e se por acaso o modelo tiver muitos contextos? Este tipo de visualização pode não ser viável.*

3. **Complexidade Estrutural de um modelo de *features***

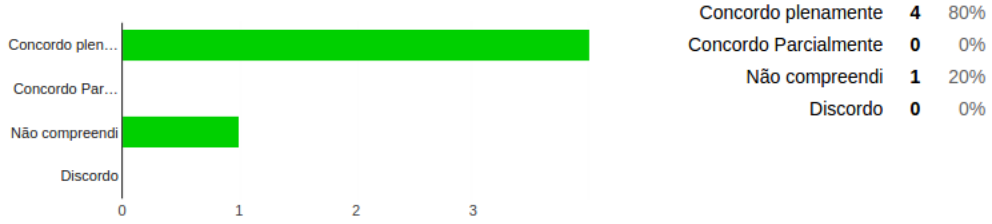
Figura 37 – Você concorda que a visualização responde a pergunta: "Qual a complexidade do modelo de *features*?"



Fonte: Elaborada pelo Autor.

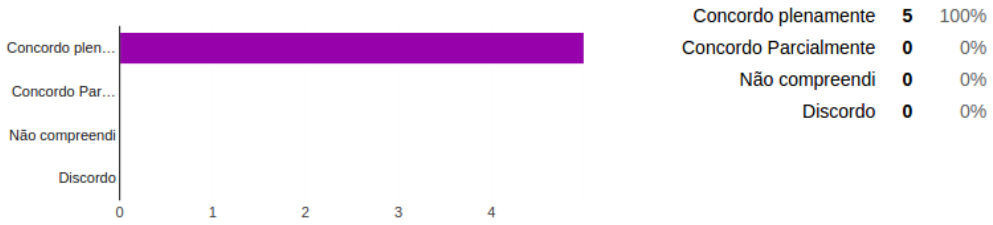
4. **Complexidade Estrutural de um conjunto de versões de um modelo**

Figura 38 – Você concorda que a visualização responde a pergunta: "Qual o impacto que a linha sofreria ao estender uma determinada *feature*?"



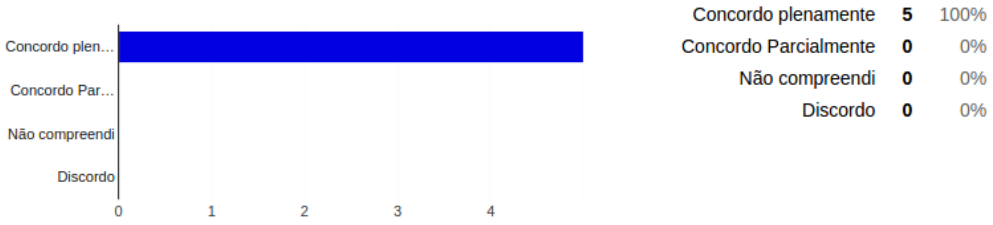
Fonte: Elaborada pelo Autor.

Figura 39 – Você concorda que a visualização responde a pergunta: “Qual o impacto a linha de produtos sofrerá quando uma mudança ocorre no modelo de *features*?”



Fonte: Elaborada pelo Autor.

Figura 40 – Você concorda que a visualização responde a pergunta: “Qual a evolução da complexidade do modelo de *features* quando uma *feature* é adicionada e removida?”

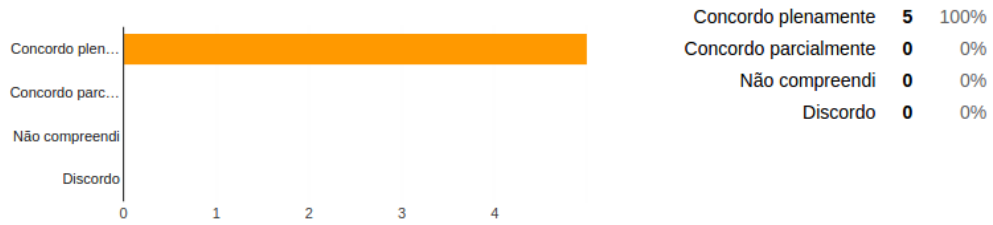


Fonte: Elaborada pelo Autor.

a) Observação 1: *Acho que a segunda pergunta engloba a primeira.*

5. **Variabilidade Estática**

Figura 41 – Você concorda que a visualização responda a pergunta: “Qual o aumento no número de configurações a partir da inclusão de novas *features*?”



Fonte: Elaborada pelo Autor.

Após a observação dos resultados da validação das visualizações, foi possível perceber que o nível de aceitação dos participantes é alto, com um total de três observações negativas em duas visualizações e um não compreendimento em uma visualização.

Para a visualização “Extensibilidade dos Modelos”, pode-se observar que seu nível de aceitação é máxima, contendo apenas uma observação positiva: (i) “Fácil visualização".

Para a visualização “Variabilidade Dinâmica”, pode-se observar que para a primeira pergunta o nível de aceitação é alta, com três respostas em “Concordo plenamente” e duas respostas em “Concordo parcialmente”. Não foi deixado nenhuma observação do porquê dos votos aplicados em “Concordo parcialmente”. Já para a segunda pergunta, o nível de aceitação é alto, com apenas uma resposta “Concordo parcialmente”, e quatro respostas “Concordo plenamente”. Duas observações foram deixadas para essas perguntas, são elas: (i) “Se variabilidade corresponde apenas ao número de *features* ativadas, a primeira pergunta responde totalmente isso. Entretanto, se há o desejo de saber “quais” *features* foram ativadas, então não.”, essa observação reflete a falta de uma característica na visualização, que seria a possibilidade de verificar quais *features* estão ativadas no modelo, contudo as medidas dos modelos de qualidade não nos trazem quais *features* estão ativadas, trazem apenas valores, como o número de *features* ativadas, por exemplo; e (ii) “No exemplo dado, temos apenas 3 contextos, e se por acaso o modelo tiver muitos contextos? Este tipo de visualização pode não ser viável.”, essa observação remete a questão de uma possível quantidade alta de contextos em um modelo, afetando a legibilidade da visualização, por existir muitos dados a serem visualizados, contudo isso não é um problema para essa visualização, pois o *TreeMap* é uma visualização que suporta muito níveis de dados. Na visualização desenvolvida na ferramenta é possível aplicar *zoom* nas camadas do *TreeMap*, isso pode ser utilizado para visualizar dados comprimidos que possam dificultar a leitura dos dados do *TreeMap*, a figura 42 ilustra a possibilidade de aplicar o *zoom*, em que é utilizada no contexto “C1”. O erro que possivelmente levou o participante a escrever essa observação foi a falta de uma ilustração que representa a funcionalidade de *zoom* para essa visualização.

Figura 42 – *Zoom* aplicado a uma camada do *TreeMap* desenvolvido.

Fonte: Elaborada pelo Autor.

Para a visualização “Complexidade Estrutural de um modelo de features”, pode-se observar que seu nível de aceitação é máxima.

Para a visualização “Complexidade Estrutural de um conjunto de versões de um modelo”, pode-se observar que para a primeira pergunta o nível de aceitação é alta, com quatro respostas “Concordo plenamente” e um “Não compreendi”. Já para a segunda e terceira o nível de aceitação é máxima. Uma observação foi deixada para essas perguntas: (i) “Acho que a segunda pergunta engloba a primeira.”, essa observação remete a questão de que a segunda pergunta é bem parecida com a primeira, essa problema poderia ter sido evitado se nos resultados obtidos no questionário A, houvesse um filtro nas perguntas para a verificação das que remetem ao mesmo significado.

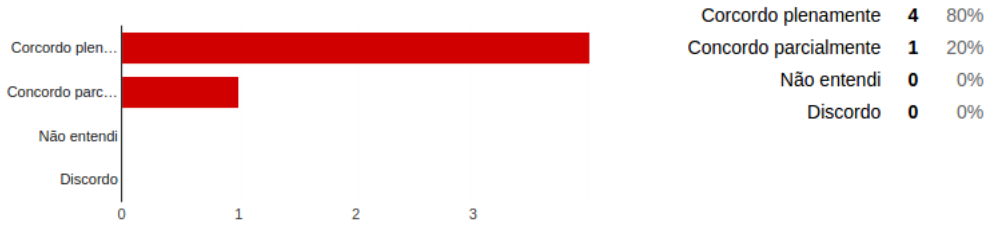
Para a visualização “Variabilidade Estática”, pode-se observar que seu nível de aceitação é máxima.

Para a validação da visualização da nova árvore, foi adicionado no mesmo questionário ilustrações que comparam a árvore antiga em relação a mais atual, e logo após são realizadas as seguintes perguntas: (i) “Você concorda que houve melhoras no entendimento em relação a antiga representação?”; (ii) “Você concorda que as novas funcionalidades garantem uma melhor usabilidade?”; (iii) “Você concorda estar satisfeito com a nova árvore de visualização?”; e (iv) “Considerações sobre a visualização e/ou as respostas acima”. Com as seguintes faixas de respostas: concordo plenamente, concordo parcialmente, não compreendi e

discordo.

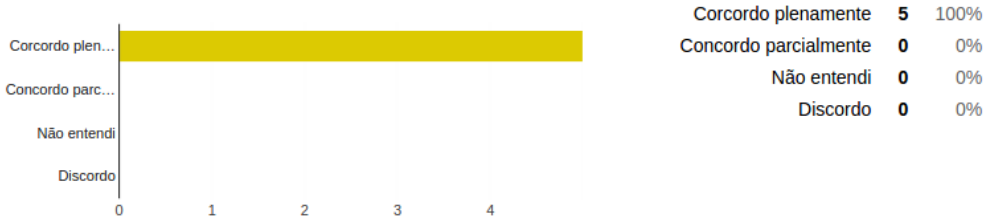
A seguir temos a relação das perguntas e respostas obtidas da validação realizada para a nova árvore do modelo de *features*.

Figura 43 – Você concorda que houve melhoras no entendimento em relação a antiga representação?



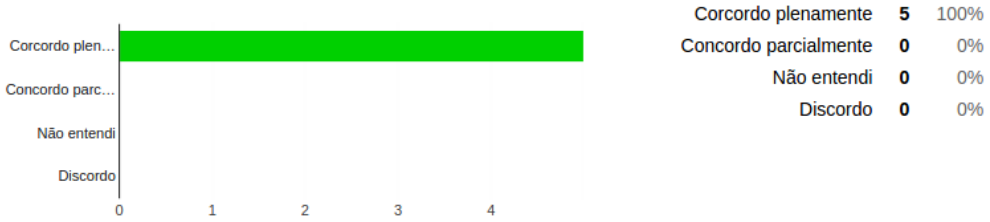
Fonte: Elaborada pelo Autor.

Figura 44 – Você concorda que as novas funcionalidades garantem uma melhor usabilidade?



Fonte: Elaborada pelo Autor.

Figura 45 – Você concorda estar satisfeito com a nova árvore de visualização?



Fonte: Elaborada pelo Autor.

Após a observação dos resultados da validação da nova árvore, foi possível perceber que o nível de aceitação é alto, com um total de 14 “Concordo plenamente” e apenas um “Concordo parcialmente” aplicada a primeira pergunta.

As perguntas dois e três tiveram aceitação máxima dos participantes, diferente da primeira que obteve um voto “Concordo parcialmente” e quatro votos “Concordo plenamente”; não foi realizada nenhuma observação para o voto realizado em “Concordo parcialmente”.

5.5 Considerações Finais

Como conclusão da validação, temos que há uma alta aceitação do trabalho desenvolvido por parte dos participantes do questionário. No entanto, são necessárias modificações em algumas visualizações das medidas de qualidade, de acordo com as observações recebidas por parte dos participantes. Pretende-se inserir essas modificações em trabalhos futuros, melhorando continuamente a ferramenta. Os resultados mostram que o que foi desenvolvido poderá auxiliar os engenheiros de domínio a melhorar a avaliação sobre as medidas de qualidades dos modelos de *features* com relação ao aspecto de manutenibilidade.

6 PROJETO E DESENVOLVIMENTO DA EXTENSÃO DA FERRAMENTA

Esta Seção apresenta os procedimentos realizados para o desenvolvimento da extensão da ferramenta DyMMer. Primeiramente são listados os requisitos funcionais que precisam ser supridos para obtemos a ferramenta proposta; em seguida é explicado um pouco da arquitetura da ferramenta estendida; logo após é explicado o desenvolvimento do repositório e finalmente das funcionalidades acrescentadas na extensão realizada.

6.1 Requisitos Funcionais

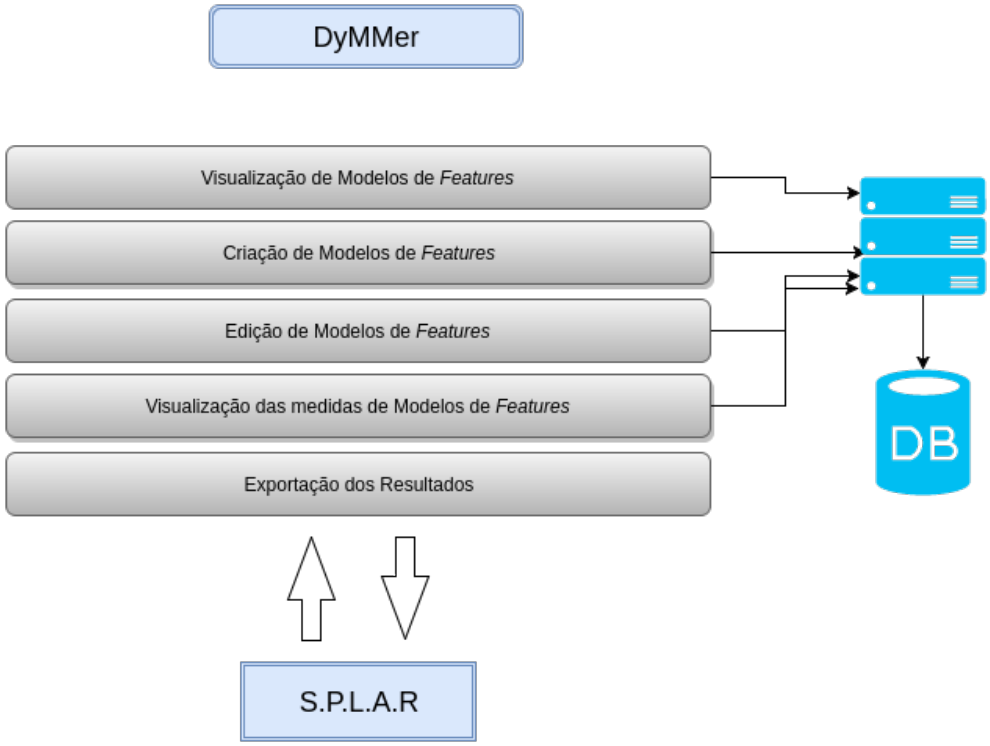
Primeiramente foram definidos os requisitos funcionais para a extensão da ferramenta. Estes requisitos foram baseados nas necessidades de visualização das medidas dos modelos de *features*, gerenciamento dos modelos de *features* e criação de novos modelos de *features*, tudo na ferramenta. Seguem os requisitos:

- Criação de uma nova forma de visualizar a árvore dos Modelo de *Features*;
- Criação de um repositório para a DyMMer em que se pode Editar, Salvar e Deletar os modelos de *features*;
- Criação de novos modelos de *features*, sendo possível a criação de um modelo estático com adição ou remoção de *features*;
- Edição de modelos de *features*, sendo possível a adição ou remoção de features em modelos já existentes; e
- Criação do módulo de visualização das medidas dos modelos de *Features*.

6.2 Arquitetura

Seguindo a base arquitetural da antiga DyMMer, o sistema foi divido nas funcionalidades: Visualização de Modelos de *Features*, Criação de Modelos de *Features*, Edição de Modelos de *Features*, Visualização das medidas de Modelos de *Features* e Exportação dos Resultados, apresentado na Figura 46.

Figura 46 – Funcionalidades da ferramenta.



Fonte: Elaborada pelo autor

6.2.1 Estruturas de Camadas

As camadas representadas são: (i) Visualização de modelos de *features*, permite-se a visualização de Modelos de *Features*, para modelos que se encontram no repositório da DyMMer ou modelos armazenados na máquina do usuário; (ii) Criação de Modelos de *Features*, permite-se criar novos modelos estáticos, que podem ser adicionados ao repositório ou armazenados na máquina do usuário; (iii) Edição de modelos de *features*, permite-se editar novos modelos, que poden ser armazenados no repositório da DyMMer ou na máquina do usuário; (iv) Visualização das medidas de modelos de *features*, permite-se a ilustração das medidas dos modelos a partir das visualizações criadas na estensão da ferramenta; e (v) Exportação dos resultados, permite a exportação dos resultados das medidas aplicadas aos modelos e seus contextos no formato *Excel*; e (vi) S.P.L.A.R, que é uma biblioteca responsável pela análise automática dos modelos de *features*, tratada com mais detalhes na Seção 2.6.

Além das camadas apresentadas, é ilustrado do lado direito da Figura 46, o repositório dos modelos de *features*. Lugar onde é possível salvar, editar e remover modelos de *features*. Explicado com mais detalhes na Seção 6.3.

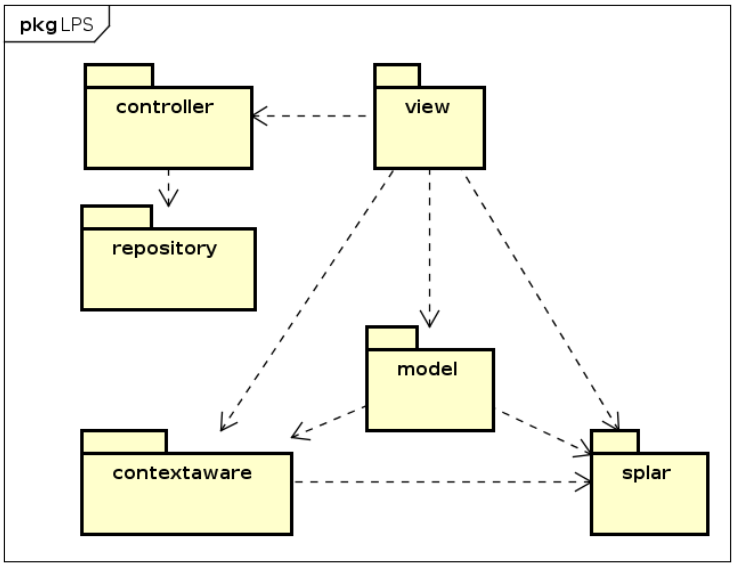
6.2.2 Metas e Restrições de Arquitetura

Para execução do sistema é necessária à instalação do JDK 1.6 ou superior, caso contrário o sistema não funcionará na máquina desejada.

6.2.3 Divisão de Pacotes

A Figura 47 ilustra os pacotes contidos no projeto da ferramenta estendida DyMMer.

Figura 47 – Diagrama de pacotes.



Fonte: Elaborada pelo autor

1. **Pacote *View***

   Responsável por receber as entradas realizadas pelo usuário e exibir diferentes dados de acordo com a ação recebida.

   - Trees: Responsável por construir as arvores dos modelos na tela.
   - Panels: Responsável por conter todos os arquivos de paineis da aplicação.
   - List: Responsável por manipular listas relacionadas ao modelo.

2. **Pacote *Model***

   Define a estrutura do modelo de *features* para o *SPLOT*.

   - Context: Responsável por definir diferentes tipos de medidas de acordo com o contexto dos modelos.
   - Normal: Responsável por definir uma interface do modelo sem contexto.
   - Xml: Responsável por traduzir um modelo de *feature* para XML em

diferentes tipos de formatos para se adequar as ferramentas que utilizam este XML.

- ContextAware: Responsável pelo gerenciamento dos estados do modelo de acordo com contexto selecionado.

3. **Pacote *Splar***

   Responsável por conter os arquivos da biblioteca S.P.L.A.R.

   - Core: Representa e implementa as restrições de integridade do modelo de feature, o próprio modelo, heurísticas e utilidades que tratam a cerca do modelo.

4. **Pacote *Controller***

   Responsável pelo controle de exportação da DyMMer, do *web content view* utilizado na ferramenta e Xmls que podem ser salvos, excluídos e editados no repositório da DyMMer;

   - Export: Responsável por exportar arquivos de acordo com a opção selecionada pelo usuário.
   - Browser: Responsável pelo controle da criação de um painel para a visualização das medidas utilizando a biblioteca *JxBrowser*;
   - Xml: Responsável pelo controle realizado no repositório para salvar os dados dos modelos de *features* no repositório da DyMMer.

5. **Pacote *repository***

   Responsável pela comunicação realizada com a API da DyMMer.

6. **Pacote *ContextAware***

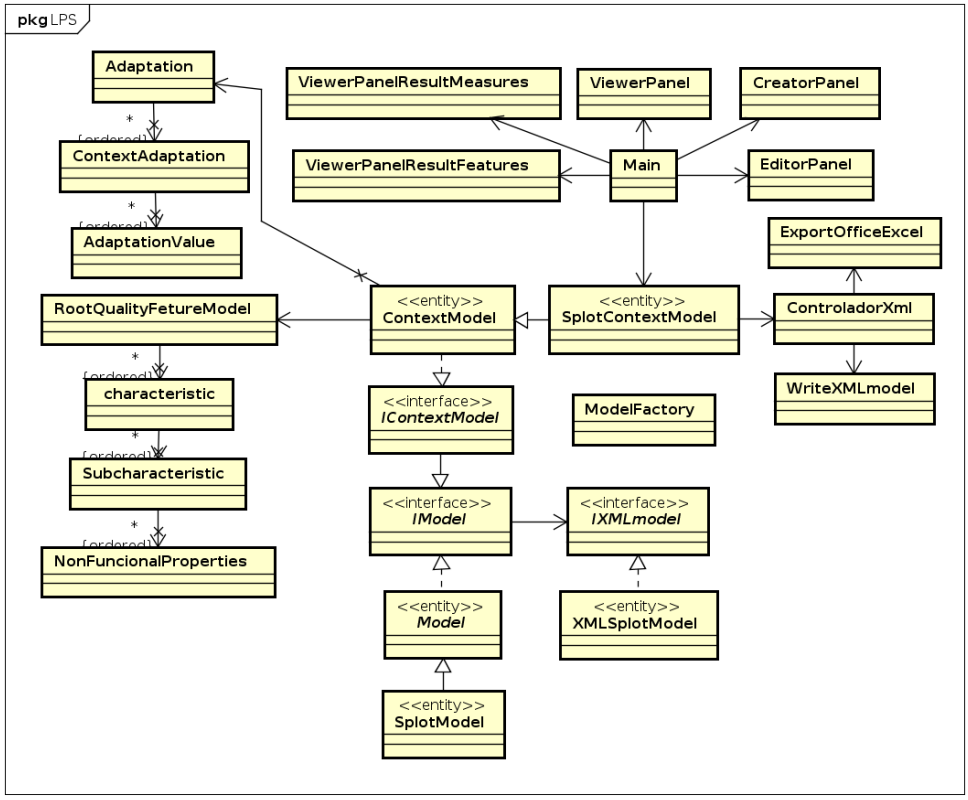
   Responsável pelo gerenciamento dos estados do modelo de acordo com contexto selecionado.

6.2.4 Diagrama de classe

A Figura 48 apresenta as novas relações existentes entre as classes da ferramenta estendida. Ao lado esquerdo do diagrama temos a relação entre as classes das árvore de adaptação e de requisitos não funcionais, que não são tradadas neste trabalho. Os painéis "ViewerPanelResultMeasures", "ViewerPanel", "CreatorPanel", "EditorPanel" e "ViewerPanelResultFeatures", tem total relação com a interação que o usuário terá com a aplicação. Ao lado direito temos o "ControladorXml", que faz o controle da escrita no XML e

exportação de dados no formato do “Excel”. Na parte central temos os modelos, que fazem o total controle sobre a lógica que ocorrem nos modelos de *features* manuseados pela ferramenta.

Figura 48 – Diagrama de classes.



Fonte: Elaborada pelo autor

6.3 Repositório da DyMMer

Primeiramente foi necessário a criação de um repositório para a DyMMer, onde é possível adicionar, editar e remover modelos de *features* de LPSDs. Os modelos de *features* que são adicionados podem ser extraídos do repositório S.L.O.T, *MAcchiaTO Repository*[1] ou criados diretamente na ferramenta DyMMer. O objetivo do repositório da DyMMer é de apenas manter os modelos de *features* criados ou importados na ferramenta pelos usuários da mesma.

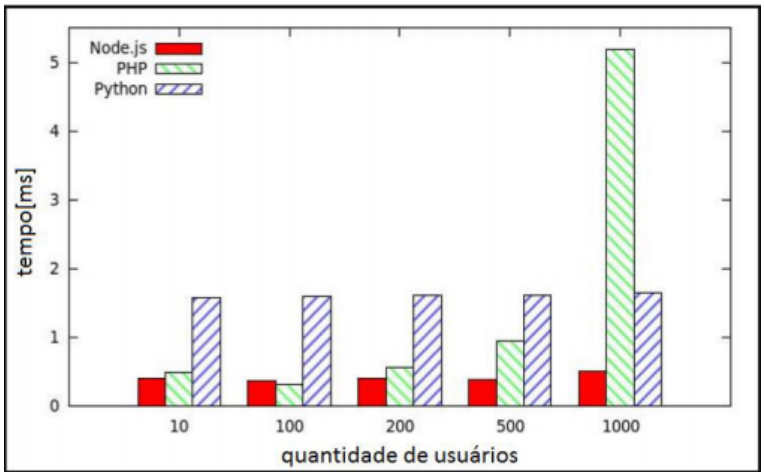
6.3.1 Linguagens para o desenvolvimento do Web Service

O lado do servidor foi desenvolvido em Node.js, que é uma plataforma para desenvolvimento de aplicações *server-side* baseadas em rede utilizando *JavaScript* e o *V8 JavaScript Engine* (LEI; MA; TAN, 2014).

[1] http://carlabezerra.great.ufc.br/macchiato/

De acordo com Lei, Ma e Tan (2014), o Node.js juntamente com o *framework express*, aumenta o nível de produtividade, diminuindo o esforço de desenvolvimento, dessa forma o trabalho realizado é finalizado mais rápido e com alta flexibilidade. Como se pode observar na Figura 49, o Node.js tem um tempo de resposta mais baixo em relação ao PHP e Python-Web, quando rodadas aplicações semelhantes; dessa forma, com o Node.js, é possível atender a mais requisições por segundo do que as demais linguagens comparadas.

Figura 49 – Desempenho do Node.js.



Adaptado de Lei, Ma e Tan (2014)

6.3.2 Banco de Dados Não Relacional MongoDB

MongoDB é um banco de dados não relacional, orientado a documentos, baseado no formato JSON. As vantagens de se utilizar um banco de dados orientado a documentos é a não necessidade de alterações de uma estrutura no banco, como é realizado em um banco de dados relacional, por exemplo. Se houver adições de novos parâmetros no banco, não é necessário uma modificação em grande escala (POULTER; JOHNSTON; COX, 2015).

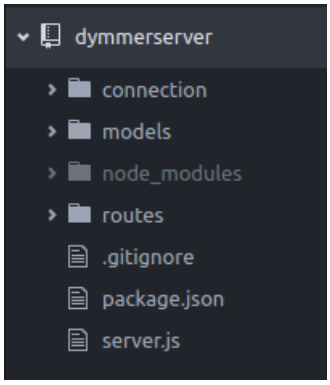
6.3.3 Web Service

Com a linguagem e o banco de dados não relacional escolhidos, foi possível realizar a implementação do *web service*[2] . O projeto foi desenvolvido utilizando o editor de texto *ATOM*[3], e foi dividido em: (i) "*connection*", onde se encontra o arquivo de conexão com o banco de dados; (ii) "*models*", onde se encontram os arquivos que criam o modelo o qual é espelho para as informações que são adicionadas no banco de dados; (iii) "*routes*", onde se encontram as rotas

[2] https://luanpereiralima@bitbucket.org/luanpereiralima/dymmerserver.git
[3] https://atom.io/

para dar acesso as requisições *HTTP* para a aplicação; (iv) ”*node_modules*”, onde se encontram as bibliotecas que são utilizadas na aplicação; (v) ”.*gitignore*”, é o arquivo responsável pelo controle de arquivos que não podem ser enviados para o repositório, controlado pelo controle de versão git; (vi) ”*package.json*”, é o arquivo de configuração do *NPM* (responsável pelo gerenciamento de dependências do projeto); e (vii) ”*server.js*”, é o arquivo responsável pela configuração geral da aplicação, como ilustrado na Figura 50.

Figura 50 – Projeto do Repositório da DyMMer.



Fonte: Elaborada pelo autor

6.3.4 Bibliotecas Utilizadas

O lado servidor necessitou de algumas bibliotecas *JavaScript* para o desenvolvimento, que são: (i) *body-parser*, responsável pelo parse que ocorre no corpo de uma requisição recebida pelo servidor, utilizado para o servidor trabalhar com recebimento de dados no formato *JSON*; (ii) *express*, é uma ferramenta robusta que fornece uma interface para a ferramenta Node.js, capaz de provê ao desenvolvedor mecanismos para manipulação de roteamento e operações HTTP, utilizado para criação da API pública; (iii) *mongoose*, é responsável pela validação dos modelos que são utilizados para o banco de dados não relacional *mongodb*; e (iv) *mongodb*, é o *driver* oficial do banco de dados não relacional *mongodb* para o *Node.js*.

6.3.5 Rotas da API

Rotas são caminhos utilizados para dar acesso a funcionalidades que uma API fornece. A API da DyMMer possui três rotas que podem ser acessadas, são elas: (i) Rota para deletar modelos de *features*, deve ser requisitada com a passagem do parâmetro "id"do modelo a

partir do método *DELETE*; (ii) Rota para acessar os modelos de *features*, deve ser requisitada a partir do método *GET*; e (iii) Rota para adicionar ou alterar um modelo de *features*, deve ser requisitada com a passagem do parâmetro "scheme"do modelo em formato de *JSON*, por meio do método *POST*. Ver Tabela 1.

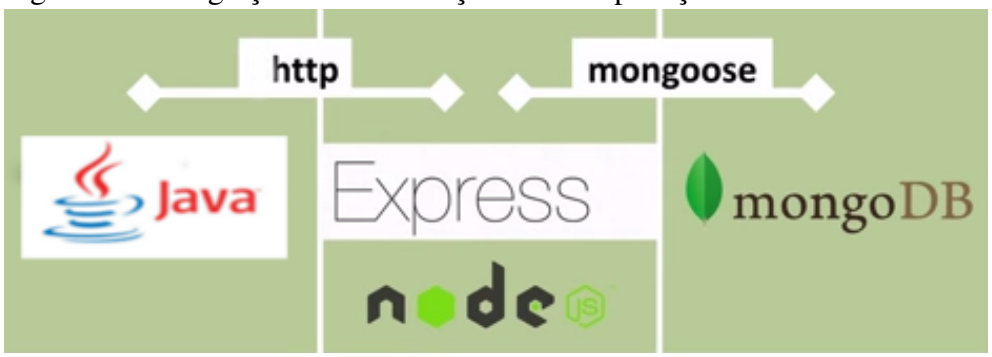
Tabela 1 – Rotas da API.

URI	Tipo de requisição	Descrição	Parâmetro	Retorno
/features	GET	Obtêm os modelos de features	Nenhum	JSON
/features	POST	Salva ou Edita um modelos de features	JSON	Nenhum
/features	DELETE	Deleta um modelo de features	_id - String	Nenhum

6.3.6 Integração e Comunicação

Como ilustrado na Figura 51, a DyMMer se comunica com o *web service* via protocolo *HTTP*, podendo acessar todas as rotas disponíveis pela API. Depois do acesso realizado pela ferramenta, o *express* juntamente com o Node.js utilizam o *mongoose* para o acesso ao banco de dados não relacional *MongoDB* e retorna os dados em *JSON* para a Aplicação.

Figura 51 – Integração e Comunicação entre a aplicação e o *web service*.



Fonte: Elaborada pelo autor

Como ilustrado na Figura 52, a esquerda da imagem são apresentados dois métodos em Java que realizam requisições na API da DyMMer, e do lado direito é ilustrado uma rota com o método *GET* que recebe a requisição que a ferramenta DyMMer realiza.

Figura 52 – Códigos da requisição realizada pela ferramenta DyMMer e recebimento na API.

```java
private OkHttpClient http;

    public RepositoryXml(OkHttpClient okHttpClient) {
        this.http = okHttpClient;
    }

    public String requestString(Request request){
        try {
            Response response = http.newCall(request).execute();
            String bodyString = response.body().string();
            response.close();
            return bodyString;
        } catch (IOException e) {
            e.printStackTrace();
            return null;
        }
    }

    public boolean requestBoolean(Request request){
        try {
            Response response = http.newCall(request).execute();
            response.close();
            return response.code()==200;
        } catch (IOException e) {
            e.printStackTrace();
            return false;
        }
    }
```

```javascript
var Feature = require('../models/feature');
var express = require('express');
var router = express.Router();

/**
 * @api {get} URL_SERVER/features/ Obter Modelos
 * @apiName Obtem os Modelos de Features
 * @apiGroup Features Model
 *
 */
router.get('/', function(req, res){

  Feature.buscar(
    function(err, features){

    if(err){
      console.log(err);
      return res.sendStatus(500);
    }

    return res.json(features);
  })

});
```

Fonte: Elaborada pelo autor

6.4 Extensão da ferramenta DyMMer

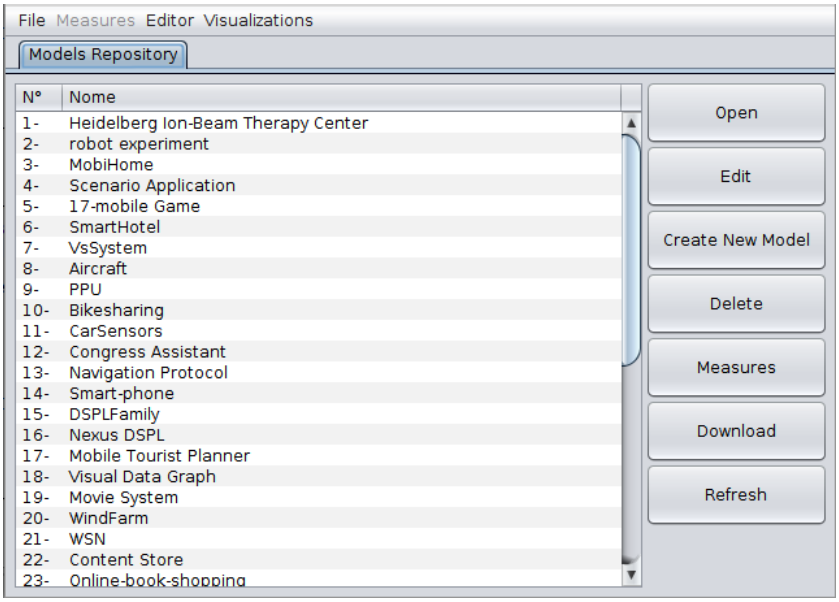
A aplicação DyMMer foi estendida para suprir os novos requisitos, especificados na Seção 6.1. nesta Subseção será abordada as adições desenvolvidas na ferramenta.

6.4.1 Listagem dos modelos de features do repositório da DyMMer

Primeiramente foi adicionado a ferramenta um painel em que fosse possível gerenciar os modelos do repositório da DyMMer, como ilustrado na Figura 53. No lado esquerdo da Figura 53 fica a lista onde é carregado todos os modelos do repositório, onde é possível selecionar um ou mais modelos para realizar ações, como: (i) abrir modelos de *features* do repositório, para isso é preciso selecionar um ou mais modelos na lista e clicar no botão "*Open*", encontrado a direita da lista de modelos do repositório, essa opção é explicada com mais detalhes na Seção 5.3; (ii) editar modelos de *features* do repositório, para isso é preciso selecionar um ou mais modelos na lista e clicar no botão "*Edit*", encontrado logo a direita da lista de modelos do repositório, essa opção é explicada com mais detalhes na Seção 6.4.2; (iii) criar novos modelos de *features* estáticos, para isso é preciso clicar no botão "*Create New Model*", encontrado logo a direita da lista de modelos do repositório, essa opção é explicada com mais detalhes na Seção 6.4.3; (iv) deletar modelos de *features* do repositório, para isso é preciso selecionar um modelo na lista e clicar no botão "*Delete*", encontrado logo a direita da lista de modelos do repositório; (v) verificar todas as medidas relacionadas a um modelo de *features* do repositório, para todos os contextos presentes no mesmo, para isso é preciso selecionar um modelo na lista e clicar no

botão "*Measures*", encontrado logo a direita da lista de modelos do repositório; (vi) Realizar *download* de modelos de *features* do repositório, para isso é preciso selecionar um modelo na lista e clicar no botão "*Download*", encontrado logo a direita da lista de modelos do repositório; e (vii) Recarregar os itens da lista de modelos de *features*, para isso é preciso selecionar um modelo na lista e clicar no botão "*Refresh*", encontrado logo a direita da lista de modelos do repositório;

Figura 53 – Visão principal da DyMMer.

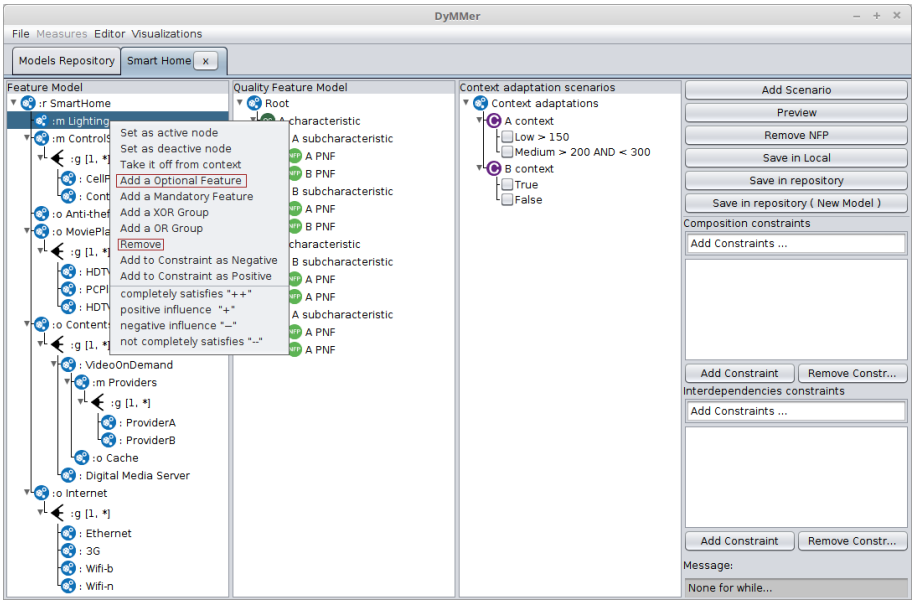


Fonte: Elaborada pelo autor

6.4.2 Edição de modelo de features já existentes

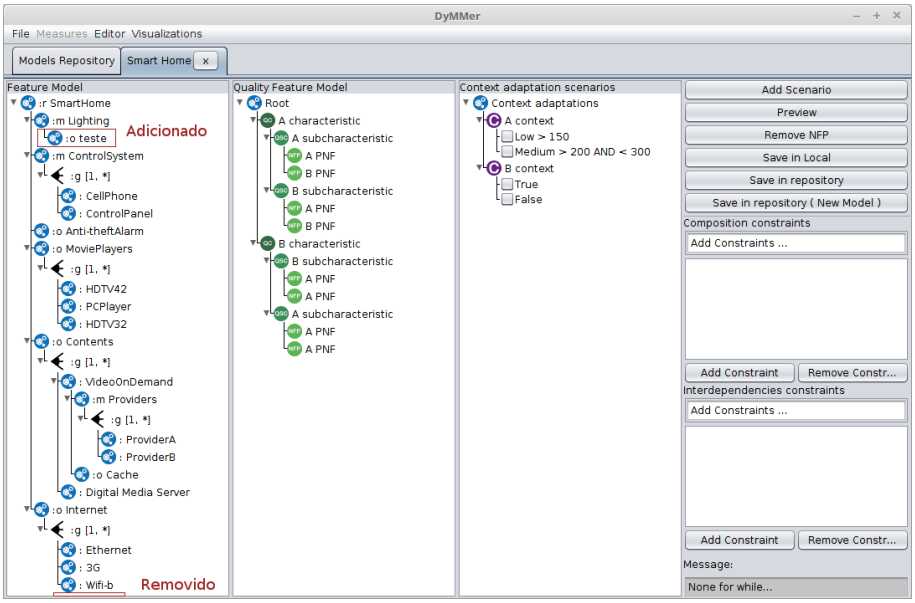
Na área de Edição estendida da ferramenta DyMMer é possível adicionar ou remover *features* opcionais, obrigatórias, agrupadas, grupos xor e grupo or de modelos já existentes; como pode-se observar na Figura 54, foi adicionado uma *feature* opcional com o nome "teste"e removida a *feature* agrupada com o nome "*Wifi-n*", o resultado é ilustrado na Figura 55.

Figura 54 – Adição e remoção de uma *feature* em um modelo de *features*.



Fonte: Elaborada pelo autor

Figura 55 – Pós Adição e remoção de uma *feature* em um modelo de *features*.



Fonte: Elaborada pelo autor

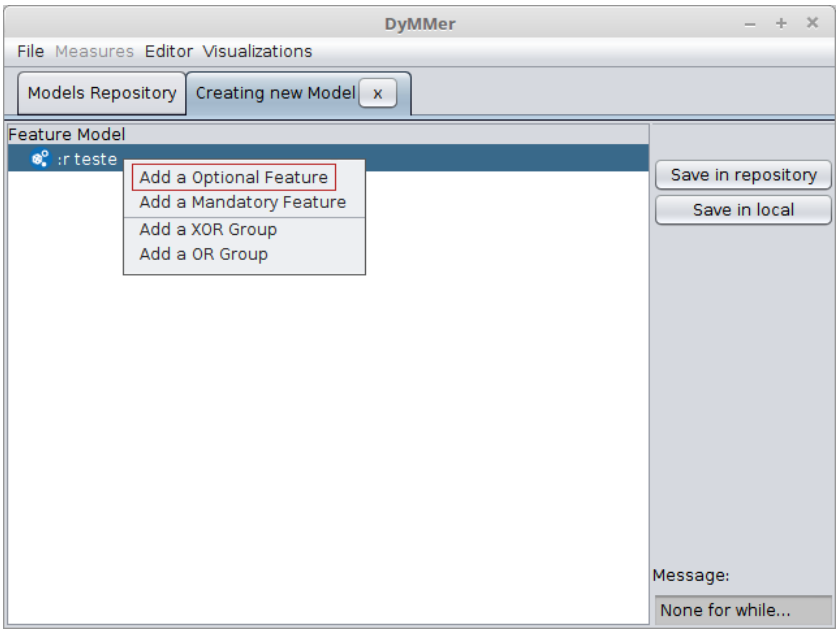
Após o modelo ser editado, pode-se realizar as seguintes operações: (i) pode-se ter um *preview* da edição realizada, se o usuário tiver a necessidade de visualizar o modelo, como mostrado na Subseção 5.3, antes de salvá-lo de alguma forma, o mesmo pode selecionar a opção "*Preview*", ilustrado do lado superior a direita da Figura 55; (ii) pode-se salvar o modelo no repositório da DyMMer, por meio do botão "*Save in repository*"é possível sobrescrever o modelo que está sendo editado no repositório, já por meio do botão "*Save in repository (New Model)*", é

possível salvar como um novo modelo o modelo que está sendo editado, os botões são ilustrados no lado superior a direita da Figura 55; e (iii) pode-se armazenar o modelo editado no diretório local do usuário, por meio do botão "*Save in Local*", ilustrado do lado superior a direita da Figura 55, é possível salvar o modelo editado em um diretório especificado pelo usuário.

6.4.3 Criação de modelos de features estáticos

Para contribuir com o avanço do repositório, a ferramenta DyMMer estendida pode criar novos modelos de *features* estáticos. Pode-se adicionar no modelo *features* opcionais, obrigatórias, agrupadas, de agrupamento *xor* e de agrupamento *or*, conforme apresentado na Figura 56. Após a finalização do modelo criado, ele pode ser adicionado no repositório da DyMMer ou armazenado na máquina do usuário.

Figura 56 – Exemplo de adição de uma *feature* opcional.



Fonte: Elaborada pelo autor

6.5 Considerações Finais

As Subseções anteriores apresentaram o desenvolvimento aplicado na ferramenta para suprir os requisitos de Criação de um repositório para a DyMMer, Criação de novos modelos de *features* e Edição de modelos de *features*. Dessa forma, a ferramenta passa a possuir uma maior gama de funcionalidades, aumentando o apoio à atividades dos engenheiros de domínio.

7 CONSIDERAÇÕES FINAIS

Esta Seção descreve as considerações do finais do trabalho. Primeiramente serão apresentadas as possíveis ameaças à validade dos resultados obtidos. Em seguida, é apresentado um levantamento sobre os possíveis trabalhos futuros. Logo após são apresentadas as contribuições deste trabalho. E finalmente, são descritas as conclusões sobre a execução deste trabalho.

7.1 Ameaças à Validade

A possível ameaça deste trabalho é sobre a efetividade do questionário realizado, uma vez que a quantidade de participantes foi baixa, contando apenas com cinco pessoas. Três participantes selecionados faziam parte do mesmo grupo de estudos do aplicador, devido às necessidade de pessoas com conhecimento prévio nos tópicos envolvidos no estudo. Dessa forma eles podem ter sido influenciados pela relação com o aplicador. Para resolver esta ameaça o estudo de observação precisa ser aplicado com outros especialistas.

Outra possível ameaça é a questão do conhecimento dos especialistas que participaram do questionário. Pode acontecer que os participantes não sabiam sobre o assunto relacionado. Para resolver esta ameaça é preciso envolver participantes mais experimentes em futuras validações.

7.2 Trabalhos Futuros

Com os resultados obtidos deste trabalho, foi possível observar melhorias que poderiam ser realizadas em trabalhos futuros.

As visualizações criadas respondem a poucas perguntas, e apenas para a característica de manutenibilidade, por isso, poderiam ser criadas novas visualizações que possam responder a mais perguntas e que remetesse a características diferentes.

A validação ocorrida para o trabalho desenvolvido, só envolvia as visualizações, por isso deveria ser realizado a validação não apenas das visualizações, mas também da usabilidade da ferramenta estendida

As visualizações criadas não contam com um sistema de ajuda, que poderia ser utilizado para explicar as visualizações para alguns usuários que não conseguem compreender a informação que a visualização deseja passar.

Após a validação realizada nas visualizações criadas, foram realizadas observações de melhorias por parte dos participantes. Dessa forma para trabalho futuro, realizar essas modificações sugeridas.

7.3 Contribuições

As principais contribuições identificadas com a realização deste trabalho foram:

- Criação de visualizações na ferramenta que ajudam no entendimento das medidas de qualidade;
- Criação de um repositório para o armazenamento e gerenciamento dos modelos de *features*;
- Extensão da árvore de representação dos modelos de *features* que ajuda a melhorar a usabilidade da ferramenta;
- Adição da funcionalidade de criação de modelos de *features* estáticos; e
- Adição da funcionalidade de edição de modelos de *features*, sendo possível a adição ou remoção de *features* em modelos já existentes.

7.4 Conclusões

As visualizações de dados realmente se mostraram importantes para um melhor entendimento das medidas de qualidade que a ferramenta dispõe, como observado na Seção de validação 5.4, o nível de aceitação dos participantes do questionário foi alto, contando apenas com a melhora em alguns pontos das visualizações criadas. A mudança ocorrida na árvore de visualização dos modelos de *features* não era crucial para o desenvolvimento do trabalho, mas mostrou bastante aprovação ao ser desenvolvida, pois além da mudança na forma de representar a árvore, há também funcionalidades na mesma, que chamaram bastante a atenção dos que à observaram.

O repositório criado para a ferramenta DyMMer vai fazer com que a mesma ganhe alto nível de interesse para os engenheiros de domínio, pois eles podem compartilhar seus trabalhos com outras pessoas, realizar buscas de modelos e visualizar medidas de seus próprios modelos em relação a outros existentes. Além de editar outros modelos e criar cópias dos mesmo no repositório como uma versão diferente.

Os pontos positivos da jornada na realização deste trabalho foram o alto aprendizado

adquirido sobre linhas de produtos de software, visualização de dados e medidas de qualidade das linhas de produtos, além de ajudar a deixar a comunidade acadêmica com algo que eles podem usar em suas pesquisas e trabalhos futuros.

A ferramenta DyMMer, utilizada como base para o desenvolvimento deste trabalho, além de não utilizar um padrão de projeto em específico, foi muito bem desenvolvida, isso facilitou no momento em que foi necessário estudá-la e entendê-la para a futura extensão.

Com o desenvolvimento deste trabalho, espera-se que as novas funcionalidades adicionadas a ferramenta ajudem os engenheiros de domínio a visualizar melhor as medidas de qualidade de manutenibilidade dos seus modelos de *features* e a árvore de modelos de *features*. Ajuda também na disseminação de conhecimento, pois o repositório de modelos de *features* é aberto a comunidade.

REFERÊNCIAS

BENAVIDES, D.; SEGURA, S.; RUIZ-CORTÉS, A. Automated analysis of feature models 20 years later: A literature review. **Information Systems**, Elsevier, v. 35, n. 6, p. 615–636, 2010.

BENCOMO, N.; HALLSTEINSEN, S.; ALMEIDA, E. S. D. A view of the dynamic software product line landscape. **Computer**, v. 45, n. 10, p. 36–41, 2012.

BEUCHE, D.; DALGARNO, M. Software product line engineering with feature models. **Overload Journal**, v. 78, p. 5–8, 2007.

BEZERRA, C. I.; ANDRADE, R. M.; MONTEIRO, J. M. S. Measures for quality evaluation of feature models. In: **Software Reuse for Dynamic Systems in the Cloud and Beyond**. Miami, FL, USA: Springer, 2015. p. 282–297.

BEZERRA, C. I.; MONTEIRO, J. M.; ANDRADE, R.; ROCHA, L. S. Analyzing the feature models maintainability over their evolution process: An exploratory study. In: ACM. **Proceedings of the Tenth International Workshop on Variability Modelling of Software-intensive Systems**. Salvador, Brazil, 2016. p. 17–24.

BEZERRA, C. I. M.; BARBOSA, J.; FREIRES, J. H.; ANDRADE, R. M. C.; MONTEIRO, J. M. S. Dymmer: A measurement-based tool to support quality evaluation of dspl feature models. In: SPRINGER. **International Conference on Software Product Lines**. [S.l.], 2016.

BORBA, C. C. Uma abordagem orientada a objetivos para as fases de requisitos de linhas de produtos de software. Universidade Federal de Pernambuco, 2009.

BOTTERWECK, G.; JANOTA, M.; SCHNEEWEISS, D. A design of a configurable feature model configurator. 2009.

CAPILLA, R.; BOSCH, J. The promise and challenge of runtime variability. **Computer**, IEEE, v. 44, n. 12, p. 93–95, 2011.

CAPILLA, R.; BOSCH, J.; TRINIDAD, P.; RUIZ-CORTÉS, A.; HINCHEY, M. An overview of dynamic software product line architectures and techniques: Observations from research and industry. **Journal of Systems and Software**, Elsevier, v. 91, p. 3–23, 2014.

CLEMENTS, P.; NORTHROP, L. **Software Product Lines: Practices and Patterns**. [S.l.]: Addison-Wesley Professional, 2001.

DURSCKI, R. C.; SPINOLA, M. M.; BURNETT, R. C.; REINEHR, S. S. Linhas de produto de software: riscos e vantagens de sua implantação. **Simpósio Brasileiro de Processo de Software. S. Paulo**, 2004.

FREITAS, C. M. D. S.; CHUBACHI, O. M.; LUZZARDI, P. R. G.; CAVA, R. A. Introdução à visualização de informações. **Revista de informática teórica e aplicada. Porto Alegre. Vol. 8, n. 2 (out. 2001), p. 143-158**, 2001.

HALLSTEINSEN, S.; HINCHEY, M.; PARK, S.; SCHMID, K. Dynamic software product lines. **Computer**, IEEE, v. 41, n. 4, p. 93–95, 2008.

HINCHEY, M.; PARK, S.; SCHMID, K. Building dynamic software product lines. **Computer**, IEEE, n. 10, p. 22–26, 2012.

LEI, K.; MA, Y.; TAN, Z. Performance comparison and evaluation of web development technologies in php, python, and node. js. In: IEEE. **Computational Science and Engineering (CSE), 2014 IEEE 17th International Conference on**. [S.l.], 2014. p. 661–668.

MENDONCA, M.; BRANCO, M.; COWAN, D. Splot: software product lines online tools. In: ACM. **Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications**. New York, NY, USA, 2009. p. 761–762.

NORTHROP, L.; CLEMENTS, P. Software product lines. **URL http://www.sei. cmu. edu/library/assets/Philips**, v. 4, n. 05, 2001.

PLEUSS, A.; BOTTERWECK, G. Visualization of variability and configuration options. **International Journal on Software Tools for Technology Transfer**, Springer, v. 14, n. 5, p. 497–510, 2012.

POHL, K.; BÖCKLE, G.; LINDEN, F. J. van der. **Software product line engineering: foundations, principles and techniques**. Berlin: Springer Science & Business Media, 2005.

POULTER, A. J.; JOHNSTON, S. J.; COX, S. J. Using the mean stack to implement a restful service for an internet of things application. In: IEEE. **Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on**. [S.l.], 2015. p. 280–285.

ROBAK, S. Feature modeling notations for system families. In: **International Workshop on Software Variability Management (SVM)**. Limerick, Ireland: [s.n.], 2003. p. 58.

SAIZ, J. E. M. **Feature Models Visualization Based on Ontology Framework**. Tese (Doutorado) — Thesis submitted to university of Vrije, Brussel, PP-22-23, 2008-09, 2009.

SILVA, F. A. P. da; NETO, P. A. d. M. S.; GARCIA, V. C.; MUNIZ, P. F. Linhas de produtos de software: Uma tendência da indústria. 2011.

URLI, S.; BERGEL, A.; BLAY-FORNARINO, M.; COLLET, P.; MOSSER, S. A visual support for decomposing complex feature models. In: IEEE. **Software Visualization (VISSOFT), 2015 IEEE 3rd Working Conference on**. Bremen, Germany, 2015. p. 76–85.

APÊNDICE A – QUESTIONÁRIO DE AVALIAÇÃO - SELEÇÃO DE VISUALIZAÇÕES PARA CARACTERÍSTICA DE MANUTENIBILIDADE DO MODELO DE FEATURES

A visualização das medidas de qualidade de modelos de features é um ponto importante para melhorar o entendimento da qualidade estrutural do modelo, dessa forma ajudando a obter uma melhor visão das estatísticas das medidas de um ou mais modelos de features. O objetivo dessa pesquisa é classificar as subcaracterísticas e medidas de qualidade que são relevantes para a visualização da manutenibilidade do modelo de features de LPSDs.

A.1 Parte 1 - Perfil do Especialista

A.1.1 Perguntas

1. **Qual o seu Nome?**
2. **Qual a sua instituição?**
3. **Qual o seu nível de formação?**
   ( ) Graduado
   ( ) Mestrado
   ( ) Doutorado
4. **Qual o tempo de experiência com Linhas de Produtos de Software?**
   ( ) Até 2 anos
   ( ) Mais de 2 e menos de 5 anos
   ( ) Mais de 5 anos
5. **Qual o tempo de experiência com Linhas de Produtos de Software Dinâmicas?**
   ( ) Até 2 anos
   ( ) Mais de 2 e menos de 5 anos
   ( ) Mais de 5 anos
6. **Como você classifica o seu grau de conhecimento em Linhas de Produtos de Software Dinâmicas?**
   ( ) Excelente
   ( ) Alto
   ( ) Bom

( ) Médio

( ) Baixo

7. **Quantos projetos (pesquisa ou indústria) envolvendo Linha de Produto de Software você já participou?**

   ( ) Até 2 projetos

   ( ) Entre 3 e 5 projetos

   ( ) Mais de 5 projetos

8. **Quantos projetos (pesquisa ou indústria) envolvendo Linha de Produto de Software Dinâmicas você já participou?**

   ( ) Até 2 projetos

   ( ) Entre 3 e 5 projetos

   ( ) Mais de 5 projetos

A.2 Parte 2 - Seleção de Visualizações

A.2.1 Definição das Subcaracterísticas de Qualidade

1. Entendibilidade / Complexidade Cognitiva: Capacidade que um produto de software tem de possibilitar ao usuário entender se o software é adequado para uso, e como ele pode ser utilizado em determinadas tarefas e condições de uso (ISO/ IEC 25010, 2011). Complexidade Cognitiva denota o quão fácil o modelo de features pode ser entendido, o que diz respeito à rastreabilidade da LPS (STUIKYS e DAMASEVICIUS, 2009).
2. Extensibilidade: Extensibilidade refere-se à capacidade de estender um modelo de *features* ao nível de esforço necessário para implementar a extensão (MONTAGUD et al., 2012).
3. Flexibilidade: Se refere à capacidade de um modelo de features para responder a possíveis mudanças internas ou externas que afetam o valor de sua entrega, em termos de tempo e de custo efetivo (DUAN et al., 2013).
4. Modularidade: O grau em que um sistema ou programa de computador é composto de componentes discretos de tal forma que uma mudança para um componente tem um impacto mínimo sobre outros componentes (ISO/ IEC 25010, 2011).
5. Complexidade Estrutural: Refere-se ao entendimento da complexidade da estrutura do modelo de *features* (BAGHERI GASEVIC, 2011; PATZKE et al., 2012; MONTAGUD et al., 2012).

6. Variabilidade estática: Variabilidade refere-se à capacidade de um artefato de ser configurado, personalizado, estendido ou modificado para o uso em um contexto específico (CHEN et al., 2009).
7. Variabilidade Dinâmica: A variabilidade dinâmica é uma característica intrínseca das LPSDs, ela se diferencia da variabilidade estática por prover mecanismos que possibilitem à variabilidade em tempo de execução.

A.2.2 Subcaracterísticas (Atributos de Qualidade) associadas as Medidas

(SUBCARACTERÍSTICA) :

- (MEDIDAS)

1. Complexidade Cognitiva/Entendibilidade
   Complexidade Cognitiva do Modelo de *Features* (CogC).
2. Extensibilidade:
   a) Extensibilidade de *feature* (FEX): Refere-se ao número de *features* que podem ser facilmente adicionadas ao modelo durante a fase de manutenção.
3. Flexibilidade:
   a) Flexibilidade de configuração (FoC): Esta é a razão entre o número de funcionalidades opcionais ao longo de todos as *features* disponíveis no modelo de *feature*. A lógica por trás disso é indicar a possibilidade de gerar configurações distintas.
4. Modularidade:
   a) *Features* dependentes de ciclos únicos (SCDF): Número de *features* que estão presentes nas restrições do modelo e filhas de pontos de variação com cardinalidade [1..1].
   b) *Features* dependentes de ciclos múltiplos (MCDF): Número de *features* que estão presentes nas restrições do modelo e filhas de pontos de variação com cardinalidade [1..*]
5. Complexidade estrutural:
   a) Número de *Features* (NF)
   b) Número de *Features* Mandatórias (NM)
   c) Número de *Features* Top (NTop)
   d) Profundidade da Árvore (DT Max, DT Mean and DT Median)
   e) Complexidade Ciclomática (CyC): O número de ciclos diferentes que podem ser

encontrados no modelo de *features*. Uma vez que os modelos de *features* são sob a forma de árvores, ciclos podem existir em um modelo. As restrições de integridade entre as *features* disponíveis podem gerar esses ciclos.

f) Complexidade Composta (ComC): Medida composta que representa o número de pontos variantes, o número de features variáveis , número de restrições de integridade e seus relacionamentos.

g) Restrições Cross-tree (CTC): A razão entre o número de *features* únicas envolvidas na restrição de integridade do modelo de *features* sobre o total do número de *features* do modelo de *features*. Essa medida representa o grau de envolvimento das *features* na definição das restrições de integridade.

h) Restrições Variáveis *Cross-tree* (CTCV): A razão entre o número de *features* únicas envolvidas na restrição de integridade do modelo de *features* sobre o total do número de *features* do modelo de *features*. Essa medida representa o grau de envolvimento das *features* na definição das restrições de integridade.

i) Taxa de Conectividade do Grafo (RCon)

j) Densidade do Grafo (RDen)

k) Coeficiente de Densidade da Conectividade (CoC)

l) Número de *Features* Agrupadas (NGF)

m) Número de Filhos por *Feature* (BF Max)

6. Variabilidade Estática:

a) Número de *Features* Opcionais (NO)

b) Single *Hotspot Features* (SHoF): Representa o número de *features* que podem ser adicionadas em tempo de execução através de pontos de variação com cardinalidade [1..1].

c) *Multiple Hotspot Features* (MHoF): Representa o número de *features* que podem ser adicionadas em tempo de execução através de pontos de variação com cardinalidade [1..*].

d) *Rigid No hotspot Features* (RNoF): O Número de *features* que podem ser adicionadas em tempo de execução, mas que não seja através de *features* de ponto variante.

e) Número de *Features* Variáveis (NVF)

f) Número de Configurações Válidas (NVC)

g) Taxa de Variabilidade (RoV): O fator médio de ramificação que a *feature* pai apresenta

no modelo de features. Em outras palavras, o número médio de filhos dos nós na árvore do modelo de *features*.

h) Número de Grupos Or (NGOr)
i) Número de Grupos XOr (NGXOr)
j) Taxa de *Features* Or (ROr)
k) Taxa de *Features* XOr (RXOr)

7. Variabilidade Dinâmica:
   a) Número de Contextos (NC)
   b) Número de *Features* Desativadas (NDF)
   c) Número de *Features* Ativadas (NAF)
   d) *Features* de Contextos em Restrições (CFC)
   e) Número de Restrições de Contexto (NCC)
   f) Número de *Features* de Contextos (CF)
   g) Número de *Features* Ativadas por Contexto (AFCA)
   h) Número de *Features* Desativadas por Contexto (DFCA)

A.2.3 Classificação das Medidas e Subcaracterísticas

As Subcaracterísticas e Medias serão classificadas com o nível de importância: menos importante, pouco importante, indiferente, importante e muito importante; dessa forma é possível obter os dados que terão maior ênfase para a criação das visualizações.

A.2.4 Identificação de subcaracterística e/ou outra medida de qualidade associada

Para a criação de possíveis candidatos a novas subcaracterística e novas medidas de qualidade é realizado uma pergunta onde é possível obter opiniões sobre os mesmos.

Por exemplo: (Subcaracterística) Extensibilidade. (Medida de qualidade associada) Extensibilidade de features.

A.2.5 Identificação de perguntas que você gostaria de responder a partir de uma visualização

As visualizações são criadas para facilitar a visualização de certos tipos de dados, e dessa forma obter respostas sobre determinadas perguntas. A seguir são realizadas

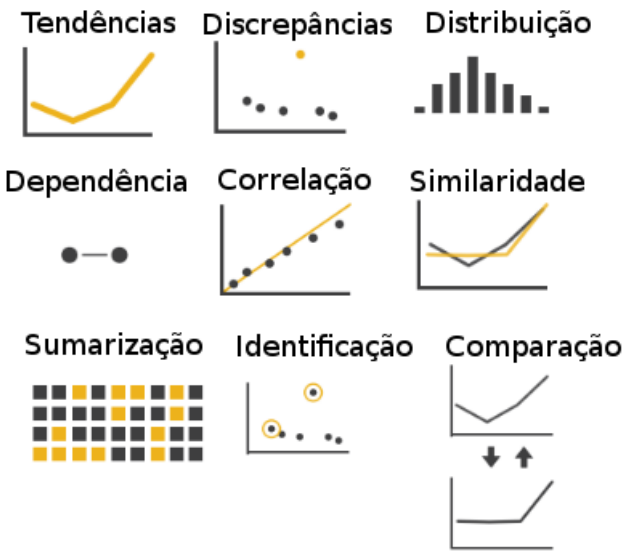
questionamentos para obter essas perguntas.

Para cada item você deve especificar a pergunta e o porque da análise.

Você também pode associar essa pergunta a uma possível visualização gráfica: diferença, similaridade, correlação, discrepância, sumarização, distribuição. Verificar : ("Informações úteis").

Além de um único modelo de *features*, podem existir combinações entre versões de um modelo ou conjunto de modelos diferentes.

A.2.6 Informações úteis: Exemplos de possíveis visualizações gráficas dos tipos de análise



A.2.7 Perguntas

1. **Qual o nível de importância de visualização que você daria para cada subcaracterística?**
   Níveis de importância: Muito Importante, Importante, Indiferente, Pouco importante, Menos importante.

   a) Complexidade cognitiva / Entendibilidade
   b) Extensibilidade
   c) Flexibilidade
   d) Modularidade

e) Complexidade estrutural
f) Variabilidade estática
g) Variabilidade dinâmica

2. **Para a característica de qualidade "Manutenibilidade"você identifica alguma outra subcaracterística e/ou outra medida de qualidade associada diferente das citadas anteriormente, que seja relevante para se avaliar a qualidade do modelo de features em LPSDs?**

3. **Quais perguntas você gostaria de responder a partir de uma visualização, das subcaracterísticas e/ou outra medidas identificadas/criadas acima?**

4. **Qual o nível de importância de visualização que você daria para a medida de qualidade associada a Subcaracterística Complexidade Cognitiva/Entendibilidade? Níveis de importância: Muito Importante, Importante, Indiferente, Pouco importante, Menos importante.**

a) Complexidade Cognitiva do Modelo de Features (CogC)

5. **Quais perguntas você gostaria de responder a partir de uma visualização, das medidas citadas acima para a Subcaracterística Complexidade Cognitiva/Entendibilidade?**

6. **Qual o nível de importância de visualização que você daria para cada medida de qualidade associada a Subcaracterística Modularidade? Níveis de importância: Muito Importante, Importante, Indiferente, Pouco importante, Menos importante.**

a) *Features* dependentes de ciclos únicos (SCDF)
b) *Features* dependentes de ciclos múltiplos (MCDF)

7. **Quais perguntas você gostaria de responder a partir de uma visualização, das medidas citadas acima para a Subcaracterística Modularidade?**

8. **Qual o nível de importância de visualização que você daria para a medida de qualidade associada a subcaracterística Extensibilidade?**
   Níveis de importância: Muito Importante, Importante, Indiferente, Pouco importante, Menos importante.
   a) Extensibilidade da Feature (FEX)

9. **Quais perguntas você gostaria de responder a partir de uma visualização, das medidas citadas acima para a subcaracterística Extensibilidade?**

10. **Qual o nível de importância de visualização que você daria para a medida de qualidade associada a subcaracterística Flexibilidade?**
    Níveis de importância: Muito Importante, Importante, Indiferente, Pouco importante, Menos importante.
    a) Flexibilidade de configuração (FoC)

11. **Quais perguntas você gostaria de responder a partir de uma visualização, das medidas citadas acima para a Subcaracterística Flexibilidade?**

12. **Qual o nível de importância de visualização que você daria para cada medida de qualidade associada a Subcaracterística Complexidade Estrutural?**

    Níveis de importância: Muito Importante, Importante, Indiferente, Pouco importante, Menos importante.

    a) Número de *Features* (NF)
    b) Número de *Features* Mandatórias (NM)
    c) Número de *Features* Top (NTop)
    d) Profundidade da Árvore (*DT Max, DT Mean e DT Median*)
    e) Complexidade Ciclomática (CyC)
    f) Complexidade Composta (ComC)
    g) Restrições *Cross-tree* (CTC)
    h) Restrições Variáveis *Cross-tree* (CTCV)
    i) Taxa de Conectividade do Grafo (RCon)
    j) Densidade do Grafo (RDen)

k) Coeficiente de Densidade da Conectividade (CoC)
l) Número de *Features* Agrupadas (NGF)
m) Número de Filhos por *Feature* (BF Max)

13. **Quais perguntas você gostaria de responder a partir de uma visualização, das medidas citadas acima para a Subcaracterística Complexidade estrutural?**

14. **Qual o nível de importância de visualização que você daria para cada medida de qualidade associada a Subcaracterística Variabilidade Estática?**
Níveis de importância: Muito Importante, Importante, Indiferente, Pouco importante, Menos importante.

a) Número de *Features* Opcionais (NO)
b) *Single Hotspot Features* (SHoF)
c) *Multiple Hotspot Features* (MHoF)
d) *Rigid No hotspot Features* (RNoF)
e) Número de *Features* Variáveis (NVF)
f) Número de Configurações Válidas (NVC)
g) Taxa de Variabilidade (RoV)
h) Número de Grupos Or (NGOr)
i) Número de Grupos XOr (NGXOr)
j) Taxa de *Features* Or (ROr)
k) Taxa de *Features* XOr (RXOr)

15. **Quais perguntas você gostaria de responder a partir de uma visualização, das medidas citadas acima para a Subcaracterística Variabilidade Estática?**

16. **Qual o nível de importância de visualização que você daria para cada medida de qualidade associada a subcaracterística Variabilidade Dinâmica?**
Níveis de importância: Muito Importante, Importante, Indiferente, Pouco importante, Menos importante.

a) Número de Contextos (NC)
b) Número de *Features* Desativadas (NDF)

c) Número de *Features* Ativadas (NAF)

d) *Features* de Contextos em Restrições (CFC)

e) Número de Restrições de Contexto (NCC)

f) Número de *Features* de Contextos (CF)

g) Número de *Features* Desativadas por Contexto (DFCA)

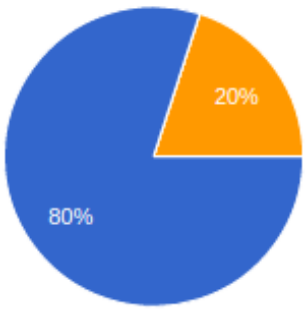
h) Número de *Features* Ativadas por Contexto (AFCA)

17. **Quais perguntas você gostaria de responder a partir de uma visualização das medidas citadas acima para a Subcaracterística Variabilidade Dinâmica?**

A.3 Resumo de Respostas

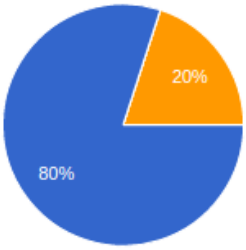
A.3.1 Perfil

1. **Nome do especialista** "*Dados Privados*"
2. **Instituição**
   a) Universidade Federal do Ceará
   b) Universidade Federal do Ceará - MDCC
   c) UFPE
   d) UFC
   e) UFC
3. **Qual o seu nível de formação?**



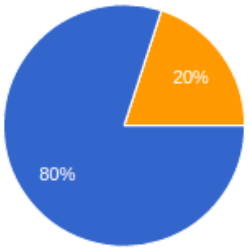
Graduado	**4**	80%
Mestrado	**0**	0%
Doutorado	**1**	20%

4. **Qual o tempo de experiência com Linhas de Produtos de Software?**



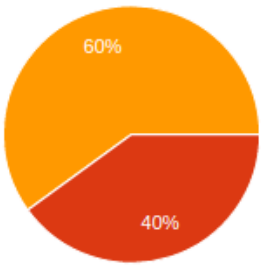
Até 2 anos	**4**	80%
Mais de 2 e menos de 5 anos	**0**	0%
Mais de 5 anos	**1**	20%

5. **Qual o tempo de experiência com Linhas de Produtos de Software Dinâmicas?**



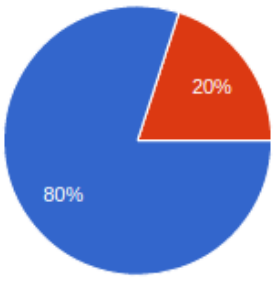
Até 2 anos	**4**	80%
Mais de 2 e menos de 5 anos	**0**	0%
Mais de 5 anos	**1**	20%

6. **Como você classifica o seu grau de conhecimento em Linhas de Produtos de Software Dinâmicas?**



Excelente	**0**	0%
Alto	**2**	40%
Bom	**3**	60%
Médio	**0**	0%
Baixo	**0**	0%

7. **Quantos projetos (pesquisa ou indústria) envolvendo Linha de Produto de Software você já participou?**



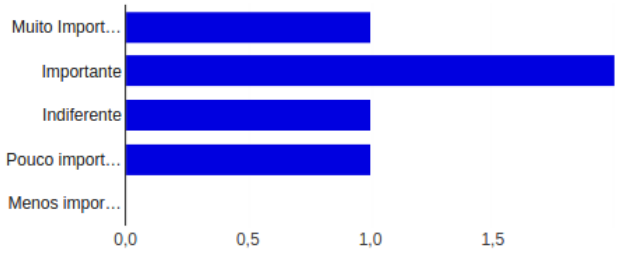
Até 2 projetos	**4**	80%
Entre 3 e 5 projetos	**1**	20%
Mais de 5 projetos	**0**	0%

8. **Quantos projetos (pesquisa ou indústria) envolvendo Linha de Produto de Software Dinâmicas você já participou?**

Até 2 projetos	5	100%
Entre 3 e 5 projetos	0	0%
Mais de 5 projetos	0	0%

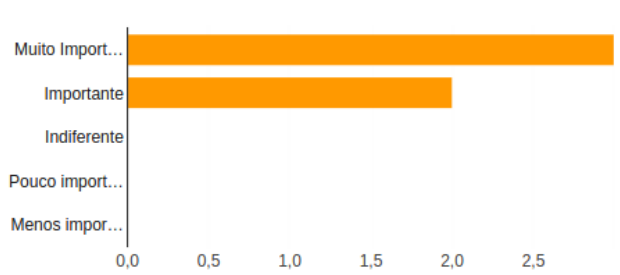
A.3.2 Seleção de Visualizações

1. **Qual o nível de importância de visualização que você daria para subcaracterística Complexidade cognitiva / Entendibilidade?**



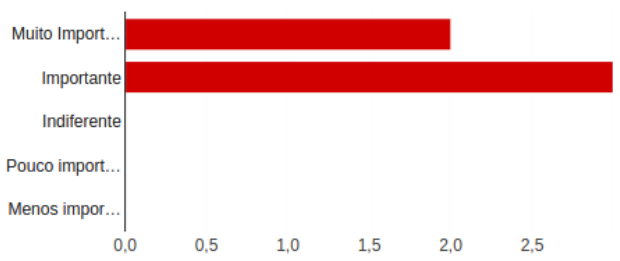
Muito Importante	1	20%
Importante	2	40%
Indiferente	1	20%
Pouco importante	1	20%
Menos importante	0	0%

2. **Qual o nível de importância de visualização que você daria para subcaracterística Extensibilidade?**



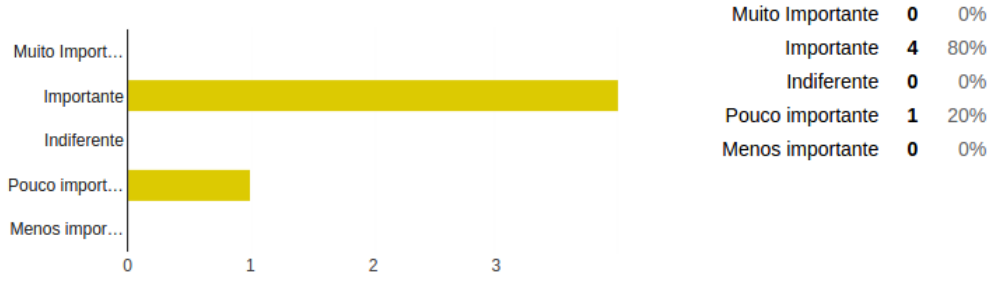
Muito Importante	3	60%
Importante	2	40%
Indiferente	0	0%
Pouco importante	0	0%
Menos importante	0	0%

3. **Qual o nível de importância de visualização que você daria para subcaracterística Flexibilidade?**

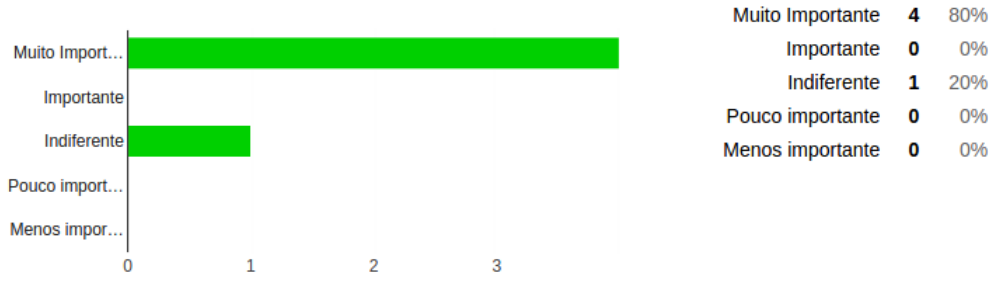


Muito Importante	2	40%
Importante	3	60%
Indiferente	0	0%
Pouco importante	0	0%
Menos importante	0	0%

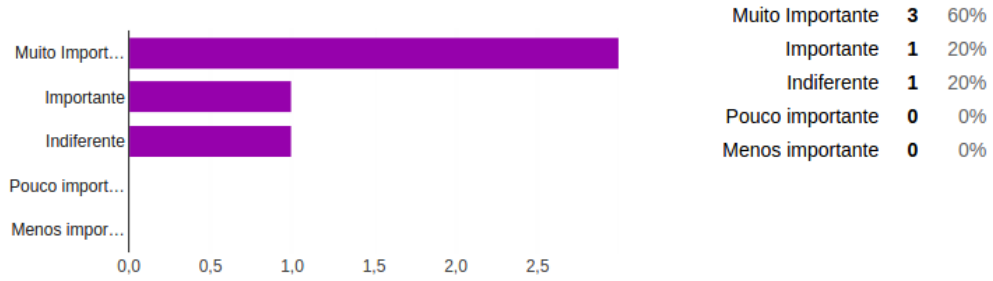
4. **Qual o nível de importância de visualização que você daria para subcaracterística Modularidade?**



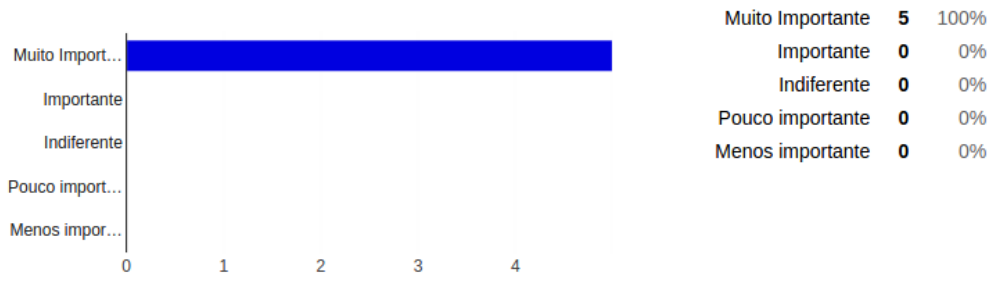
5. **Qual o nível de importância de visualização que você daria para subcaracterística Complexidade estrutural?**



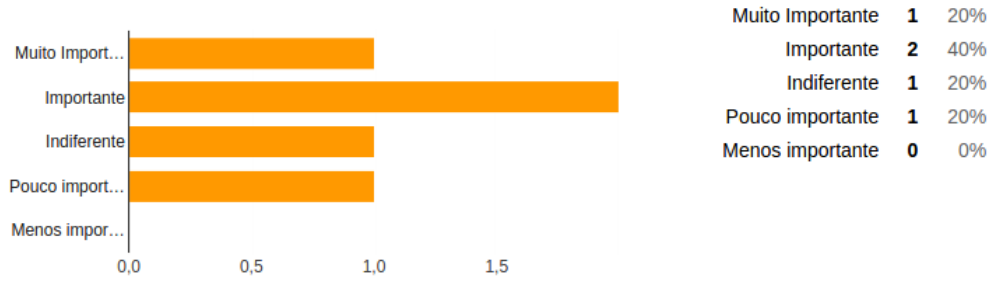
6. **Qual o nível de importância de visualização que você daria para subcaracterística Variabilidade estática?**



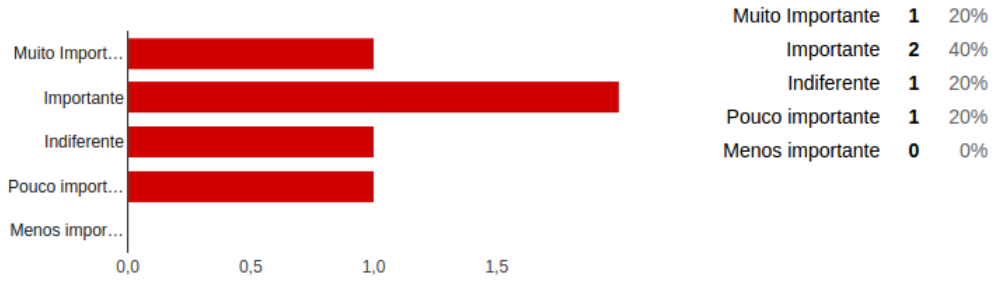
7. **Qual o nível de importância de visualização que você daria para subcaracterística Variabilidade dinâmica?**



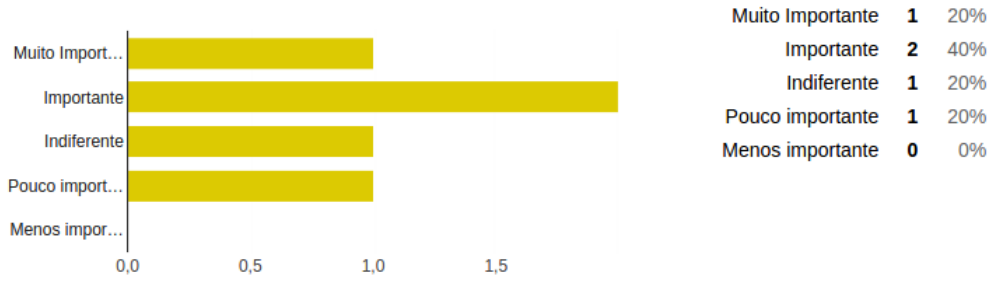
8. **Quais perguntas você gostaria de responder a partir de uma visualização, das subcaracterísticas e/ou outra medidas identificadas/criadas acima?**
    a) Quais os pontos mais flexíveis do modelo para adicionar novas features sem haver grandes impactos na arquitetura?
    b) Qual o impacto a linha de produtos sofrerá quando uma mudança ocorre no modelo de features?
    c) A relação entre a complexidade estrutural e a complexidade cognitiva, ou seja, se quantos mais complexo o modelo de features é mais difícil de entender ele fica. A relação entre modularidade e a extensibilidade do modelo, ou seja, se um modelo possui uma alta modularidade isso significa que ele tem uma alta extensibilidade? Se eu tenho muitas features dependentes de ciclos únicos (SCDF) a extensibilidade aumenta ou diminui ? Se eu tenho muitas features dependentes de ciclos únicos (SCDF) a flexibilidade aumenta ou diminui ?
9. **Qual o nível de importância de visualização que você daria para a medida de qualidade Complexidade Cognitiva do Modelo de Features (CogC) associada a subcaracterística Complexidade Cognitiva/Entendibilidade?**



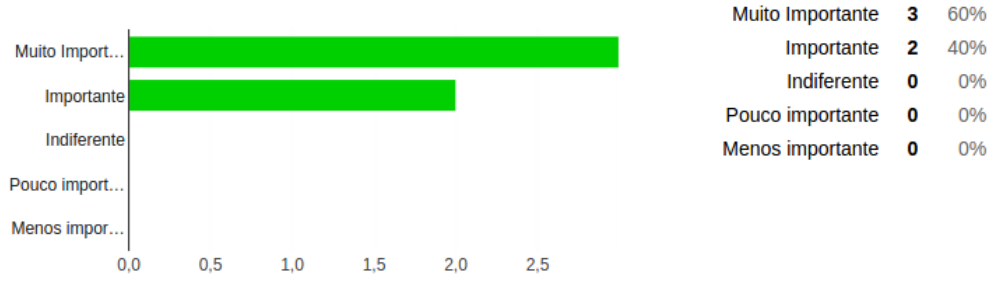
10. **Quais perguntas você gostaria de responder a partir de uma visualização, das medidas citadas acima para a subcaracterística Complexidade Cognitiva/Entendibilidade?**
    a) Uma determinada configuração da linha de produto seria útil para um determinado contexto, a partir do conjunto de *features* selecionadas?
    b) Quais modelos do conjunto de modelas possuem uma maior Complexidade Cognitiva/Entendibilidade?
11. **Qual o nível de importância de visualização que você daria para a medida de qualidade Features dependentes de ciclos únicos (SCDF) associada a subcaracterística Modularidade?**



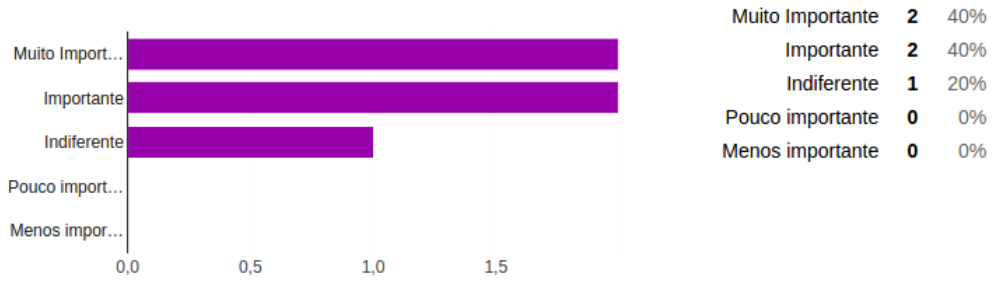
12. **Qual o nível de importância de visualização que você daria para a medida de qualidade Features dependentes de ciclos únicos (SCDF) associada a subcaracterística Modularidade?**



13. **Qual o nível de importância de visualização que você daria para a medida de qualidade Extensibilidade da Feature (FEX) associada a subcaracterística Extensibilidade?**



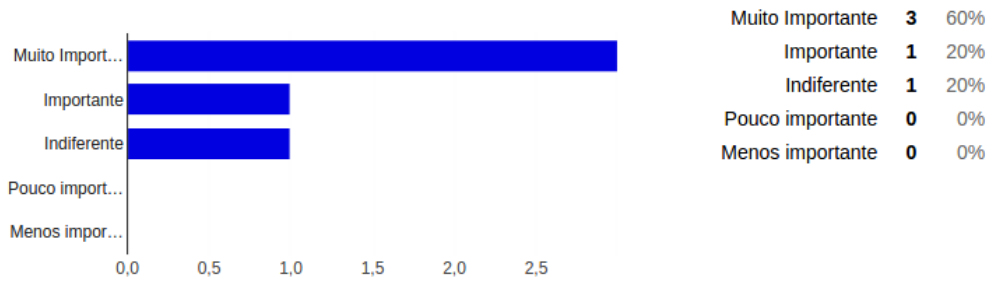
14. **Quais perguntas você gostaria de responder a partir de uma visualização, das medidas citadas acima para a subcaracterística Extensibilidade?**
    a) Qual o impacto que a linha sofreria ao extender uma determinada feature?
15. **Qual o nível de importância de visualização que você daria para a medida de qualidade Flexibilidade de configuração (FoC) associada a subcaracterística Flexibilidade?**



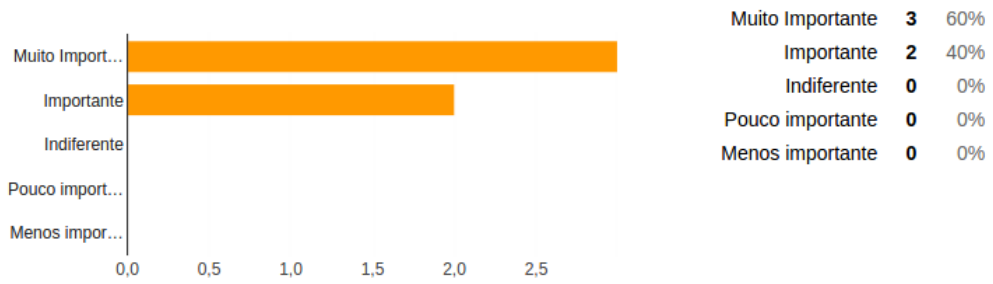
16. **Quais perguntas você gostaria de responder a partir de uma visualização, das medidas citadas acima para a subcaracterística Flexibilidade?**

    a) Qual o aumento no número de configurações a partir da inclusão de novas features? No caso usando a visualização de comparação.

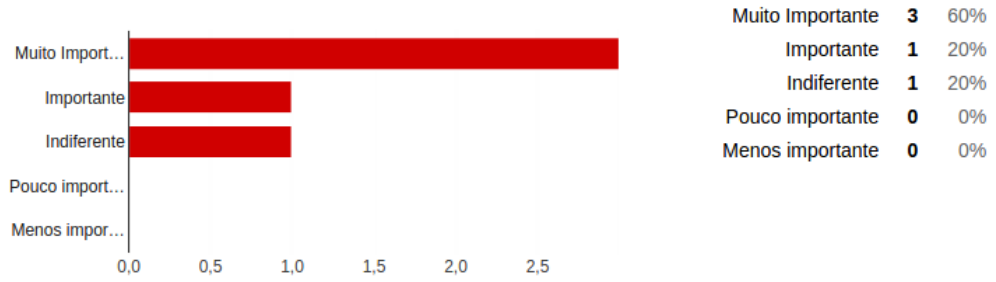
17. **Qual o nível de importância de visualização que você daria para a medida de qualidade Número de Features (NF) associada a subcaracterística Complexidade Estrutural?**



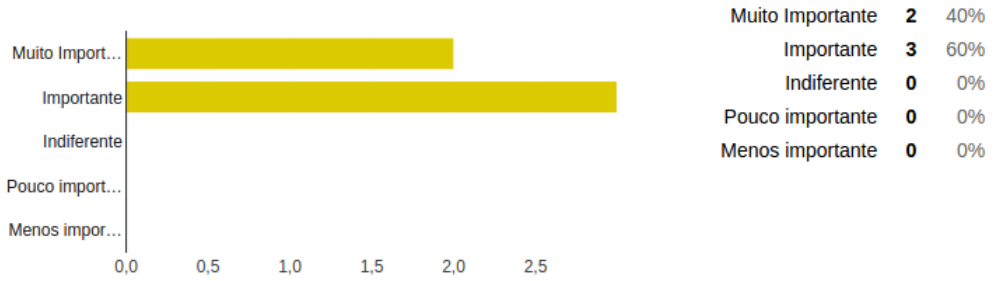
18. **Qual o nível de importância de visualização que você daria para a medida de qualidade Número de Features Mandatórias (NM) associada a subcaracterística Complexidade Estrutural?**



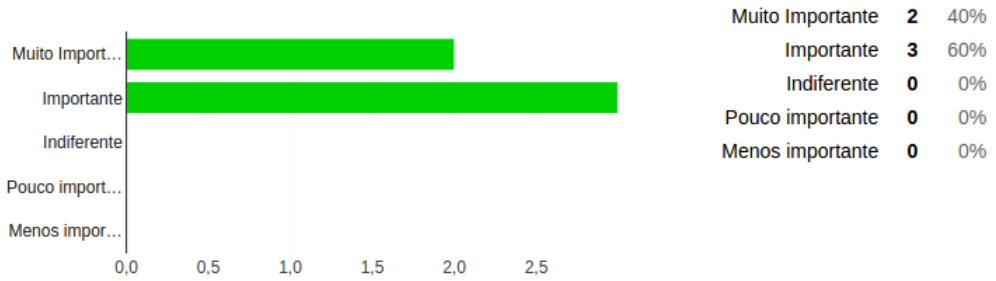
19. **Qual o nível de importância de visualização que você daria para a medida de qualidade Número de *Features* Top (NTop) associada a subcaracterística Complexidade Estrutural?**



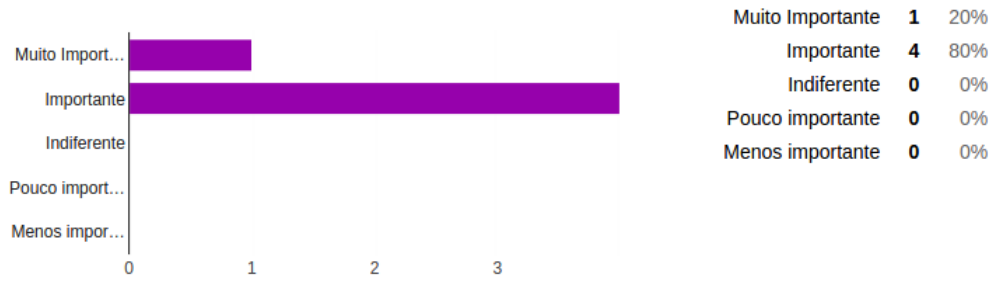
20. **Qual o nível de importância de visualização que você daria para a medida de qualidade Profundidade da Árvore (DT Max, DT Mean e DT Median) associada a subcaracterística Complexidade Estrutural?**



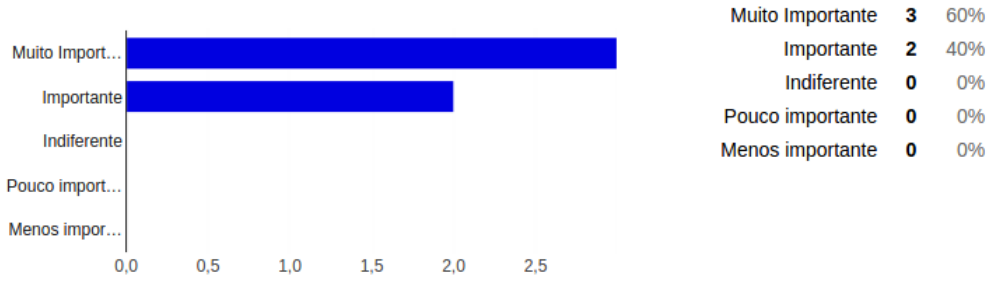
21. **Qual o nível de importância de visualização que você daria para a medida de qualidade Complexidade Ciclomática (CyC) associada a subcaracterística Complexidade Estrutural?**



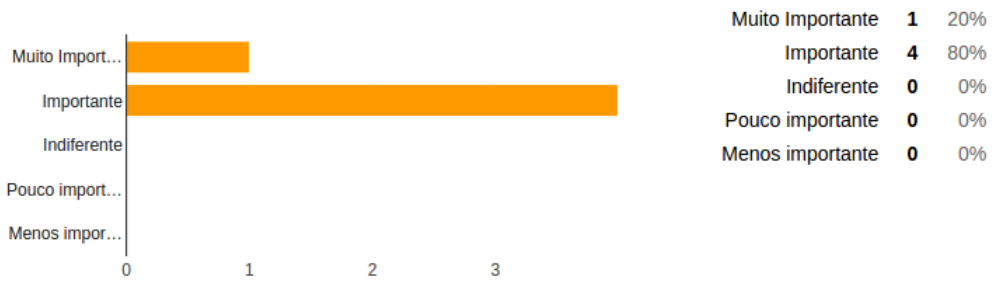
22. **Qual o nível de importância de visualização que você daria para a medida de qualidade Complexidade Composta (ComC) associada a subcaracterística Complexidade Estrutural?**



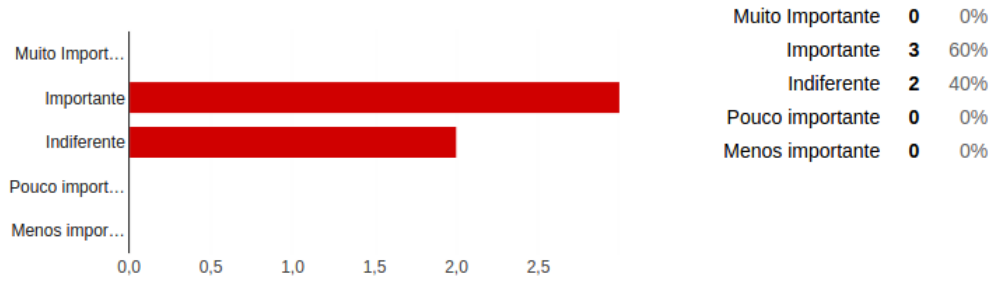
23. **Qual o nível de importância de visualização que você daria para a medida de qualidade Restrições Cross-tree (CTC) associada a subcaracterística Complexidade Estrutural?**



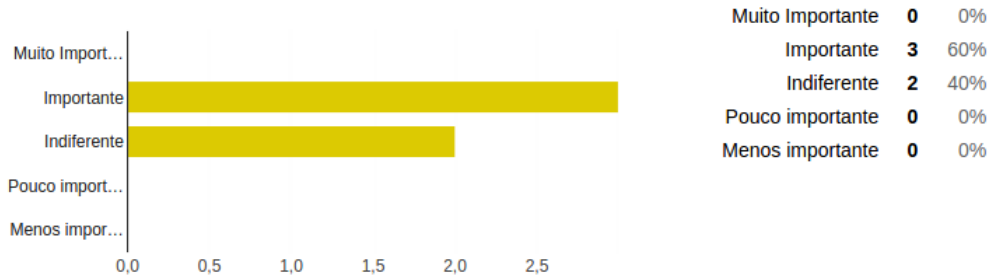
24. **Qual o nível de importância de visualização que você daria para a medida de qualidade Restrições Variáveis Cross-tree (CTCV) associada a subcaracterística Complexidade Estrutural?**



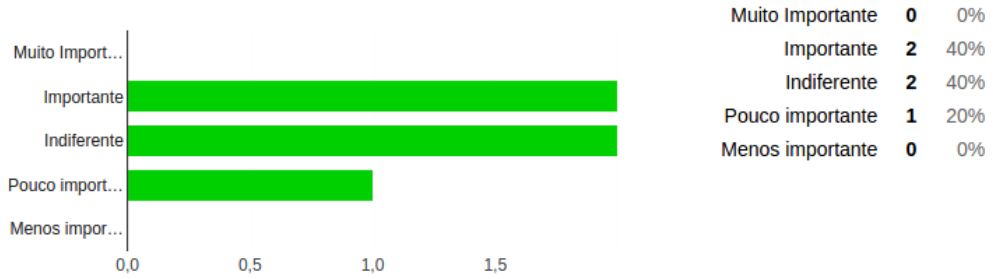
25. **Qual o nível de importância de visualização que você daria para a medida de qualidade Taxa de Conectividade do Grafo (RCon) associada a subcaracterística Complexidade Estrutural?**



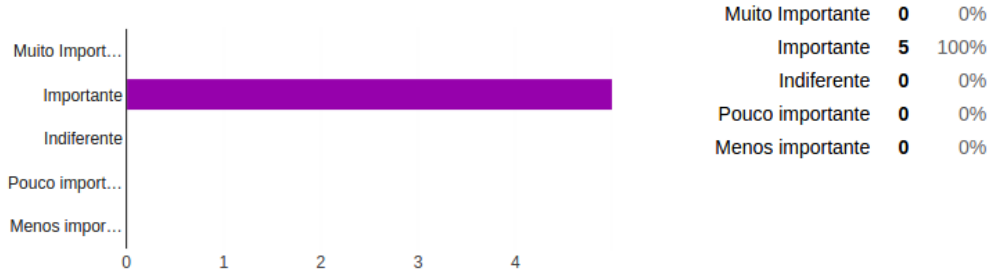
26. **Qual o nível de importância de visualização que você daria para a medida de qualidade Densidade do Grafo (RDen) associada a subcaracterística Complexidade Estrutural?**



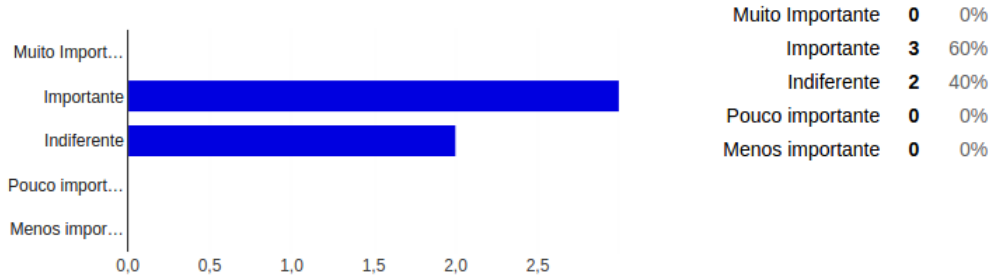
27. **Qual o nível de importância de visualização que você daria para a medida de qualidade Coeficiente de Densidade da Conectividade (CoC) associada a subcaracterística Complexidade Estrutural?**



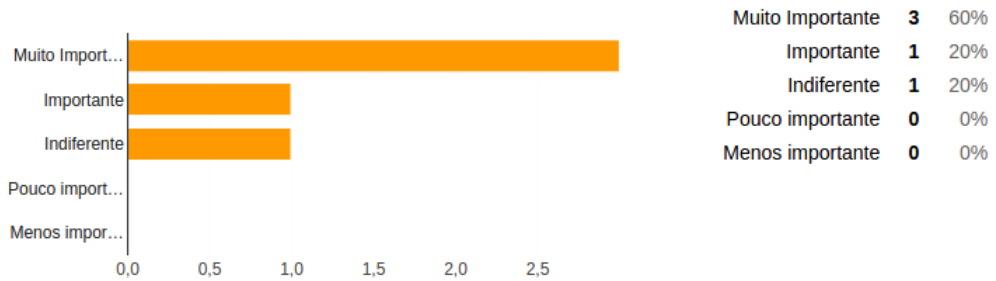
28. **Qual o nível de importância de visualização que você daria para a medida de qualidade Número de Features Agrupadas (NGF) associada a subcaracterística Complexidade Estrutural?**



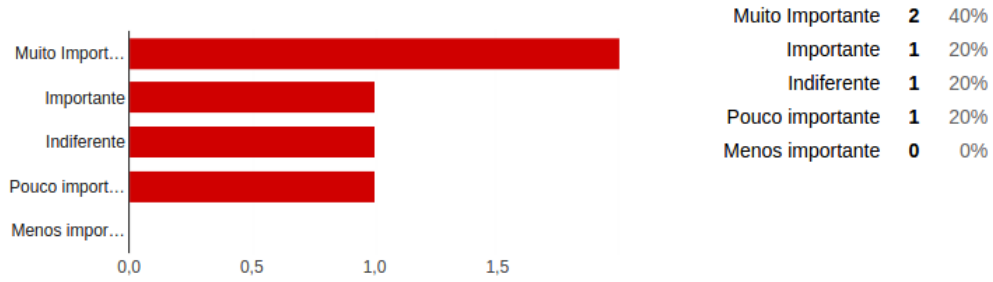
29. **Qual o nível de importância de visualização que você daria para a medida de qualidade Número de Filhos por Feature (BF Max) associada a subcaracterística Complexidade Estrutural?**



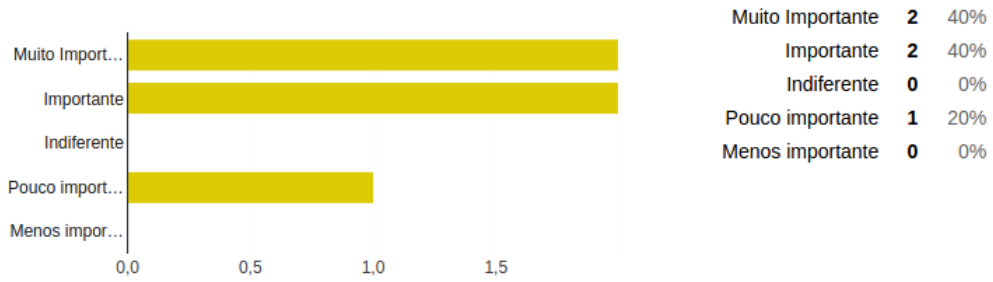
30. **Quais perguntas você gostaria de responder a partir de uma visualização, das medidas citadas acima para a subcaracterística Complexidade estrutural?**
    a) A evolução da complexidade do modelo de features quando uma feature é add e removida.
31. **Qual o nível de importância de visualização que você daria para a medida de qualidade Número de Features Opcionais (NO) associada a subcaracterística Variabilidade Estática?**



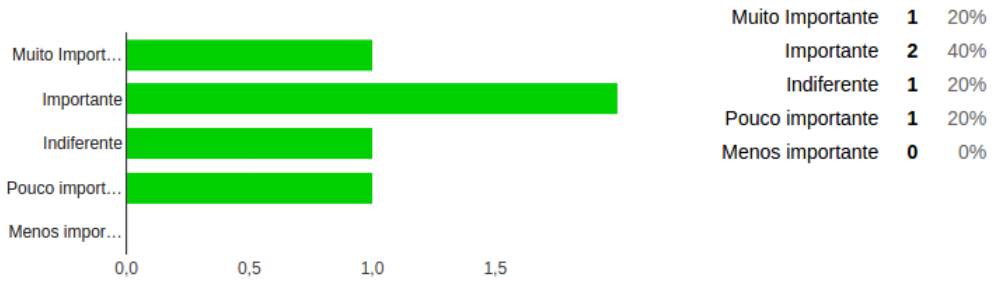
32. **Qual o nível de importância de visualização que você daria para a medida de qualidade Single Hotspot Features (SHoF) associada a subcaracterística Variabilidade Estática?**



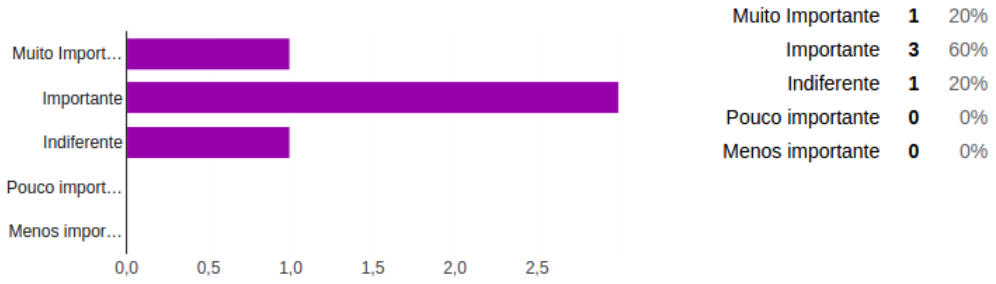
33. **Qual o nível de importância de visualização que você daria para a medida de qualidade Multiple Hotspot Features (MHoF) associada a subcaracterística Variabilidade Estática?**



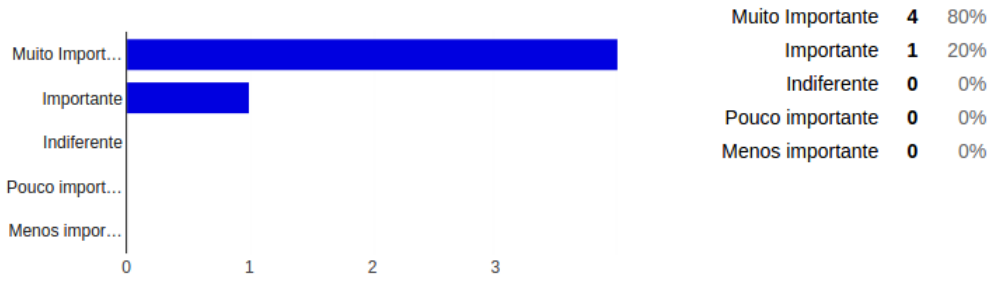
34. **Qual o nível de importância de visualização que você daria para a medida de qualidade Rigid No hotspot Features (RNoF) associada a subcaracterística Variabilidade Estática?**



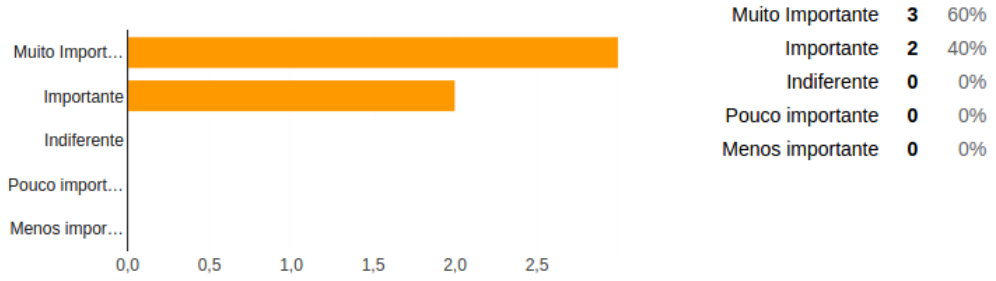
35. **Qual o nível de importância de visualização que você daria para a medida de qualidade Número de Features Variáveis (NVF) associada a subcaracterística Variabilidade Estática?**



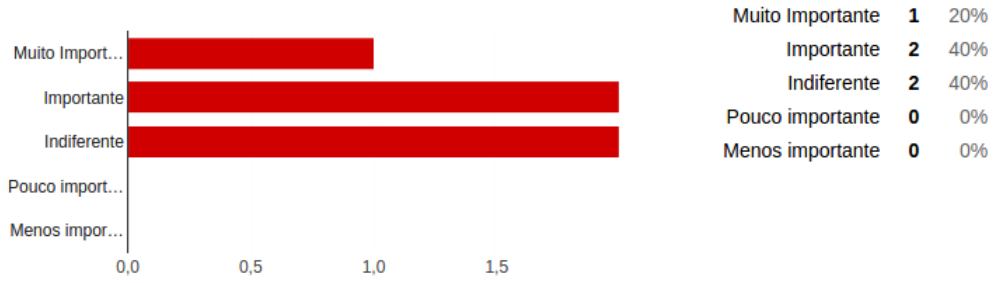
36. **Qual o nível de importância de visualização que você daria para a medida de qualidade Número de Configurações Válidas (NVC) associada a subcaracterística Variabilidade Estática?**



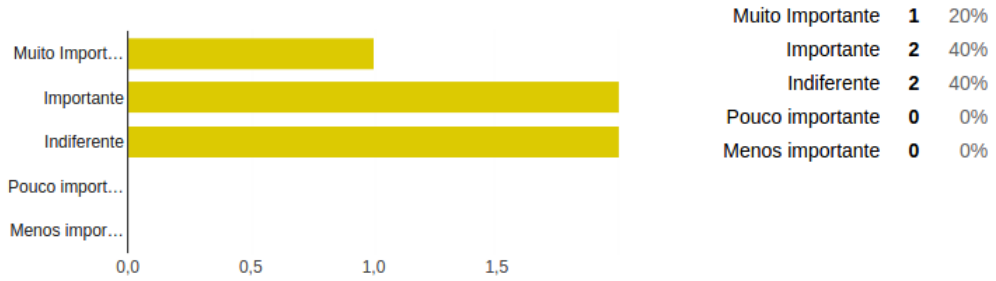
37. **Qual o nível de importância de visualização que você daria para a medida de qualidade Taxa de Variabilidade (RoV) associada a subcaracterística Variabilidade Estática?**



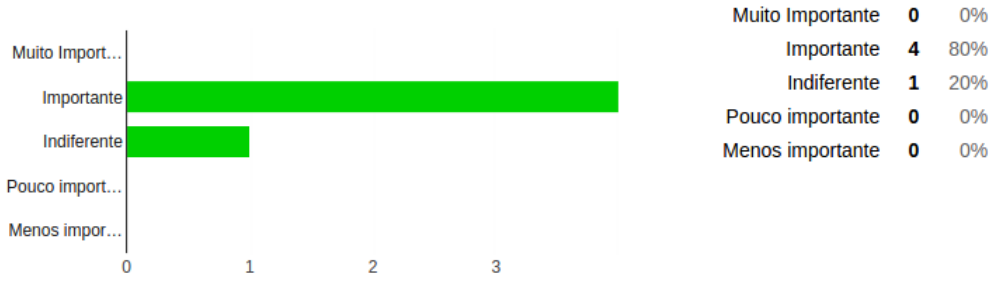
38. **Qual o nível de importância de visualização que você daria para a medida de qualidade Número de Grupos Or (NGOr) associada a subcaracterística Variabilidade Estática?**



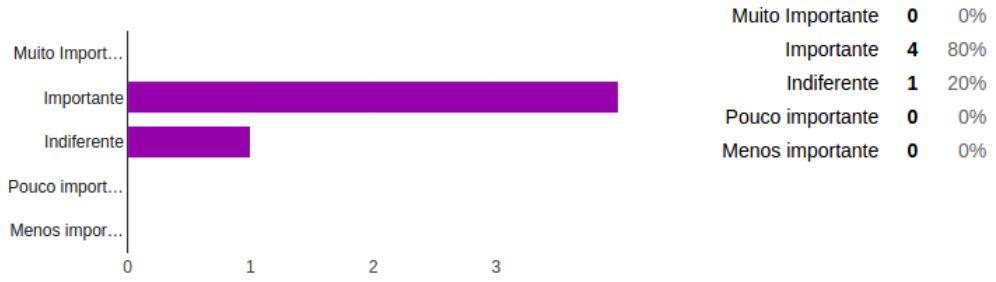
39. **Qual o nível de importância de visualização que você daria para a medida de qualidade Número de Grupos XOr (NGXOr) associada a subcaracterística Variabilidade Estática?**



40. **Qual o nível de importância de visualização que você daria para a medida de qualidade Taxa de Features Or (ROr) associada a subcaracterística Variabilidade Estática?**



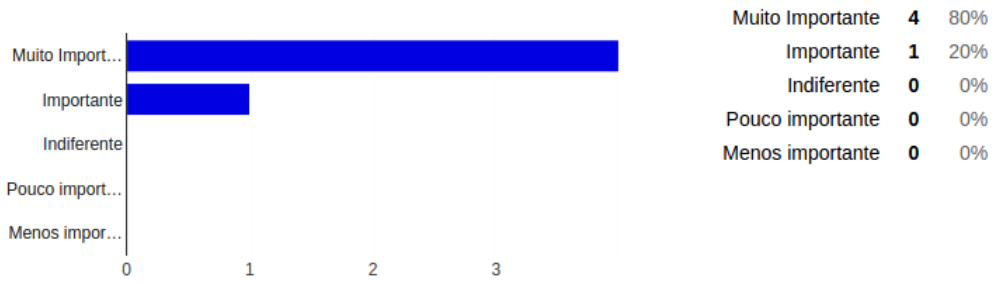
41. **Qual o nível de importância de visualização que você daria para a medida de qualidade Taxa de Features XOr (RXOr) associada a subcaracterística Variabilidade Estática?**



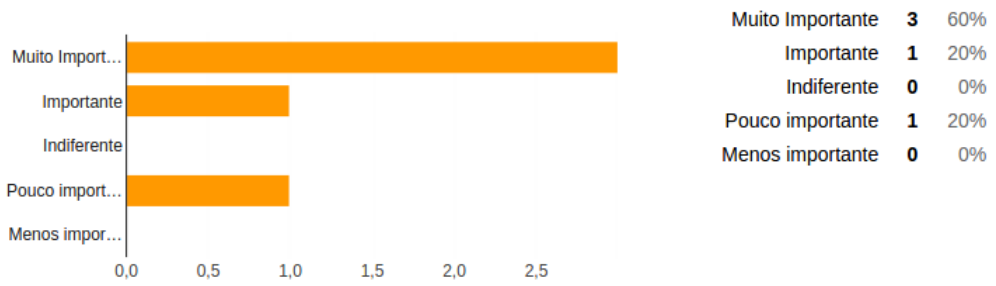
42. **Quais perguntas você gostaria de responder a partir de uma visualização, das medidas citadas acima para a subcaracterística Variabilidade Estática?**

    a) A relação entre a taxa de variabilidade com o numero de grupos XOR e OR. Se quanto mais grupos eu tenho a taxa de variabilidade aumenta? ou se eu tenho mais *features* opcionais a taxa de variabilidade é maior.

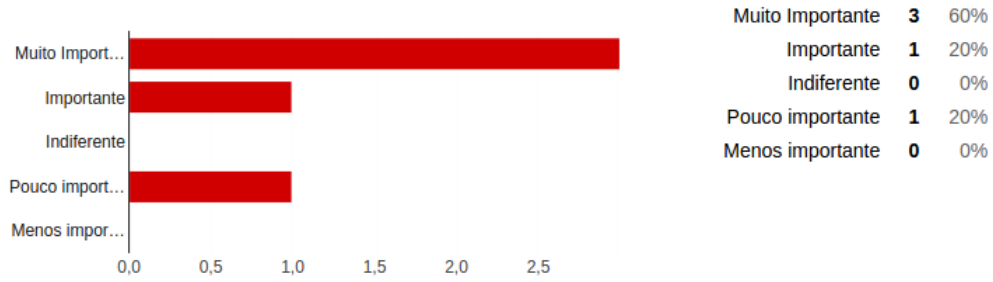
43. **Qual o nível de importância de visualização que você daria para a medida de qualidade Número de Contextos (NC) associada a subcaracterística Variabilidade Dinâmica?**



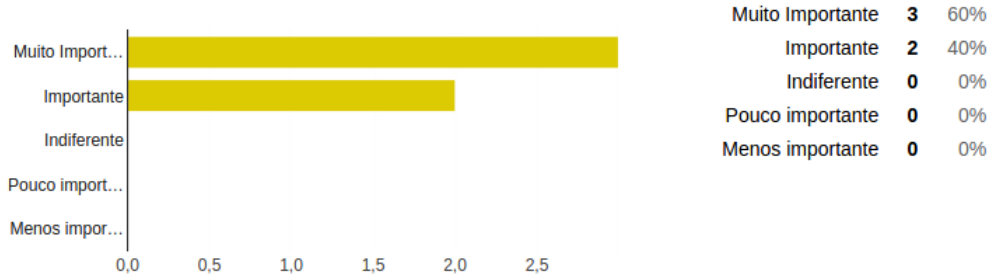
44. **Qual o nível de importância de visualização que você daria para a medida de qualidade Número de Features Desativadas (NDF) associada a subcaracterística Variabilidade Dinâmica?**



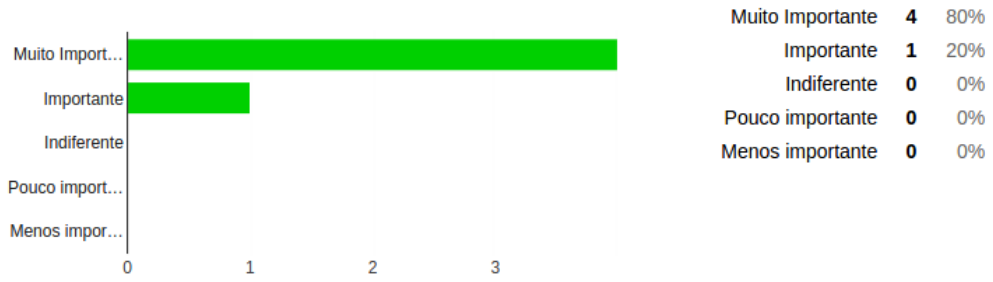
45. **Qual o nível de importância de visualização que você daria para a medida de qualidade Número de Features Ativadas (NAF) associada a subcaracterística Variabilidade Dinâmica?**



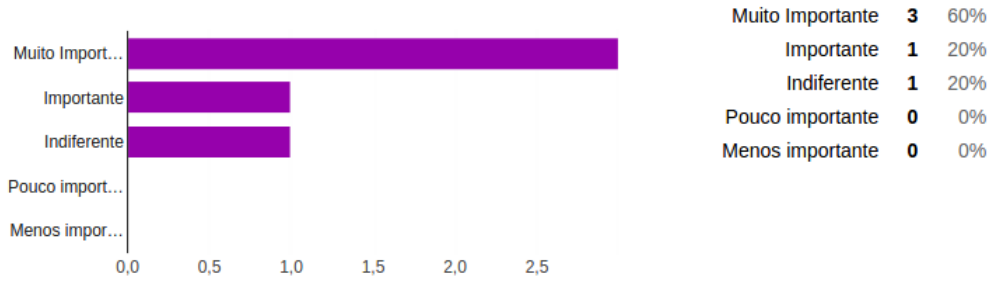
46. **Qual o nível de importância de visualização que você daria para a medida de qualidade Features de Contextos em Restrições (CFC) associada a subcaracterística Variabilidade Dinâmica?**



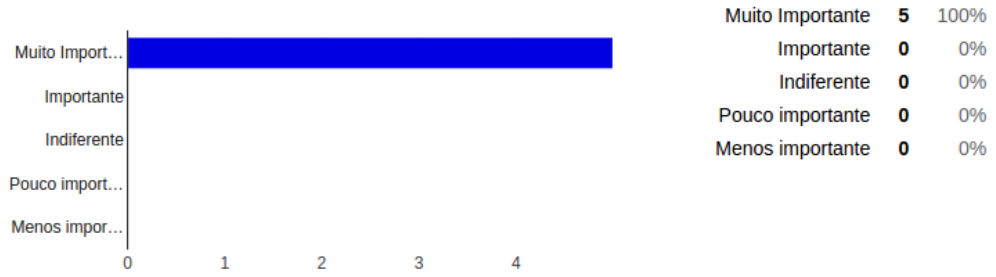
47. **Qual o nível de importância de visualização que você daria para a medida de qualidade Número de Restrições de Contexto (NCC) associada a subcaracterística Variabilidade Dinâmica?**



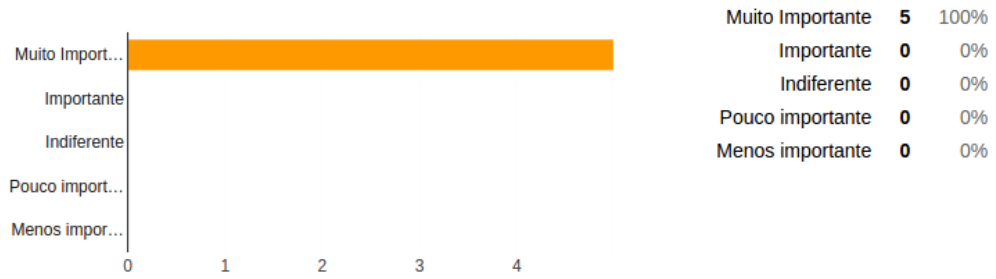
48. **Qual o nível de importância de visualização que você daria para a medida de qualidade Número de Features de Contextos (CF) associada a subcaracterística Variabilidade Dinâmica?**



49. **Qual o nível de importância de visualização que você daria para a medida de qualidade Número de Features Desativadas por Contexto (DFCA) associada a subcaracterística Variabilidade Dinâmica?**



50. **Qual o nível de importância de visualização que você daria para a medida de qualidade Número de Features Ativadas por Contexto (AFCA) associada a subcaracterística Variabilidade Dinâmica?**



51. **Quais perguntas você gostaria de responder a partir de uma visualização das medidas citadas acima para a subcaracterística Variabilidade Dinâmica?**
    a) Quais features serão ativadas/desativadas em uma mudança de contexto? Quais features são associadas a restrições?
    b) Qual dos contextos presentes em um modelo de features, possui mais dinamicidade em termos de ativação e desativação ?
    c) Qual a complexidade do modelo de features? Qual a variabilidade dinâmica do modelo de features? O modelo de features é de fácil extensibilidade?

APÊNDICE B – VALIDAÇÃO DAS VISUALIZAÇÕES DESENVOLVIDAS

Esse questionário tem como objetivo validar as visualizações desenvolvidas a partir das perguntas obtidas no questionário: (Seleção de Visualizações para Característica de Manutenibilidade do Modelo de *Features*)

B.1 Parte 1

B.1.1 Perguntas

1. **Qual o seu nome?**

B.1.2 Sobre o questionário anterior: Seleção de Visualizações para Característica de Manutenibilidade do Modelo de Features.

Depois da análise realizada no questionário anterior foram obtidas as seguintes Subcaracterísticas e Medidas como importantes:

1. Variabilidade dinâmica
    a) Número de Contextos (NC)
    b) Número de *Features* Desativadas por Contexto (DFCA)
    c) Número de *Features* Ativadas por Contexto (AFCA)
    d) Número de Restrições de Contexto (NCC)
    e) Número de *Features* Desativadas (NDF)
    f) Número de *Features* Ativadas (NAF)
    g) *Features* de Contextos em Restrições (CFC)
    h) Número de *Features* de Contextos (CF)
2. Complexidade estrutural
    a) Número de *Features* (NF)
    b) Número de *Features* Mandatórias (NM)
    c) Número de *Features* Top (Ntop)
    d) Restrições *Cross-tree* (CTC)
3. Extensibilidade

    Extensibilidade da *Feature* (FEX)

4. Variabilidade estática
    a) Número de Configurações Válidas (NVC)
    b) Taxa de Variabilidade (RoV)
    c) Número de *Features* Opcionais (NO)

Por serem Subcaracterísticas e Medidas selecionadas como importantes, elas serão utilizadas gerar as visualizações.

Dessa forma, as perguntas que foram selecionadas para serem respondidas por meio de uma visualização tem relação direta com essas Subcaracterísticas e Medidas. As perguntas foram as seguintes:

1. **Qual o impacto a linha de produtos sofrerá quando uma mudança ocorre no modelo de *features*?**
2. **Qual o impacto que a linha sofreria ao estender uma determinada *feature*?**
3. **Qual o aumento no número de configurações a partir da inclusão de novas *features*?**
4. **Qual a evolução da complexidade do modelo de *features* quando uma *feature* é adicionada e removida?**
5. **Qual dos contextos presentes em um modelo de *features*, possui mais dinamicidade em termos de ativação e desativação ?**
6. **Qual a complexidade do modelo de *features*?**
7. **Qual a variabilidade dinâmica do modelo de *features*?**
8. **O modelo de features é de fácil extensibilidade?**

Com essas perguntas em mãos, foram desenvolvidas visualizações que poderiam responde-las, que podem ser vistas mais abaixo.

B.1.3 Como responder esse questionário?

Uma visualização criada pode responder uma ou mais das perguntas selecionadas.

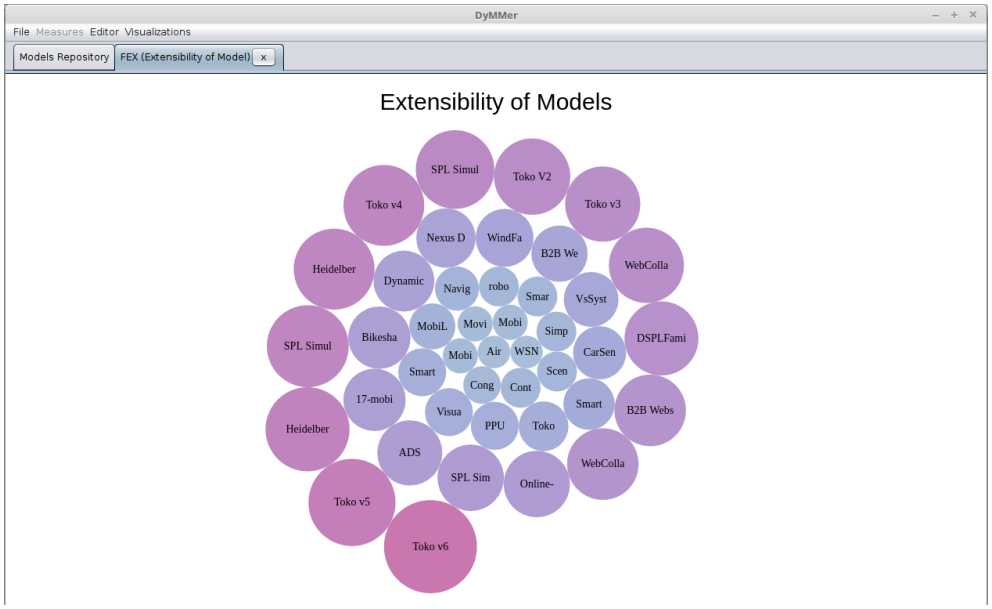
Para cada visualização adicionada abaixo, existe uma explicação para a mesma e as perguntas que podem ser respondidas por ela.

Dessa forma você deve analisar a visualização e a partir da análise realizada selecionar para cada pergunta um dos níveis: Concordo plenamente, Concordo parcialmente, Não compreendi e Discordo; dessa forma pode-se obter o nível de satisfação da visualização em relação as perguntas.

Se for necessário escrever algum comentário, existe um campo: "Considerações sobre a visualização e/ou resposta acima"para cada visualização.

B.1.4 Extensibilidade de todos os Modelos

Visualização que representa a extensibilidade de todos os modelos que existem no repositório da DyMMer. Os Modelos com extensibilidade menores são encontrados mais centralizados, com cor mais clara e com tamanho menores, enquanto os com extensibilidade maior são encontrados mais distantes do centro, cor mais escura e com tamanho maiores. As medida utilizada foi a Extensibilidade da Feature (FEX). O gráfico de Bolha foi utilizado para essa visualização.



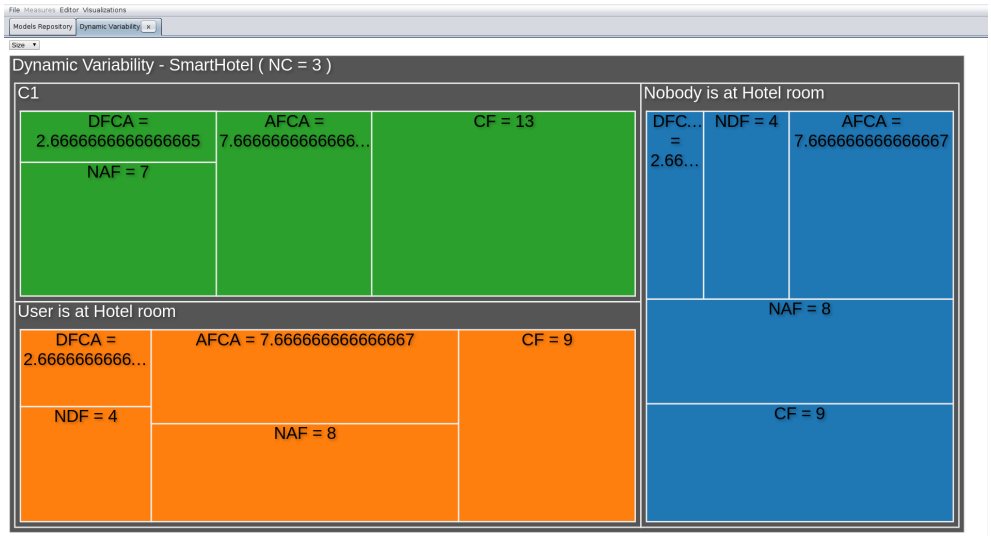
Faixa de repostas: Concordo plenamente, Concordo parcialmente, Não compreendi, Discordo.

1. **Você concorda que a visualização acima mostra qual modelo de features é de fácil extensibilidade?**
2. **Considerações sobre a visualização e/ou resposta acima.**

B.1.5 Variabilidade Dinâmica de um Modelo

Visualização que representa a variabilidade dinâmica de um modelo de *features*. Os contextos do modelo são recuperados e adicionados na visualização, como se pode observar em: "C1", "*Nobody is at Hotel room*"e "*User is at Hotel room*"; para cada contexto que existe no

modelo são ilustrados as medidas: Número de Contextos (NC), Número de *Features* Ativadas, (NaF), Número de *Features* Desativada (NdF), Número de *Features* de Contextos (CF), *Features* de Contextos em Restrições (CFC), Número de Features Ativadas por Contexto (AFCA), Número de *Features* Desativadas por Contexto (DFCA); as medidas que não são ilustradas em algum modelo estão com valor zero. O *TreeMap* foi utilizado para essa visualização.

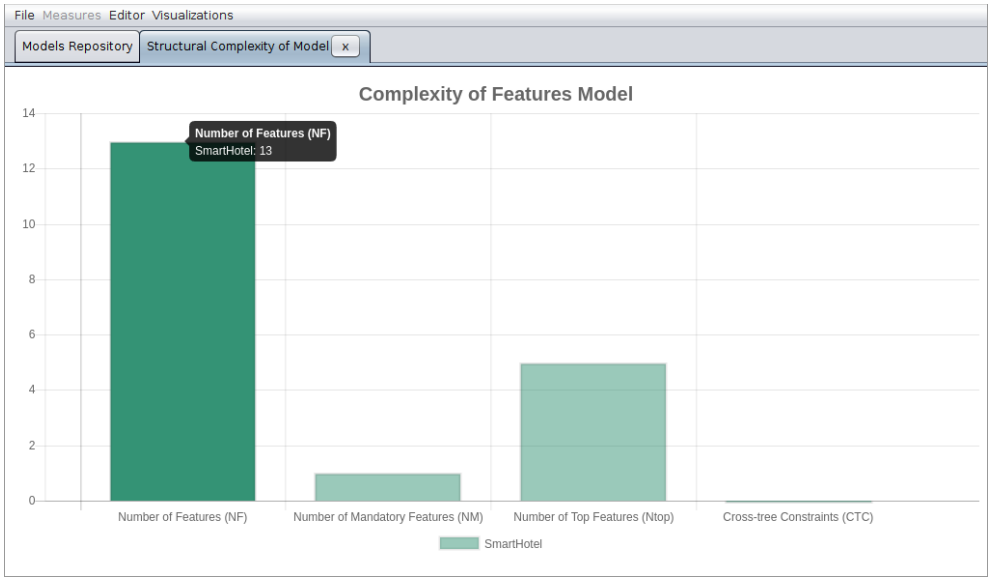


Faixa de repostas: Concordo plenamente, Concordo parcialmente, Não compreendi, Discordo.

1. **Você concorda que a visualização acima reponde a pergunta: "Qual a variabilidade dinâmica do modelo de *features*?"?**
2. **Você concorda que a visualização acima responde a pergunta: "Qual dos contextos presentes em um modelo de *features*, possui mais dinamicidade em termos de ativação e desativação?"?**
3. **Considerações sobre a visualização e/ou resposta acima.**

B.1.6 Complexidade Estrutural de um modelo

Visualização que representa a Complexidade Estrutural de um modelo. As medidas utilizadas foram: Número de *Features* (NF), Número de *Features* Mandatórias (NM), Número de *Features* Top (Ntop), Restrições *Cross-tree* (CTC); são obtidas do modelo e mostrados em um gráfico de barras.

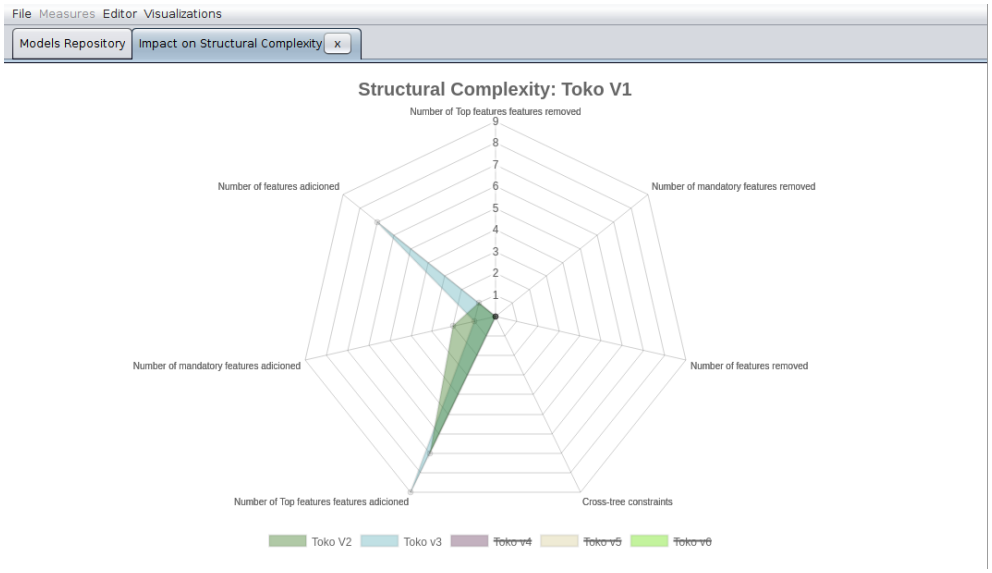


Faixa de repostas: Concordo plenamente, Concordo parcialmente, Não compreendi, Discordo.

1. **Você concorda que a visualização acima responde a pergunta: "Qual a complexidade do modelo de *features*?"**
2. **Considerações sobre a visualização e/ou resposta acima.**

B.1.7 Complexidade Estrutural entre Versões de Modelos

Visualização que ilustra a complexidade estrutural de um modelo de acordo com suas versões. Versões de um modelo são selecionadas, e desse conjunto é obtido: O Número de *Features* adicionadas e removidas, o Número de *Features* Mandatórias adicionadas e removidas, o Número de *Features* Top adicionadas e removidas, e as Restrições *Cross-tree* das versões. Na visualização pode-se observar que o Toko V4, Toko v5 e Toko V6 não estão selecionados, isso pode ser realizado dinamicamente na ferramenta, para assim ser possível observar melhor a relação entre Toko V2 e Toko V3; com apenas esses dois sendo ilustrados na visualização, pode-se observar que da Toko V2 a Toko V3 ouve um aumento de 7 *features*, 1 *feature* obrigatória, 9 *features* top; e nenhuma remoção. O gráfico radar foi utilizado para essa visualização.

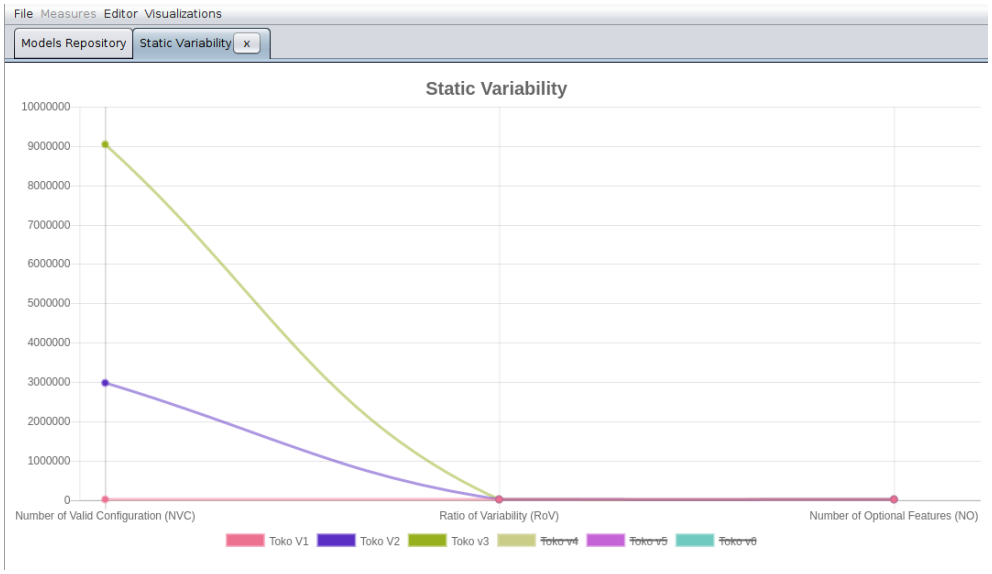


Faixa de repostas: Concordo plenamente, Concordo parcialmente, Não compreendi, Discordo.

1. **Você concorda que a visualização acima responde a pergunta: "Qual o impacto que a linha sofreria ao estender uma determinada *feature*?"**
2. **Você concorda que a visualização acima responde a pergunta: "Qual o impacto a linha de produtos sofrerá quando uma mudança ocorre no modelo de *features*?"**
3. **Você concorda que a visualização acima responde a pergunta: "Qual a evolução da complexidade do modelo de *features* quando uma *feature* é adicionada e removida?"**
4. **Considerações sobre a visualização e/ou resposta acima.**

B.1.8 Variabilidade Estática

Visualização que mostra a medidas relacionadas a variabilidade estática do modelo de *features*. Versões de um modelo são selecionadas, e desse conjunto é obtido: Número de configurações válidas (NVC), Taxa de variabilidade (RoV) e Número de *features* opcionais (NO); é ilustrado na visualização a relação entre Toko V1, Toko v2 e Toko v3, onde Toko v3 tem um número de configurações maior que Toko v2 e Toko v3, mas as outras medidas são pareadas. Foi utilizado para a visualização o gráfico de linhas.



Faixa de repostas: Concordo plenamente, Concordo parcialmente, Não compreendi, Discordo.

1. **Você concorda que a visualização acima responda a pergunta: "Qual o aumento no número de configurações a partir da inclusão de novas *features*?"?**
2. **Considerações sobre a visualização e/ou resposta acima.**

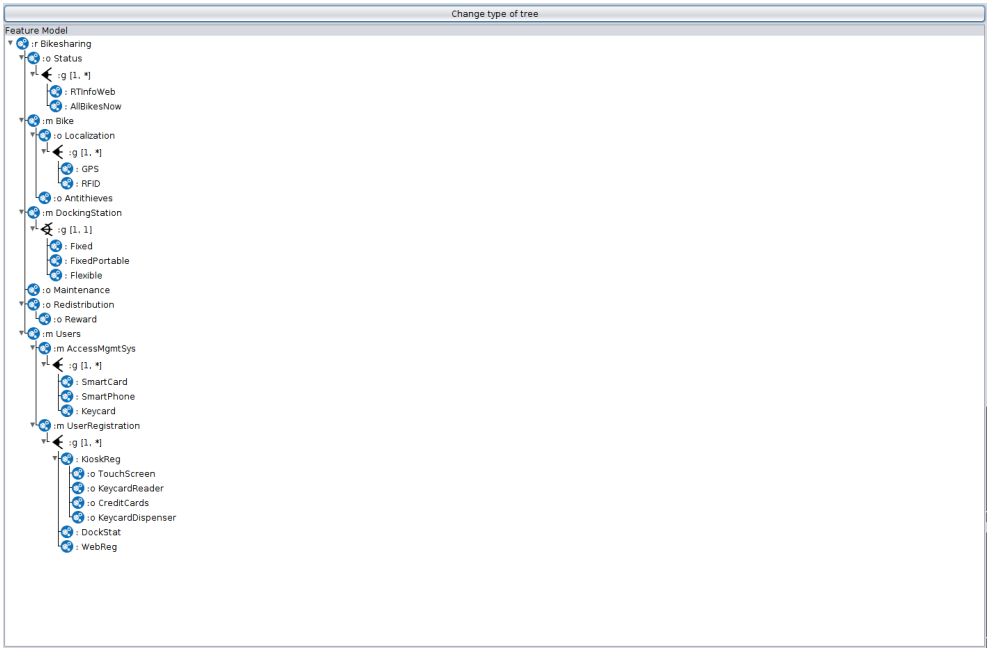
B.2 Validação da nova árvore criada para ilustrar os modelos de features na ferramenta DyMMer.

Esta Seção tem como objetivo validar a nova visualização criada para ilustrar um modelo de features e suas propriedades.

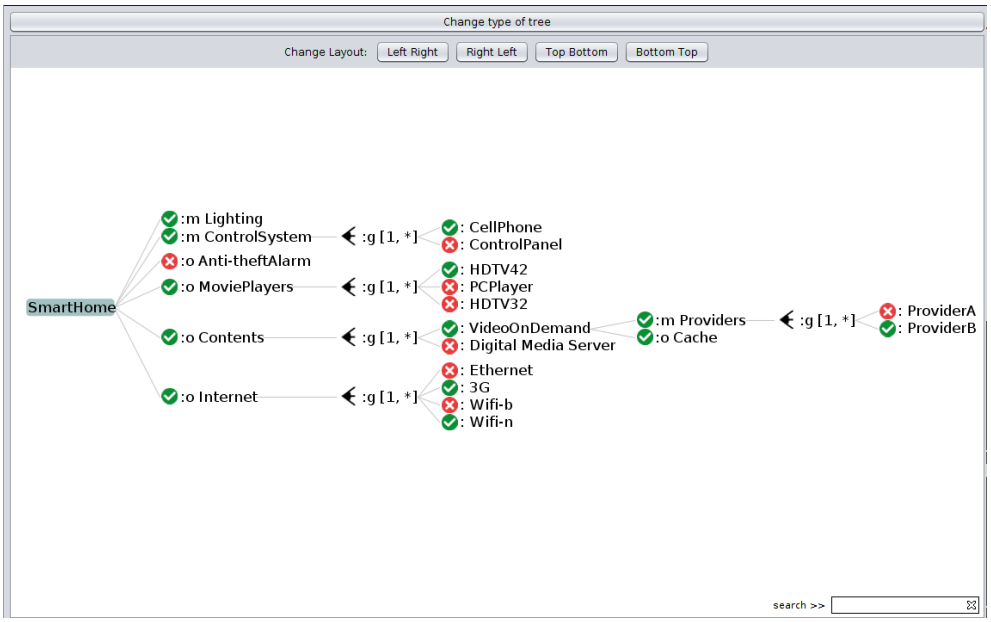
B.2.1 Como responder essa Seção?

Será ilustrado a visualização da árvore antiga e da nova árvore da ferramenta DyMMer, e será perguntado opiniões sobre a nova árvore em comparação a antiga, com os seguintes níveis de resposta: concordo plenamente, concordo parcialmente, não entendi e discordo. No final existe um campo para considerações finais das respostas fornecidas.

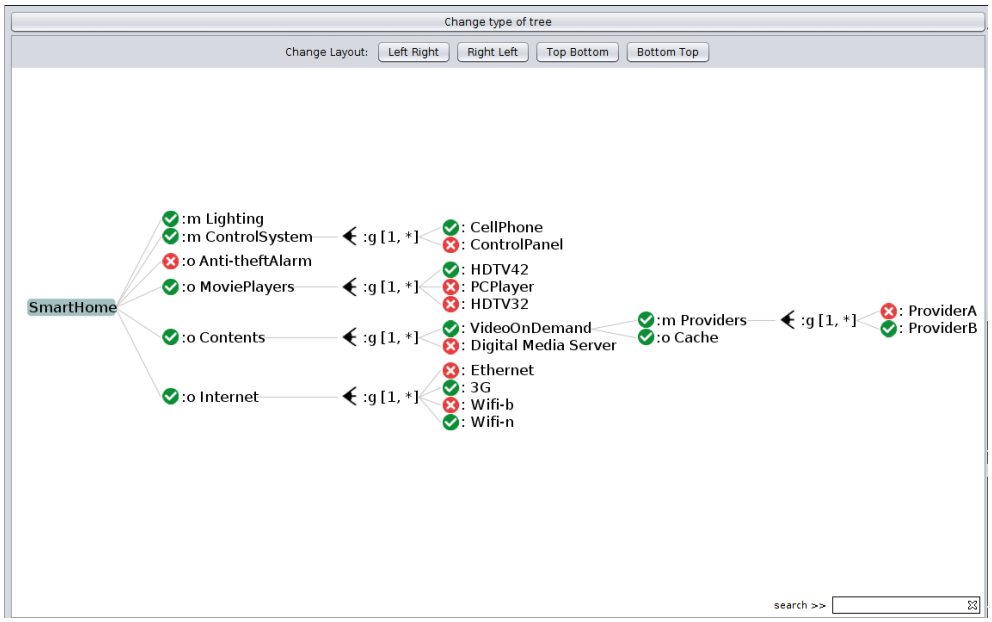
B.2.2 Antiga representação dos modelos de features



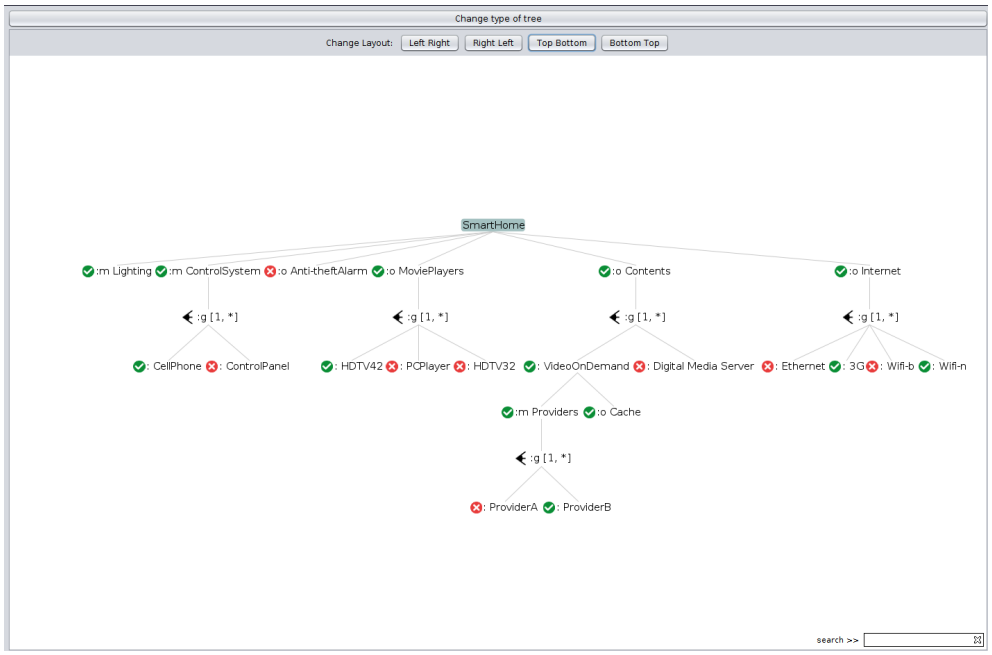
B.2.3 Nova árvore de representação (orientação Esquerda para Direita)



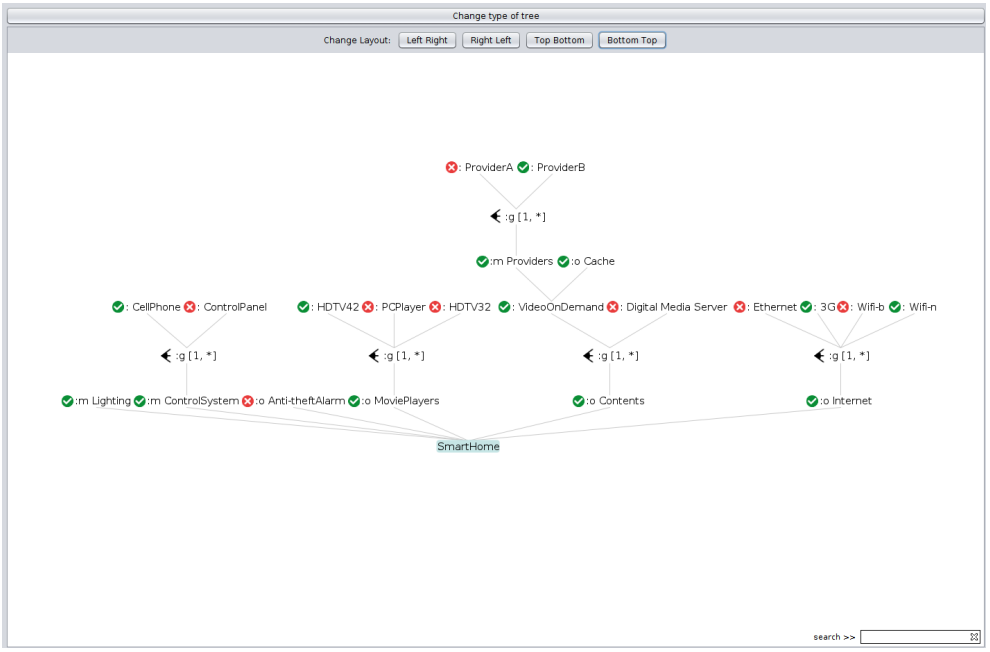
B.2.4 Nova árvore de representação (orientação Direita para Esquerda)



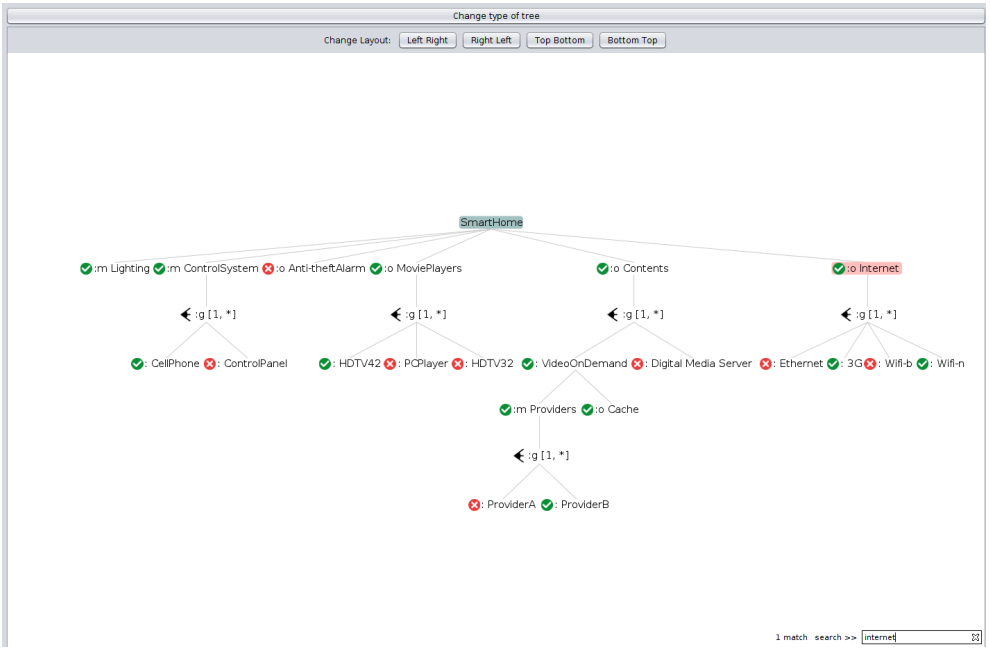
B.2.5 Nova árvore de representação (orientação Cima para Baixo)



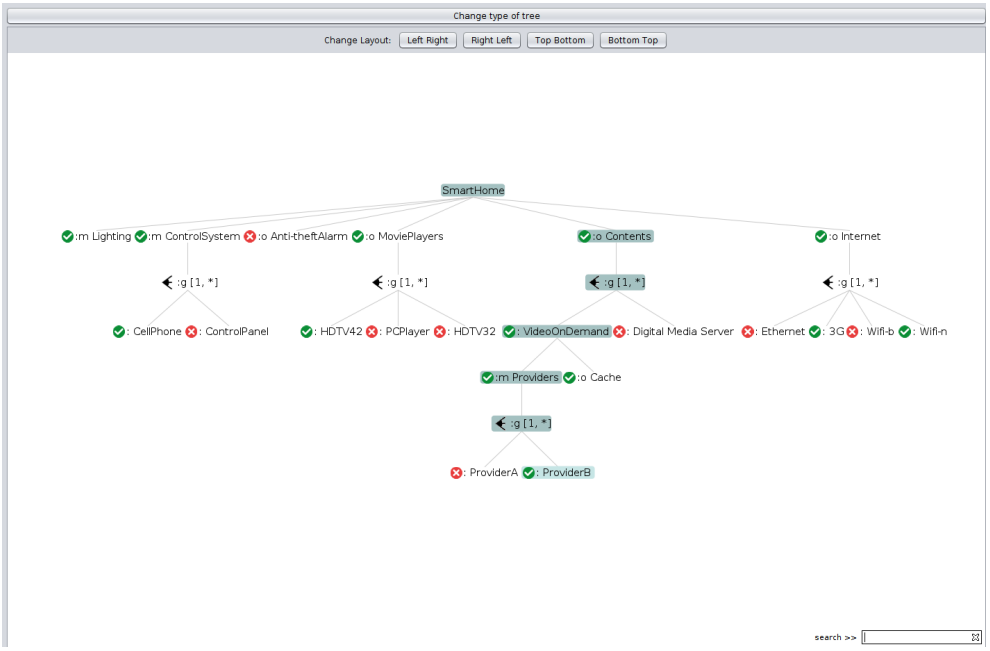
B.2.6 Nova árvore de representação (orientação Baixo para Cima)



B.2.7 Nova árvore de representação (Utilizando a funcionalidade de busca)



B.2.8 Nova árvore de representação (Utilizando a funcionalidade de criar um percurso de uma feature selecionada até a raiz do modelo)



1. **Com base nas imagens acima**

   Faixa de repostas: Concordo plenamente, Concordo parcialmente, Não compreendi, Discordo.

   a) **Você concorda que houve melhoras no entendimento em relação a antiga representação?**

   b) **Você concorda que as novas funcionalidades garantem uma melhor usabilidade?**

   c) **Você concorda estar satisfeito com a nova árvore de visualização?**

   d) **Considerações sobre a visualização e/ou as respostas acima.**