Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Programa de Pós-Graduação em Ciência da Computação

Renata Mendonça Saraiva

Um Método para Medição de Software com Foco na Interpretação de Medidas

Campina Grande, Paraíba, Brasil

2020

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

Um Método para Medição de Software com Foco na Interpretação de Medidas

Renata Mendonça Saraiva

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

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

Hyggo Almeida
(Orientador)

Campina Grande, Paraíba, Brasil

S243m Saraiva, Renata Mendonça.
Um método para medição de software com foco na interpretação de medidas / Renata Mendonça Saraiva. - Campina Grande, 2020.
190 f. : il. Color.

Tese (Doutorado em Ciência da Computação) - Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2020.
"Orientação: Prof. Dr. Hyggo Oliveira de Almeida.
Referências.

1. Método para Medição de Software. 2. Redes Bayesianas. 3. Tomada de Decisão. I. Almeida, Hyggo Oliveira de. II. Título.

CDU 004.41(43)

FICHA CATALOGRÁFICA ELABORADA PELO BIBLIOTECÁRIO GUSTAVO DINIZ DO NASCIMENTO CRB-15/515

UM MÉTODO PARA MEDIÇÃO DE SOFTWARE COM FOCO NA INTERPRETAÇÃO DE MEDIDAS

RENATA MENDONÇA SARAIVA

TESE APROVADA EM 20/08/2020

HYGGO OLIVEIRA DE ALMEIDA, Dr., UFCG
Orientador(a)

KYLLER COSTA GORGÔNIO, Dr., UFCG
Examinador(a)

DANILO FREIRE DE SOUZA SANTOS, Dr., UFCG
Examinador(a)

ROSSANA MARIA DE CASTRO ANDRADE, Dra., UFC
Examinador(a)

LENARDO CHAVES E SILVA, Dr., UFERSA
Examinador(a)

CAMPINA GRANDE - PB

Resumo

Com a evolução da tecnologia e a alta competitividade, é cada vez maior a necessidade das empresas de software de reduzir custos, melhorar a produtividade e a qualidade dos produtos entregues. Neste contexto, medidas de software são recursos essenciais para se alcançar tais objetivos. Estudos indicam que medidas, quando usadas precocemente no ciclo de desenvolvimento de software, ajudam a corrigir falhas de requisitos e prevenir erros, diminuindo o custo de desenvolvimento e prevenindo efeitos colaterais oriundos de modificações tardias. Outros estudos revelam que, a partir dos dados fornecidos pela medição, o usuário de medição pode ter uma visão concreta do andamento dos projetos e, desta forma, consegue tomar decisões baseadas em informações objetivas. Contudo, apesar dos benefícios descritos, é importante destacar que um processo de medição mal elaborado pode acarretar vários problemas, como a coleta de dados irrelevantes, redundantes, incompletos ou de baixa qualidade, tendo como consequências o desperdício de esforço e a análise de dados inconclusiva ou errônea. Neste contexto, para garantir que o usuário de medição irá tomar a decisão com base em informações coerentes e representativas da situação, é imprescindível que ele entenda o valor atribuído à medida, considerando os fatores que influenciam neste valor. Assim, dado esse objetivo e tendo como base o processo de medição abordado na ISO/IEC 15939:2007, o escopo desta tese centra-se sobre quatro etapas que impactam a interpretação de medidas: seleção de medidas, validação de medidas, validação de dados e definição de *thresholds* ou valores de referência. Na literatura, é possível encontrar diversas pesquisas abordando medidas de software, inclusive pesquisas que abrangem as etapas supracitadas. Contudo, a maioria delas, além de não ter esse viés voltado para a interpretação de medidas, trata de uma ou outra etapa de forma isolada. Com base nisto, neste trabalho de tese propõe-se um método para medição de software, que contempla as quatros etapas descritas de forma integrada, com foco no suporte à interpretação de medidas. Para fins de aplicação do método e considerando a incerteza envolvida nesta problemática, utilizou-se Redes Bayesianas. A validação do trabalho foi realizada a partir de um estudo de caso e de um grupo de foco com praticantes da indústria, evidenciando que o método em questão é útil para uma tomada de decisão mais assertiva.

Abstract

With the evolution of technology and high competitiveness, there is an increasing need for software companies to reduce costs and improve the productivity and the quality of the products delivered. In this context, software measures are essential resources to achieve these objectives. Studies indicate that measures, when used early in the software development cycle, help correct requirements flaws, prevent errors, decrease the development cost, and prevent side effects from late changes. Other studies reveal that, from the data provided by the measurement, the measurement user can view the progress of the projects and make decisions based on objective information. However, despite the benefits described, it is essential to highlight that a poorly designed measurement process can lead to several problems: the collection of irrelevant, redundant, incomplete, or low-quality data, resulting in wasted effort and inconclusive or erroneous data analysis. In this context, to ensure that the measurement user will make the decision based on coherent and representative information of the situation, he/she must understand the value attributed to the measure, considering the factors that influence this value. Thus, given this objective and based on the measurement process covered in ISO/IEC 15939:2007, this thesis's scope focuses on four steps that impact the interpretation of measures: measure selection, measure validation, data validation, and definition of thresholds or reference values. In the literature, it is possible to find several studies addressing software measures, including ones that cover the steps mentioned earlier. However, most studies, besides not focusing on the interpretation of measures, deal with one or another step in isolation. Based on this, this thesis proposes a method for measuring software, which contemplates the four steps described in an integrated manner, with a focus on supporting the interpretation of measures. Further, it models the uncertainty involved in this problem using Bayesian Networks. The validation of the work was carried out based on a case study and a focus group with industry practitioners. The results show that the method in question is useful for a more confident decision.

Conteúdo

1 Introdução **1**

1.1 Problemática 4

1.2 Objetivo 6

1.3 Metodologia 7

1.4 Contribuições 9

1.5 Estrutura do documento 10

2 Fundamentação Teórica **12**

2.1 Medição de Software 12

2.1.1 ISO/IEC 15939:2007 15

2.1.2 Seleção de Medidas 17

2.1.3 Validação de Medidas 19

2.1.4 Validade de Dados 24

2.1.5 Definição de *thresholds* 27

2.2 *Goal/Question/Metric* 32

2.2.1 Plano de Medição GQM 32

2.2.2 Aplicação do Plano GQM 34

2.3 Redes Bayesianas 36

3 Trabalhos Relacionados **43**

4 Método Proposto baseado em Redes Bayesianas **48**

4.1 Seleção de Medidas 50

4.2 Validação de Medidas 55

4.3 Validação de Dados 60

4.4 Definição de *thresholds* 67
4.5 Aplicação do Método e Condução da Medição 79
4.6 Considerações Finais do Capítulo 81

5 Validação do Método **82**
5.1 Caracterização do VIRTUS 85
5.2 Estudo de Caso 86
5.2.1 Protocolo 86
5.2.2 Resultados 90
5.2.3 Ameaças à validade 97
5.2.4 Conclusão 98
5.3 Ferramenta SAM 98
5.4 Grupo de Foco 100
5.4.1 Protocolo 105
5.4.2 Resultados 111
5.4.3 Ameaças à validade 118
5.4.4 Conclusão 120
5.5 Discussão dos Resultados 121

6 Considerações Finais **123**

A Manual do Usuário da Ferramenta SAM **142**

B Manual do Desenvolvedor da Ferramenta SAM **162**

Lista de Símbolos

AHP - *Analytic Hierarchy Process*

BPJ - *Best Professional Judgement*

DAG - *Direct Acyclic Graph*

GQM - *Goal/Question/Metric*

GQM-DSFMS - *GQM-Decision Support Framework for Metric Selection*

HP - *Historical Precedence*

MCDA - *Multi-Criteria Decision Analysis*

OO - *Object-Oriented*

QFD - *Quality Function Deployment*

ROC - *Receiver Operating Characteristic)*

SAM - *Software Attributes Measurement*

SQM - *Software Quality Metrics*

TAM - *Technology Acceptance Model*

TUAUT - Teoria Unificada de Aceitação e Uso de Tecnologia

WQM - *Web Quality Model*

Lista de Figuras

1.1 *Design science* conceitual do problema enfrentado. 7

2.1 Processo de Medição ISO/IEC 15939:2007 15
2.2 Etapas do método de Perkusich *et al.* [102] 25
2.3 Taxonomia para classificação de *threshold*. 28
2.4 Exemplo de uma folha de abstração. 34
2.5 Estrutura hierárquica do modelo GQM 35
2.6 Exemplo de rede Bayesiana 37
2.7 Expressão idiomática de causa e consequência 39
2.8 Expressão idiomática de medição 39
2.9 Expressão idiomática de síntese 40
2.10 Engenharia de Conhecimento de rede Bayesiana baseada em Especialistas proposto por Mendes [86]. 42

4.1 Visão geral do método 49
4.2 Modelo de folha de abstração. 51
4.3 Instância de folha de abstração. 53
4.4 Estrutura hierárquica do modelo GQM 54
4.5 Rede Bayesiana considerando objetivo, questões e medidas do foco de qualidade 55
4.6 Entrada e saída da etapa de seleção de medidas 55
4.7 Mapeamento entre critérios de validação associados à vantagem *Quality-focused* [88] 57
4.8 Rede Bayesiana calibrada com função de probabilidade e variância 59
4.9 Entrada e saída da etapa de validação de medidas 60

4.10 Adaptação do método de Perkusich *et al.* [102] 62
4.11 Rede Bayesiana com a confiança da medida modelada 66
4.12 Entrada e saída da etapa de validação de dados 67
4.13 Exemplo do impacto do fator de contexto *Criticidade do projeto* na interpretação das medidas *Número de alertas de análise estática* e *Cobertura de código* 69
4.14 Modelo para definição de *thresholds* 70
4.15 Seis tipos de funções de pertinência *Fuzzy*: (A) triangular, (B) z-shape, (C) trapezoidal, (D) s-shape, (E) sigmoid, e (F) Gaussiana. 72
4.16 Escala verbal para definição de probabilidades [108] 74
4.17 Exemplos de funções de pertinência calculadas por meio de interpolações cúbica, linear, quadrática e polinomial. 77
4.18 Funções de pertinência definidas para a medida *Cobertura de código*. . . . 78
4.19 Entrada e saída da etapa de definição de *thresholds* 79
4.20 Fluxo de execução do método proposto 81

5.1 Modelo de TUAUT. 82
5.2 Procedimento aplicado a unidade de análise do estudo de caso. 88
5.3 Estrutura hierárquica do modelo GQM 91
5.4 DAG completo construído para o objetivo *Avaliar a qualidade do sistema* . 92
5.5 DAG completo construído para o objetivo *Avaliar o valor entregue por sprint* 93
5.6 Medidas coletadas e interpretadas para o objetivo *Avaliar a qualidade do sistema* 93
5.7 Medidas coletadas e interpretadas para o objetivo *Avaliar o valor entregue por sprint* 94
5.8 Valores calculados para o projeto de acordo com os objetivos 94
5.9 Arquitetura da ferramenta SAM. 99
5.10 Mapeamento dos fatores de sucesso de programas medição e o TUAUT. . . 104
5.11 Estado inicial do quadro de termômetro de humor utilizado para coleta de dados no grupo de foco. 111

5.12 Resultado da votação dos fatores de sucesso para um programa de medição efetivo, com foco na interpretação de medidas. O fator em amarelo significa que ele foi elicitado duranta o grupo de foco. 113
5.13 Quadro de termômetro de humor preenchido. 116
5.14 Mapa conceitual demostrando quais fatores de sucesso para programas de medição efetivos com foco na interpretação de medidas são satisfeitos pelo método proposto. 117

Lista de Tabelas

2.1 Lista dos 47 critérios de validação [88] 21
2.2 Vantagens de medidas [88] 22
2.3 Mapeamento entre critérios de validação e vantagens [88] 22
2.4 Características de trabalhos relacionados a *thresholds* 31

3.1 Comparação entre os trabalhos relacionados e o método proposto 46

4.1 Exemplo de tabela verdade 59
4.2 Cenários *"what-if"* para a medida *Cobertura de código* em uma escala verbal. 73
4.3 Cenários *"what-if"* para a medida *Cobertura de código* em uma escala numérica. 74
4.4 Cenários *"what-if"* para a medida *Cobertura de código* em uma escala numérica normalizada. 75
4.5 TPN resultante para a medida *Cobertura de código* = 0,65 (*i.e.,* 65%). . . . 79

5.1 Fatores de sucesso para programas de medição efetivos considerados inicialmente no estudo. 102
5.2 Expressão GQM para o grupo de foco sobre as necessidade da indústria (**QP_GF_1**). 107
5.3 Expressão GQM para o grupo de foco sobre a avaliação do método proposto à luz dos fatores de sucesso para implantar sistemas de medição (**QP_GF_2**). 107
5.4 Dados demográficos dos participantes do grupo de foco. 108

Capítulo 1

Introdução

Medição tem um papel fundamental nas engenharias. Na Engenharia de Software, especificamente, medir consiste em uma atividade desafiadora em virtude da natureza abstrata do software [74]. Contudo, antes de explanar sobre os desafios atrelados a este processo, é necessário o entendimento do que ele representa e dos benefícios que ele traz no contexto da Engenharia de Sotware.

De acordo com Finkelstein e Leaning [43], medição é a representação objetiva de um conhecimento empírico acerca de uma entidade do mundo real. Em outras palavras, medição consiste na atribuição de números ou símbolos a atributos de entidades. Um software, por exemplo, é composto por uma série de atributos, tais como tamanho, confiabilidade, manutenabilidade e eficiência. Segundo Mathias *et al.* [80], quando se atribui um valor a um atributo, tem-se uma medida e, ao combinar esta medida com uma informação útil, tem-se uma métrica (*e.g.,* média de números de defeitos por módulo). Na prática, tais termos são frequentemente usados de forma intercambiável. Porém, neste trabalho, utiliza-se o termo "medida" conforme terminologia descrita na ISO/IEC 15939:2007 [123].

Os benefícios da medição e a importância desta atividade vêm sendo abordados na literatura há décadas. A partir dela, é possível planejar, monitorar, controlar e avaliar processos, produtos e recursos de software [66, 15, 49, 132]. Neste contexto, de acordo com Pressman [106], a partir dos dados fornecidos pela medição, o líder do projeto ou outro usuário de medição pode ter uma visão concreta do andamento dos projetos e, desta forma, tomar decisões baseadas em informações objetivas. Corroborando com Pressman, García *et al.* [46] afirmam que o uso da medição ajuda no enfrentamento de questões relacionadas ao desen-

volvimento e manutenção do software, uma vez que facilita a tomada de decisão.

Na prática, o uso de medidas em grandes organizações, como HP, Motorola e NASA, foi avaliado por Ordonez e Haddad [100]. O resultado deste estudo indicou que medidas, quando usadas precocemente no ciclo de desenvolvimento de software, ajudam a corrigir falhas de requisitos e prevenir erros. Além disso, os autores constataram que encontrar potenciais problemas diminui o custo de desenvolvimento e previne efeitos colaterais oriundos de modificações tardias.

De acordo com a ISO/IEC 15939:2007 [123], um processo de medição visa estabelecer, planejar, executar e avaliar a medição dentro de um projeto geral, empresa ou estrutura organizacional. Tal processo é composto por quatro atividades: i) *estabelecer e manter o comprometimento com a medição*, que envolve definir escopo, contexto, stakeholders e forma de comunicação do processo de medição; ii) *planejar o processo de medição*, que inclui todo o plano para execução do processo, desde a forma de seleção das medidas até ferramentas para viabilizar a medição; iii) *executar o processo de medição*, que envolve a execução do plano; e iv) *avaliar a medição*, guiando a melhoria contínua do processo.

Embora a referida ISO sirva de guia para o desenvolvimento e implantação de um processo de medição, este documento não contempla detalhes de como executar as atividades descritas. Neste contexto, vale ressaltar o estudo realizado por Oliveira, Oliveira e Belchior [27], no qual é apresentado um comparativo de outras documentações que abrangem o processo de medição, como CMMI-SW, IEEE Std 1061, Six Sigma, PSM (*Practical Software Measurement*), além da própria ISO/IEC 15939:2007 [27]. Como resultado deste estudo, os autores constataram que, embora esteja clara na literatura a importância de tal processo, informações sobre como executá-lo ainda são escassas. Além disso, Meding e Staron, em [84], pontuam sobre a dificuldade em ler e entender tais documentos.

Nesse sentido, é importante destacar que um processo de medição mal elaborado pode acarretar vários problemas, como a coleta de dados irrelevantes, redundantes, incompletos ou de baixa qualidade, tendo como consequências o desperdício de esforço e a análise de dados inconclusiva ou errônea. Implementar um programa de medição de software não é uma tarefa simples. Existem muitos desafios envolvidos na engenharia de medidas úteis e na garantia de um nível satisfatório de confiabilidade dos dados envolvidos [59]. Mais de 80% das iniciativas de medição de software falham dentro dos primeiros 18 meses e uma possível

justificativa para este fenômeno é a dificuldade de entender e usar medidas [144].

Medidas devem gerar valor, ao invés de dados [95]. Para Mendonça e Basili [87], uma estrutura de medição (ou programa de medição) deve ser sólida (aderir ao ambiente aplicável), completa (orientada aos objetivos do usuário), enxuta (medir apenas o necessário) e consistente (atender às metas do usuário).

Em [40], Fenton e Neil afirmam que medidas têm sido usadas com sucesso para a quantificação, mas elas não têm sido usadas adequadamente para apoiar a tomada de decisão. Posteriormente, em [36], Fenton afirma que muitos programas de medição são usados para medir o que é conveniente ou fácil de mensurar, ao invés de medir o que é necessário. Tais programas frequentemente falham porque os dados resultantes não são úteis para os desenvolvedores e mantenedores do software. Para ele, a razão chave para a grande lacuna entre como é feita a medição e como poderia ser feita é a ausência de uma coordenada, de um arcabouço que possibilite entender e usar esse processo. Vale ressaltar que a falta de dados sobre programas de medição e a dificuldade de entender e usar medidas também foram observadas por Wallace e Sheetz, em [144].

Em uma revisão sistemática realizada por Tahir, Rasool e Gencel [132], publicada em 2016, os autores afirmam que empresas de software implementam programas de medição como parte de seus processos de medição, e que, apesar da importância de tais programas, o sucesso de sua implementação ainda permanece um desafio. Os autores apontam que a falta de consenso entre pesquisadores é outro fator que dificulta ainda mais a adoção de tais programas. Aliado a isto, Perkusich *et al.* [102] acreditam que uma das razões por trás deste desafio é a dificuldade em interpretar as medidas.

Desta forma, pode-se dizer que, apesar das diversas pesquisas e publicações sobre medidas de software realizadas nos últimos anos, a aceitação e o uso delas na prática ainda são preocupações constantes, principalmente quando se consideram os avanços na área de Engenharia de Software, como a prática de GSD (*Global Software Development*) [137], a necessidade de sistemas de *Big Data* e, consequentemente, o uso de modelos de decisão e algoritmos cada vez mais complexos [73, 127].

É neste contexto de medição de software que se insere este trabalho. Mais especificamente, o foco é na utilização e interpretação de medidas para auxílio à tomada de decisão.

1.1 Problemática

Um programa de medição deve fornecer informações atualizadas e confiáveis [126], qualidades inerentes a um dos mais importantes fatores de sucesso (fidedignidade de medidas) para se estabelecer de forma efetiva tal programa [107]. De acordo com Ram *et al.*, em [107], este é um dos fatores menos conhecidos na operacionalização bem-sucedida e no uso a longo prazo de um programa de medição. Uma prova disto é que tal fator não é discutido nem por Meneely, Smith e Williams, em [88], nem por Tahir *et al.*, em [132].

Dado que medidas são usadas para auxiliar no processo de tomada de decisão, a interpretação delas deve ser correta e confiável. Interpretar uma medida significa entender o valor atribuído a ela, considerando os fatores que influenciam neste valor.

Neste contexto de interpretação de medidas, pode-se elencar quatro etapas principais do processo de medição, definido na ISO/IEC 15939:2007, que afetam esta interpretação:

- **Seleção de medidas**, que consiste em selecionar medidas adequadas para representar os atributos de uma entidade de interesse;
- **Validação de medidas**, que consiste em determinar se as medidas selecionadas medem o que se propõem a medir;
- **Validação de dados**, que consiste em se definir o grau de confiança que se tem na acurácia dos valores atribuídos às medidas;
- **Definição de *thresholds***, que consiste em se definir valores que auxiliam na classificação das medidas.

Na literatura, há vários trabalhos envolvendo estas quatro etapas. No contexto de seleção de medidas, destacam-se os trabalhos de Niessink e Van Vliet [95], Gopal *et al.* [50], Basili [141, 8], Briand, Morasca e Basili [16], Bukhari, Yahaya e Deraman [17] e Tahir *et al.* [132]. Neste último, os autores afirmam que, embora ainda não haja um consenso sobre a definição de medida, o paradigma GQM (*Goal-Question-Metric*) vem sendo amplamente adotado e recomendado para este fim. Além disso, vale ressaltar que GQM é citado como um dos fatores de sucesso para programas de medição efetivos [95, 50]. Tais fatores foram discutidos em [53, 87, 60, 126, 107].

A validação de medidas, por sua vez, foi abordada em detalhes nos trabalhos de Schneidewind [118], Kitchenham, Pfleeger e Fenton [66], Basili, Briand e Melo [9], Meneely, Smith e Williams [88], Srinivasan e Devi [124] e Antinyan *et al.* [6]. Assim como na etapa de seleção, não há um consenso entre os pesquisadores sobre como validar medidas. No entanto, as pesquisas de Meneely, Smith e Williams [88] e de Antinyan *et al.* [6] mostram-se como bons guias neste processo, uma vez que apresentam um passo a passo para a validação de medidas.

Em relação à validação de dados, tem-se o trabalho de Perkusich *et al.* [102], que apresenta uma abordagem para auxiliar a interpretação de medidas de software com base em redes Bayesianas. Embora este trabalho tenha forte relação com a presente pesquisa, tanto ele como os demais trabalhos descritos abordam uma ou até duas atividades que têm influência na interpretação de medidas. No entanto, para garantir que o usuário de medição tome decisões mais assertivas, baseadas na interpretação correta das medidas, todas as etapas supracitadas devem ser consideradas em conjunto.

Por fim, no âmbito de definição de *thresholds*, ou valores de referência, pode-se destacar as pesquisas de: Alves, Ypma e Visser [3], Shatnawi *et al.* [121, 120], Herbold, Grabowski e Waack [55], Ferreira *et al.* [42], Zhang *et al.* [150], Oliveira *et al.* [98, 96, 97], Foucault *et al.* [45], Boucher e Badri [12] e Manzano *et al.* [78]. Apesar de apresentarem boas contribuições, estes trabalhos possuem limitações, incluindo a ausência de validação e de fatores que influenciam nos *thresholds*, como o contexto dos projetos. É importante destacar que a acurácia da avaliação baseada em medidas é fortemente influenciada pela calibração de seus *thresholds* [119].

Embora os trabalhos encontrados na literatura abordem as quatro etapas isoladamente, não foram encontradas abordagens que integrem tais etapas para suporte à tomada de decisão, com foco na interpretação das medidas. Com base no exposto, enuncia-se o problema abordado neste trabalho de tese: como integrar as quatro etapas relacionadas à interpretação de medidas de software para prover suporte ao usuário no processo de tomada de decisão?

1.2 Objetivo

O objetivo geral no presente trabalho é a concepção de um método de medição de software, iterativo e incremental, para auxiliar usuários de medição no processo de tomada de decisão com foco na interpretação de medidas, considerando a integração das etapas de seleção de medidas, validação de medidas, validação de dados e definição de *thresholds*.

Para viabilizar a integração das etapas e execução do método, focando na interpretação de medidas, utilizam-se redes Bayesianas, uma vez que elas possibilitam a modelagem da incerteza de dados envolvida na problemática em questão, um maior entendimento por parte dos usuários de medição, dada a representação gráfica de sua estrutura, e facilitam as análises, modificações e interpretação de resultados [152, 140, 35], principalmente quando se considera a combinação de diferentes fontes de conhecimento.

Com base no objetivo principal, os seguintes objetivos específicos foram definidos:

- Definir etapas do método tendo como foco a interpretação de medidas;
- Definir uma abordagem para modelar a validade de medidas utilizando redes Bayesianas;
- Adaptar GQM ao processo de construção de redes Bayesianas proposto por Perkusich *et al.* [102], que tem foco na interpretação de medidas;
- Construir um processo para definição de *thresholds* baseado em conhecimento de especialistas e fatores de contexto;
- Integrar e operacionalizar as etapas do método proposto a partir da utilização de redes Bayesianas;
- Desenvolver um protótipo de sistema de medição para apoiar a validação do método proposto;
- Validar o método proposto a partir de um caso de estudo e grupo de foco.

1.3 Metodologia

Uma vez que se tem a intenção de projetar um artefato que melhore algo para as partes interessadas, bem como de investigar empiricamente o desempenho de tal artefato em um contexto, foi utilizada neste trabalho de tese a metodologia de *Design Science* [146]. De acordo com tal metodologia e com o objetivo apresentado, tem-se que o problema de pesquisa em questão é do tipo problema de *design*, uma vez que a problemática levantada exige uma mudança real e que pode haver diferentes soluções para seu enfrentamento. Além disso, a avaliação dessas soluções é feita pela utilidade delas com respeito aos objetivos dos *stakeholders* e não há uma única melhor solução [146]. O *design science* conceitual acerca da problemática desta pesquisa pode ser visualizado na Figura 1.1.

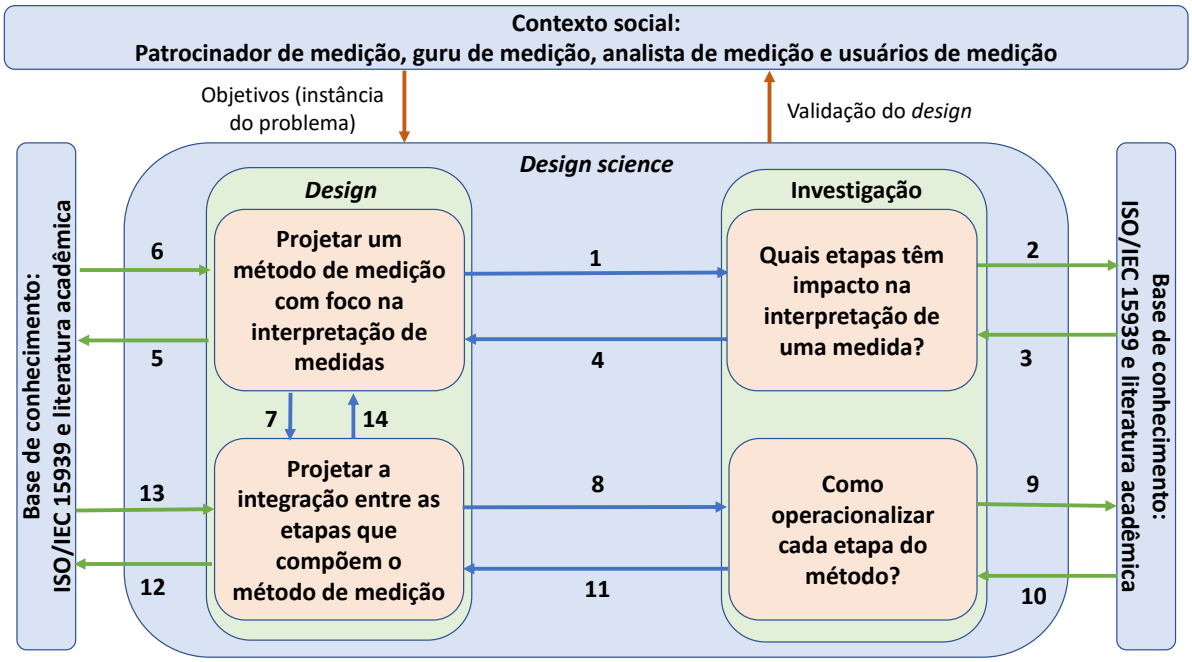


Figura 1.1: *Design science* conceitual do problema enfrentado.

Como ilustrado na Figura 1.1, as setas laranjas representam a entrada e saída do *design science*, que correspondem, respectivamente, aos objetivos para o desenvolvimento do projeto de pesquisa, bem como a validação do *design* aos *stakeholders*, que é o projeto propriamente dito. As setas verdes representam a busca por conhecimento em uma base de conhecimento, bem como a inserção de novos conhecimentos a esta base. As setas azuis,

por sua vez, representam os artefatos e contextos que motivam uma investigação, bem como o conhecimento adquirido e novos problemas de *design* identificados. De uma maneira geral, as setas enumeradas representam o fluxo de atividades seguido.

Com base neste *design* conceitual, o **objetivo de pesquisa** e o **problema de *design*** foram definidos:

Objetivo de pesquisa:

- Prover um método para auxiliar a tomada de decisão com foco na interpretação de medidas.

Problema de *design*:

Melhorar a interpretabilidade de medidas de software
Por prover um método de medição
Que satisfaça as 4 etapas que influenciam na interpretação de medidas
A fim de aumentar a assertividade nas decisões tomadas pelos usuários de medição.

Com base no objetivo e no problema de *design*, foram definidas questões de pesquisas abertas e descritivas, as quais são listadas a seguir:

- **QP1:** Quais etapas têm impacto na interpretação de uma medida?
- **QP2:** Como operacionalizar cada etapa do método?
- **QP3:** Como integrar as etapas do método viabilizando sua execução?

Tais questões foram respondidas mediante a realização das seguintes atividades:

1. Identificação e seleção de etapas de um processo de medição que tem influência na interpretação de medidas de software, tendo como material base a ISO/IEC 15939:2007. Esta atividade está relacionada à questao de pesquisa **QP1**.
2. Identificação e seleção de abordagens descritas na literatura para operacionalizar cada etapa identificada em **QP1**. Esta atividade está relacionada à questão de pesquisa **QP2**.
3. Definição de uma técnica que possibilitasse integrar as etapas identificadas em **QP1**, considerando a incerteza envolvida em cada etapa. Esta atividade está relacionada à questão de pesquisa **QP3**.

4. Desenvolvimento de um protótipo que viabilizasse a execução do método proposto a fim de validá-lo. Esta atividade está relacionada à questão de pesquisa **QP3**.

A validação do trabalho foi realizada a partir de um estudo de caso e de um grupo de foco com praticantes da indústria. O grupo de foco seguiu uma metodologia baseada no arcabouço da Teoria Unificada de Aceitação e Uso de Tecnologia (TUAUT) [142]. Mais especificamente, utilizou-se como base para avaliação da utilidade prática do método proposto o TUAUT e um mapeamento entre as características do referido método com fatores de sucesso para implantação efetiva de método de medição apresentados na literatura.

O grupo de foco foi realizado com a participação de um moderador e quatro profissionais do VIRTUS[1] - Núcleo de Pesquisa, Desenvolvimento e Inovação em Tecnologia da Informação, Comunicação e Automação – um órgão suplementar da Universidade Federal de Campina Grande (UFCG) vinculado ao Centro de Engenharia Elétrica e Informática (CEEI). Os resultados evidenciaram que o método proposto tem um efeito positivo na melhoria da interpretabilidade das medidas de um programa de medição.

1.4 Contribuições

A principal contribuição deste trabalho de tese é a descrição de um arcabouço metodológico, conceitual e prático, acerca de medição de atributos de software, tendo como foco a interpretação de medidas para auxílio à tomada de decisão. Tal arcabouço possibilita ao usuário de medição um melhor entendimento sobre o uso e interpretação de medidas e, consequentemente, uma tomada de decisão mais assertiva. Neste contexto, pode-se citar outras contribuições menores, como:

- proposição de um método iterativo e incremental para construir sistemas de medição de software com foco na interpretação de medidas constituído de quatro etapas: seleção de medidas baseada em GQM, validação de medidas, validação de dados e definição de *thresholds*.

- demonstração de como utilizar conceitos de validação de medidas para modelar a incerteza dos nós associados às medidas em questão em redes Bayesianas [116].

[1] https://www.virtus.ufcg.edu.br/

- evolução da abordagem previamente apresentada por Perkusich *et al.* [102] para gerenciar os riscos relacionados à validação de dados.

- apresentação de resultados de um estudo de caso para avaliar a utilidade prática da abordagem definida para gerenciar os riscos relacionados à validação de dados [115].

- apresentação de uma taxonomia para classificação de *thresholds*.

- demonstração de como fatores de contexto influenciam na definição de *thresholds* [114].

- demonstração de como integrar os conceitos de lógica *Fuzzy* e redes Bayesianas para o tratamento de incertezas na interpretação de medidas [117].

- demonstração de como aplicar a Teoria Unificada de Aceitação e Uso de Tecnologia (TUAUT) [142] para guiar o processo de validação de um método (ou programa) de medição.

- disponibilização de um protótipo[2] para apoiar a adoção e operacionalização do método proposto.

1.5 Estrutura do documento

O restante deste documento está organizado da seguinte forma:

- No Capítulo 2, é apresentada a fundamentação teórica que embasou a presente pesquisa. Neste contexto, são apresentados conceitos fundamentais acerca de medição de software, bem como um *overview* sobre o paradigma GQM e sobre redes Bayesianas.

- No Capítulo 3, são apresentados trabalhos relacionados à temática discutida nesta tese, bem como uma comparação entre eles.

- No Capítulo 4, o método proposto é apresentado, considerando as quatro etapas que o compõem.

[2]https://github.com/isevirtus/SAM

- No Capítulo 5, são apresentadas as validações do método como um todo, ou seja, considerando a integração das quatro etapas que o compõem.

- No Capítulo 6, as conclusões da presente pesquisa são apresentadas.

Por fim, são apresentados como anexo o manual do usuário e o manual do desenvolvedor da ferramenta desenvolvida no contexto desta pesquisa, SAM.

Capítulo 2

Fundamentação Teórica

Este capítulo é composto pelas seguintes seções: medição de software, o paradigma *Goal Question Metric* e redes Bayesianas. Na primeira seção, são descritos conceitos importantes acerca de medição de software, considerando uma visão geral da ISO/IEC 15939:2007 e as quatro etapas que compõem o método proposto: seleção de medidas, validação de medidas, validação de dados e definição de *thresholds*. Na segunda seção, o paradigma GQM é abordado, considerando seu plano de medição e aplicabilidade. Por fim, na terceira seção, são apresentados conceitos acerca de redes Bayesianas.

2.1 Medição de Software

Uma das áreas mais importantes da Engenharia de Software é a qualidade de software. Avaliar a qualidade de um processo ou de um produto de software é uma das razões de se utilizar a medição. Basili [8] lista uma série de outras razões para se medir o processo de desenvolvimento de software e o produto, como: auxiliar no planejamento do projeto, determinar os pontos fortes e fracos do processo e do produto, e avaliar o impacto de determinada técnica utilizada.

Corroborando com Basili, Shatnawi [120] afirma que medidas de software são ferramentas imprescindíveis durante o ciclo de vida de um projeto. As medidas funcionam como indicadores de qualidade e, portanto, um de seus maiores usos é encontrar onde a qualidade pode ser melhorada.

De acordo com Finkelstein e Leaning [43], medição é a representação objetiva de um co-

nhecimento empírico acerca de uma entidade do mundo real. Em outras palavras, medição consiste na atribuição de números ou símbolos a atributos de entidades. De acordo com Fenton [35], essa atribuição deve preservar observações empíricas e intuitivas sobre os atributos e entidade, podendo variar de pessoa para pessoa. Segundo ele, uma forma de se lidar com esse problema é a construção de um modelo capaz de refletir um ponto de vista específico.

Ainda de acordo com Fenton [35], após identificar o uso que se quer dar ao processo de medição (*e.g.,* avaliação ou predição), deve-se identificar quais entidades são os objetos de interesse e quais atributos dessas entidades são significantes. As entidades de interesse são classificadas em três tipos: processos, produtos e recursos. O primeiro corresponde a quaisquer atividades relacionadas ao software que ocorrem ao longo do tempo (*e.g.,* planejamento de *sprints*). O segundo tipo de entidade corresponde a quaisquer artefatos que resultam dos processos (*e.g., sprint backlog*). O terceiro consiste em itens que são as entradas para os processos (*e.g.,* time de desenvolvedores).

Assim como as entidades, as medidas dos atributos também são classificáveis [58] [35].

- Medida direta (básica): é aquela que pode ser mensurada puramente em termos de sua entidade (*e.g.,* tamanho de código).

- Medida indireta (derivada): é aquela que não pode ser mensurada diretamente a partir da observação de um atributo, mas sim da combinação de outras medidas (*e.g.,* tempo de resposta).

- Medida objetiva: consiste na contagem absoluta de um atributo da entidade e independe do autor da mensuração (*e.g.,* número de defeitos reportados).

- Medida subjetiva: consiste na classificação de um atributo a partir do julgamento humano, ou seja, pessoas diferentes podem medir valores diferentes para um mesmo atributo (*e.g.,* satisfação do usuário).

- Medida preditiva: é aquela cujo objetivo é prever com antecedência determinados aspectos do desenvolvimento (*e.g.,* velocidade da equipe).

- Medida exploratória: é aquela cujo intuito é caracterizar objetivamente os eventos (*e.g.,* alertas de análise estática).

Vale ressaltar que não há medidas definitivas que possam ser prescritas para cada objetivo aplicado em qualquer área de aplicação. Além disto, embora muitas medidas tenham sido propostas ao longo dos anos, há evidências de que muitas delas são incompletas, ambíguas e abertas a diferentes interpretações [14]. Métodos de seleção de medidas são discutidos na Subseção 2.1.2.

Uma vez selecionadas as medidas, é necessário validá-las. De acordo com Schneidewind [118], medidas deveriam ser validadas a fim de determinar se elas medem o que se propõem a medir. A validação de medidas é discutida na Subseção 2.1.3.

Após a definição e validação de medidas é importante estabelecer qual tipo de escala utilizar de acordo com a natureza dos dados a serem coletados [82]. A escala é um conjunto ordenado de valores, contínuos ou discretos, ou um conjunto de categorias a qual o atributo é mapeado. O tipo da escala, por sua vez, depende da natureza da relação entre os valores da escala. Em teoria de medição, mais especificamente em relação a software, os tipos de escalas geralmente usadas são [123, 74, 33]:

- nominal: os valores da medição são categóricos. Por exemplo, os tipos de defeitos não implicam ordem nas categorias;

- ordinal: os valores da medição são ordenados. Por exemplo, a quantidade de defeitos pode ser ordenada de acordo com o tipo de criticidade;

- intervalar: os valores de medição têm distâncias iguais correspondendo a quantidades iguais do atributo. Por exemplo, a complexidade ciclomática tem um valor mínimo de 1, mas cada incremento representa um caminho adicional. Complexidade ciclomática é uma medida de software usada para indicar a complexidade de um programa de computador. Mede a quantidade de caminhos de execução independentes a partir de um código fonte. O valor de zero não é possível.

- proporcional: os valores de medição têm distâncias iguais correspondendo a quantidades iguais do atributo, onde o valor de zero corresponde a nada do atributo. Por exemplo, o tamanho de um módulo de software em termos de LOC (linhas de código) é uma escala proporcional porque o valor de zero corresponde a nenhuma linha de código e cada incremento adicional representa a quantidade igual de código.

Outras atividades importantes atreladas à medição de software e que merecem destaque nesta pesquisa são a validação de dados e a definição de *thresholds* ou valores de referência. Tais atividades são discutidas, respectivamente, na Seção 2.1.4 e na Seção 2.1.5.

2.1.1 ISO/IEC 15939:2007

A ISO/IEC 15939:2007 [123] aborda o processo de medição dentro da Engenharia de Software e Sistemas. Tal processo é composto por atividades e tarefas, conforme apresentado na Figura 2.1.

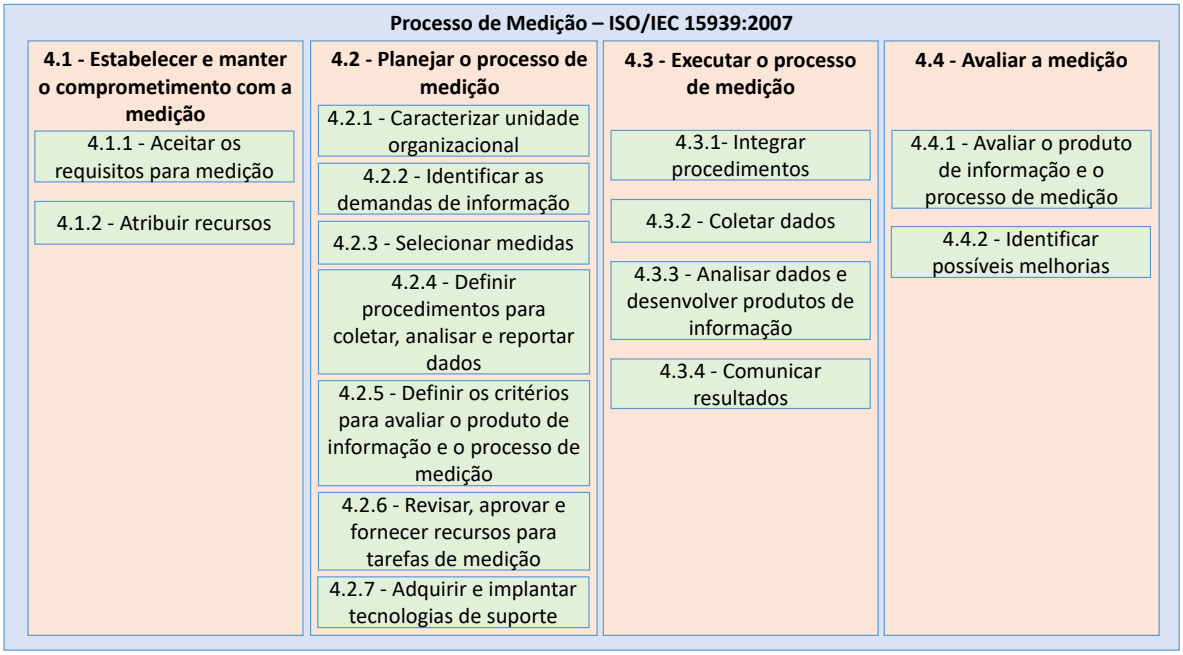


Figura 2.1: Processo de Medição ISO/IEC 15939:2007

A primeira atividade deste processo consiste em estabelecer e manter o comprometimento com a medição. Para isto, é necessário identificar o escopo de medição, considerando o contexto onde este processo será executado, os *stakeholders* envolvidos, o propósito de medição, o comprometimento e a atribuição de papéis de todos os envolvidos neste processo, e a comunicação deste comprometimento a todos que compõem a unidade organizacional.

A segunda atividade corresponde ao planejamento do processo de medição e contempla grande parte das tarefas que compõem tal processo. São elas:

- 4.2.1 - Caracterização da unidade organizacional, que consiste em detalhar mais o

escopo de medição;

- 4.2.2 - Identificação da demanda de informação, que consiste em realizar um levantamento das informações necessárias à unidade de organização, baseado em objetivos, restrições, riscos e problemas da referida unidade, priorizar tais informações e selecionar aquelas que serão enfrentadas ao longo do processo de medição, comunicando esta decisão a todos os *stakeholders*;

- 4.2.3 - Seleção de medidas, que consiste em identificar medidas candidatas para satisfazer as informações selecionadas na tarefa anterior e, a partir disto, selecionar aquelas que sejam prioritárias, com base em critérios como viabilidade e facilidade de coleta de dados. Por fim, as medidas selecionadas devem ser documentadas, considerando seu nome, unidade de medição, definição formal, método de coleta e a relação entre elas e a informação necessária;

- 4.2.4 - Definição de procedimentos para coletar, analisar e registrar dados, que consiste em definir procedimentos para especificar como os dados serão coletados e como e onde eles serão armazenados, além de procedimentos que especifiquem o método de análise dos dados (ferramentas a serem utilizadas), o formato e métodos para registro do resultado da análise (produto de informação) e procedimentos de gerenciamento de configuração;

- 4.2.5 - Definição de critérios para avaliar o produto de informação e o processo de medição, que consiste em definir critérios que permitam determinar se os dados necessários foram coletados e analisados com qualidade suficiente para satisfazer a demanda de informação, como a confiabilidade no método de medição, e critérios para avaliar o processo de medição, como *timeliness* e eficiência;

- 4.2.6 - Revisão, aprovação e fornecimento de recursos para tarefas de medição, que consiste em revisar e aprovar as tarefas que compõem o plano de medição, bem como prover recursos para que tal plano seja implementado;

- 4.2.7 - Aquisição e implantação de tecnologias de suporte, que consiste em avaliar, selecionar e disponibilizar tecnologias, ferramentas, dados e até cursos de treinamento que viabilizem a execução do plano de medição.

A terceira atividade do processo de medição corresponde à execução do referido processo, que consiste em integrar os procedimentos de coleta, geração, análise e registro de dados aos processos relevantes da unidade organizacional; comunicar tais procedimentos aos provedores dos dados; coletar, armazenar, verificar e analisar os dados; interpretar o resultado da análise realizada, considerando o contexto das medidas; revisar e documentar o produto de informação gerado, comunicando isto a todos os usuários de medição.

Por fim, a quarta atividade do processo corresponde à avaliação do produto de informação e do processo de medição, com base nos critérios selecionados na segunda atividade, bem como a identificação de possíveis melhorias.

As temáticas abordadas na literatura, detalhadas no Capítulo 3, estão alinhadas com as atividades que compõem o processo de medição da referida ISO. Neste contexto, e tendo como foco a interpretação de medidas, o estado da arte sobre as quatro etapas que compõem o método prosposto é apresentado nas subseções seguintes.

2.1.2 Seleção de Medidas

Programas de medição bem estabelecidos são altamente influenciados pela seleção de medidas de software [47]. Centenas de medidas de software já foram propostas na literatura. No entanto, na maioria dos casos, elas foram propostas em situações *ad hoc*, não havendo um guia de como medi-las e utilizá-las [17]. Bouwers, Deursen e Visser [13] afirmam que a escassez de informações sobre como escolher e selecionar as melhores medidas tem deixado pesquisadores e engenheiros em constante questionamento. A fim de unificar tais informações de modo a facilitar esta escolha, Bukhari, Yahaya e Deraman [17] discutem os pontos fortes e fracos dos seguintes métodos de seleção de medidas de software:

- *Best Professional Judgement (BPJ)*: consiste em selecionar medidas a partir da combinação de vários fatores, como: conhecimento do especialista, critério de decisão e objetivos do projeto [76];
- *Historical Precedence (HP)*: consiste na avaliação de medidas utilizadas em projetos equivalentes [22];
- *Web Quality Model (WQM)*: consiste na classificação de medidas Web a partir de características Web (*e.g.,* conteúdo e navegação), de características de qualidade (*e.g.,*

usabilidade e manutenabilidade) e do processo do ciclo de vida do projeto (*e.g.,* desenvolvimento e manutenção) [18];

- *Meta-metrics*: consiste em selecionar medidas baseadas em características específicas delas mesmas (*e.g.,* facilidade de uso e custo) [129];

- *Multi-Criteria Decision Analysis (MCDA)*: similar a abordagem BPJ, uma vez que tem como base a definição de fatores ponderados, opiniões de especialistas e preferências de *stakeholders* [77]. No entanto, a abordagem MCDA é mais sistemática;

- *GQM-Decision Support Framework for Metric Selection (GQM-DSFMS)*: consiste em processo de seleção de medidas baseado em objetivos e iterativo, incluindo um mecanismo de tomada de decisão, um repositório de medidas e atributos predefinidos, e um modelo rastreável entre elementos GQM [47].

Em uma revisão sistemática da literatura, Tahir, Rasool e Gencel [132] discutem métodos de seleção de medidas no contexto de programas de medição orientados a objetivo. Eles categorizam os métodos em: (*i*) uso de padrões (*e.g.,* ISO/IEC 15939:2007, ISO/IEC/IEEE 24765:2010 e ISO/IEC 9126-x), (*ii*) uso de especialista em medição e (*iii*) uso de ferramentas automatizadas. Em relação a (*i*), os autores afirmam que não há um consenso na comunidade sobre que padrão utilizar e que a maioria dos padrões apresentam somente medidas de qualidade ou medidas básicas de gerenciamento de projeto, como "tamanho". Em relação a (*ii*), os autores enfatizam a importância de um especialista, capaz de selecionar o número apropriado de medidas em relação aos objetivos da organização. Em relação a (*iii*), os autores categorizam as ferramentas automatizadas em dois tipos: aquelas que foram desenvolvidas especificamente para o planejamento de programa de medição e aquelas que as empresas utilizam como parte do seu processo de gerenciamento.

Ainda de acordo com os autores supracitados, medição orientada a objetivo é considerada um fator de sucesso para a implementação de programas de medição. Eles observaram que 83% dos modelos de planejamento de medição e 90% das ferramentas estenderam ou melhoraram o modelo GQM original, ou propuseram uma nova abordagem baseada em objetivo [132].

2.1.3 Validação de Medidas

A validade de uma medida é definida como o grau em que ela representa um determinado atributo. Sendo assim, o uso de medidas inválidas compromete o programa de medição, uma vez que decisões podem ser tomadas com base em dados que não são representativos da entidade objeto de estudo.

De acordo com Schneidewind [118], medidas deveriam ser validadas para determinar se elas medem o que se propõem a medir. De nada adianta selecionar medidas se elas não forem representativas do atributo quantificado. De acordo com Kitchenham [65], qualquer resultado empírico obtido a partir de medidas inválidas é inútil. Portanto, medidas inválidas devem ser removidas dos *suites* de medidas e nenhum estudo de validação empírica incluindo tais medidas deveria ser publicado [65].

Na literatura, é possível encontrar muitas pesquisas que abordam a questão da validação de medidas [118, 35, 66, 65, 88, 124, 6]. Ao longo dos últimos 40 anos, pesquisadores têm debatido sobre o que constitui uma medida válida [88]. Uma vez que este debate tem como ponto central critérios de validação de medidas de software, Meneely, Smith e Williams [88] realizaram uma revisão sistemática acerca dos critérios de validação encontrados na literatura acadêmica.

Embora ainda não haja um consenso entre os pesquisadores sobre como validar medidas, a pesquisa de Meneely, Smith e Williams [88] mostra-se como um bom guia nesse processo, uma vez que apresenta um passo a passo para a validação de medidas a partir de 47 critérios de validação encontrados na literatura, ilustrados na Tabela 2.1, que representam uma visão diversa do que constitui uma medida válida.

Outra pesquisa que representa um bom guia neste contexto de validação de medidas é a de Antinyan *et al.* [6], que baseia-se no sistemático processo de pesquisa-ação típico de definir, refinar e redefinir o processo de validação com os profissionais, o que permite moldar a medida pretendida e aceitá-la ou rejeitá-la para aplicação posterior.

Em [88], é apresentada uma análise da categorização dos critérios, considerando conflitos, temas em comum e motivações filosóficas existentes por trás de cada um deles. De acordo com essas motivações, os autores afirmam que uma medida pode ser usada tanto como uma forma pragmática de melhorar o software e seus processos circundantes, que

corresponde à filosofia dirigida a objetivo, ou mais rigorosamente como um meio para a compreensão da própria natureza do software, que corresponde à filosofia dirigida à teoria.

De acordo com Srinivasan e Devi [124], há dois tipos de validações para medidas de software: teórica e empírica. Enquanto alguns autores referem-se à "validação teórica" como "validação interna" e "validação empírica" como "validação externa", Meneely, Smith e Williams [88] distinguem as ideias. Para eles, validação interna e externa referem-se ao que está sendo validado, enquanto validação teórica e empírica referem-se a como uma medida é validada. De forma mais específica, validação interna refere-se a quão bem uma medida mede um atributo, enquanto a validação externa relaciona uma medida a um fator de qualidade (*e.g.,* uma outra medida). Por sua vez, a validação teórica usa lógica para argumentar formalmente se uma medida é válida ou não, enquanto a validação empírica emprega análises de dados a partir de experimentações ou observações. Vale destacar ainda a validade de construção (*construct validity*), que difere da validade interna e externa uma vez que trata sobre como uma medida é implementada e estudada [88].

Definir o objetivo de utilização de uma medida é uma etapa crítica para validá-la. Além disto, ao tomar essa decisão, o usuário de medição pode especificar propriedades da medida que são mais apropriadas ao seu uso. Os autores supracitados chamam isto de "vantagem". Mostrar que uma medida é uma representação significante de determinado atributo é uma vantagem (*Meaningfulness*) e mostrar que ela pode ser aplicada a um processo de desenvolvimento é outra vantagem (*Practicality*). Para demonstrar que uma medida tem vantagens específicas, o usuário de medição precisa usar os critérios de validação que oferecem estas vantagens [88]. Neste contexto, a partir dos 47 critérios encontrados durante a pesquisa, foram observadas 11 vantagens comuns, ilustradas na Tabela 2.2. A partir do mapeamento entre vantagens e critérios de validação, ilustrados na Tabela 2.3, Meneely, Smith e Williams [88] apresentam um processo de aplicação de critérios capaz de auxiliar líderes de projetos na escolha dos critérios apropriados para uma medida específica.

Suponha que um líder de projeto propõe usar a medida "anos de experiência", referente ao somatório dos anos de experiência de cada integrante do time, para avaliar a habilidade do time em desenvolver software de alta qualidade. O líder acredita que ao utilizar tal medida, tempo e dinheiro serão poupados. A vantagem *Efficiency* se aplica ao caso, uma vez que ela possibilita que usuários de medição e pesquisadores poupem tempo e energia a partir

Tabela 2.1: Lista dos 47 critérios de validação [88]

A priori validity	*Monotonicity*
Actionability	*Metric Reliability*
Appropriate Continuity	*Non-collinearity*
Appropriate Granularity	*Non-exploitability*
Association	*Non-uniformity*
Attribute validity	*Notation validity*
Causal model validity	*Permutation validity*
Causal relationship validity	*Predictability*
Content validity	*Prediction system validity*
Construct validity	*Process or Product Relevance*
Constructivenessy	*Protocol validity*
Definition validityy	*Rank Consistency*
Discriminative power	*Renaming insensitivity*
Dimensional consistency	*Repeatability*
Economic productivity	*Representation condition*
Empirical validity	*Scale validity*
External validity	*Stability*
Factor independence	*Theoretical validity*
Improvement validity	*Trackability*
Instrument validity	*Transformation invariance*
Increasing growth validity	*Underlying theory validity*
Interaction sensitivity	*Unit validity*
Internal consistency	*Usability*
Internal validity	

Tabela 2.2: Vantagens de medidas [88]

Mathematical Soundness	*Meaningfulness*
Practicality	*Decision-Informing*
Correctness	*Quality-Focused*
Efficiency	*Theory-Building*
Difference-Detecting	*Consensus Contribution*
Hypothesis-Strengthening	

da escolha de uma medida simples para um dado objetivo de medição. Desta forma, o analista de medição verifica que os critérios de validação associados à vantagem *Efficiency* são: *Improvement Validity* e *Usability*. Uma medida tem *Improvement Validity* se a medida é uma melhoria de medidas existentes. Esta melhoria pode se referir à facilidade de medição, por exemplo. Uma medida tem *Usability* se ela pode ser implementada de forma rentável em um programa de garantia de qualidade. Uma vez que o líder de projeto demonstra que a referida medida segue ambos os critérios de validação, pode-se afirmar que esta medida é válida para o propósito pretendido.

Tabela 2.3: Mapeamento entre critérios de validação e vantagens [88]

#	Critérios	*Mathematical Soundness*	*Practicality*	*Correctness*	*Efficiency*	*Hypothesis-Strengthening*	*Meaningfulness*	*Decision-Informing*	*Quality-Focused*	*Theory-Building*	*Consensus Contribution*	*Difference-Detecting*
1	*A Priori Validity*					X						
2	*Actionability*		X					X				
3	*Appropriate Continuity*	X					X				X	
4	*Appropriate Granularity*											X
5	*Association*							X		X		
6	*Attribute Validity*						X				X	
7	*Causal Model Validity*		X					X		X		
8	*Causal Relationship Validit*		X					X		X		

Continua na próxima página

#	Critérios	*Mathematical Soundness*	*Practicality*	*Correctness*	*Efficiency*	*Hypothesis-Strengthening*	*Meaningfulness*	*Decision-Informing*	*Quality-Focused*	*Theory-Building*	*Consensus Contribution*	*Difference-Detecting*
9	*Content Validity*						X				X	
10	*Construct Validity*			X							X	
11	*Constructiveness*							X		X		
12	*Definition Validity*										X	
13	*Discriminative Power*		X					X		X		
14	*Dimensional Consistency*	X					X					
15	*Economic Productivity*		X					X				
16	*Empirical Validity*					X				X		
17	*External Validity*							X	X	X		
18	*Factor Independence*					X	X					
19	*Improvement Validity*				X							
20	*Instrument Validity*	X		X								
21	*Increasing Growth Validity*	X									X	
22	*Interaction Sensitivity*	X									X	
23	*Internal Consistency*						X					
24	*Internal Validity*						X					
25	*Monotonicity*	X										
26	*Metric Reliability*			X								
27	*Non-collinearity*					X			X			
28	*Non-exploitability*		X									
29	*Non-uniformity*	X					X					X
30	*Notation Validity*			X							X	
31	*Permutation Validity*						X					
32	*Predictability*		X						X	X		
33	*Prediction System Validity*		X					X	X			
34	*Product or Process Relevance*		X									
35	*Protocol Validity*			X							X	
36	*Rank Consistency*					X		X	X			
37	*Renaming Insensitivity*						X					
38	*Repeatability*											
39	*Representation Condition*					X				X		

Continua na próxima página

#	Critérios	*Mathematical Soundness*	*Practicality*	*Correctness*	*Efficiency*	*Hypothesis-Strengthening*	*Meaningfulness*	*Decision-Informing*	*Quality-Focused*	*Theory-Building*	*Consensus Contribution*	*Difference-Detecting*
40	*Scale Validity*	X										
41	*Stability*			X								X
42	*Theoretical Validity*						X					
43	*Trackability*					X		X				
44	*Transformation Invariance*											X
45	*Underlying Theory Validity*									X		
46	*Unit Validity*						X					
47	*Usability*		X		X				X			

2.1.4 Validade de Dados

A validação de dados refere-se à confiança que se tem na acurácia da medida, ou seja, o grau de confiança de que o valor coletado de um medida representa o valor real dela [74]. Por exemplo, se o "número de erros" em um determinado sistema é 0, mas nenhum teste foi realizado, conclui-se que esta medida não é confiável.

Neste contexto, há dois passos para se atribuir o valor de uma medida, a partir da coleta ou do registro de dados. Para validar estes dados, deve-se considerar riscos (*e.g.,* métodos falsos) e/ou fatores (*e.g.,* prioridade dos testes) que influenciam no processo de coleta e no registro de dados.

Na literatura, é possível encontrar algumas pesquisas que combinam medidas de software com fatores subjetivos na construção de modelos para auxiliar a tomada de decisão. Fenton e Neil [38], por exemplo, apresentaram uma abordagem para predizer defeitos e recursos considerando fatores subjetivos (*e.g.,* qualidade de funcionários e ferramentas) e medidas de software. Ahmed e Muzaffar [1], por sua vez, propuseram um arcabouço baseado em lógica *Fuzzy* tipo 2 para predizer esforço de desenvolvimento de software, tendo como entradas medidas e opiniões de especialista. Em [143], Wagner apresentou um modelo baseado em

atividade para avaliar e predizer qualidade de software usando medidas de software como indicadores das atividades e fatores do modelo.

Uma limitação das pesquisas descritas é que elas não consideram riscos inerentes ao processo de coleta. Tais riscos influenciam na validade dos dados e, consequentemente, na sua interpretação. Em [102], Perkusich *et al.* afirmam que as abordagens existentes para auxiliar na interpretação de medidas consistem na definição de *thresholds* para determinar se o valor de uma medida é confiável ou não. No entanto, tais abordagens não são suficientes para garantir uma correta interpretação da medida, uma vez que elas não consideram riscos e fatores que influenciam esse processo. Com base nisto, os autores propuseram um método para construir modelos, baseados em redes Bayesianas, para auxiliar na interpretação de medidas considerando a influência de riscos e fatores subjetivos sobre este processo.

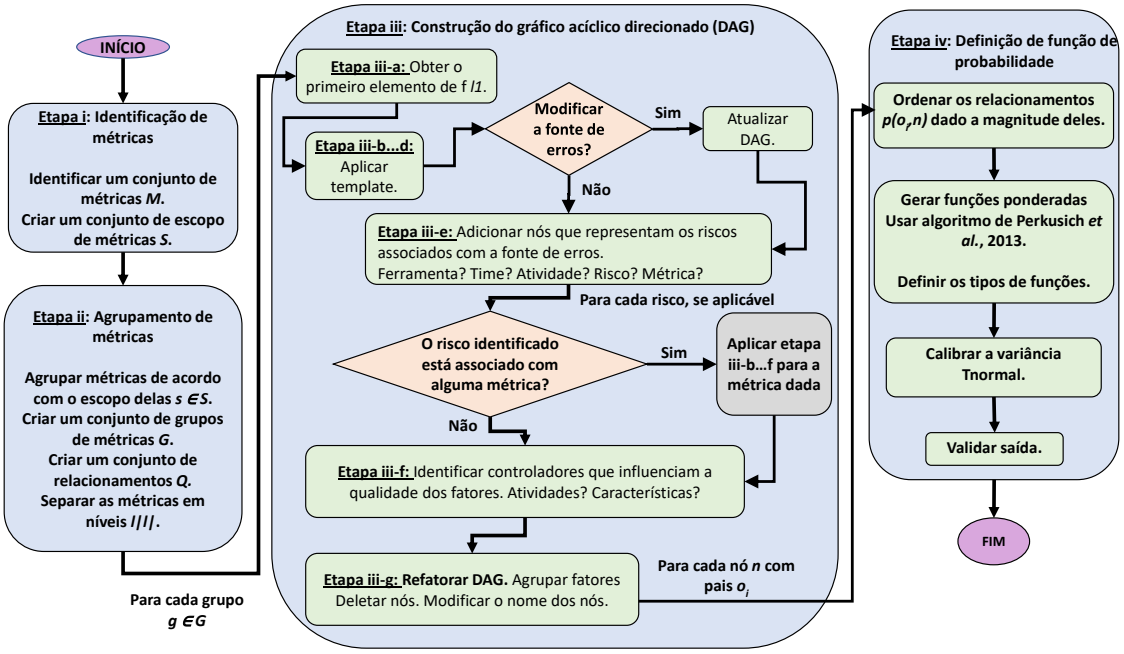


Figura 2.2: Etapas do método de Perkusich *et al.* [102]

O método de Perkusich *et al.* [102], ilustrado na Figura 2.2, é composto pelas seguintes etapas: (i) identificação de medidas, (ii) agrupamento de medidas; (iii) construção de um grafo acíclico direcionado (*DAG*) e (iv) definição de funções de probabilidade.

Na primeira etapa, identifica-se um conjunto de medidas e os escopos em que elas estão inseridas. Na segunda etapa, as medidas são agrupadas de acordo com os escopos aos quais pertencem. Medidas têm mesmo escopo se elas têm alguma influência na interpretação umas

das outras. Por exemplo, medidas relacionadas a *bugs* e testes, tais como *cobertura de teste*, *número de bugs abertos* e *status do teste* deveriam pertencer ao mesmo escopo, uma vez que *cobertura de teste* e *status do teste* influenciam na interpretação de *número de bugs abertos*. Ainda nesta etapa, as medidas de um mesmo grupo devem ser separadas em níveis, de acordo com a dependência existente entre elas.

Na terceira etapa do referido método, um DAG é construído para cada grupo de medidas. Para isto, deve-se aplicar um *template* no qual é criado um nó que representa a interpretação da medida, outro que representa a medida coletada, e mais um que representa o *Error*, ou seja, a incerteza da medição. A este nó podem ser adicionados alguns nós pais, como: *Qualidade do procedimento de coleta de medida*, *Qualidade do procedimento de registro de medida* e *Uso indevido da medida*. Riscos e controladores para mitigá-los também podem ser adicionados. Por fim, o DAG é refatorado.

A quarta e última etapa do método em questão corresponde à definição das funções de probabilidade, bem como a calibração da variância e a validação da saída. Para este propósito, os autores utilizaram AgenaRisk[1], ferramenta que possibilita executar redes Bayesianas e calcular as probabilidades. AgenaRisk usa a distribuição normal duplamente truncada (distribuição TNormal) para representar nós ordinais [41]. A distribuição TNormal é caracterizada por dois parâmetros: média (μ) e variância (σ^2). μ é calculada pela expressão ponderada que reflete a influência dos pais de um dado nó sobre ele mesmo. Assim, os autores utilizaram três tipos de funções ponderadas suportadas pelo AgenaRisk: média ponderada, mínimo ponderado e máxima ponderada [39]. O mínimo ponderado deve ser usado em situações em que se qualquer nó pai é *Baixo*; o nó filho tende a ser *Baixo* também. A máxima ponderada deve ser usada em situação em que se qualquer nó pai é *Alto*; o nó filho tende a ser *Alto* também. Em uma situação diferente dessas, usa-se a média ponderada.

Por fim, vale ressaltar que a validação do método apresentado em [102] consistiu de um estudo de caso realizado em três projetos de desenvolvimento. Dado isto, os autores afirmam que, apesar de não poderem concluir que o método é útil para a interpretação de medidas, os resultados mostraram que a abordagem é promissora.

[1] www.agenarisk.com

2.1.5 Definição de *thresholds*

Thresholds, ou valores de referência, correspondem à semântica de uma medida, uma vez que particiona a escala da medida em grupos, como discutido nos trabalhos de Foucault *et al.* [45] e Shatnawi [120]. Na ausência de um consenso na academia com relação aos conceitos relacionados à definição de *thresholds*, conforme explicado a seguir, construiu-se, como parte deste trabalho de tese, a taxonomia apresentada na Figura 2.3. De acordo com a taxonomia definida, os *Thresholds* podem ser classificados de acordo com quatro facetas, descritas a seguir.

- **Escala semântica**: corresponde a escala a ser utilizada para representar a semântica da medida. Os tipos mais comuns de escalas semânticas de medida são: *Boolean* e ordinal [117]. A medida *Cobertura de Código*, por exemplo, poderia ter uma escala semântica do tipo *Boolean* com os seguintes valores: *OK* e *Não OK*. Tal medida também poderia ter uma escala semântica ordinal com os valores: *Ruim*, *Razoável* e *Bom*.

- **Conhecimento**: representa os tipos de fonte de conhecimento utilizados para definir os *thresholds*. A fonte de conhecimento pode ser baseada em dados históricos de projetos (*e.g.,* código fonte e números de defeitos) [3, 42, 45, 96, 98, 120, 121, 150] ou baseada em especialistas [78].

- **Mapeamento**: corresponde ao mapeamento entre a escala original da medida e a escala semântica. Tal mapeamento pode ser de dois tipos, absoluto ou probabilístico, sendo o primeiro mais comum. No caso do mapeamento absoluto, um valor da medida sempre irá pertencer a um e somente um valor possível da escala semântica associada, seguindo os princípios da lógica clássica. Por exemplo, a escala original da medida *Cobertura de Código* é contínua no intervalo $[0, 1]$. Ao se realizar o mapeamento desta escala para a escala semântica do tipo *Boolean*, tem-se que o valor atribuído à *Cobertura de Código* será associado a *OK* ou a *Não OK*, nunca a ambos simultaneamente. No caso do mapeamento probabilístico, o valor da medida pode ser associado com mais de um valor da escala semântica, tendo uma probabilidade (*i.e.,* grau de pertencimento) associada com cada valor possível da escala semântica. Por exemplo, o valor $0,8$ da medida *Cobertura de Código* poderia ser mapeado para uma escala semântica

do tipo *Boolean* tendo 80% de pertencimento ao valor *OK*, e 20% de pertencimento ao valor *Não OK*. Como exemplos de implementação de tal mapeamento, tem-se o uso de *boxplot* [79, 64] e lógica *Fuzzy* [117, 151].

- **Regra**: corresponde às regras utilizadas para definir os valores que compõem os *thresholds*. Tais regras podem ser classificadas em dois tipos: marginal e intervalar [79]. Em relação à marginal, há duas regras possíveis: **MaiorQue**(δ) e **MenorQue**(δ), onde δ é o valor do *threshold*. O *threshold* intervalar, por sua vez, pode ter três ou mais regras do tipo **Entre**(γ, λ), onde γ é o limite inferior, podendo ser igual a δ_i e λ é o limite superior, podendo ser igual a δ_{i+1}. Em escalas do tipo *Boolean*, utilizam-se apenas regras marginas. Em escalas ordinais, utilizam-se ambas as regras.

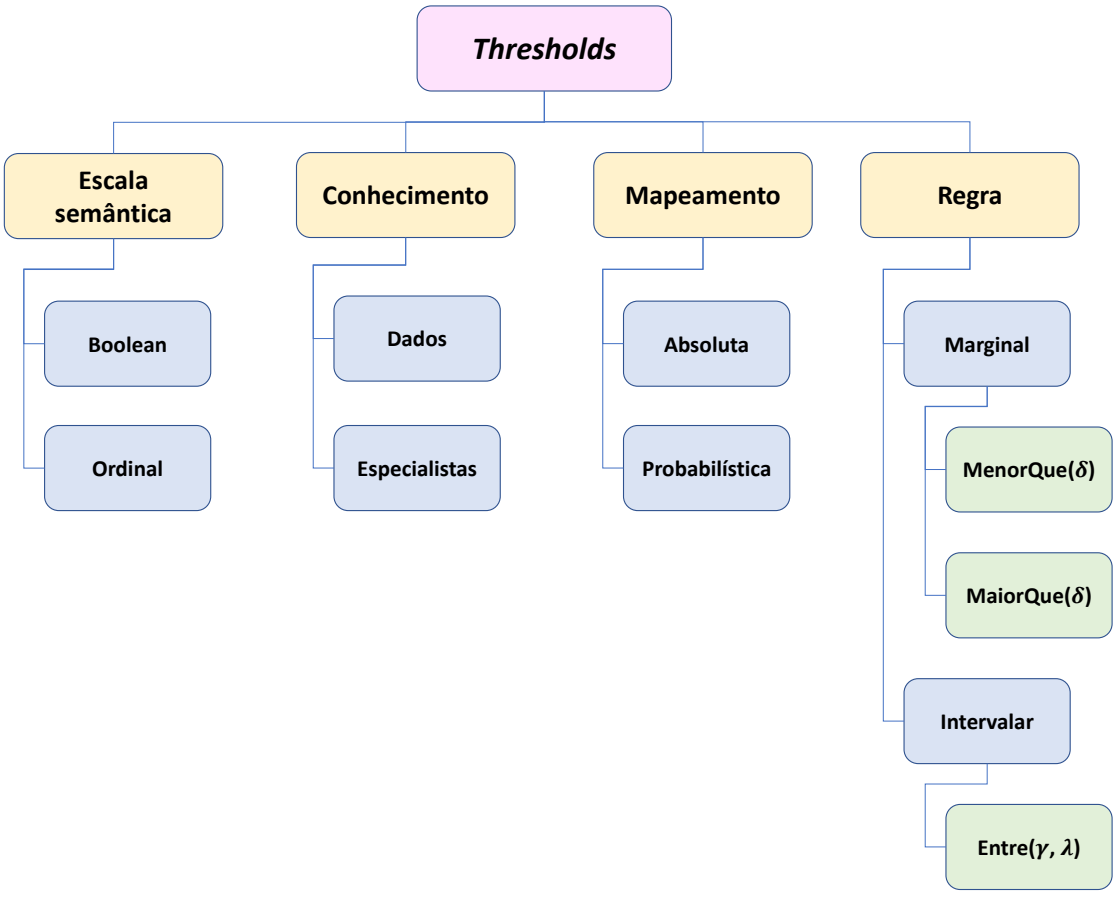


Figura 2.3: Taxonomia para classificação de *threshold*.

Desta forma, pode-se representar um *threshold* $t_m = M(R, S)$, onde m é a medida referente ao *threshold*, R refere-se ao conjunto de intervalos que compõem as regras dos

thresholds, S refere-se ao conjunto de valores que compõem a escala semântica da medida m, e M corresponde ao mapeamento (1:1) entre R e S.

De acordo com Alves, Ypma e Visser [3], o uso eficaz de medidas de software é dificultado pela falta de *thresholds* significativos. Na literatura, poucas medidas possuem *thresholds* conhecidos e determinados. Neste contexto, abordagens para identificar tais valores têm sido discutidas por vários pesquisadores [3, 121, 42, 113, 150, 98, 96, 45, 97, 120].

Em [3], Alves, Ypma e Visser apresentam uma abordagem para derivar *thresholds* de medidas baseada em dados de medição, obtidos a partir de um conjunto representativo de sistemas orientados a objeto. Em [121], Shatnawi *et al.* utilizaram curvas *Receiver Operating Characteristic* (ROC) para identificar *thresholds* para algumas medidas de Orientação a Objeto (OO). Eles acreditam que valores úteis e significativos devem ser claramente e explicitamente associados com fatores do projeto de interesse. Por exemplo, para reduzir a probabilidade de erros em um módulo, os *thresholds* para as medidas do módulo devem estar associados com a probabilidade dos erros deste módulo. Desta forma, este método contribuiu para a investigação relacionada à associação do valor da medida aos erros encontrados, seja pela presença ou ausência de erros, ou pelo nível do erro (baixo, médio ou alto).

Assim como Shatnawi *et al.*[121], Sánchez *et al.* [113] também utilizaram curvas ROC para a identificação de *thresholds*. No entanto, eles compararam a técnica de curvas ROC com a do método Bender. Ambos utilizam uma abordagem baseada em duas etapas. O primeiro passo é estimar a função discriminadora e o segundo é a determinação dos limites. Para ambos os métodos, a regressão logística é utilizada para estimar uma função discriminadora. Como conclusão deste estudo, os autores afirmaram que curvas ROC obtêm *thresholds* mais acurados.

Também focando em medidas OO, Ferreira *et al.*, em [42], usaram a ferramenta *EasyFit* a fim de encontrar a distribuição que mais se assemelha à distribuição da medida. Com isto, se a distribuição encontrada tem um valor médio representativo, esse valor é usado como *thresholds*, caso contrário, ocorre a quantificação da distribuição em: boa, moderada ou ruim. A partir da validação, os autores perceberam que tais valores não expressam as melhores práticas de Engenharia de Software, porém eles expõem um padrão da maioria dos sistemas de software.

De acordo com Zhang *et al.*, em [150], uma falha do trabalho de Ferreira *et al.* [42] é que

eles dividiram todos os sistemas de software usando os fatores de contexto, sem examinar se esses fatores afetam a distribuição das medidas ou não. Isto resultou em uma alta proporção de *thresholds* duplicados. Para reduzir duplicações e maximizar as amostras de sistemas de software de medição, uma divisão só é necessária quando um fator de contexto afeta a distribuição do valor de medida. Com base nisto, Zhang *et al.*, em [150], investigaram se a distribuição de valores de medidas de manutenabilidade variava de acordo com os seguintes fatores de contexto: domínio de aplicação, linguagem de programação, idade, tempo de vida, número de alterações e número de downloads. Como conclusão, os autores afirmaram que todos os seis fatores impactaram a distribuição de valores de 51% das medidas, sendo a linguagem de programação o fator com maior índice de impacto, impactando a distribuição de valores de 90% das medidas.

O conceito de *thresholds* relativos e de uma ferramenta para a extração desses valores são propostos por Oliveira *et al.*, em [98, 96, 97]. Já Foucault *et al.*, em [45], destacam-se por considerar o contexto dos projetos no cálculo de *thresholds*. A abordagem apresentada pelos autores foi definida por meio da aplicação de dois métodos estatísticos: (i) *Double sampling* [136], que consiste em selecionar aleatoriamente amostras de projetos; e o (ii) *Bootstrap*, que estima os *thresholds* baseados em quartis. O processo de validação consistiu em um teste para identificar a melhor configuração para a abordagem, uma vez que os dois métodos estatísticos são amplamente utilizados.

Em [120], Shatnawi *et al.* propuseram uma abordagem que inicialmente transforma a distribuição das medidas em uma distribuição logarítmica, deixando-a mais próxima possível da distribuição normal. Em seguida, o *threshold* temporário (T') é coletado utilizando a média (M) e o desvio padrão (SD), de forma que $T' = M + SD$ ou $T' = M - SD$. Por fim, o T' é convertido para a distribuição original, por meio de exponenciação do T', gerando o *threshold* final.

Em uma pesquisa mais recente, Manzano *et al.* [78] utilizaram redes Bayesianas para criar indicadores estratégicos no contexto de desenvolvimento de um software específico. Em [78], Manzano *et al.* definiram os *thresholds* das medidas *Qualidade do Código*, *Estabilidade do Código* e *Qualidade do Produto*, todas tendo valores contidos no intervalo $[0, 1]$. Nesse estudo, definiu-se como escala semântica, para todas as medidas, uma escala ordinal de cinco pontos: *Muito Baixo*, *Baixo*, *Médio*, *Alto* e *Muito Alto*. Como consequência, foram

definidos cinco *thresholds*. Para a medida *Qualidade do Código*, foram definidos os seguintes *thresholds*: ([0 - 0,805), *Muito Baixo*), ([0,805 - 0,807), *Baixo*), ([0,807 - 0,808), *Médio*), ([0,808 - 0,9), *Alto*), e ([0,9 - 1], *Muito Alto*). Não foram apresentados detalhes do processo de elicitação de conhecimento dos especialistas de domínio para definir os *thresholds*.

As principais características das pesquisas descritas estão consolidadas na Tabela 2.4.

Autor/Referência	**Características da pesquisa**
Alves, Ypma e Visser [3]	• A abordagem proposta não é impulsionada pela opinião de especialistas; • É repetível, transparente e fácil de realizar; • Os *thresholds* correspondem aos cortes nos percentis dos dados, classificados em baixo risco, risco moderado, alto risco e altíssimo risco.
Ferreira *et al.* [42]	• Considera o contexto; • Os *thresholds* são classificados em bom, moderado e ruim.
Zhang *et al.* [150]	• Considera o contexto; • Provê um guia de como agrupar os sistemas de software de acordo com seis fatores de contexto.
Oliveira *et al.* [98, 96, 97]	• Introduziu o conceito de *thresholds* relativos.
Foucault *et al.* [45]	• Considera o contexto; • Usou Bootstrap para estimar os *thresholds*
Shatnawi *et al.*[121] Sánchez *et al.* [113]	• Avaliou a associação do valor da medida aos erros.
Shatnawi *et al.* [120]	• Realiza uma transformação logarítmica nos dados, deixando-os o mais próximo de uma distribuição normal.
Manzano *et al.* [78]	• Utiliza redes Bayesianas para criar indicadores estratégicos no contexto de desenvolvimento de um software específico.

Tabela 2.4: Características de trabalhos relacionados a *thresholds*

2.2 *Goal/Question/Metric*

Goal Question Metric (GQM) é um paradigma orientado a objetivo proposto em 1984 [10] para definir um programa de medição baseado em objetivos explícitos e precisamente definidos. De acordo com Briand, Differding e Rombach [15], GQM garante planos de medições adequados, consistentes e completos; ajuda a gerenciar a complexidade de programas de medição; e estimula uma discussão estruturada sobre medição.

2.2.1 Plano de Medição GQM

Briand, Differding e Rombach, em [15], definem um plano de medição GQM na segunda etapa do processo de medição orientada a objetivo. Neste processo, a primeira etapa corresponde à caracterização do ambiente, onde são identificadas as características relevantes da organização e da entidade a ser medida. A segunda etapa, além de compor a definição do plano GQM, compõe também a identificação dos objetivos mensuráveis. Esta identificação é realizada com base nas informações reunidas durante a primeira etapa.

GQM provê *templates* para definir objetivos mensuráveis baseado em cinco aspectos: (i) objeto de estudo, corresponde ao principal alvo do estudo (*i.e.,* produto); (ii) propósito, corresponde à razão pela qual o objeto deve ser analisado; (iii) foco de qualidade, consiste no atributo particular do objeto de estudo que será analisado (*i.e.,* custo, confiabilidade e corretude); (iv) ponto de vista, representa o papel da pessoa que irá utilizar a saída do programa de medição; e (v) contexto, o ambiente em que o estudo será realizado. Na prática, os objetivos são limitados aos recursos dedicados à melhoria e medição do processo, à profundidade da compreensão dos processos correntes, à estabilidade dos processos e pontos de vista envolvidos no programa de medição [15].

O plano de medição GQM contém a informação necessária ao planejamento da medição e à realização da análise dos dados. Ele é composto de: objetivo, que corresponde ao nível conceitual, definido por um objeto de medição como produtos, processos e recursos; questão, que corresponde ao nível operacional em que um conjunto de questões é usado para decidir se o objetivo específico foi alcançado; e medida, que corresponde ao nível quantitativo em que um conjunto de dados é coletado para responder às questões do nível operacional de forma quantitativa, lembrando que definir uma medida inclui definir sua escala e variação.

Além disto, o plano de medição contém modelos que usam dados coletados como entrada para gerar respostas às questões. De acordo com Briand, Differding e Rombach [15], é importante definir o tipo de modelo a ser usado durante a definição do plano GQM, uma vez que ele pode influenciar na definição das medidas e dos procedimentos de coleta de dados. Vale ressaltar também que questões e medidas podem ser relacionadas a fatores de qualidade e a fatores que afetam estes fatores de qualidade, como características do processo e do produto (*i.e.,* experiência dos desenvolvedores e complexidade do código).

Para definir um plano GQM, faz-se necessária a coleta de dados a partir de especialistas. Folhas de abstração ou, do inglês, *abstraction sheets*, que são usadas como aquisição de conhecimento durante as entrevistas [15], são popularmente usadas com este propósito. Elas são compostas de um objetivo e de quatro quadrantes.

O objetivo é baseado em cinco aspectos: (i) o objeto de estudo, que consiste no alvo principal do estudo; (ii) o propósito, que representa o porquê de o objeto de estudo ser analisado; (iii) o foco de qualidade, que consiste no atributo particular do objeto de estudo que será utilizado para o propósito definido (iv) o ponto de vista, que representa o papel ou posição da pessoa que irá usar a saída do programa de medição; e (v) o contexto, que se refere ao ambiente em que o estudo será executado.

Em relação aos quadrantes, têm-se:(i) foco de qualidade, que captura a intuição do especialista sobre o foco da qualidade como definição operacional. Em outras palavras, lista um conjunto de indicadores que deverá ser usado para medir o aspecto de qualidade no foco de interesse; (ii) fatores de variação, que capturam a intuição do especialista relacionada aos fatores que impactam o foco da qualidade (*i.e.,* experiência dos desenvolvedores); (iii) hipótese de *baseline*, que especifica a expectativa do especialista com a respeito às medidas e modelos que definem o foco de qualidade (*i.e.,* é esperado ter trinta falhas detectadas); e (iv) hipótese de variação, que captura o impacto esperado dos fatores de variação sobre os fatores de qualidade (*i.e.,* quanto maior a qualidade dos casos de teste, maior as falhas detectadas). Ambos, foco de qualidade e fatores de variação, desencadeiam perguntas e podem ser medidas. Um exemplo de uma folha de abstração pode ser visualizado na Figura 2.4.

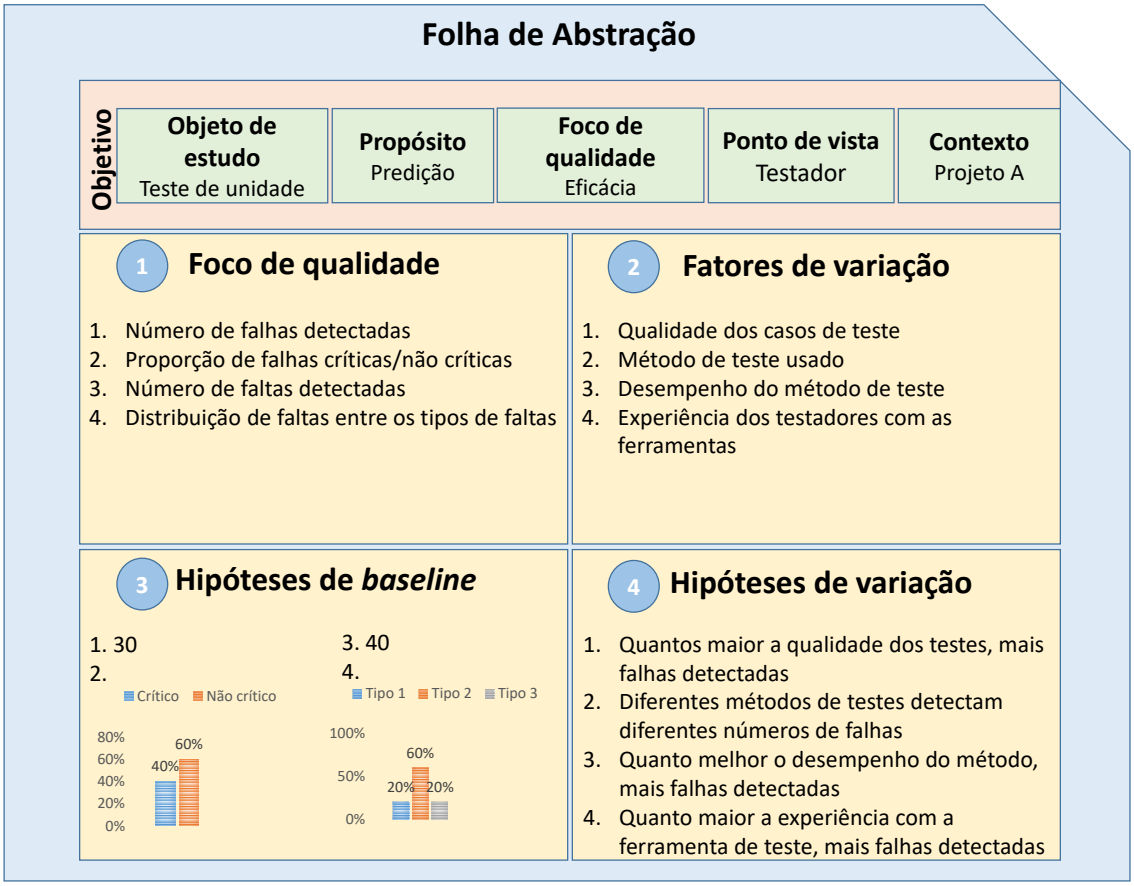


Figura 2.4: Exemplo de uma folha de abstração.

2.2.2 Aplicação do Plano GQM

O procedimento para a coleta de dados, correspondente à terceira etapa do processo de medição orientada a objetivo, deve ser especificado com base no plano GQM. Nesta etapa, o momento, a pessoa responsável e os meios para a coleta de dados são definidos. Há vários riscos associados a esta etapa. Para definir a pessoa responsável, por exemplo, critérios como experiência, viés e motivação devem ser considerados. Além disto, a definição de instrumentos de medição é crucial para se obter dados confiáveis. Estes instrumentos são caracterizados em três categorias: ferramentas, questionários e entrevistas estruturadas. As ferramentas podem ser utilizadas para medir artefatos objetivos, enquanto os questionários e entrevistas podem medir processos e artefatos subjetivos.

Uma vez que o procedimento de coleta de dados é definido, a próxima etapa corresponde

à coleta, análise e interpretação dos dados. É importante destacar que GQM é usado de forma *bottom-up*, ou seja, de baixo para cima, durante sessões de *feedback* e a partir de modelos, para analisar a avaliação do objetivo. As medidas coletadas devem responder às questões levantadas e o alcance do objetivo pode ser finalmente analisado. A estrutura hierárquica do modelo GQM é ilustrada na Figura 2.5.

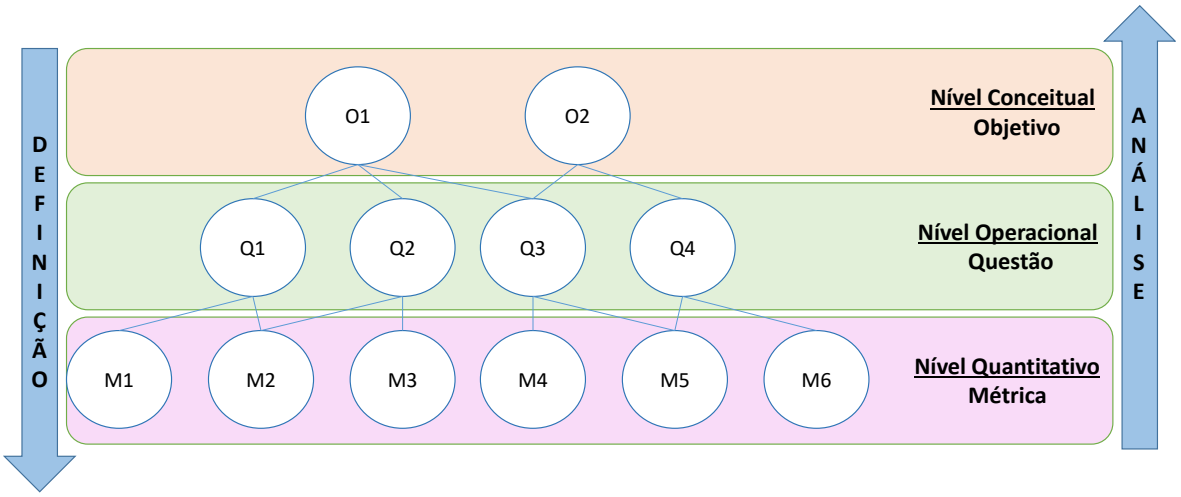


Figura 2.5: Estrutura hierárquica do modelo GQM

Ainda no contexto da análise e interpretação dos dados medidos, deve-se levar em conta uma série de importantes requisitos para o sucesso do programa de medição, como: a confiança mútua entre gerente e desenvolvedores, o comprometimento dos desenvolvedores e as ferramentas utilizadas no processo. Se estes requisitos não forem atendidos, o programa não obterá sucesso devido a comportamentos humanos indesejados. Se os desenvolvedores não forem motivados e treinados para seguir o plano GQM, eles podem produzir erros de medição devido à falta de experiência ou na tentativa de burlar os resultados para apresentar números favoráveis ao gerente. Além disto, se os desenvolvedores não forem treinados para usar as ferramentas requisitadas ou estas não forem confiáveis, os dados coletados não serão confiáveis. No contexto de GQM, onde a tomada de decisão é baseada na medição, medidas não confiáveis podem levar a decisões erradas.

2.3 Redes Bayesianas

Redes Bayesianas constitui um mecanismo para construção executável de modelos que lidam com incerteza e que auxiliam o processo de tomada de decisão [93]. Elas são aplicadas em vários contextos da Engenharia de Software, como gerenciamento de riscos [32] [37] [62] [56], predição de qualidade [11] [61] e gerenciamento de processos [104].

Uma rede Bayesiana é um grafo direcionado cujos nós representam as variáveis aleatórias e cujas arestas representam os relacionamentos causais entre essas variáveis. De acordo com Nielsen e Jensen [94], uma rede Bayesiana consiste de:

- um conjunto de variáveis e um conjunto de arestas entre as variáveis;
- um conjunto finito de estados mutuamente exclusivos para cada variável;
- um grafo acíclico direcionado, do inglês *direct acyclic graph (DAG)*, formado pela junção entre variáveis e arestas direcionadas;
- uma função de probabilidade associada a cada variável A com pais $B_1, \ldots, B_n$, $P(PA|B_1, \ldots, B_n)$, onde esta função é, geralmente, representada por uma tabela de probabilidade condicional, também chamada de tabela de probabilidade de nó (TPN). Se A não tiver pais, então a tabela se reduz à tabela de probabilidade incondicional $P(A)$.

Um exemplo de rede Bayesiana é ilustrado na Figura 2.6. Elipses representam os nós das variáveis e as arestas são representadas pelos arcos. As funções de probabilidade são representadas por tabelas e geralmente são definidas a partir do conhecimento do especialista do domínio. Embora os arcos representem a direção de causalidade entre as variáveis, a propagação da informação pode ocorrer em qualquer direção [101].

De acordo com Ziv e Richardson [152], incerteza é inerente e inevitável em um produto e processo de desenvolvimento de software. Segundo eles, as incertezas de software devem ser modeladas e gerenciadas explicitamente por meio de técnicas de modelagem de incerteza, como redes Bayesianas. Redes Bayesianas oferecem um modelo computacional matemático para raciocínio de incerteza e a representação gráfica de sua estrutura combina com sistemas

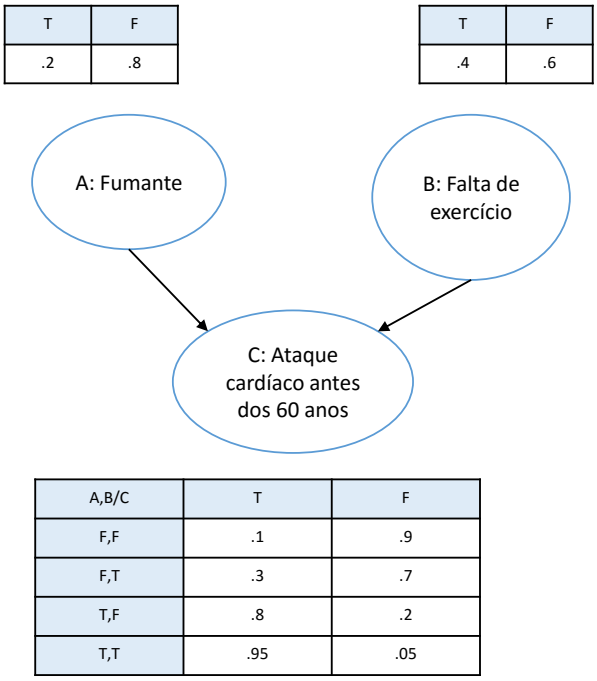


Figura 2.6: Exemplo de rede Bayesiana

de software [152]. Logo, uma vez que os profissionais de software têm experiência com estruturas de árvores, representar redes Bayesianas graficamente facilita a comunicação entre eles e facilita análises e modificações. Além destas vantagens, redes Bayesianas se destacam pela adequação a conjuntos de dados pequenos ou incompletos, pela possibilidade de combinação de diferentes fontes de conhecimento e pelo fornecimento de respostas rápidas [140].

Em relação às desvantagens de se usar redes Bayesianas, pode-se listar as seguintes: a discretização de variáveis contínuas, a estruturação do conhecimento de especialistas e ciclos de *feedback* [140]. Além disto, para um problema relativamente grande, a construção de um grafo estruturado e as definições de funções de probabilidade podem aumentar a complexidade do entendimento dos profissionais [93]. Neste contexto, existem várias abordagens que simplificam funções de probabilidade. Uma delas é a partir do uso de nós ranqueados, baseados em uma distribuição Normal truncada denotada por TNormal (μ, σ^2, 0, 1), onde μ corresponde à média, σ^2 corresponde à variância e [0,1] corresponde ao intervalo de truncamento. Os estados dos nós ranqueados são expressados em uma escala ordinal, podendo ser mapeados em uma escala numérica limitada, contínua e ordenada uniformemente [41].

A abordagem baseada em nós ranqueados consiste basicamente em se definir a probabi-

lidade dos nós filhos a partir de uma função ponderada dos valores dos nós pais, μ, sendo esta classificada em:

- *Mean Average*, que consiste na média ponderada;
- *Minimum*, que consiste no mínimo ponderado;
- *Maximum*, que consiste no máximo ponderado;
- *MixMinMax*, que consiste na mistura das funções *Minimum* e *Maximum*.

A escolha de qual função utilizar é feita pelo desenvolvedor do modelo com base na tabela verdade, construída pelo especialista do domínio. A variância, σ^2, é um indicador de quanto uma distribuiçao pode variar os valores. Assim, quanto menor o valor da variância, mais confiança o especialista tem no resultado. Neste trabalho de tese, a identificação dos valores da variância será abordada na atividade de validação de medidas.

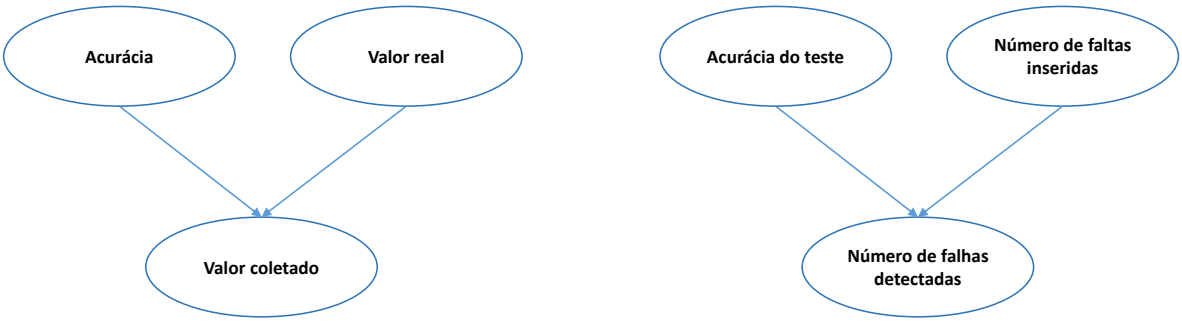
Em relação à construção de um grafo estruturado, Fenton e Neil [39] apresentam um conjunto de expressões idiomáticas. Uma expressão idiomática é um fragmento de rede Bayesiana que representa a parte gráfica de tipos genéricos de raciocínio de incerteza. De acordo com os autores, usar tais expressões acelera o processo de desenvolvimento da rede e melhora a qualidade da mesma. Neste contexto, eles apresentam quatro expressões idiomáticas comuns:

- De causa e consequência: modela a incerteza de um processo causal com consequências observáveis;
- De medição: modela a incerteza sobre a acurácia de qualquer tipo de medição;
- De síntese: modela a síntese ou combinação de muitos nós dentro de um nó para o propósito de organizar a rede Bayesiana; também modela a definição determinística ou incerta entre as variáveis;
- De indução: modela a incerteza relacionada ao raciocínio indutivo com base na população de membros semelhantes ou permutáveis.

As formas básicas e instâncias das expressões idiomáticas que são utilizadas no presente trabalho (causa e consequência, medição e síntese) são ilustradas nas Figuras 2.7, 2.8 e 2.9, respectivamente.

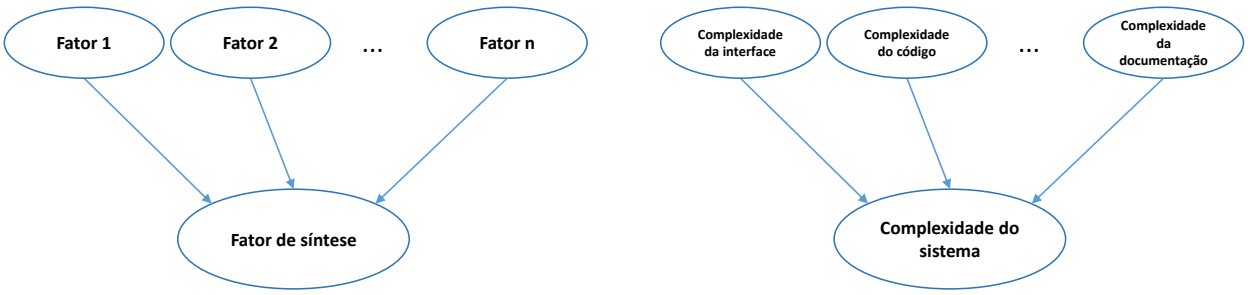
(a) Forma básica (b) Instância

Figura 2.7: Expressão idiomática de causa e consequência



(a) Forma básica (b) Instância

Figura 2.8: Expressão idiomática de medição



(a) Forma básica (b) Instância

Figura 2.9: Expressão idiomática de síntese

Para este trabalho de tese, é importante destacar que um processo popular para a construção de redes Bayesianas, a partir de conhecimento de especialistas, é o EKEBN, processo de Engenharia de Conhecimento de Rede Bayesiana baseada em Especialistas, proposto por Mendes [86] e ilustrado na Figura 2.10.

O EKEBN é composto por quatro etapas: construção da estrutura, combinação da estrutura, quantificação da incerteza e validação do modelo. Na primeira etapa do processo, são identificadas as variáveis, seus estados e os relacionamentos entre elas. Ao final da primera etapa, tem-se o grafo da rede Bayesiana definido. Na segunda etapa, referente à combinação da estrutura, todas essas variáveis, estados e relacionamentos são combinados em um único modelo de rede Bayesiana. Esta etapa só é aplicável para empresas nas quais há diferentes grupos construindo estruturas separadas de redes Bayesianas e que estão dispostos a combinar tais estruturas em uma só, tendo como resultado um modelo global para a organização. A terceira etapa corresponde a quantificação da incerteza, ou seja, as probabilidades a serem utilizadas no modelo devem ser obtidas a partir da elicitação do conhecimento de especialistas, dados, literatura, ou uma combinação dessas fontes de conhecimento.

Por fim, a quarta etapa do processo consiste em verificar se a rede construída necessita de revisão em alguma das etapas anteriores. Geralmente, utiliza-se dois métodos diferentes de validação: explicação do modelo passo a passo e a acurácia preditiva. A primeira forma de validação representa o uso de cenários de casos reais preparados para avaliar se as previsões fornecidas por uma rede Bayeasiana correspondem às previsões que os especialistas escolheriam com base em seus próprios conhecimentos. A acurácia preditiva, por sua vez, usa

dados históricos para obter previsões. Assim, dada uma variável de destino (*e.g.,* qualidade, esforço), o sucesso da primeira abordagem de validação é medido com base na frequência probabilística do valor previsto pela rede corresponder à avaliação dos próprios especialistas. Em relação a segunda abordagem de validação, o sucesso é medido com base na frequência probabilística com a qual o valor previsto da rede corresponde ao valor real dos dados anteriores.

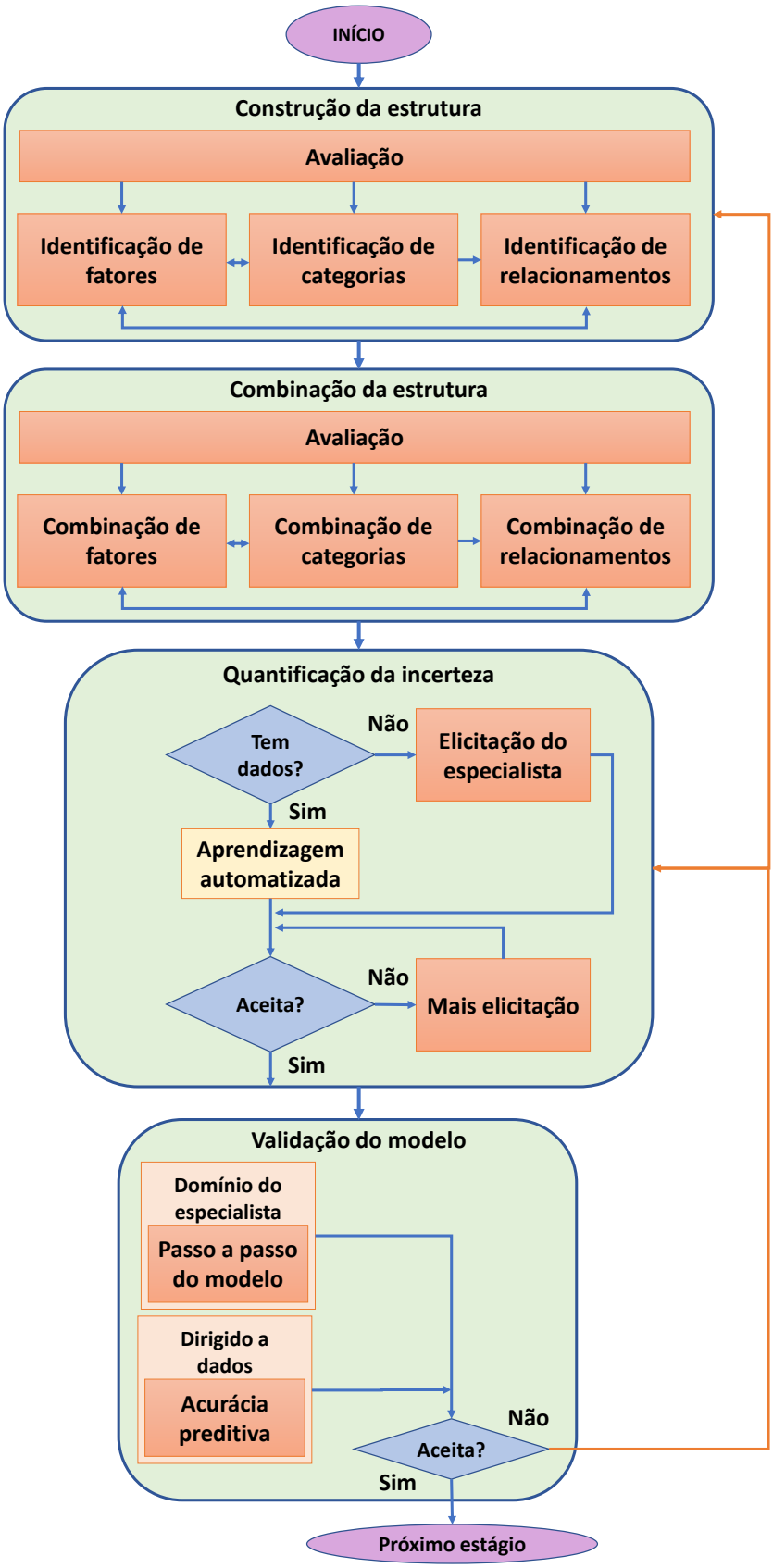


Figura 2.10: Engenharia de Conhecimento de rede Bayesiana baseada em Especialistas proposto por Mendes [86].

Capítulo 3

Trabalhos Relacionados

Os trabalhos relacionados à presente pesquisa são vários, uma vez que cada etapa do método proposto aborda uma temática diferente no contexto de medição de atributos de software, como: seleção de medidas, validação de medidas, validação de dados e definição de *thresholds*. Dado que não foi encontrado na literatura trabalhos que tenham foco na interpretação de medidas, o presente capítulo contempla trabalhos relacionados às temáticas descritas, bem como à temática referente à programas de medição ou, mais especificamente, sistemas de medição, uma vez que espera-se que tais trabalhos abordem mais de uma das etapas contempladas no método proposto.

Vale ressaltar que, embora a ISO/IEC 15939:2007 [123] não discuta sobre o conceito de programa de medição, Staron e Meding, em [127], afirmam que um programa de medição é um conjunto sociotécnico de sistemas de medição e seus usuários. Sistema de medição, por sua vez, consiste em um sistema técnico que coleta dados, executa cálculos e apresenta o resultado aos *stakeholders* [127].

Em relação à temática "seleção de medidas", destaca-se o trabalho de Bukhari, Yahaya e Deraman [17], na qual os autores discutem os pontos fortes e fracos de seis métodos de seleção de medidas de software. Huda *et al.*, em [57], abordam esta temática em um contexto mais específico, voltado para a predição de defeitos de software. Eles utilizam uma abordagem híbrida baseada em filtros e em uma combinação de *Support Vector Machines* (SVM) e *Artificial Neural Network* (ANN). Neste contexto, a validade das medidas é discutida dentro da etapa de seleção. Farshchi *et al.*, em [34], também abordam a temática em um contexto específico, voltado para sistema de computação na nuvem. Neste trabalho, os autores apre-

sentam uma abordagem de seleção de medidas baseada em análise de regressão. Os autores também consideram a validação das medidas dentro da etapa de seleção.

Em relação à temática "validade de medidas", destaca-se o trabalho de Schneidewind [118], no qual uma metodologia de validação de medidas, baseada em seis critérios de validade, é apresentada. Outros tabalhos que também merecem destaque nesta temática são: o trabalho de Meneely, Smith e Williams [88], no qual é apresentada uma revisão sistemática acerca dos critérios de validação encontrados na literatura acadêmica; e o trabalho de Antinyan *et al.* [6], que baseia-se no sistemático processo de pesquisa-ação típico de definir, refinar e redefinir o processo de validação com os profissionais, o que permite moldar a medida pretendida e aceitá-la ou rejeitá-la para aplicação posterior.

Em relação à temática "validade de dados", o único trabalho encontrado que apresenta uma discussão mais profunda sobre o assunto é o de Perkusich *et al.*, em [102]. Nesta pesquisa, os autores apresentam uma abordagem para auxiliar a interpretação de medidas de software com base em redes Bayesianas.

Em relação à temática "definição de *thresholds*", destaca-se o trabalho de Alves, Ypma e Visser [3], no qual é apresentada uma abordagem para derivar *thresholds* de medidas baseada em dados de medição, obtidos a partir de um conjunto representativo de sistemas orientados a objeto. Shatnawi *et al.*, em [121], utilizaram curvas *Receiver Operating Characteristic* (ROC) para identificar *thresholds* para algumas medidas de Orientação a Objeto (OO). Neste trabalho, os autores selecionam as medidas a partir de um conjunto pré-definido, e um dos critérios para a seleção é justamente a medida ser válida. Por fim, Sánchez *et al.* [113] focaram na realização de uma comparação entre a técnica de curvas ROC e o método Bender para definição de *thresholds*, concluindo que curvas ROC obtêm *thresholds* mais acurados. Neste trabalho, os autores afirmaram utilizar medidas já validadas empiricamente.

Considerando a temática programa de medição e sistemas de medição, tem-se o trabalho de Zhang *et al.* [151], na qual é proposto um método baseado em *Fuzzy* para avaliar a confiabilidade dos processos de software. Uma das partes do método corresponde a um sistema de medição de confiabilidade, no qual as medidas e seus pesos são obtidos a partir do método AHP (*Analytic Hierarchy Process*). Não há menção sobre a validade das medidas definidas, nem sobre a validade dos dados obtidos. Em relação à definição de *thresholds*, supõe-se que os autores utilizaram lógica *Fuzzy*. Também não há menção sobre a validação/avaliação do

referido sistema.

Em [52], os autores discutem sobre um processo específico para construir modelos para análise de qualidade de software por meio de um processo automático de definição de métricas baseado em GQM. Neste trabalho, não há menção sobre a validade das medidas, a validade dos dados e a definição de *thresholds*. Os autores indicam o uso de ontologias e sistemas multi-agentes, mas a abordagem proposta não foi avaliada.

Em [109], os autores apresentam um refinamento e extensão do método para definição de medidas válidas (*SMDM - Software Measure Definition Method*). Neste trabalho, GQM é utilizado para a seleção de medidas. Para validá-las, são consideradas etapas de validação teórica e empírica. Não há menção sobre as demais etapas. Os autores apresentam um exemplo da aplicação do método proposto para diagramas UML.

Staron*et al.*, em [128], apresentam um processo industrial, também com base na ISO/IEC 15939:2007, para o desenvolvimento de sistemas de medição, incluindo artefatos e produtos importantes para uma implantação bem-sucedida de sistemas de medição na indústria. Os autores utilizam um processo em V contendo etapas para identificação de indicadores, medidas derivadas e medidas base. A elicitação do conjunto de medidas é feita a partir do paradigma GQM. O processo apresentado também inclui etapas de validação de medidas, considerando a validação teórica e empírica. Em relação à etapa de validade de dados, os autores apenas citam que as fontes de informações devem ser confiáveis, utilizando-se de instrumentos precisos, atualizados e sem viés. Em relação à definição de *thresholds*, apesar dos autores citarem esta atividade, eles não apresentam instruções sobre como executá-la. Por fim, não é especificado nenhum modelo de análise, apenas citam a utilização do Microsoft Excel. O processo foi aplicado por 3 anos na Ericsson.

Em [78], Manzano *et al.* propõem uma abordagem, baseada em redes Bayesianas, para estimar indicadores estratégicos relevantes para uma tomada de decisão significativa. Este trabalho foi realizado no contexto do projeto Q-Rapids *(Quality-aware Rapid Software Development)*[1]. No Q-Rapids, as medidas são definidas a partir de dados extraídos de várias ferramentas diferentes. Estas medidas são então agrupadas em fatores de produto e de processo. Estes, por sua vez, são agrupados em indicadores estratégicos. Não há menção sobre a validação dessas medidas. A validade de dados é discutida de forma superficial, a par-

[1] https://www.q-rapids.eu/

tir da metodologia EKEBN *(Expert-based Knowledge Engineering of Bayesian Networks)* e do *Outcome Adequacy*. Por fim, a definição de *thresholds* é atribuída ao especialista do domínio.

Uma comparação dos trabalhos descritos pode ser visualizada na Tabela 3.1, sendo o último trabalho correspondente a presente pesquisa. Além de nenhum dos trabalhos descritos ter como foco a interpretação de medidas, nenhum deles discute todas as etapas que têm influência nesta interpretação. Sendo assim, fica claro que o presente trabalho de tese consiste em uma importante contribuição para a literatura acedêmica, uma vez que possibilita um melhor entendimento acerca da utilização e interpretação de medidas e, consequentemente, amplia a confiança do usuário de medição quanto à tomada de decisão.

Tabela 3.1: Comparação entre os trabalhos relacionados e o método proposto

Artigo	**Seleção**	**Validação de medidas**	**Validação de dados**	**Definição de thresholds**	**Técnicas de modelagem**	**Avaliação/ Validação do sistema**
[17]	Sim	Não	Não	Não	—-	—-
[57]	Sim	Sim	Não	Não	Filter, SVM e ANN	Sim
[34]	Sim	Sim	Não	Não	Análise de regressão	Sim
[118]	Não	Sim	Não	Não	Método não paramétrico	Não
[88]	Não	Sim	Não	Não	——	——
[6]	Não	Sim	Não	Não	——	Sim
[102]	Não	Não	Sim	Não	Redes Bayesianas	Sim
[3]	Não	Não	Não	Sim	Não	Sim
[121]	Não	Não	Não	Sim	Curvas ROC	Sim
[113]	Não	Não	Não	Sim	Curvas ROC Método Bender	Sim
[151]	Sim	Não	Não	Sim	AHP e Fuzzy	Não

[52]	Sim	Não	Não	Não	Ontologia e sistemas multi-agentes	Não
[109]	Sim	Sim	Não	Não	Não	Sim
[128]	Sim	Sim	Não	Sim	Não	Sim
[78]	Não	Não	Sim	Sim	Redes Bayesianas	Sim
Saraiva	Sim	Sim	Sim	Sim	Redes Bayesianas	Sim

Capítulo 4

Método Proposto baseado em Redes Bayesianas

Dado que o objetivo do presente trabalho de tese é a construção de um método que possa auxiliar usuários de medição quanto à tomada de decisão, neste capítulo apresenta-se, de forma integrada, as quatro etapas que compõem o método e que têm impacto sobre a interpretação de medidas: seleção de medidas, validação de medidas, validação de dados e definição de *thresholds*. Uma ilustração do método proposto é apresentada na Figura 4.1.

As etapas descritas foram elencadas com base em tarefas e subtarefas presentes na atividade de Planejamento de Processo de Medição da ISO/IEC 15939:2007. Além disso, é importante destacar os papéis desempenhados pelos indivíduos que estão associados a execução do método proposto, são eles:

- Patrocinador da medição: indivíduo responsável por autorizar e suportar o estabelecimento de um processo de medição;
- Analista de medição: indivíduo responsável por planejar, desempenhar, avaliar e melhorar o processo de medição;
- Guru de medição (ou *Champion*): indivíduo especialista em medição, geralmente externo à organização;
- Usuário de medição: indivíduo que faz uso do produto de informação, geralmente é o gerente e/ou líder técnico do projeto;

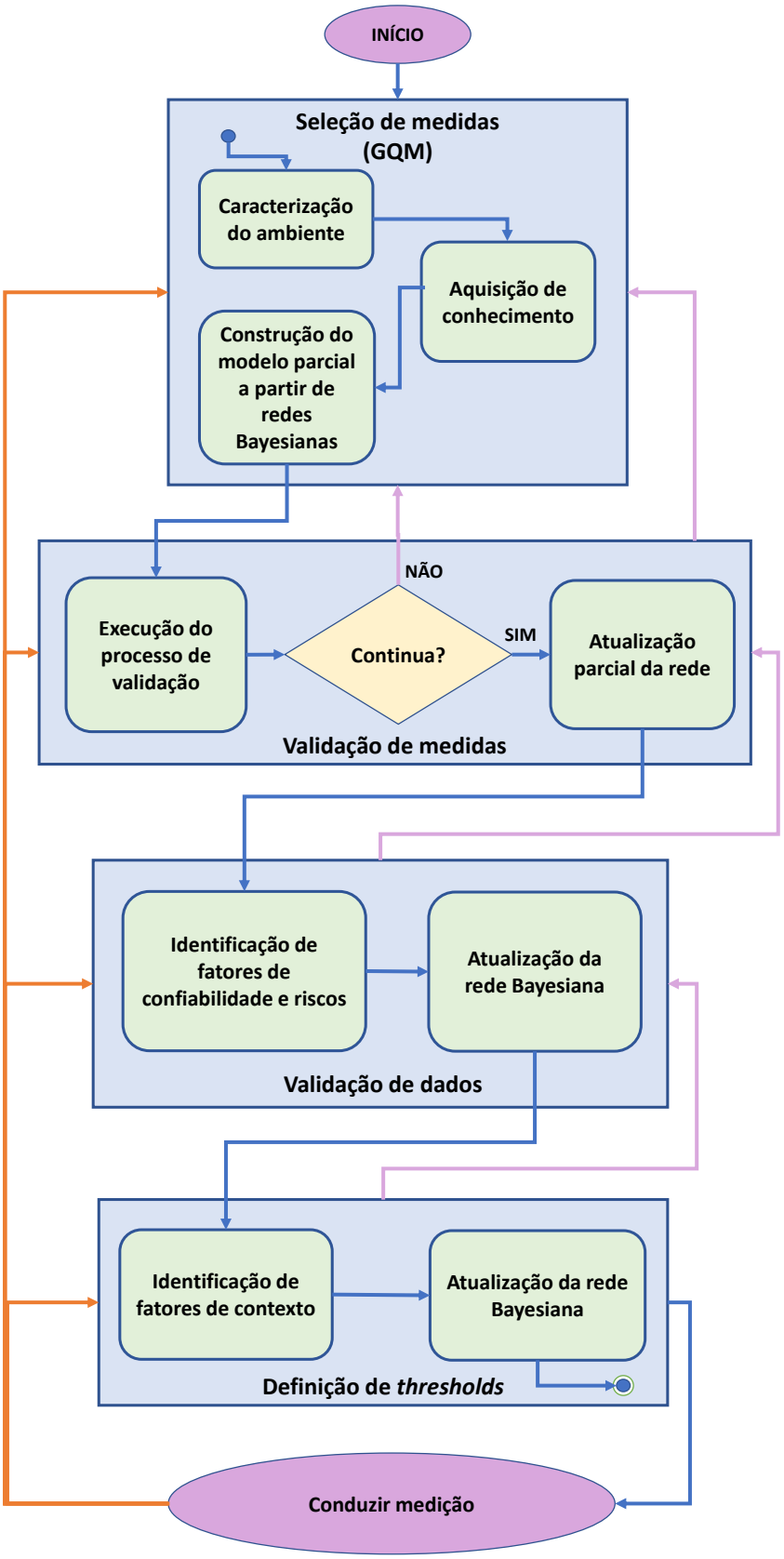


Figura 4.1: Visão geral do método

- Especialista: indivíduo que possui habilidades ou conhecimentos especiais ou excepcionais em determinada atividade. Assim, ao longo deste trabalho de tese, tal termo pode se referir ao detentor de maior conhecimento a cerca de medição, do projeto de software em si, ou de redes Bayesianas.

É importante destacar também que dadas as vantagens de redes Bayesianas, apresentadas na Seção 2.3, elas são utilizadas para construção do referido método considerando a sequência das etapas descritas e sumarizadas a seguir:

- **Seleção de medidas**: consiste na seleção de medidas adequadas para representar os atributos de uma entidade de interesse;
- **Validação de medidas**: consiste na verificação da validade das medidas selecionadas, ou seja, verifica-se se elas medem o que se propõem a medir;
- **Validação de dados**: consiste na definição do grau de confiança que se tem na acurácia dos valores atribuídos às medidas;
- **Definição de *thresholds***: consiste na definição de valores que auxiliam na classificação das medidas.

A entrada para a etapa de seleção de medidas consiste na identificação de um fator externo ao programa de medição, mas essencial ao sucesso dele, chamado de proposição de valor [107] ou propósito [95]. O patrocinador da medição precisa ter em mente o valor que a aplicação do metódo de medição trará à organização. Exemplos de proposição de valor são: monitoramento de desempenho e saúde organizacional [95]. Para o exemplo construído ao longo das seções subsequentes, supõe-se que a proposição de valor é o monitoramento de desempenho.

4.1 Seleção de Medidas

A etapa de seleção de medidas é realizada com base em GQM. A partir de GQM é possível discutir medição de forma estruturada. Desta forma, o analista de medição deve realizar entrevistas estruturadas com usuários de medição (*e.g.,* líderes de projetos) a fim de se

identificar o contexto dos projetos e caracterizar o ambiente (*e.g.,* tipo de projeto, modelo de processo de desenvolvimento, problemas existentes, etc). Em um segundo momento, tais usuários são questionados pelo analista de medição sobre o objetivo e as questões, bem como sobre medidas associadas com o foco de qualidade declarado no objetivo. Uma forma de adquirir este conhecimento é a partir da utilização de folhas de abstração, tal como ilustrado na Figura 4.2.

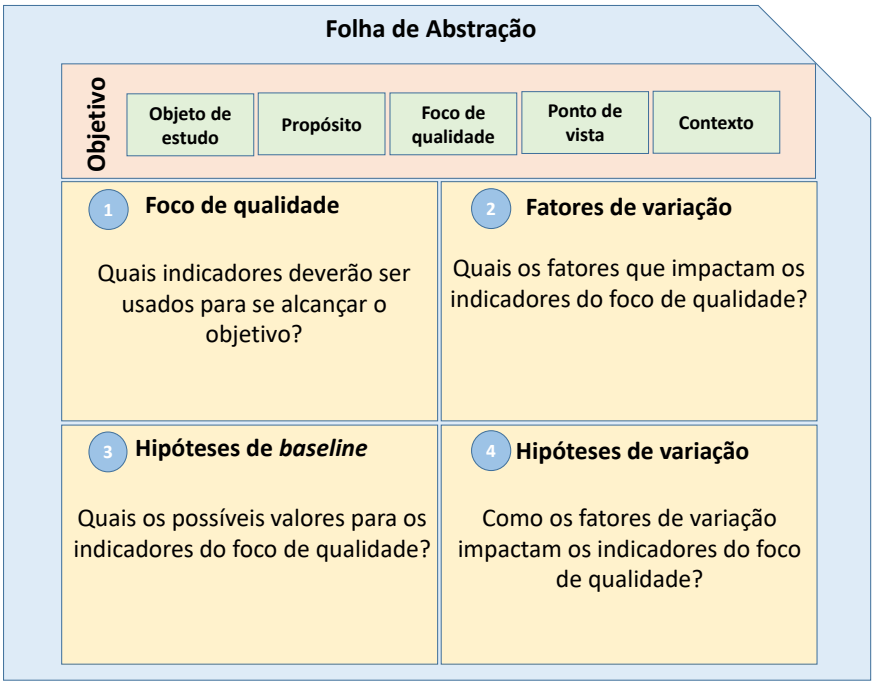


Figura 4.2: Modelo de folha de abstração.

Considerando o uso de folhas de abstração, a aquisição do conhecimento se dá da seguinte forma: para cada objetivo deve haver uma folha de abstração. Cada objetivo deve considerar o objeto de estudo (*e.g.,* produto de software), o propósito da medição (*e.g.,* caracterizar), o foco de qualidade (*e.g.,* qualidade externa), o ponto de vista (*e.g.,* desenvolvedor) e o contexto (*e.g.,* projeto A). Assim, tem-se um exemplo de objetivo: "**Analisar** o **produto de software** com respeito a sua **qualidade externa** para o propósito de **caracterização** a partir do ponto de vista do **desenvolvedor**, no contexto do **projeto A**".

O primeiro quadrante da folha de abstração, correspondente ao foco de qualidade, captura a intuição do usuário de medição sobre o foco da qualidade definido no objetivo. Neste

momento, deve-se fazer a seguinte pergunta ao referido usuário: “Quais indicadores deverão ser usados para se alcançar o objetivo?”. A partir desses indicadores, um conjunto de questões e um conjunto de medidas devem ser extraídos.

O segundo quadrante da folha de abstração, correspondente aos fatores de variação, captura a intuição do usuário de medição sobre os fatores que impactam os indicadores definidos no quadrante anterior. Neste momento, deve-se fazer a seguinte pergunta ao referido usuário: “Quais os fatores que impactam os indicadores do foco de qualidade?”. A partir destes fatores, devem ser extraídos um novo conjunto de questões e um novo conjunto de medidas, agora relacionadas aos fatores de variação.

O terceiro quadrante, correspondente às hipóteses de *baseline*, captura a expectativa do usuário de medição a respeito das medidas extraídas. Neste momento, deve-se fazer a seguinte pergunta ao referido usuário: “Quais os possíveis valores para os indicadores do foco de qualidade?”. A partir desses valores é possível demonstrar a utilidade do processo de medição, uma vez que se pode observar as discrepâncias entre expectativas e realidade, sendo esta obtida a partir da correta coleta e análise de dados.

Por fim, o último quadrante da folha de abstração, correspondente às hipóteses de variação, captura o impacto dos fatores de variação, mais precisamente das medidas extraídas no quadrante 2, sobre os indicadores do foco de qualidade, mais precisamente sobre as medidas extraídas a partir do quadrante 1. Neste momento, deve-se fazer a seguinte pergunta ao referido usuário: “Como os fatores de variação impactam os indicadores do foco de qualidade?”.

Vale ressaltar que a descrição deste impacto serve como motivação para a inclusão do fator de variação na folha de abstração e, consequentemente, no plano GQM. Se o usuário de medição não sabe informar o impacto de um fator de variação, este fator deve ser excluído da folha.

Uma instância de folha de abstração referente ao objetivo supracitado pode ser visualizada na Figura 4.3.

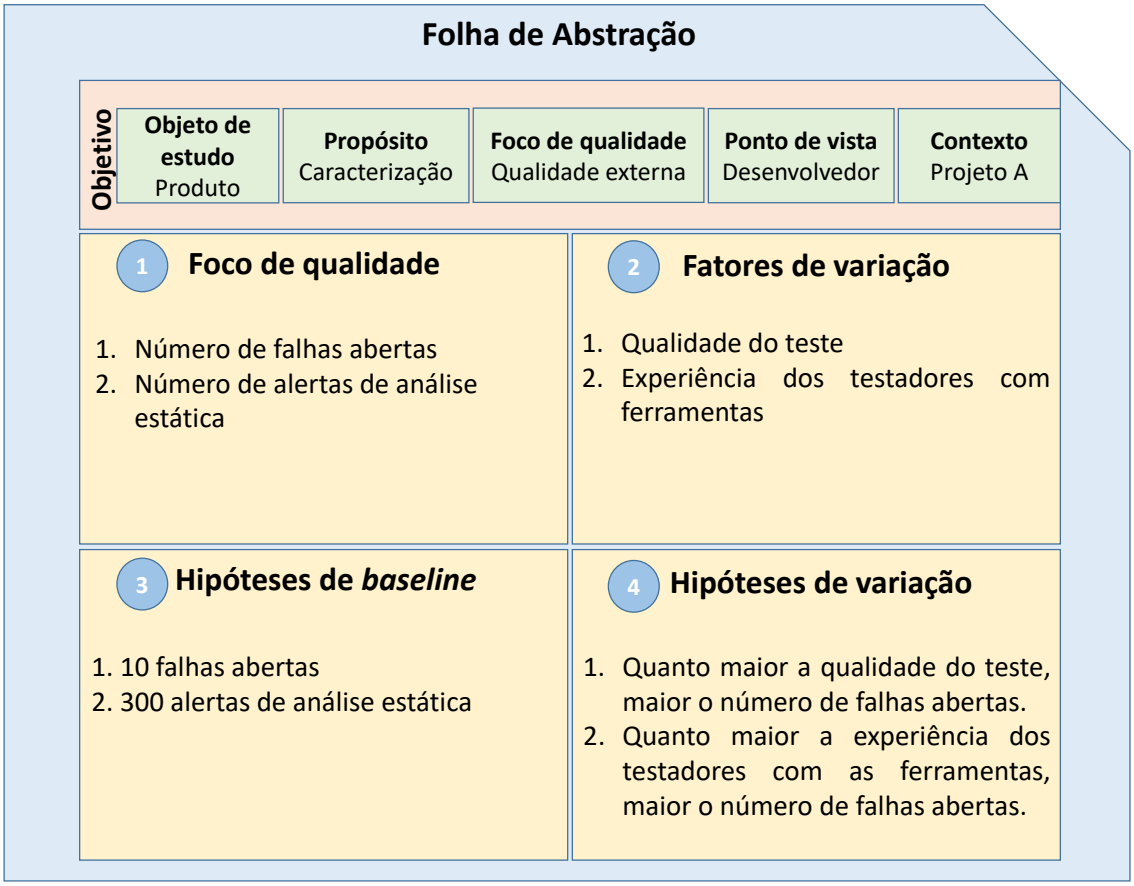


Figura 4.3: Instância de folha de abstração.

Supõe-se que, a partir do primeiro quadrante desta folha de abstração, que corresponde ao foco de qualidade, o analista de medição extraiu a questão *"Quantos comportamentos indesejados o produto tem?"* e as medidas *número de falhas abertas* e *número de alertas de análise estática*. A partir do segundo quadrante, que corresponde aos fatores de variação, ele extraiu a questão *"Qual é a qualidade do teste?"* e as medidas *cobertura de código* e *experiência do testador*.

Uma vez que cada questão extraída deve estar associada no mínimo a um objetivo, um conjunto de relacionamentos entre objetivos e questões deve ser criado, assim como um conjunto de relacionamentos entre questões e medidas. A estrutura hierárquica do modelo GQM referente ao exemplo descrito é ilustrada na Figura 4.4.

Como hipóteses de *baseline*, o analista de medição registrou que o desenvolvedor espera ter 10 falhas abertas e 300 alertas de análise estática. Como hipóteses de variação, registrou

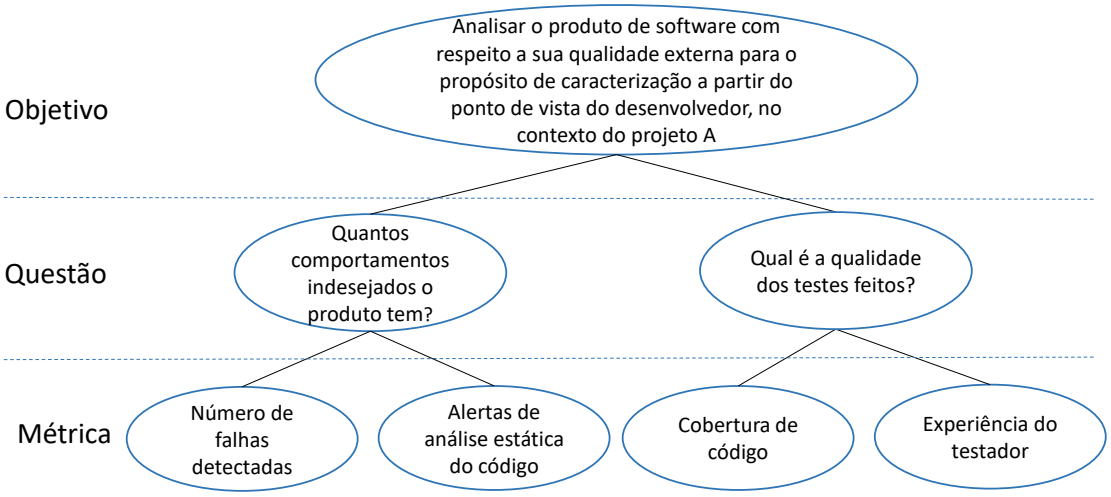


Figura 4.4: Estrutura hierárquica do modelo GQM

que, para o desenvolvedor, quanto maior a qualidade dos testes e quanto maior a experiência dos testadores com as ferramentas, maior o número de falhas abertas.

Vale ressaltar que as questões e medidas extraídas a partir dos fatores de variação serão utilizadas apenas na etapa de validação de dados, uma vez que tais informações tem forte relação com a acurácia das medidas.

Uma vez que o objetivo, questões e medidas referentes ao foco de qualidade são definidos, a rede Bayesiana pode ser construída, conforme Figura 4.5. Dado que se deseja representar conhecimentos subjetivos, nós ranqueados são utilizados. No caso de medidas quantitativas, estas são abordadas após a definição dos *thresholds*.

Após a construção da rede, deve-se calibrar o nó referente ao objetivo. Logo, deve-se construir uma tabela verdade, baseada em informações providas pelo(s) especialista(s) (*e.g.,* analista de medição e/ou usuário de medição), a fim de se identificar qual função de probabilidade será associada ao nó em questão e quais os pesos dos nós pais.

Para o exemplo dado, uma vez que há uma relação de 1 para 1 entre objetivo e questão, a tabela de probabilidade de nó (TPN) deve ser calibrada como uma matriz identidade, caso não haja incerteza associada. Se houver, a TPN deve ser calibrada conforme análise do especialista. A entrada e saída da etapa de seleção de medidas podem ser visualizadas na Figura 4.6.

Figura 4.5: Rede Bayesiana considerando objetivo, questões e medidas do foco de qualidade

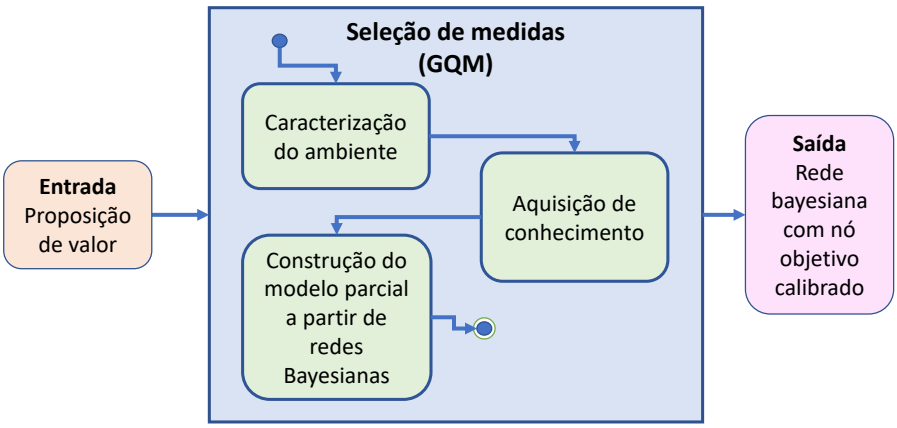


Figura 4.6: Entrada e saída da etapa de seleção de medidas

4.2 Validação de Medidas

Uma vez que as medidas são selecionadas e modeladas na rede Bayesiana, deve-se validá-las a fim de garantir que elas sejam representativas dos atributos medidos. Nesta pesquisa, a etapa de validação de medidas tem como base os critérios de validação ilustrados na Tabela 2.1. Desta forma, para cada medida referente ao foco de qualidade será utilizado o

processo de aplicação de critérios descrito abaixo [88]:

1. Determinar o objetivo de utilização da medida. Tal objetivo varia de acordo com contextos práticos ou científicos específicos.

2. Destacar as vantagens, ilustradas na Tabela 2.2, que são apropriadas para o objetivo de utilização da medida. O usuário de medição pode decidir sobre as vantagens que ele deseja apresentar ao time de desenvolvimento.

3. Observar os critérios de validação que estão vinculados às vantagens a partir da Tabela 2.3.

4. Escolher cuidadosamente os critérios de validação, considerando o objetivo de utilização da medida, relacionamentos e motivações entre os critérios aplicáveis. Vale destacar que um critério deve ser escolhido somente pela adequação dele ao objetivo de utilização da medida.

5. Demonstrar que uma medida segue ou não o critério de validade escolhido.

Ao aplicar este processo considerando o exemplo definido na etapa anterior, tem-se que o objetivo de utilização das medidas *número de falhas abertas* e *número de alertas de análise estática* é identificar a qualidade do software.

Dado isto, supõe-se o seguinte cenário. Para a medida *número de falhas abertas*, considerando o objetivo de utilização desta medida, o usuário de medição, juntamente com um guru de medição [53], destacou a vantagem de *Quality-focused*, que indica que uma medida é útil para melhorar a vida de pessoas a partir de um software de alta qualidade [88]. Seguindo o processo, observa-se a partir da Tabela 2.3 os critérios de validade que estão associados à vantagem *Quality-focused*. Para facilitar a visualização e entendimento, é apresentado na Figura 4.7 o mapeamento entre os critérios de qualidade associados à referida vantagem.

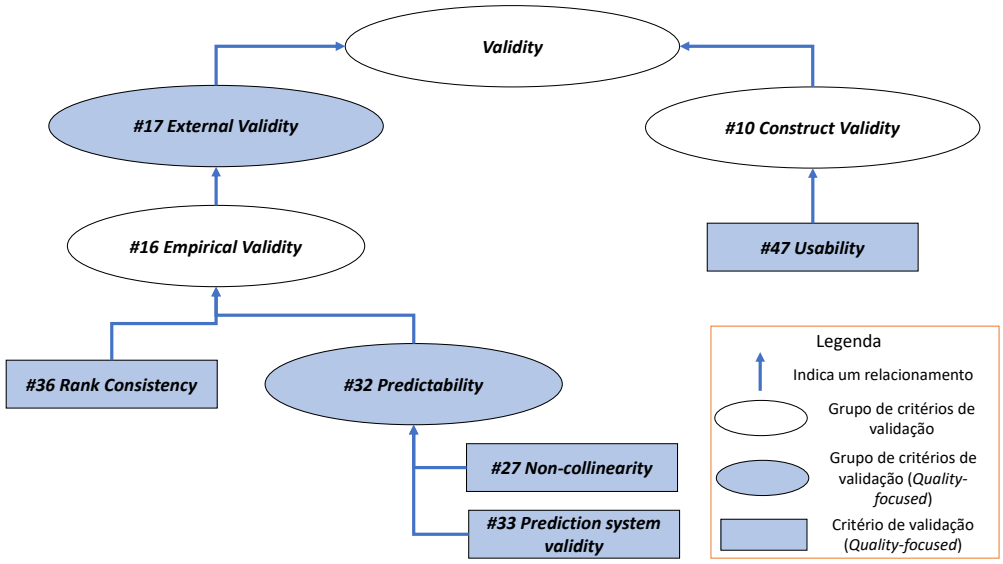


Figura 4.7: Mapeamento entre critérios de validação associados à vantagem *Quality-focused* [88]

Pode-se observar na Figura 4.7 que as elipses representam grupos de critérios, enquanto os retângulos representam critérios específicos. Além disso, tanto as elipses como os retângulos que estão em azul representam os critérios de validade associados à vantagem *Quality-focused*.

Supõe-se que o usuário e o guru de medição analisam tais critérios, considerando o objetivo de utilização da medida, o tipo de validação e suas motivações filosóficas, e optam por escolher apenas o critério de validade *Usability*. Uma medida tem *Usability* se ela pode ser implementada de maneira econômica em um programa de garantia de qualidade. Deve ser viável coletá-la dentro de um processo. Por exemplo, uma medida que requer vários meses de computação pode não ser considerada utilizável.

Para finalizar o processo, o usuário e guru de medição demonstram que a medida *número de falhas abertas* pode ser aplicada ao processo de desenvolvimento de software, uma vez que ela pode ser coletada por meio de ferramentas já utilizadas no contexto do projeto em questão.

Espera-se que ao aplicar o processo descrito para a outra medida em questão, *número de alertas de análise estática*, o usuário de medição tenha ganho mais confiança sobre as medidas selecionadas para identificar a qualidade do software. A partir disto, essa confiança pode ser modelada com redes Bayesianas, a partir do conceito de variância do nó.

Para o exemplo ilustrado na Figura 4.5, a variância do nó referente à questão "Quantos comportamentos indesejados o produto tem?" será definida com base na validação das medidas *número de falhas abertas* e *número de alertas de análise estática*. Quanto maior a confiança na validade da medida, menor a variância de seu nó filho.

Para se definir esta variância, o usuário e guru de medição podem definir o valor direto na rede, baseando-se no conhecimento prévio e adquirido ao longo da aplicação do processo de validação descrito, ou podem utilizar tal conhecimento na elaboração de regras do tipo:

Regra 1: Se 100% dos critérios de validação escolhidos foram demonstrados, a variância deve ser igual a $5^{10^{-4}}$;

Regra 2: Se 50% a 99% dos critérios de validação escolhidos foram demonstrados, a variância deve ser igual a $5^{10^{-3}}$;

Regra 3: Se 1% a 49% dos critérios de validação escolhidos foram demonstrados, a variância deve ser igual a $5^{10^{-2}}$;

Regra 4: Se não for possível demonstrar qualquer critério de validação escolhido, deve-se analisar a possibilidade de retornar à etapa de seleção de medidas ou manter a medida com uma variância igual a $5^{10^{-1}}$;

Vale ressaltar que, para fins de simulação do cenário apresentado, o intervalo de valores referente à variância [$5^{10^{-4}}$, $5^{10^{-1}}$] foi escolhido com base na ferramenta AgenaRisk[1], utilizada para modelar redes Bayesianas.

Vale ressaltar também que, antes de se executar o modelo do exemplo em questão, a função de probabilidade do nó referente à questão deve ser definida. Para isto, sendo este nó representado por Z, os nós referentes às medidas X e Y, onde X corresponde à medida *número de falhas abertas* e Y corresponde à medida *número de alertas de análise estática*, e a tabela verdade representada pela Tabela 4.1, a função de probabilidade mais adequada seria a *Minimum*.

[1] www.agenarisk.com

X	Y	Z
Muito bom	Muito bom	Muito bom
Muito ruim	Muito ruim	Muito ruim
Muito bom	Muito ruim	Ruim
Muito ruim	Muito bom	Muito Ruim

Tabela 4.1: Exemplo de tabela verdade

Na Figura 4.8 é ilustrado o resultado da rede Bayesiana calibrada com a função *Minimum* e variância $5^{10^{-3}}$. Os nós *número de falhas abertas* e *número de alertas de análise estática* foram calibrados, respectivamente, com pesos 2 e 1, uma vez que *número de falhas abertas* é mais importante para a qualidade do software que *número de alertas de análise estática*. Nota-se que tais valores correspondem a uma simulação de decisões feitas pelo usuário e guru de medição.

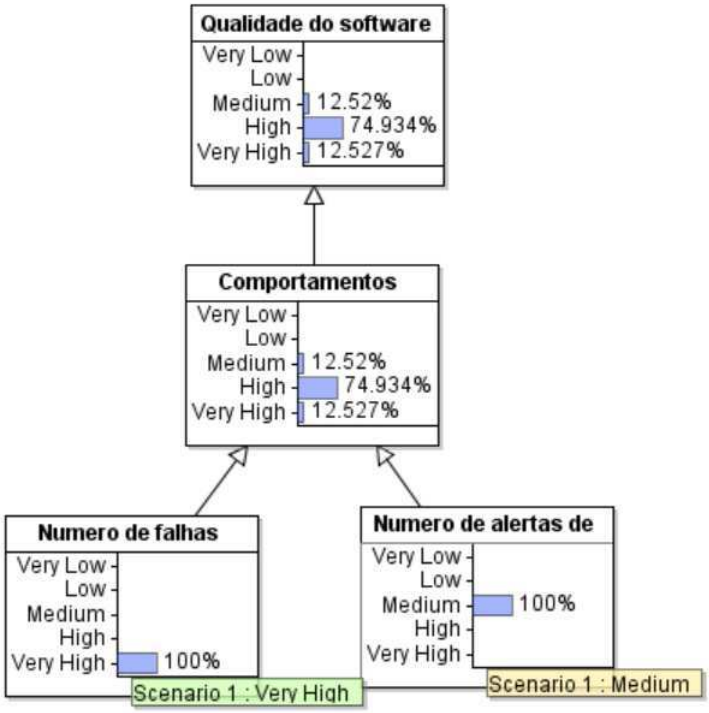


Figura 4.8: Rede Bayesiana calibrada com função de probabilidade e variância

Por fim, vale salientar que o papel do guru de medição [53] é fortemente recomendado

nesta etapa, dada a complexidade do processo de validação. Salienta-se também a importância de ciclos de *feedback* com o propósito de refinar as medidas [6].

A entrada e saída da etapa de validação de medidas são ilustradas na Figura 4.9.

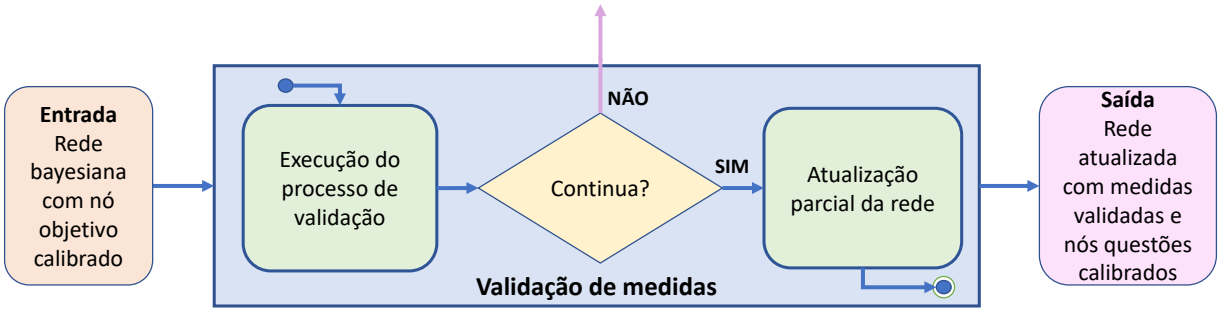


Figura 4.9: Entrada e saída da etapa de validação de medidas

4.3 Validação de Dados

A validação de dados consiste em se definir o grau de confiança que se tem na acurácia dos valores atribuídos às medidas. Para isto, deve-se identificar os fatores de confiabilidade. Estes, por sua vez, capturam a intuição do especialista quanto aos fatores que afetam a confiança do usuário de medição em relação à acurácia de uma medida.

Tendo como base o exemplo simulado ao longo deste capítulo, cujas questões e medidas foram extraídas a partir da folha de abstração ilustrada na Figura 4.3, tem-se que: as medidas *número de falhas abertas* e *número de alertas de análise estática* correspondem ao foco de qualidade, e as medidas *cobertura de código* e *experiência do testador* correspondem aos fatores de variação. Com isto em mente, supõe-se o seguinte cenário: se o *número de falhas abertas* e o *número de alertas de análise estática* são próximos ou iguais a zero e a *cobertura de código* é alta, isto aumenta a confiança na acurácia das medidas e, consequentemente, aumenta a confiança de que o software se comporta como o esperado. Se a *experiência do testador* é alta, a confiança nas medidas e consequentemente na qualidade do produto aumenta ainda mais.

Considerando este cenário, um outro fator de variação que poderia ter sido levantado é a *experiência dos desenvolvedores*, uma vez que quanto maior a *experiência dos desenvolvedores*, espera-se que menores sejam o *número de falhas abertas* e o *número de alertas de*

análise estática. Neste caso, o fator de variação tem uma influência sobre o que se espera da medida, mas não sobre a confiança no valor atribuído a ela. Logo, tem-se que o *número de falhas abertas* e *número de alertas de análise estática*, além de serem fatores de variação, são fatores de confiabilidade, diferentemente da medida *experiência dos desenvolvedores*. Assim, considerar os fatores de confiabilidade aumenta a probabilidade do analista e usuários de medição em tomar decisões mais assertivas, uma vez que eles terão mais confiança sobre a acurácia das medidas coletadas.

Para possibilitar uma maior confiança na tomada de decisão, além de considerar fatores de confiabilidade, é necessário considerar os riscos inerentes ao processo de medição (*e.g.,* erros humanos e limitações de ferramentas). Sendo assim, um ponto de partida para validação de dados é identificar os fatores de confiabilidade com base na análise dos fatores de variação, extraídos a partir do segundo quadrante da folha de abstração. Em seguida, o analista e usuário de medição devem se perguntar quais outros fatores de confiabilidade e quais riscos podem influenciar na confiança sobre a acurácia dos valores das medidas.

Uma vez que os fatores de confiabilidade e os riscos foram levantados, deve-se atualizar a rede Bayesiana. Para tal, realizou-se uma adaptação do método proposto por Perkusich *et al.* [102], ilustrado na Figura 2.2. Basicamente, as adaptações foram feitas na primeira etapa do método, que passou a utilizar GQM para identificação de objetivos, questões e medidas, e em uma subetapa referente à etapa 2, que corresponde a construção do gráfico acíclico direcionado (DAG). Estas adaptações são ilustradas em amarelo na Figura 4.10 e explanadas ao longo desta subseção.

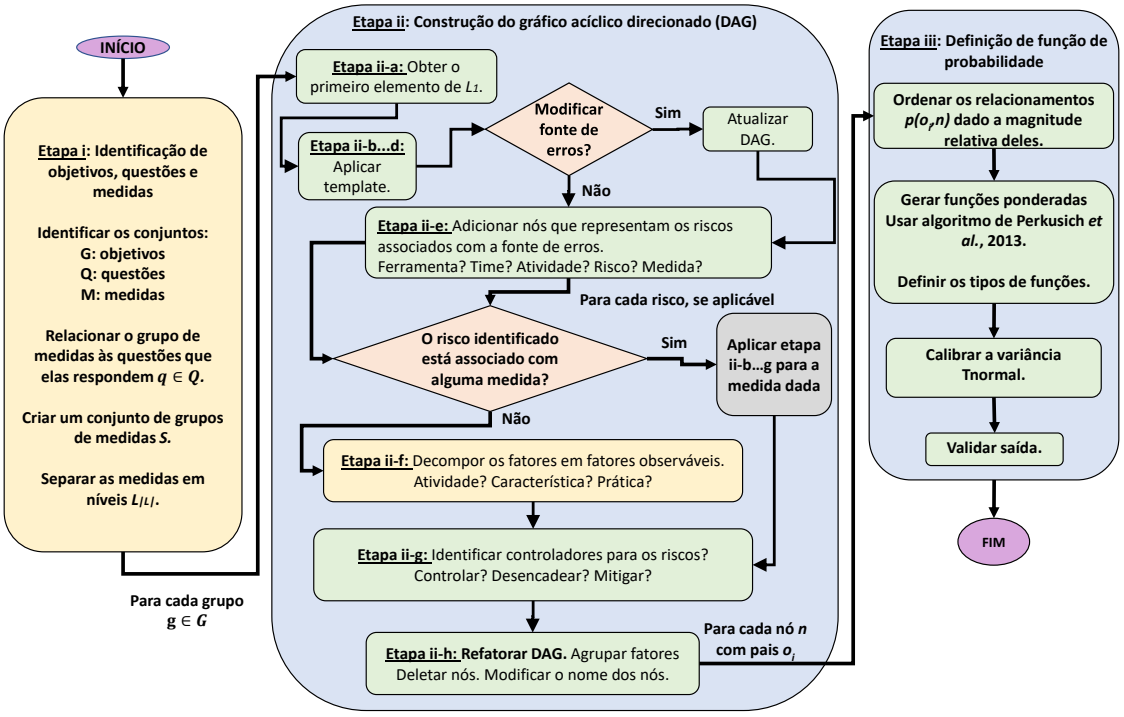


Figura 4.10: Adaptação do método de Perkusich *et al.* [102]

Vale ressaltar que, de acordo com o exemplo simulado ao longo deste capítulo e considerando que o objetivo, questões e medidas referentes ao foco de qualidade já estavam modelados na rede Bayesiana, conforme ilustrado na Figura 4.5, bastaria refinar a rede considerando os fatores de confiabilidade e riscos levantados. Contudo, para um melhor entendimento da adaptação feita no contexto do exemplo simulado, considera-se a execução das etapas (*i*) e (*ii*) do método adaptado (Figura 4.10), descritas a seguir.

Na etapa (*i*) são identificados conjuntos de objetivos, questões e medidas relacionadas a um projeto. O conjunto de objetivos é definido como $G = \{g_1, \ldots, g_{|G|}\}$, onde g_i representa um objetivo do projeto e $|G|$ representa o número de objetivos definidos no projeto. Em seguida, define-se um conjunto de questões $Q = \{q_1, \ldots, q_{|Q|}\}$, onde q_i representa uma questão relacionada a, no mínimo, um objetivo do projeto e $|Q|$ corresponde ao total de questões. Dado que cada questão deve ser associada a no mínimo um objetivo, cria-se um conjunto de relacionamentos entre objetivos e questões T, no qual $t(g, q)$ significa que o objetivo g e a questão q estão relacionados. Finalmente, define-se um conjunto de medidas $M = \{m_1, \ldots, m_{|M|}\}$, onde m_i é uma medida e $|M|$ é o número de medidas usadas no projeto.

As medidas pertencentes a M devem ser agrupadas em relação à questão extraída do foco de qualidade. Cada grupo de medida é representado como um conjunto e a mesma medida pode pertencer a múltiplos conjuntos, pois ela pode ser usada para responder múltiplas questões. Por exemplo, medidas relacionadas aos comportamentos indesejados do produto, tais como: *número de falhas abertas*, *cobertura do código* e *experiência do testador* podem pertencer ao mesmo grupo.

Formalmente, pode-se representar o conjunto de grupos de medidas $S = \{S_1, \ldots, S_{|Q|}\}$, onde S_i representa um grupo de medidas. Cada grupo contém somente medidas que correspondem a uma única questão do foco de qualidade. O relacionamento entre um grupo de medidas e uma questão do foco de qualidade é representado por $r(s, q')$, que significa que as medidas contidas no grupo s estão relacionadas à questão q'. Um conjunto A que contém cada relacionamento entre grupos de medidas e questões deve ser criado. Então, $\forall_{s \subseteq S} \forall_{q' \in Q'} \ (r(s, q') \in A)$. Finalmente, cada grupo $s \subseteq S$ precisa ser dividido em níveis de acordo com a dependência entre medidas e a importância delas no que diz respeito às questões que elas respondem. Por exemplo, o *número de falhas abertas* depende da *cobertura do código*. Espera-se que quanto maior for a *cobertura do código*, maior a confiança no valor referente ao *número de falhas abertas*. Para se fazer esta divisão, um conjunto de níveis $L = \{L_1, \ldots, L_{|L|}\}$ deve ser criado, onde $|L|$ é o número de níveis hierárquicos que o grupo s tem. Para cada nível $L_i \subseteq L$, deve-se garantir que cada medida pertencente ao nível pertença também ao grupo S_i, então $\forall_{L_i \subseteq L} \forall_{m \in L_i} \ (m \in S_i)$.

Na prática, os níveis podem ser definidos usando uma simples regra: se a medida está relacionada ao foco de qualidade, M', coloque-a no primeiro nível; caso contrário, se ela está relacionada a fatores de confiabilidade, M'', coloque-a no segundo nível. Desta forma, com mais análises, é possível decompor s em mais níveis. Para o exemplo dado, uma vez que o *número de falhas abertas* depende da qualidade do teste (*e.g.,* fator de confiabilidade), *número de falhas abertas* $\in L_1$ e *cobertura do código* e *experiência do testador* $\in L_2$. O *número de falhas abertas* pertence ao primeiro nível porque ele é mais importante para a questão "Quantos comportamentos indesejados o produto tem?" do que as outras medidas que estão contidas no grupo.

Além da integração de GQM, que possibilita a identificação de medidas com base no objetivo do usuário de medição, outro ponto adaptado ao método de Perkusich *et al.* [102]

é a etapa de construção do DAG. Então, para cada $s \in S$, um DAG deve ser construído da seguinte forma:

Até que cada s esteja vazio, deve-se:

1. Remover cada elemento $m_i \in s$, a começar pelo nível L_1.

2. Adicionar o relacionamento $r(m_i, q')$, que indica que o nó referente à medida m_i é pai do nó referente à questão q'.

3. Adicionar o relacionamento $r(q', g)$, que indica que o nó referente à questão q' é pai do nó referente ao objetivo g.

4. Criar um nó I_{m_i} que representa o valor real de m_i, ou seja, sua interpretação. A interpretação de uma medida representa o valor dela considerando os fatores que influenciam nesse valor.

5. Criar os nós I_{m_i-C} e I_{m_i-A}, onde I_{m_i-C} representa a medida coletada e I_{m_i-A} representa a acurácia da medida. Para representar as medidas coletadas de forma quantitativa, deve-se utilizar valores de referência (*thresholds*), definidos pelo método descrito posteriormente.

6. Se fizer sentido, deve-se aplicar a expressão idiomática de síntese e decompor a acurácia em: *qualidade do processo de coleta da medida*, representado pelo nó F_pc e *riscos*, representado pelo nó R, sendo estes relacionados a *falhas duplicadas* e *falhas que são requisitos*.

7. Para o nó F_pc, pai de I_{m_i-A}, deve-se criar um conjunto de nós f referentes aos fatores de confiabilidade F, extraídos da folha de abstração, e para cada um desses nós, um relacionamento $r(f, F_pc)$. Neste momento, o líder de projeto deve ser questionado em relação aos fatores que influenciam na acurácia da medida. Por exemplo, a experiência do desenvolvedor é um fator de variação, pois impacta na quantidade de falhas abertas. Quanto maior a experiência, menor o número de falhas que se espera encontrar. No entanto, a experiência do desenvolvedor não influencia no número encontrado, o que influencia é a experiência do testador.

O resultado do DAG relativo ao modelo GQM, ilustrado na Figura 4.4, pode ser visualizado na Figura 4.11. O nó com contorno verde refere-se à interpretação que se dá à medida, considerando riscos e fatores que afetam a confiança que se tem nela. O nó com contorno vermelho refere-se à acurácia da medida, sendo esta dependente dos riscos e da qualidade do processo de coleta. Por fim, os nós com contorno azul referem-se aos fatores de confiabilidade extraídos da folha de abstração, mais especificamente, do quadrante referente aos fatores de variação.

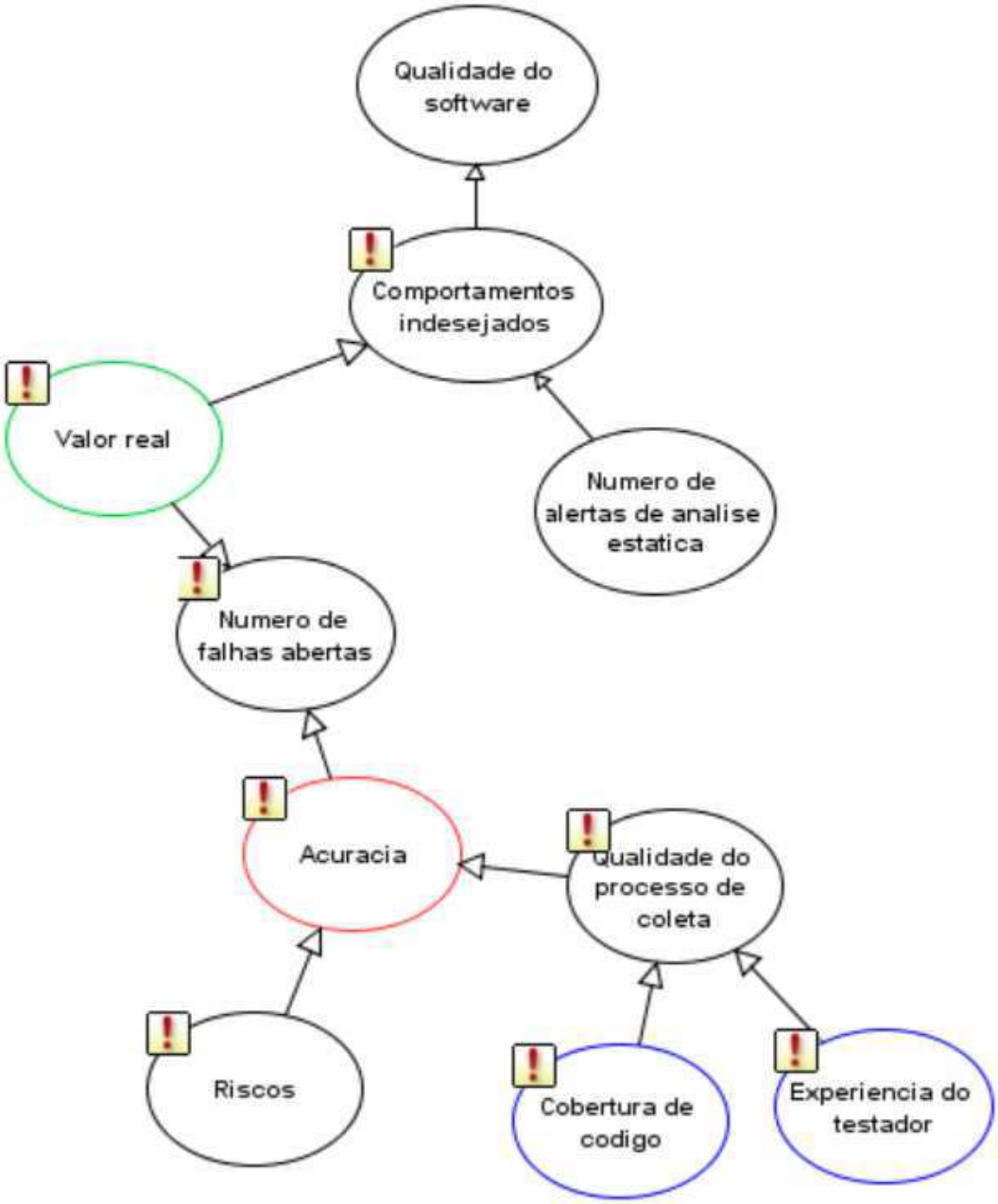


Figura 4.11: Rede Bayesiana com a confiança da medida modelada

Para calibrar as tabelas de probabilidade dos nós, executa-se a etapa *iv* do método de Perkusich *et al.* [102], descrita no Capítulo 2.

A entrada e saída da etapa de validação de dados são ilustradas na Figura 4.12.

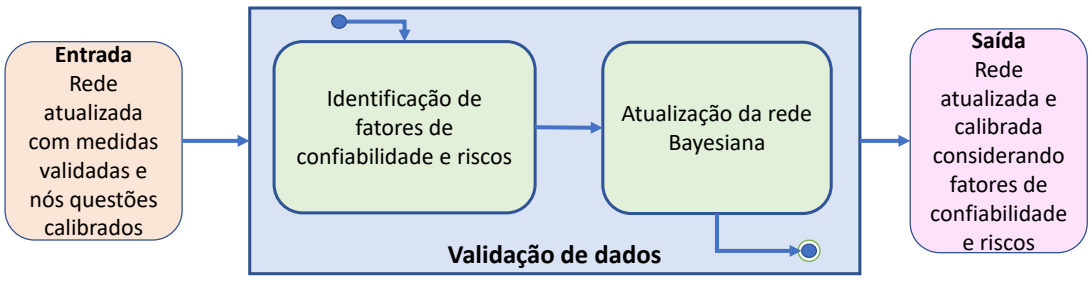


Figura 4.12: Entrada e saída da etapa de validação de dados

4.4 Definição de *thresholds*

Dado que as medidas referentes ao foco de qualidade e aos fatores de confiabilidade foram validadas, assume-se que o tipo de escala dos dados a serem coletados [82] foi definido. Com isto, dá-se seguimento ao método em questão, mais precisamente à etapa de definição de *thresholds*. Para esta etapa, utilizou-se uma abordagem baseada em fatores de contexto [114] e conhecimento de especialistas (*e.g.,* analista de medição).

Fatores de contexto capturam a intuição do usuário de medição sobre os fatores que influenciam os *thresholds* das medidas. Assume-se que as medidas para as quais os fatores devem ser definidos estão representados inicialmente em uma escala númerica, contínua (*e.g., cobertura de código* [148] e *estimativa de esforço*) ou discreta (*e.g., número de defeitos abertos* e *número de alertas de análise estática*). Para estas medidas, tem-se só e somente só dois valores extremos possíveis: o menor (α) e o maior valor (β). Assim, o valor de cada medida está contido no intervalo $[\alpha, \beta]$. Por exemplo, para *número de alertas de análise estática*, tem-se o intervalo $[0, \infty]$; para *cobertura de código*, tem-se $[0, 1]$.

Dado isso, os fatores de contexto podem ser classificados em dois tipos: decrescentes e crescentes. Os primeiros influenciam o valor do *threshold* (δ) a tender para α ($\delta \rightarrow \alpha$), enquanto os fatores crescentes influenciam o valor do *threshold* (δ) a tender para β ($\delta \rightarrow \beta$).

Considerando o exemplo ilustrado ao longo deste capítulo, tem-se que a "criticidade do projeto" constitui um fator de contexto decrescente para a medida *Número de alertas de análise estática*, uma vez que quanto maior a criticidade de um projeto, menor deve ser o valor do *threshold* da medida em questão (*i.e.,* espera-se mais rigor na qualidade do produto).

Por outro lado, tem-se o fator "tempo de mercado" como um fator de contexto crescente, uma vez que quanto mais crítico for atender o tempo de mercado para o projeto, maior o valor do *threshold* da medida em questão. Em específico, essa dicotomia com relação à gestão de qualidade e tempo de mercado é tratada na literatura como parte do problema de gestão de débito técnico [72, 138, 26].

Neste contexto, é importante destacar que, diferentemente dos fatores de variação, os fatores de contexto têm influência sobre a semântica da medida e não sobre o valor dela. Por exemplo, *experiência da equipe* e *maturidade do processo de desenvolvimento* são fatores de variação, uma vez que podem impactar o valor da medida *número de alertas de análise estática*. Quanto menor a *experiência da equipe* e a *maturidade do processo de desenvolvimento*, maior o valor esperado para a medida *número de alertas de análise estática*, considerando que ferramentas de análise estática são executadas no projeto em questão. Apesar disso, esses fatores não influenciam o valor dos *thresholds* referentes à medida *número de alertas de análise estática* e, portanto, não devem ser considerados para a definição dos *thresholds* da referida medida.

Há diversos estudos na literatura que apontam para a relevância de considerar tais fatores de contexto na definição dos *thresholds* de uma medida [45, 150, 98]. Para confirmar tais achados, como parte deste trabalho de tese, foi desenvolvido um estudo em parceria com um aluno de mestrado do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande. Para tal, foi realizado um experimento utilizando métodos estatísticos (*Double sampling* e *Bootstrap*) [45] e transformação logarítmica [120] para avaliar os efeitos de fatores de contexto nos *thresholds* de medidas *Chidamber-Kemerer* [19]. O resultado da pesquisa [114] confirmou resultados de estudos passados [45, 150, 98], evidenciando a influência de fatores de contexto sobre a definição dos *thresholds* de medidas de software.

Para exemplificar a aplicação de fatores de contexto na definição de *thresholds* de medidas de software, considera-se o exemplo simulado, utilizado ao longo deste capítulo. Supondo que tem-se como objetivo interpretar os valores coletados das medidas *Número de alertas de análise estática* e *Cobertura de código*, decidiu-se utilizar uma escala semântica ordinal de três pontos (*i.e., Bom, Razoável, Ruim*) para *Número de alertas de análise estática*. Para *Cobertura de código*, decidiu-se utilizar uma escala semântica do tipo *Boolean*

(*i.e., Ruim, Bom*).

Para *Número de alertas de análise estática*, definiu-se os *thresholds*: ([0 - 10), *Bom*), ([10 - 20), *Razoável*), e ([20 - ∞), *Ruim*). Para *Cobertura de código*, definiu-se os *thresholds*: ([0 - 0,7), *Ruim*) e ([0,7 - 1), *Bom*). Os *thresholds* definidos para ambas as medidas são apresentados visualmente na parte superior da Figura 4.13.

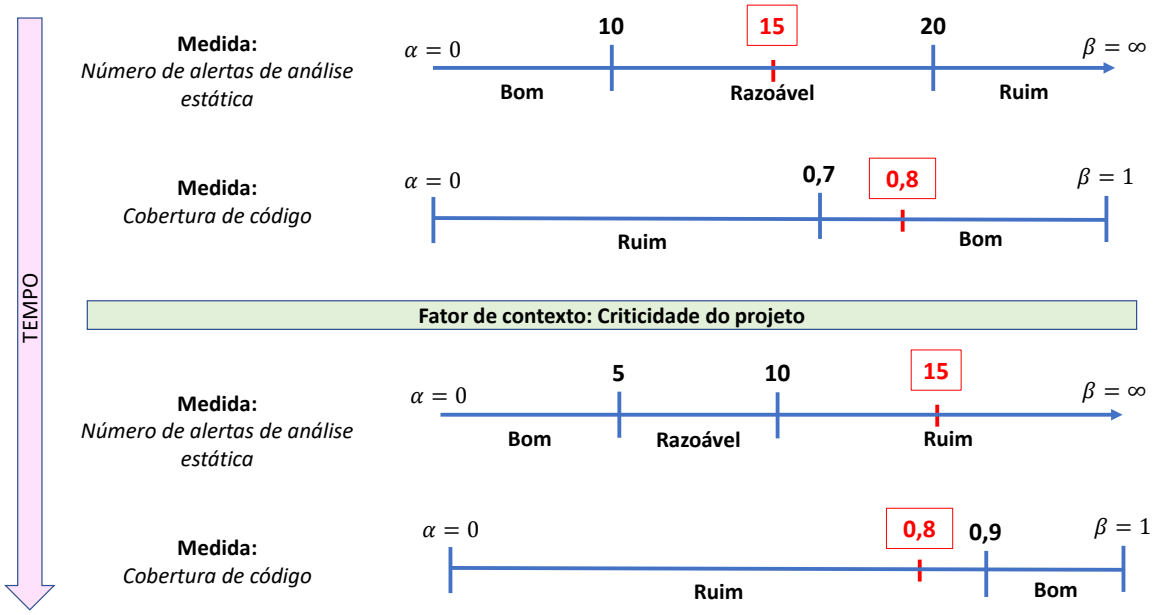


Figura 4.13: Exemplo do impacto do fator de contexto *Criticidade do projeto* na interpretação das medidas *Número de alertas de análise estática* e *Cobertura de código*

Desta forma, supondo-se que o valor coletado para a medida *Número de alertas de análise estática* seja 15, tem-se que o valor associado na escala semântica é *Razoável*. Supondo-se que o valor coletado para a medida *Cobertura de código* seja $0,8$, tem-se que o valor associado na escala semântica é *Bom*.

Contudo, assume-se que o *stakeholder* do projeto em questão resolveu adicionar um módulo crítico ao sistema, aumentando o nível de criticidade do projeto, um dos fatores de contexto que podem afetar os *thresholds* de ambas as medidas. Com este novo requisito levantado, o analista de medição verifica a necessidade de alterar os *thresholds* de ambas as medidas. Como resultado, para *Número de alertas de análise estática*, definiu-se os *thresholds*: ([0 - 5), *Bom*), ([5 - 10), *Razoável*), e ([10 - ∞), *Ruim*). Para *Cobertura de*

código, definiu-se os *thresholds*: ([0 - 0,9), *Ruim*) e ([0,9 - 1), *Bom*) (ver parte inferior da Figura 4.13). Assim, interpretação da medida *Número de alertas de análise estática* $= 15$, que era *Razoável*, passou a ser *Ruim*; e a interpretação da medida *Cobertura de código* $= 0,8$, que era *Bom*, passou a ser *Ruim*. Nota-se então que a mudança no contexto não alterou os valores das medidas, mas sim o entendimento que se tem sobre elas. Os valores que eram considerados aceitáveis, a partir da mudança do contexto, não são mais.

No método proposto, executa-se quatro subetapas para definir os *thresholds* das medidas: *Caracterização dos thresholds*, *Mapeamento dos thresholds*, *Avaliação dos thresholds* e, por fim, transfere-se os *thresholds* para a rede Bayesiana. As etapas em questão são apresentadas na Figura 4.14.

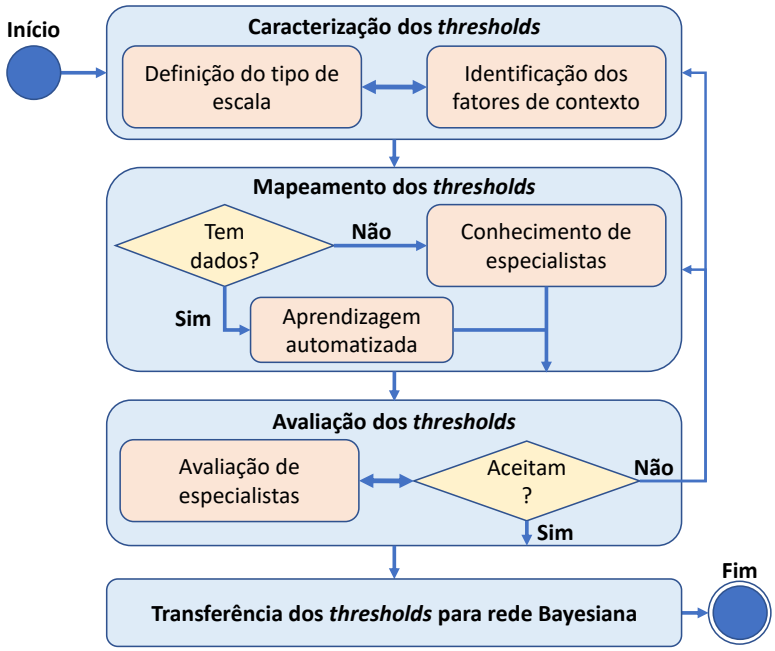


Figura 4.14: Modelo para definição de *thresholds*

Na primeira subetapa (*Caracterização dos thresholds*), o objetivo é caracterizar os *thresholds* por meio da identificação de fatores de contexto. Inicia-se definindo a escala semântica a ser utilizada para interpretar as medidas em questão. Os tipos de escala semânticas mais comuns são a ordinal e *Boolean* [117].

Posteriormente, os fatores de contexto devem ser identificados, juntamente com os seus respectivos tipos (*i.e.,* decrescente ou crescente). No caso de haver múltiplos especialistas de domínio envolvidos nesta subetapa, para evitar viés, o analista de medição deve executá-la

de forma independente com eles.

A seguir, o analista de medição, em conjunto com os especialistas de domínio, executa uma variação do *Planning Poker* [54], no qual cada especialista segura duas cartas: *Concordo* e *Discordo*. O analista de medição media a cerimônia garantindo uma discussão estruturada com relação aos *thresholds* e fatores de contexto a serem considerados. Assim como em uma cerimônia de *Planning Poker*, cada alternativa deve ser votada individualmente e de forma independente por cada especialista, e todos os participantes devem revelar a carta simultaneamente. Caso não haja consenso, os especialistas devem discutir acerca da alternativa em questão e reiniciar o processo de votação, até que o consenso seja alcançado por todos os especialistas. Ao final desta subetapa, os *thresholds* e os fatores de contexto relacionados para todas as medidas devem estar definidos.

Na segunda subetapa (*Mapeamento dos thresholds*), após haver um melhor entendimento sobre a semântica das medidas em questão, deve-se mapear os valores da escala original das medidas para a escala semântica (*i.e.,* os *thresholds* definidos previamente), levando em consideração os fatores de contexto identificados. Para tal, caso haja dados suficientes disponíveis, pode-se utilizar alguma técnica baseada em dados [3, 42, 45, 96, 98, 120, 121, 150]. Caso contrário, deve-se elicitar conhecimento dos especialistas de domínio.

Há dois tipos de mapeamentos possíveis: absoluto e probabilístico. A escolha de qual tipo de mapeamento a ser utilizado dependerá dos dados disponíveis ou da necessidade julgada pelo analista de medição. O mapeamento probabilístico tem como resultado uma representação mais precisa da escala semântica. Por outro lado, para o caso dos *thresholds* serem definidos a partir do conhecimento de especialistas, adiciona-se a complexidade de necessitar aplicar conceitos de lógica *Fuzzy*.

Independente do tipo de mapeamento a ser utilizado, para elicitar o conhecimento dos especialistas de domínio, deve-se executar um grupo de foco. No que se segue, descreve-se como executar tal grupo de foco para definir mapeamentos probabilísticos com lógica *Fuzzy*. O processo de definição para mapeamentos absolutos é executado de forma análoga.

No contexto de lógica *Fuzzy*, esta etapa de mapeamento resume-se a "fuzzificar" os valores da medida em termos linguísticos *Fuzzy* (*i.e.,* escala semântica). Para tal, há diversos tipos de funções de pertinência que podem ser utilizados. Na Figura 4.15 são apresentados seis tipos de funções de pertinência populares.

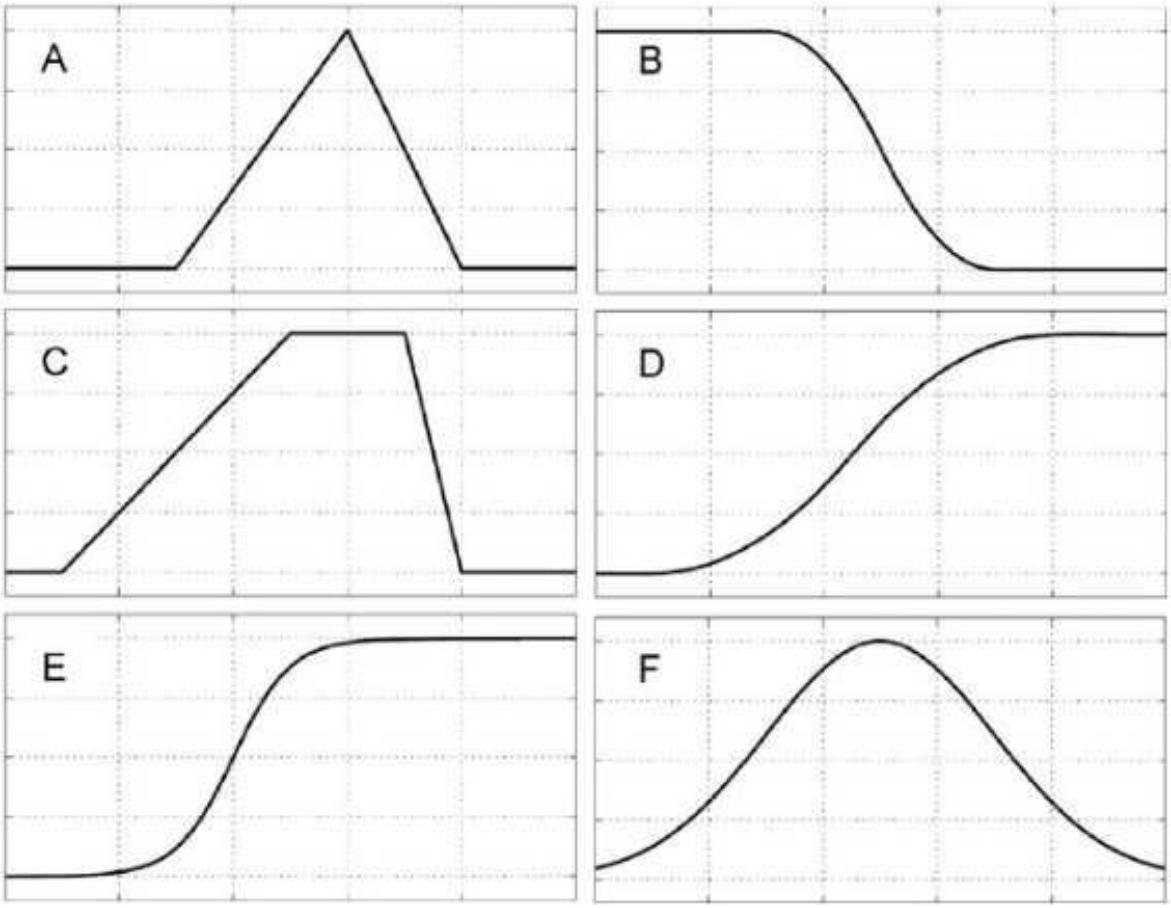


Figura 4.15: Seis tipos de funções de pertinência *Fuzzy*: (A) triangular, (B) z-shape, (C) trapezoidal, (D) s-shape, (E) sigmoid, e (F) Gaussiana.

Uma função de pertinência deve ser definida para cada valor possível da escala semântica alvo. Então, por exemplo, para uma medida com escala semântica do tipo *Boolean*, é necessário definir duas funções de pertinência.

O maior desafio desta subetapa é definir os parâmetros das funções de pertinência. Para tal, os especialistas devem utilizar as suas experiências em projetos passados para definir cenários *"what-if"*. Cada cenário é composto de um valor possível da medida em questão (*e.g.,* o valor $0,1$ para a medida *Cobertura de código*), e a probabilidade dele ser interpretado para cada valor possível da escala semântica (*e.g.,* para a medida *Cobertura de código*, poderia ser *Not OK* e *OK*). Por exemplo, para a medida *Cobertura de código*, considerando-se uma escala semântica do tipo *Boolean* (*i.e., Not OK* e *OK*), poderiam ser elicitados os cenários apresentados na Tabela 4.2.

Na Tabela 4.2, observa-se que ao invés de definir as probabilidades diretamente, eles

devem utilizar a escala verbal proposta por Renooij e Witteman [108] e apresentada na Figura 4.16. Vale salientar que, para esta etapa, é essencial que os especialistas expliquem o porquê da sua decisão, entrando em detalhes sobre as razões pelas quais eles selecionaram a probabilidade em questão, citando experiências passadas, sempre que possível. Em geral, no contexto de elicitação de conhecimento, perguntar o porquê da decisão dos especialistas é recomendado para incentivá-los a discutir o raciocínio por trás da decisão. De acordo com Mendes [86], frequentemente, ao solicitar aos especialistas mais detalhes sobre o raciocínio por trás de uma decisão, os especialistas mudam sua decisão como resultado de terem se aprofundado na explicação dela. Mais especificamente, ao discutir os cenários *"what-if"*, devem ser considerados os fatores de contexto elicitados previamente, bem como a probabilidade de ocorrência deles.

Tabela 4.2: Cenários *"what-if"* para a medida *Cobertura de código* em uma escala verbal.

Cobertura de código	**Not OK**	**OK**
0,1	Certo	Impossível
0,2	Certo	Impossível
0,3	Certo	Impossível
0,4	Provável	Improvável
0,5	Esperado	Incerto
0,6	Cinquenta-cinquenta	Cinquenta-cinquenta
0,7	Incerto	Esperado
0,8	Improvável	Provável
0,9	Improvável	Certo
1	Impossível	Certo

Posteriormente, a escala verbal é convertida em uma numérica, seguindo as regras apresentadas na Figura 4.16. Como resultado, para o exemplo apresentado previamente (ver Tabela 4.2), tem-se os valores descritos na Tabela 4.3.

Figura 4.16: Escala verbal para definição de probabilidades [108]

Tabela 4.3: Cenários *"what-if"* para a medida *Cobertura de código* em uma escala numérica.

Cobertura de código	**Not OK**	**OK**
0,1	1	0
0,2	1	0
0,3	1	0
0,4	0,85	0,15
0,5	0,75	0,25
0,6	0,5	0,5
0,7	0,25	0,75
0,8	0,15	0,85
0,9	0,15	1
1	0	1

Posteriormente, os valores coletados são utilizados para configurar as tabelas de probabilidade dos nós referentes às medidas em questão. Desta forma, para cada linha, os valores coletados para as colunas referentes à escala semântica devem somar um (1). Para a Tabela 4.3, observa-se que o somatório dos valores coletados para *Cobertura de código* = 0,9 é 1,15. Como o valor do somatório é maior do que um (1), é necessário normalizar os da-

dos em questão. Para tal, divide-se cada valor coletado da escala semântica pelo valor do somatório. Para *Not OK* tem-se:

$$v_{NOT} = \frac{0,15}{1,15} = 0,13. \tag{4.1}$$

Para *OK*, tem-se:

$$v_{OK} = \frac{1}{1,15} = 0,87. \tag{4.2}$$

Após realizar a normalização para todos os valores coletados, tem-se o conjunto de dados necessários para guiar a configuração das funções de pertinência das medidas em questão. Os valores atualizados após a normalização para os cenários *"what-if"* previamente apresentados na Tabela 4.3 são apresentados na Tabela 4.4.

Tabela 4.4: Cenários *"what-if"* para a medida *Cobertura de código* em uma escala numérica normalizada.

Cobertura de código	**Not OK**	**OK**
0,1	1	0
0,2	1	0
0,3	1	0
0,4	0,85	0,15
0,5	0,75	0,25
0,6	0,5	0,5
0,7	0,25	0,75
0,8	0,15	0,85
0,9	0,13	0,87
1	0	1

Uma vez que os cenários de *"what-if"* foram coletados, o analista de medição, tendo como base os dados gerados pelos referidos cenários, juntamente com as regras dos *thresholds* e seu conhecimento prévio em tipos de funções de pertinência, deve apresentar visualmente para os especialistas as opções de funções para guiá-los na tomada de decisão.

Há três opções não-excludentes para gerar tais funções. Uma opção é utilizar os dados coletados para realizar o ajuste de uma curva comum em lógica *Fuzzy* [147], como as apresentadas na Figura 4.15. Por exemplo, para *thresholds* marginais, poderiam ser realizados ajustes de curva para funções do tipo z-shape, sigmoid, ou s-shape. Para *thresholds* intervalares, os especialistas poderiam realizar ajustes de curva para funções do tipo triangular, trapezoidal ou Gaussiana.

Outra possibilidade seria utilizar algoritmos de interpolação ou extrapolação (*e.g.,* linear, quadrática, cúbica ou polinomial) para calcular a função de pertinência. Algoritmos de interpolação podem ser utilizados para inferir funções de pertinência para medidas contendo escalas com limite inferior α e superior β bem definidos, caso da medida *Cobertura de código*, tendo $\alpha = 0$ e $\beta = 1$. Algoritmos de extrapolação podem ser utilizados para inferir funções de pertinência para medidas contendo escalas com α ou β não sendo bem definidos, caso da medida *Número de alertas de análise estática*, tendo $\alpha = 0$ e $\beta = \infty$. A terceira possibilidade seria o especialista definir os parâmetros das opções de funções de pertinência manualmente.

Independente do meio para calcular opções de funções de pertinência, elas devem ser apresentadas visualmente para os especialistas. Por exemplo, na Figura 4.17 são apresentadas funções de pertinência calculadas por meio de interpolações cúbica, linear, quadrática e polinomial para os dados apresentados na Tabela 4.3, para o valor semântico *Not OK*.

Depois de apresentadas as opções de funções de pertinência, o analista de medição colabora com os especialistas para realizar as devidas calibrações nas funções e definir as funções de pertinência para cada valor semântico das medidas em questão. Ao final desta etapa, todos os *thresholds* estão definidos.

A próxima etapa (*Avaliação dos thresholds*) só necessita de execução caso os *thresholds* tenham sido definidos baseados em aprendizagem automática. Neste caso, os especialistas devem avaliar, por meio de cenários *"what-if"* (processo de elicitação de conhecimento similar ao passo anterior), se os *thresholds* estão consistentes com a realidade do projeto em questão. Caso os *thresholds* não estejam consistentes, os ajustes devem ser executados. Vale salientar que, como o método proposto é iterativo, há ciclos de feedback que são executados como parte da condução da medição (ver Seção 4.5) e que podem resultar em ajustes nos *thresholds*.

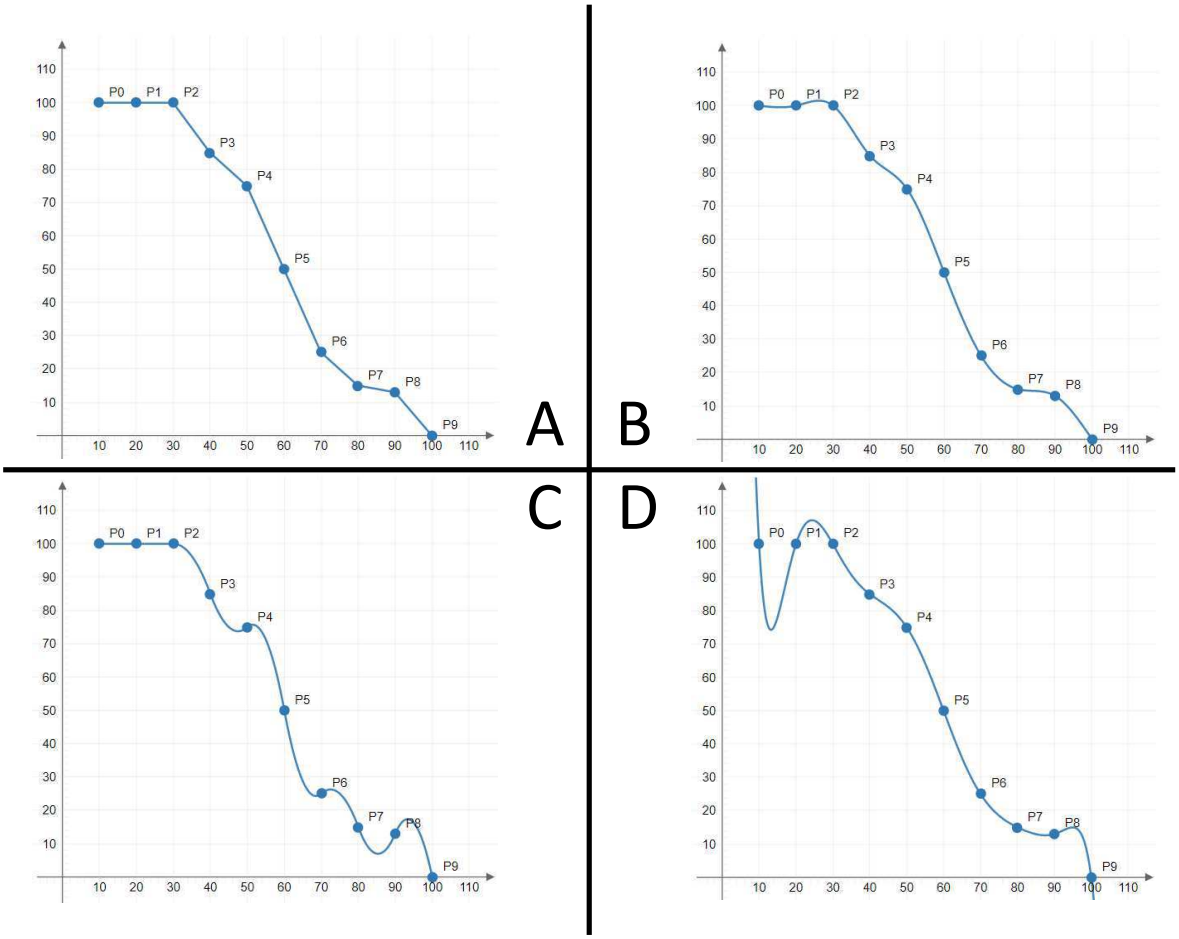


Figura 4.17: Exemplos de funções de pertinência calculadas por meio de interpolações cúbica, linear, quadrática e polinomial.

Finalmente, após os *thresholds* estarem devidamente definidos e validados, parte-se para a *transferência dos valores para a rede Bayesiana*. Tal processo é direto e resume-se a transformar as regras de *thresholds* em TPN. No que se segue, é demonstrado como realizar tal transferência para o caso de regras probabilísticas. Para regras absolutas, o processo é análogo.

O processo de transformação das regras de *thresholds* em TPN é externo à rede Bayesiana em si. Desta forma, assumindo-se que para a medida *Cobertura de código* com escala semântica do tipo *Boolean* foi definido, em função dos dados apresentados na Tabela 4.4, utilizar a interpolação quadrática como meio para calcular as funções de pertinência dos valores semânticos. As funções de pertinência em questão são apresentadas na Figura 4.18.

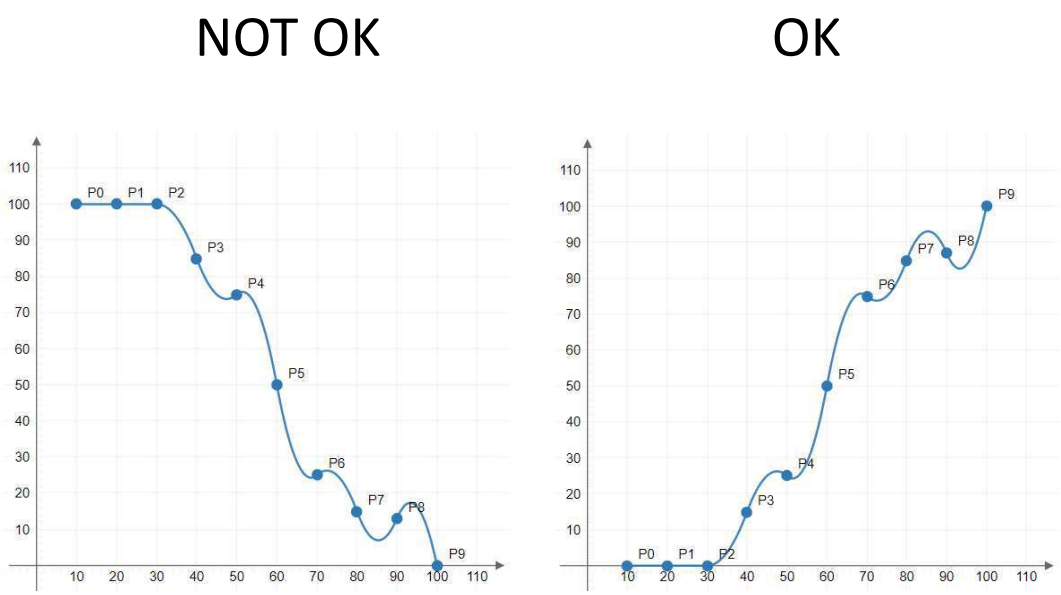


Figura 4.18: Funções de pertinência definidas para a medida *Cobertura de código*.

Dado isso, supondo-se que foi coletado o valor $0,65$ (*i.e.,* 65%) para a medida *Cobertura de código*, o primeiro passo é utilizar as funções de pertinência (ver Figura 4.18) para calcular os valores resultantes para *Not OK* e *OK*. Neste caso, os resultados são *Not OK* = 0,29 (*i.e.,* 29%) e *OK* = 0,71 (*i.e.,* 71%). Como a escala semântica da medida em questão é do tipo *Boolean*, tal medida é representada como uma variável (nó) *Boolean* na rede Bayesiana, tendo dois estados: *Not OK* e *OK*. Então, os valores calculados para *Not OK* e *OK* de acordo com as suas respectivas funções de pertinência são utilizadas para definir a TPN da medida

Tabela 4.5: TPN resultante para a medida *Cobertura de código* = 0,65 (*i.e.,* 65%).

Estado	**Probabilidade**
NOT OK	0,29
OK	0,71

em questão. Como resultado, tem-se a TPN apresentada na Tabela 4.5. Para cada valor coletado, utilizam-se as funções de pertinência para calcular a evidência a ser utilizada como entrada no nó da rede Bayesiana referente à medida em questão.

A entrada e saída da etapa de definição de *thresholds* são ilustradas na Figura 4.19.

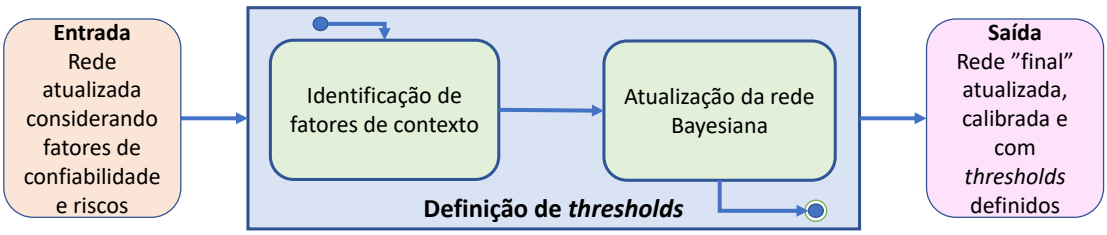


Figura 4.19: Entrada e saída da etapa de definição de *thresholds*

4.5 Aplicação do Método e Condução da Medição

Como explanado ao longo deste capítulo, a entrada para o método proposto consiste na identificação de um fator externo ao programa de medição, mas essencial ao sucesso dele, chamado de "proposição de valor" [107] ou "propósito" [95]. O patrocinador da medição precisa ter em mente o valor que a aplicação do metódo de medição trará a organização. Além disso, é importante estabelecer e manter o comprometimento dos envolvidos com a medição [123].

Cada etapa do método deve ser executada de forma sequencial, conforme ilustrado pela seta "azul" na Figura 4.20. Contudo, pode-se retornar a qualquer etapa, a qualquer momento da execução do método, com o objetivo de refinar ou substituir a abordagem utilizada na etapa em questão, ou ainda refinar os artefatos produzidos ao longo dela. Estes retornos são representados pelas setas "lilás". Vale salientar que todas as etapas devem ser execu-

tadas tendo como foco a interpretação de medidas. Portanto, não é recomendado alterar as abordagens utilizadas nas etapas com o intuito apenas de facilitar ou agilizar a execução do método.

Vale salientar também que redes Bayesianas foram utilizadas como instrumento para viabilizar a execução do método proposto, uma vez que elas possibilitam a modelagem da incerteza de dados envolvida na problemática em questão, bem como um maior entendimento por parte dos usuários de medição, dada a representação gráfica de sua estrutura. Além disso, a utilização de redes Bayesianas facilita as análises, modificações e interpretação de resultados [152, 140, 35], principalmente quando se considera a combinação de diferentes fontes de conhecimento. Por todas estas vantagens e considerando o foco sobre a interpretação de medidas, recomenda-se fortemente a utilização desses modelos.

Sendo assim, dado que a saída de cada etapa constitui uma rede Bayesiana atualizada e calibrada, e que é possível retornar às etapas e adicionar mais nós e idiomas de medição, é importante destacar a necessidade de uma análise mais aprofundada sempre que a rede for reformulada, de modo a não deixá-la tão complexa a ponto de dificultar seu uso para auxílio à tomada de decisão.

Após a execução de todas as etapas, tendo como saída a rede "final" atualizada, calibrada e com *thresholds* definidos, tem-se a condução da medição. Neste momento, os dados coletados são utilizados na execução da rede a fim de obter um produto de informação [123] para tomada de decisão. Caso haja alguma dúvida em relação aos valores obtidos, pode-se retornar a uma das etapas do método para refinamento ou alterar manualmente as tabelas de probabilidades geradas.

Por fim, vale salientar que o modelo gerado ao longo da execução do método pode ser reutilizado. Neste caso, é importante identificar as semelhanças e discrepâncias do modelo em relação ao cenário que se deseja aplicar o método proposto. A partir disto, deve-se fazer as adequações necessárias no modelo considerando as especificidades do novo cenário.

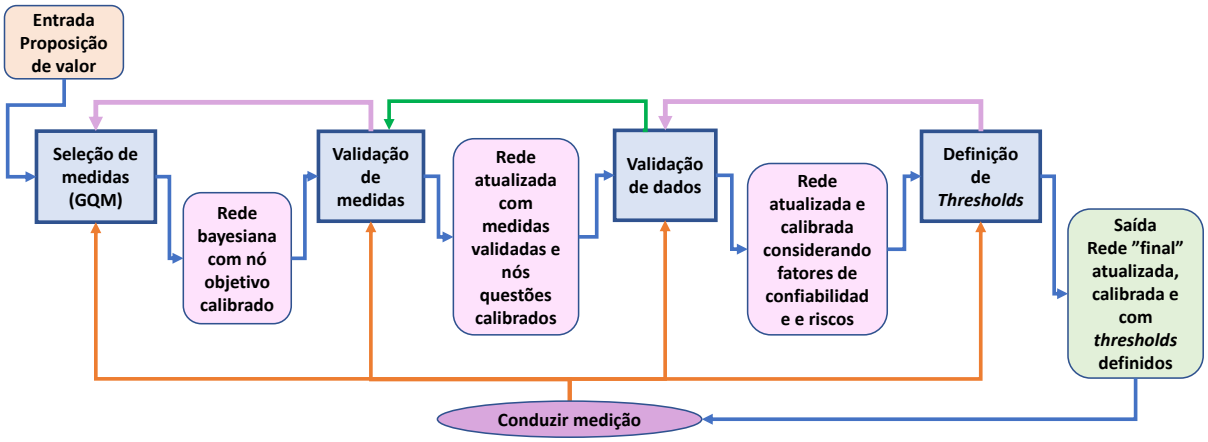


Figura 4.20: Fluxo de execução do método proposto

4.6 Considerações Finais do Capítulo

Neste capítulo foi apresentado um método de medição de software, baseado em redes Bayesianas, com foco na interpretação de medidas. O objetivo de tal método é possibilitar aos usuários de medição uma assertividade maior na tomada de decisão a partir de uma interpretação mais correta das medidas utilizadas.

O método é composto de quatro etapas, são elas: seleção de medidas, validação de medidas, validação de dados e definição de *thresholds*. Estas etapas foram extraídas do padrão ISO/IEC 15939:2007. Mais especificamente, buscou-se nesta documentação atividades e tarefas que têm maior impacto na interpretação de medidas.

Para cada etapa do método, correspondente a uma seção deste capítulo, foi apresentada uma descrição da etapa, os artefatos de entrada e saída da etapa, e uma descrição sobre como operacionalizar a etapa a partir da utilização de redes Bayesianas. Neste contexto, para facilitar o entendimento das etapas, bem como a integração delas, utilizou-se um exemplo simulado composto de um único objetivo.

Por fim, na Seção 4.5 são descritas informações necessárias para o entendimento e aplicação do método proposto, bem como sobre a condução da medição em si, considerando o refinamento do método e do modelo, bem como a possibilidade de reúso deste para cenários similares.

Capítulo 5

Validação do Método

Neste capítulo são apresentados a metodologia e os resultados da validação do método proposto. O estudo foi conduzido com o intuito de analisar o método proposto à luz da sua utilidade prática. Para definir a metodologia do estudo, foi utilizado como arcabouço a Teoria Unificada de Aceitação e Uso de Tecnologia (TUAUT) [142]. Na TUAUT, há quatro construtos exógenos (*i.e., Expectativa de Desempenho*, *Expectativa de Esforço*, *Influência Social* e *Condições Facilitadoras*) e dois endógenos (*i.e., Intenção Comportamental de Uso* e *Comportamento de Uso Real*). A relação entre os construtos é apresentada na Figura 5.1.

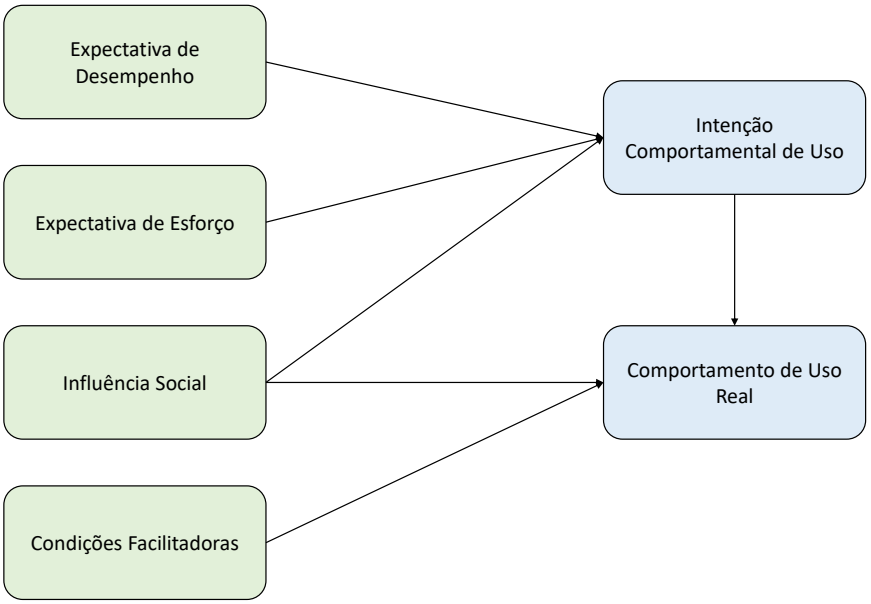


Figura 5.1: Modelo de TUAUT.

As definições de cada construto, de acordo com Venkatesh *et al.* [142], são apresentadas no que se segue.

- **Expectativa de Desempenho**: o grau no qual um indivíduo acredita que utilizar a tecnologia irá ajudá-lo a obter benefícios na sua performance no trabalho. Tal construto é o preditor mais relevante de *Intenção Comportamental de Uso*.
- **Expectativa de Esforço**: o grau da facilidade de uso associada com a utilização da tecnologia.
- **Influência Social**: o grau no qual o indivíduo percebe que outras pessoas importantes creem que ele ou ela deve utilizar o sistema.
- **Condições Facilitadoras**: o grau no qual o indivíduo acredita que há uma organização e infraestrutura técnica para dar suporte à tecnologia.
- **Intenção Comportamental de Uso**: a intenção do indivíduo em utilizar a tecnologia. É um preditor de *Comportamento de Uso Real*.
- **Comportamento de Uso Real**: a utilização de fato da tecnologia pelo indivíduo.

De acordo com a perspectiva de um usuário final, a transferência de tecnologia apenas é realizada quando um produto vai para o mercado [146]. Desta perspectiva, a variável *Comportamento de Uso Real*, do TUAUT, só pode ser avaliada após a transferência tecnológica, que não faz parte de projetos de pesquisa [7, 145]. Na pesquisa, apenas é possível prever o *Comportamento de Uso Real*, baseando-se em validações ou avaliações estáticas ou dinâmicas [51]. Seguindo este raciocínio, definiu-se a seguinte questão de pesquisa para guiar a validação do método proposto:

QP Qual o potencial de adoção do método proposto?

Para definir a estratégia para responder esta questão de pesquisa, foi utilizado como base o construto proposto pela TUAUT. Inicialmente, foi realizado um estudo de caso com uma versão parcial do método proposto, contemplando apenas as etapas *Seleção de Medidas* e *Validação de Dados*. Posteriormente, uma ferramenta que operacionaliza o método proposto foi desenvolvida. Trata-se de um sistema Web denominado *Software Attributes Measurement*

(SAM). O desenvolvimento de tal ferramenta, contemplando as quatro etapas do método proposto, os modelos conceituais subjacentes, e o relacionamento entre as entidades de informação, pode ser considerado, em efeito, uma demonstração semi-formal da viabilidade de aplicação prática dos conceitos relacionados à interpretação de medidas inerentes ao método proposto.

Finalmente, o método completo foi avaliado por meio de um grupo de foco com praticantes da indústria. Tradicionalmente, estudos baseados em modelos de aceitação de tecnologia (*e.g.,* TAM e TUAUT) utilizam-se de questionários para realizar a coleta de dados. Por outro lado, foi decidido utilizar um estudo de caso e o grupo de foco devido a serem canais mais ricos de comunicação [4], possibilitando uma análise mais profunda dos dados [71]. Outra possibilidade, ao invés do grupo de foco, seria entrevistar os praticantes da indústria, mas o grupo de foco vem se mostrando uma técnica com melhor custo-benefício, tanto do lado do pesquisador quanto pelos participantes [71].

É importante destacar que o planejamento inicial para realizar o terceiro estudo era realizar uma validação dinâmica por meio de um estudo de caso no qual o método proposto seria aplicado e avaliado em projetos reais de software de uma organização parceira. Porém, a pandemia do COVID-19 e o consequente distanciamento social inviabilizou a realização de tal estudo de caso. O impacto da pandemia em pesquisas acadêmicas, principalmente no que se refere à mães acadêmicas, como é o caso da pesquisadora em questão, é apresentado por Staniscuaski *et al.* em [125]. Desta forma, para o método completo, foi realizada uma validação estática por meio de um grupo de foco, executado de forma remota.

Ambos os estudos (*i.e.,* estudo de caso e grupo de foco) foram realizados em parceria com uma organização, mais especificamente, o VIRTUS[1]. No que se segue, apresenta-se a descrição do contexto do VIRTUS (Seção 5.1), o protocolo e resultados do estudo de caso (Seção 5.2), uma visão geral do SAM (Seção 5.3), o protocolo e resultados do grupo de foco (Seção 5.4) e a discussão da questão de pesquisa (**QP**) à luz dos resultados de ambos os estudos (Seção 5.5).

[1]https://www.virtus.ufcg.edu.br/

5.1 Caracterização do VIRTUS

Nesta seção são apresentados dados do VIRTUS seguindo as facetas de contexto apresentadas por Petersen e Wohlin [105]: *produto*, *processos*, *práticas e técnicas*, *pessoas*, *organização* e *mercado*.

O VIRTUS é um núcleo de pesquisa, desenvolvimento e inovação em tecnologia da informação, comunicação e automação. Trata-se de um órgão suplementar da Universidade Federal de Campina Grande (UFCG). A organização gerencia centenas de engenheiros e pesquisadores, co-localizados na sua sede em Campina Grande, na Paraíba. Vale salientar que, no momento que o grupo de foco foi realizado, todas as atividades do VIRTUS estavam sendo executadas remotamento devido ao recomendado distanciamento social causado pela pandemia do COVID-19.

O VIRTUS foi fundado em 2015 por pesquisadores da UFCG com mais de 15 anos de experiência em projetos de pesquisa e desenvolvimento e executa projetos em diversos domínios tecnológicos incluindo ciência de dados, sistemas Web, sistemas Mobile, inteligência artificial, realidade aumentada, sistemas embarcados e hardware. Além disso, os projetos focam em diversos segmentos de mercado incluindo segurança, biometria e *business intelligence*. Os projetos tipicamente duram entre dez e dezoito meses e são frutos de mecanismos de incentivo como Lei de Informática, EMBRAPII, ANP, dentre outros. Tipicamente, são executados entre 40 e 50 projetos por ano, tendo parceiros da indústria como a Asus, Compal, Envision, Ericsson, dentre outras empresas de tecnologia de pequeno, médio e grande porte.

Do ponto de vista estrutural, pode-se considerar o VIRTUS como sendo hierárquico e orientado a projeto. Por outro lado, deve-se considerar que o VIRTUS contém um departamento de qualidade, responsável por definir diretrizes para o processo de qualidade executados pelos projetos - incluindo o sistema de medição - e auditá-los. No momento da execução do estudo, o VIRTUS não era certificado com modelos de qualidade (*e.g.,* ISO 9001) ou maturidade (*e.g.,* CMMI ou MPS.Br). Por outro lado, a sua carteira de parceiros é um indicador da qualidade e maturidade do seu processo de desenvolvimento.

Em geral, os projetos são executados utilizando abordagens ágeis de gerenciamento como Scrum ou Kanban. As práticas e ferramentas de desenvolvimento adotadas pelos projetos se-

guem as diretrizes definidas pela organização, mas são adaptadas de acordo com o contexto (*e.g.,* linguagem de programação e tipo de sistema). Para auxiliar na gestão dos projetos, utiliza-se uma ferramenta própria, que integra o gerenciamento de requisitos, testes, *issues*, repositórios e construção de sistemas. Tal ferramenta habilita a rastreabilidade entre os artefatos de desenvolvimento seguindo um modelo similar ao *Agile Traceability Information Model*, popular em ferramentas de gestão ágil (*e.g.,* Rally[2]).

5.2 Estudo de Caso

De acordo com Kitchenham, Pickard e Pfleeger [67], estudos de caso mostram os efeitos de um projeto em uma situação específica, não podendo ser generalizado para outras situações possíveis. Então, a conclusão deste estudo é limitada ao contexto do VIRTUS.

Vale ressaltar que este estudo de caso foi realizado em parceria com um aluno de mestrado do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande. Dado isto, uma vez que as etapas de validação de medidas e definição de *thresholds* ainda estavam sendo definidas, o estudo de caso contemplou uma versão parcial do método proposto, contemplando apenas as etapas *Seleção de Medidas* e *Validação de Dados*. Sendo assim, no restante desta seção, a palavra "método" é utilizada considerando apenas as duas etapas supracitadas. No que se segue, apresenta-se o protocolo do estudo de caso (Seção 5.2.1), os resultados (Seção 5.2.2), ameaças à validade (Seção 5.2.3) e conclusões do estudo (Seção 5.2.4).

5.2.1 Protocolo

Para a execução do estudo de caso, foi seguido o guia apresentado em [112]. No que se segue, são apresentados os objetivos do estudo de caso (Seção 5.2.1.1), a caracterização das unidades de análise (Seção 5.2.1.2) e sujeitos, e procedimentos (Seção 5.2.1.3).

5.2.1.1 Objetivos

O objetivo do estudo de caso foi (*i*) analisar o método com respeito à intenção de sua adoção sob o ponto de vista do líder de projeto. Para este estudo, foi utilizado como base para

[2]http://www.rallydev.com

definir as questões de pesquisa o construto de utilidade do *Technology Acceptance Model* (TAM) [24]. O TAM é composto de dois construtos: *Utilidade percebida* e *Facilidade de uso percebida*, os quais, em relação ao TUAUT são relacionados, respectivamente, com os construtos exógenos *Expectativa de desempenho* e *Expectativa de esforço*. Assim, pode-se considerar o construto do TAM como um subconjunto do TUAUT. Dado isto, foram definidas as seguintes questões de pesquisa:

QP_EC_1 Qual é o benefício de utilizar o método para apoiar o líder de projeto na tomada de decisão gerencial no VIRTUS?

QP_EC_2 Qual é o esforço percebido para adotar o método no VIRTUS?

5.2.1.2 Caracterização das unidades de análise e sujeitos

O referente estudo de caso contemplou uma unidade de análise, que correspondeu a um projeto de desenvolvimento de software, com 12 membros, sendo dois líderes de projetos, oito desenvolvedores e dois testadores. Neste contexto, foi utilizado um processo de desenvolvimento iterativo e incremental, bem como o arcabouço de gerenciamento *Scrum*, para o desenvolvimento de uma aplicação baseada em Android para dispositos móveis (*smartphones* e *tablets*), integrado a dispositivos portáteis, como relógios inteligentes e pulseiras inteligentes.

Vale ressaltar que, antes de aplicar o método, a equipe já tinha um conjunto de medidas quantitativas utilizadas para apoiar a tomada de decisão, as quais não foram definidas utilizando-se o paradigma GQM. As medidas utilizadas foram: *número de erros*, *cobertura de código*, *alertas de análise estática*, *alertas de Checkstyle* e *Sprint burndown* (*e.g.,* velocidade).

Vale ressaltar também que o sujeito do estudo de caso, líder do projeto em questão, foi o responsável pela maior quantidade de módulos e componentes desenvolvidos no referido projeto, além de ter gerenciado o projeto sozinho por um certo período. Portanto, entende-se que este sujeito tinha maior conhecimento acerca do projeto como um todo.

5.2.1.3 Procedimentos

Neste estudo de caso, realizado de forma presencial, com duração de aproximadamente dois meses, utilizou-se o AgenaRisk para executar a inferência na rede Bayesiana, bem como o procedimento apresentado na Figura 5.2, que é dividido em duas fases.

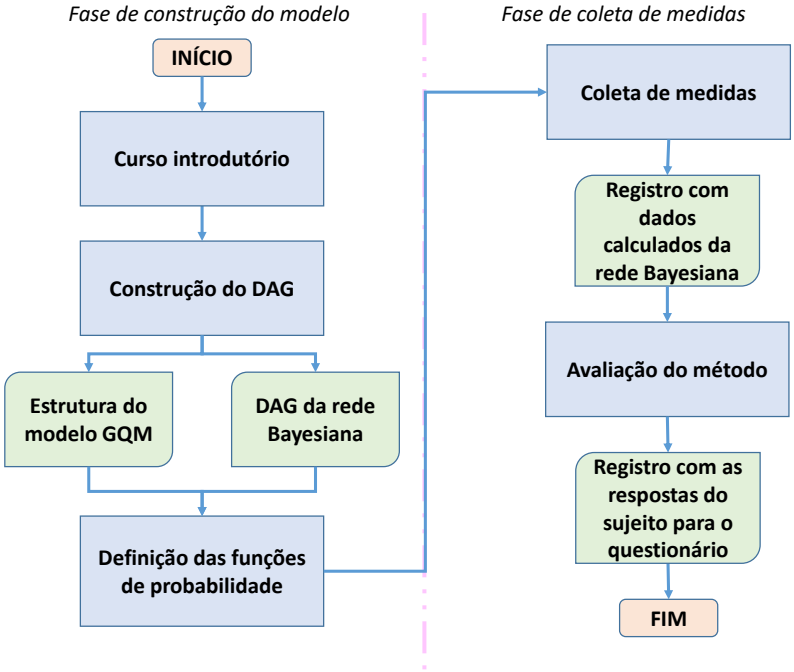


Figura 5.2: Procedimento aplicado a unidade de análise do estudo de caso.

Fase de construção do modelo

Durante esta fase, foi realizada uma capacitação do sujeito com relação aos conceitos básicos relacionados ao método. Em seguida, com a assistência dos pesquisadores envolvidos, o sujeito construiu o modelo GQM e aplicou o método para construir a rede Bayesiana. As atividades contempladas nesta fase foram:

Curso introdutório - foram explicados os conceitos necessários para aplicar o método, (*e.g.,* GQM e redes Bayesianas). Após apresentar conceitos básicos e exemplos, algumas questões foram abordadas, como: “Como definir objetivos:”, “Dados os objetivos, como definir questões e medidas?”, "O que é uma rede Bayesiana?", "Como aplicar o método?"e "Quando parar a decomposição dos fatores?". Esta atividade durou 30 minutos.

Construção do DAG - o sujeito foi auxiliado na execução das etapas (*i*) e (*ii*) do método, que são, respectivamente, construir a estrutura do modelo hierárquico GQM e o DAG da rede Bayesiana. Inicialmente, dado que a equipe já usava medidas no projeto, foi discutido o raciocínio por trás de tais medidas. Com isto, a estrutura do modelo GQM foi construída. Não houve modificações no conjunto de medidas inicial. Esta etapa durou 50 minutos.

Definição das funções de probabilidade - nesta atividade, a etapa (*iii*) do método foi executada. O sujeito respondeu um questionário para quantificar a magnitude relativa dos relacionamentos de cada nó filho no modelo. Esta atividade durou 14 minutos.

Fase de coleta de medidas

Esta fase durou 45 dias, ou três iterações (*i.e., sprints*). Durante esta fase, o sujeito inseriu as evidências (*i.e.,* dados coletados) no modelo e avaliou o método proposto. Esta fase contemplou as seguintes atividades:

Coleta de medidas - ao final de cada iteração, o sujeito apresentou as medidas coletadas do projeto e as utilizou como entrada no modelo. Além disso, o sujeito forneceu evidências sobre os fatores de confiabilidade e os controladores de riscos (ou seja, nós de folhas). Em seguida, a rede Bayesiana foi executada e os dados foram analisados. Esta reunião durou, em média, 25 minutos.

Avaliação das medidas - foi realizada uma entrevista estruturada, na qual o sujeito avaliou o método.

As variáveis independentes das configurações do estudo foram: o AgenaRisk, a complexidade da aplicação do método e o nível de experiência do líder do projeto na construção do modelo. As variáveis dependentes estavam relacionadas à medição da *Expectativa de Desempenho* e *Expectativa de Esforço* do uso do método. Estas medidas são descritas a seguir:

Expectativa de Desempenho corresponde a extensão em que uma pessoa pensa que o uso do sistema melhora seu desempenho no trabalho [110]. Para medir o benefício, durante entrevistas estruturadas realizadas ao final da fase de coleta de medidas (*i.e., post mortem*), foram feitas duas perguntas: “O uso do método aumenta a confiança nas decisões tomadas

em relação aos objetivos avaliados?" e "O uso do método auxilia na sua tomada de decisões para continuamente melhorar o processo?". Para ambas as perguntas, foram coletadas respostas abertas (*i.e.,* dados qualitativos). Além disso, levou-se em consideração os artefatos construídos pelo sujeito durante a execução do estudo de caso.

Expectativa de Esforço consiste na análise do esforço (*i.e.,* custo) referente à adoção do método. Foram coletados dados para essa medida durante a entrevista estruturada realizada ao final da fase de coleta (*i.e., post mortem*). Neste contexto, foi feita a seguinte pergunta: "Como você avalia o esforço para adotar o método proposto?". Com isto, coletou-se uma resposta aberta (*i.e.,* dados qualitativos). Além disso, levou-se em consideração o tempo requerido pelo sujeito para executar as tarefas associadas à aplicação do método (*i.e.,* dado quantitativo).

O método de coleta de dados incluiu entrevistas, questionários e medidas do projeto. Foi realizada uma entrevista não estruturada para coletar informações sobre o domínio do projeto e construir a estrutura hierárquica do modelo GQM e o DAG da rede Bayesiana. Foi aplicado um questionário para coletar dados e definir as funções de probabilidade, seguindo o método apresentado em [103]. O sujeito, por sua vez, coletou dados do projeto para inserir como evidências no modelo construído. Por fim, foi realizada uma entrevista estruturada para coletar dados do sujeito sobre a avaliação do método.

5.2.2 Resultados

Nesta seção, são apresentados os resultados da execução do método, seguindo o procedimento apresentado na Figura 5.2. No que se segue, são apresentados os modelos construídos (Seção 5.2.2.1), os resultados relacionados à **QP_EC_1** (Seção 5.2.2.2), e os resultados relacionados à **QP_EC_2** (Seção 5.2.2.3).

5.2.2.1 Modelos construídos

A rede Bayesiana gerada ao fim deste procedimento era composta de 56 nós. Este número é um reflexo do número de medidas escolhidas e dos detalhes do processo que o profissional modelou seguindo as etapas do método, ilustrado na Figura 4.10.

A estrutura hierárquica do modelo GQM construído para o projeto em questão é ilustrada na Figura 5.3. Os DAG que foram construídos ao longo da execução do método são

ilustrados nas Figuras 5.4 e 5.5. Por fim, nas Figuras 5.6 e 5.7, são ilustrados os dados coletados e os dados calculados levando em consideração a incerteza do processo de coleta deles (*i.e.,* "interpretados") para as medidas independentes do estudo. Nas Figuras 5.6 e 5.7, os dados coletados são representados pelas barras coloridas com cinza escuro, e os dados "interpretados" pelas barras coloridas com cinza claro.

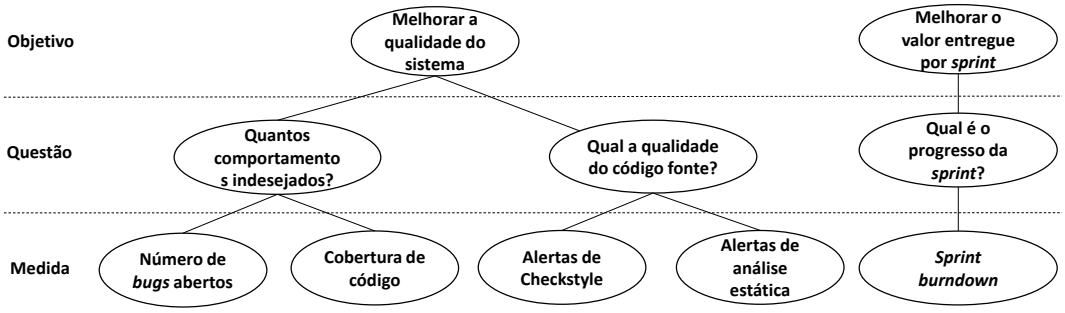


Figura 5.3: Estrutura hierárquica do modelo GQM

Ao analisar a Figura 5.6, conclui-se que a confiança relacionada à medida *Número de Bugs* é baixa a ligeiramente moderada. Esta confiança reflete a falta de boas práticas de teste e problemas com relatórios de erros que foram evidenciados pelo líder do projeto. A confiança em relação a *Alertas de análise estática* é um pouco alta. Por fim, a confiança correspondente a *Alertas de Checkstyle* é moderada. Esta confiança, por sua vez, reflete problemas com a configuração das ferramentas. Este nível de confiança é refletido nos valores calculados para o objetivo *Avaliar o valor entregue por sprint*, conforme ilustrado na Figura 5.8. Isto significa que, dada a situação do projeto, o líder deve ser cuidadoso ao responder às perguntas *Quantos comportamentos indesejados?* e *Qual é a qualidade do código fonte?* e, com isto, avaliar o objetivo *Avaliar qualidade do sistema*.

Ao analisar a Figura 5.7, conclui-se que a confiança em relação a *Sprint Burndown* é alta. Este nível de confiança é refletido nos valores calculados para o objetivo *Avaliar o valor entregue por sprint*, ilustrado na Figura 5.8. Portanto, o líder deve ter confiança em responder à pergunta *Qual é o progresso do sprint?*, e avaliar objetivo *Avaliar o valor entregue por sprint*.

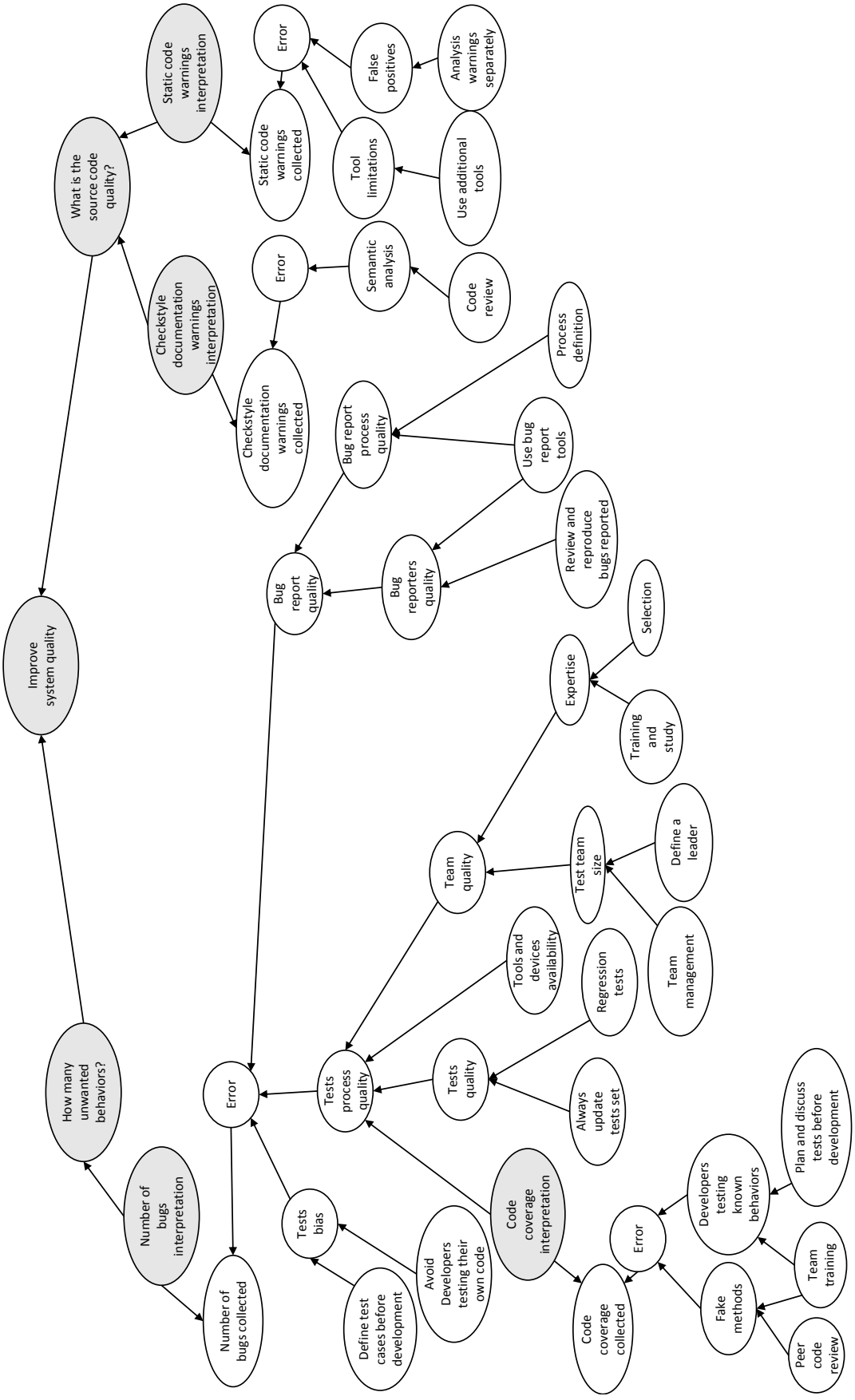


Figura 5.4: DAG completo construído para o objetivo *Avaliar a qualidade do sistema*

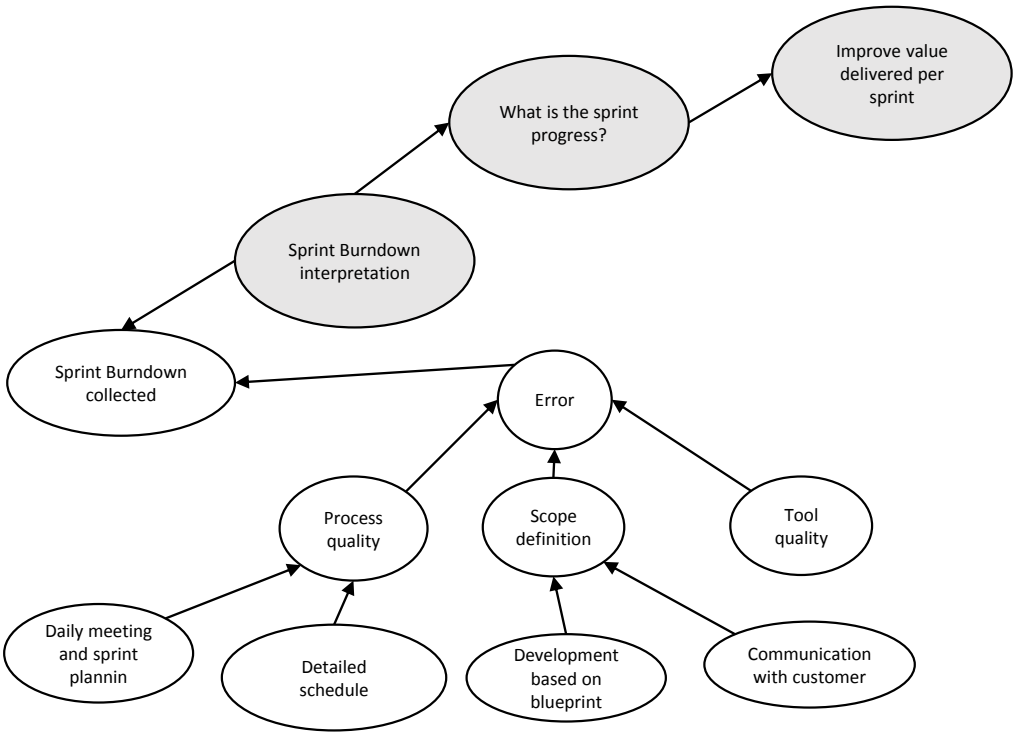


Figura 5.5: DAG completo construído para o objetivo *Avaliar o valor entregue por sprint*

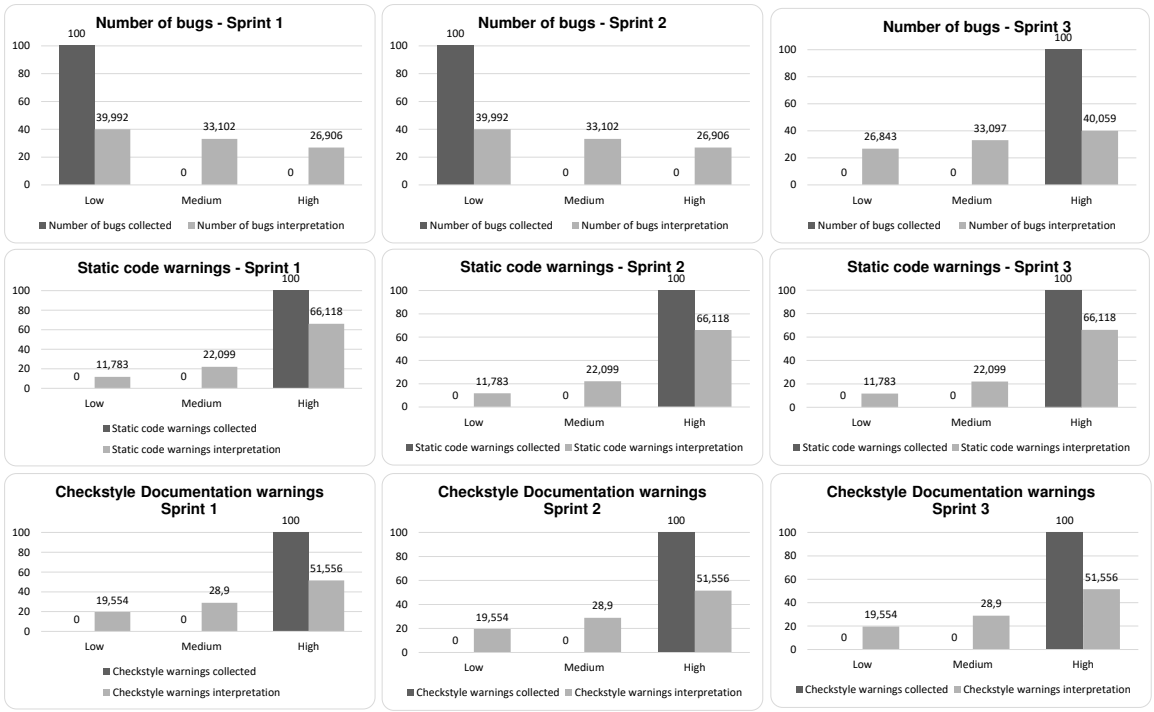


Figura 5.6: Medidas coletadas e interpretadas para o objetivo *Avaliar a qualidade do sistema*

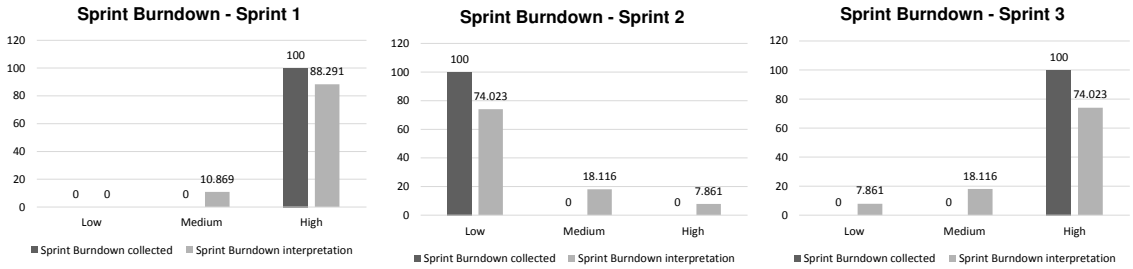


Figura 5.7: Medidas coletadas e interpretadas para o objetivo *Avaliar o valor entregue por sprint*

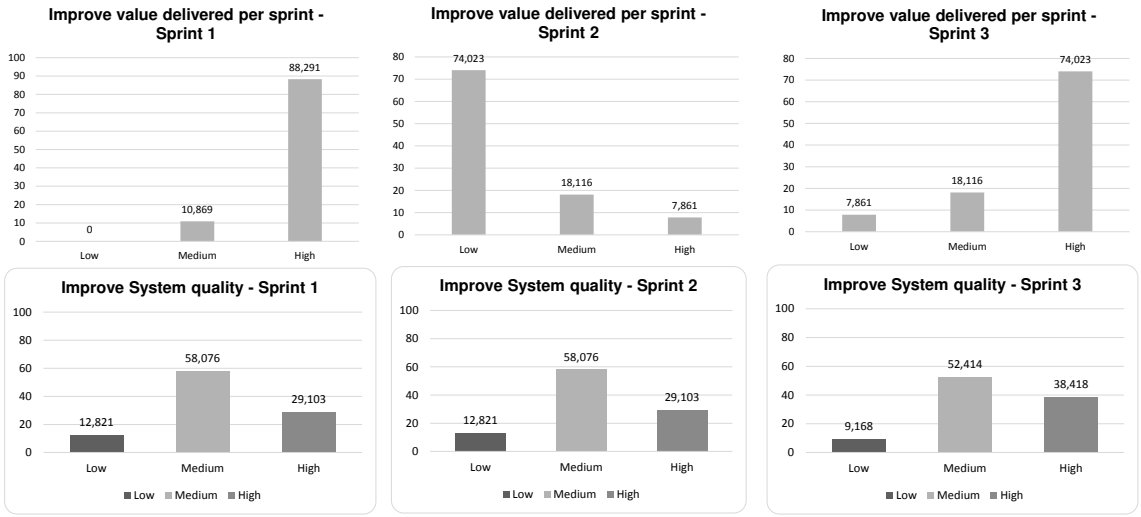


Figura 5.8: Valores calculados para o projeto de acordo com os objetivos

5.2.2.2 Benefícios de aplicar o método (QP_EC_1)

Na **QP_EC_1** foram explorados os benefícios do método proposto. Como discutido na Seção 5.2.1.3, após a construção dos modelos, foram realizadas duas perguntas para o sujeito do estudo por meio de uma entrevista estruturada. As perguntas seguidas das respostas são apresentadas no que se segue.

Em relação à pergunta "O uso do método aumenta a confiança nas decisões tomadas em relação aos objetivos avaliados?", ele respondeu: *"Com o método, temos dados concretos. O método mostra explicitamente oportunidades de melhoria. Por exemplo, sabíamos que tínhamos problemas com os testes, mas não conseguimos identificar objetivamente as causas principais. Agora, está claro que o problema é que os membros da equipe não têm experiência em escrever testes."*

Em relação à pergunta "O uso do método auxilia na sua tomada de decisões para continuamente melhorar o processo?", ele respondeu: *"Nós podemos identificar onde estão os principais problemas e elaborar pontos de ação para melhorar o processo."*

Após a entrevista, ele reportou que antes de aplicar o método não confiava na medida "*Número de bugs abertos*". Ao aplicar o método e analisar os dados calculados, foi apto a desenvolver uma análise de causa raiz e identificar oportunidades de melhoria na área de teste. Além disso, ele afirmou que o modelo ajuda na visualização do processo como um todo e auxilia na interpretação das medidas e, consequentemente, no alcance do objetivo. Por fim, ele acrescentou que, dada a análise realizada construindo os modelos e analisando os dados calculados, ele começou a pesquisar por melhores ferramentas para gerenciar o processo de desenvolvimento de software e realizar a análise estática do código.

Primeiramente, analisando a primeira resposta utilizando uma abordagem de codificação indutiva [130], pode-se classificar o benefício relacionado como sendo *Externalização*, no sentido de transformar o conhecimento tácito, com relação ao processo e as medidas, em conhecimento explícito, por meio de sua formalização em uma rede Bayesiana. Analisando a resposta da segunda pergunta, pode-se classificar o benefício de *Apoio ao diagnóstico*, ou seja, o método auxiliou o líder e sua equipe a diagnosticar as tomadas de decisão.

Além disso, analisando as informações complementares reportadas pelo sujeito do estudo, pode-se concluir que outro benefício inerente à aplicação do método é a *Melhoria*

na interpretação das medidas, que por sua consequência, causa a *Melhoria na tomada de decisões*. Este benefício também é consequência lógica da análise dos dados coletados e interpretados (ver Figuras 5.6 e 5.7), discutidos na Seção 5.2.2.1.

***QP_EC_1**: O método tem como benefícios a Externalização, Apoio ao diagnóstico, Melhoria na interpretação das medidas e Melhoria na tomada de decisões.*

5.2.2.3 Esforço ao adotar o método (QP_EC_2)

Na **QP_EC_2** foi explorado o esforço percebido pelo sujeito com relação à adoção do método. Como discutido na Seção 5.2.1.3, após a construção dos modelos, foi realizada uma pergunta para o sujeito do estudo por meio de uma entrevista estruturada. Pergunta e resposta são apresentadas no que se segue.

Em resposta à pergunta "Como você avalia o esforço para adotar o método proposto?", ele respondeu: *"O tempo investido na construção das redes Bayesianas e os dados introduzidos foram superestimados pela análise realizada. Agora, temos mais dados para ajudar nas tomadas de decisões e definir ações corretivas para melhorar o desempenho da equipe. Se a equipe melhora, o processo melhora, o controle sobre os resultados melhora, a qualidade do produto aumenta e, finalmente, a satisfação do cliente aumenta."*

Nota-se que ao responder à pergunta, o sujeito fez menção à informações relacionadas à **QP_EC_2** e **QP_EC_1**. A primeira sentença da resposta afirma que o *"tempo investido ... foi superestimado pela análise realizada"*, ou seja, ao construir os modelos, ele teve a sensação de um alto grau de esforço, mas que depois ele julgou ser compensado pelos resultados alcançados. Tal fenômeno pode ser explicado pela necessidade de um esforço inicial para a construção da rede Bayesiana, que durou em torno de 54 minutos. Posteriormente, foi registrada a necessidade de 25 minutos por iteração para utilizar a rede Bayesiana construída. Considerando que ele trabalha em um regime comum de 40 horas por semana, tem-se que a utilização do método ocupou $0,5\%$ da sua capacidade de trabalho. Nota-se que, durante o estudo, os dados relacionados a cada variável foram introduzidas no modelo manualmente.

***QP_EC_2**: O esforço inicial para aplicar o método é considerável devido à necessidade de construir os modelos. Por outro lado, o esforço de utilização dos modelos é baixo, mesmo no caso de coleta manual dos dados.*

5.2.3 Ameaças à validade

As ameaças à validade dos resultados referentes ao estudo de caso são discutidas à luz da classificação apresentada em [112, 111]. Então, as ameaças à validade referentes a um estudo de caso foram classificadas como interna, de construção, externa e de confiabilidade.

A validade interna do estudo é relacionada ao quanto a condução do estudo garante que o relacionamento causal entre tratamento e resultado, se houver, não é resultado de um fator o qual o pesquisador não tem controle. Neste caso, durante atividade de construção de DAG, devido a sua duração, o sujeito pode ter se cansado e, consequentemente, construído modelos de baixa qualidade. Para minimizar esta ameaça, foram realizadas pausas quando necessário e utilizada a experiência da pesquisadora para decompor os fatores da rede Bayesiana. Um viés do estudo relacionado à validade interna está relacionado ao fato de haver a presença da pesquisadora auxiliando na aplicação do método, em especial, na construção dos modelos. Por outro lado, para a aplicação do método proposto, assume-se que o analista de medição tenha conhecimento sobre redes Bayesianas.

A validade de construção de um estudo é referente ao relacionamento entre teoria e observação, garantindo que o tratamento reflete o construto da causa e que o resultado reflete o construto do efeito. Em outras palavras, a validade de construção refere-se ao quanto o estudo de caso mediu o que deveria medir. As conclusões do estudo foram baseadas na análise da entrevista e no período registrado para as atividades relacionadas à utilização do método. Para minimizar esta ameaça, as transcrições da entrevista foram revisadas por outro pesquisador independente.

A validade externa do estudo refere-se à habilidade de generalizar os resultados a partir da amostra da população. Como discutido previamente, os resultados do estudo de caso não podem ser generalizados.

A confiabilidade é referente ao quanto os dados e a análise são dependentes dos pesquisadores. Para minimizar esta ameaça, todas as perguntas da entrevista foram revisadas, em termos de clareza, por outro pesquisador independente. Além disto, todos os dados coletados foram revisados pelo sujeito do estudo. Com relação à codificação, os códigos gerados foram revisados por outro pesquisador.

5.2.4 Conclusão

Ao analisar as respostas às questões de pesquisa **QP_EC_1** e **QP_EC_2**, pode-se concluir que, da perspectiva do líder de projeto do VIRTUS, sujeito do estudo de caso, a aplicação do método traz os benefícios de *Externalização*, *Apoio ao diagnóstico*, *Melhoria na interpretação das medidas* e *Melhoria na tomada de decisões*.

De acordo com os dados coletados, pode-se concluir que o método apoia o gerenciamento de riscos relacionados ao processo de medição e, consequentemente, ao processo de desenvolvimento. Parte deste benefício deve ser atribuído a utilização de redes Bayesianas, que já teve sua capacidade de auxiliar na gerenciamento de riscos demonstrada em outros estudos [40, 39, 36, 104].

Com relação ao esforço, o necessário esforço de modelagem inicial e da potencial necessidade da coleta manual de dados (*i.e.,* a depender das variáveis escolhidas para o modelo e os dados disponíveis e coletados automaticamente) é minimizado pelo reuso dos modelos construídos durante o projeto. No contexto de múltiplos projetos, a depender do contexto deles, pode-se reutilizar os modelos, reduzindo assim o esforço de modelagem inicial.

Então, à luz do construto de utilidade do TAM, pode-se concluir que há altas chances de haver uma alta intenção de adoção do método proposto. Embora o estudo não possa ser generalizado, as descobertas podem interessar outras empresas com projetos similares à unidade de análise em questão, composta por uma pequena equipe (com 12 membros) que desenvolveu um aplicativo para dispositivos móveis. Vale ressaltar que, além de ter sido útil como uma primeira validação para o presente trabalho de tese, tal estudo consiste em uma importante contribuição pra literatura, uma vez que estudos com foco na interpretação de medidas ainda são escassos.

5.3 Ferramenta SAM

A ferramenta SAM foi implementada em parceria com alunos de graduação vinculados ao laboratório Embedded[3], da UFCG, por meio de um projeto de capacitação em cooperação com uma empresa. A principal função do SAM na perspectiva desta pesquisa é ser uma prova

[3]https://www.embedded.ufcg.edu.br/

de conceito para implementar um software que dê suporte ao método proposto. O SAM possibilita a operacionalização do método proposto nesta pesquisa, facilitando o processo de transferência de tecnologia para a indústria.

A ferramenta SAM foi desenvolvida utilizando a pilha MEAN.JS[4] para desenvolvimento Web. Para realizar as inferências Bayesianas, utilizou-se a *engine* SMILE[5] e o KaizenRNM[6]). No que se segue, apresenta-se uma visão geral da ferramenta, incluindo suas principais funcionalidades e como elas atendem ao método proposto. Mais detalhes com relação à ferramenta SAM são apresentados nos Apêndices A e B.

A arquitetura da ferramenta SAM é apresentada na Figura 5.9. Como pode ser observado, o SAM delega a responsabilidade de se comunicar com as fontes de dados para uma ferramenta de *Business Intelligence*. Na prática, foram realizadas conexões com a ferramenta Pentaho[7]. Por outro lado, em teoria, qualquer ferramenta que exporte os dados via JSON poderia ser conectado facilmente à ferramenta SAM.

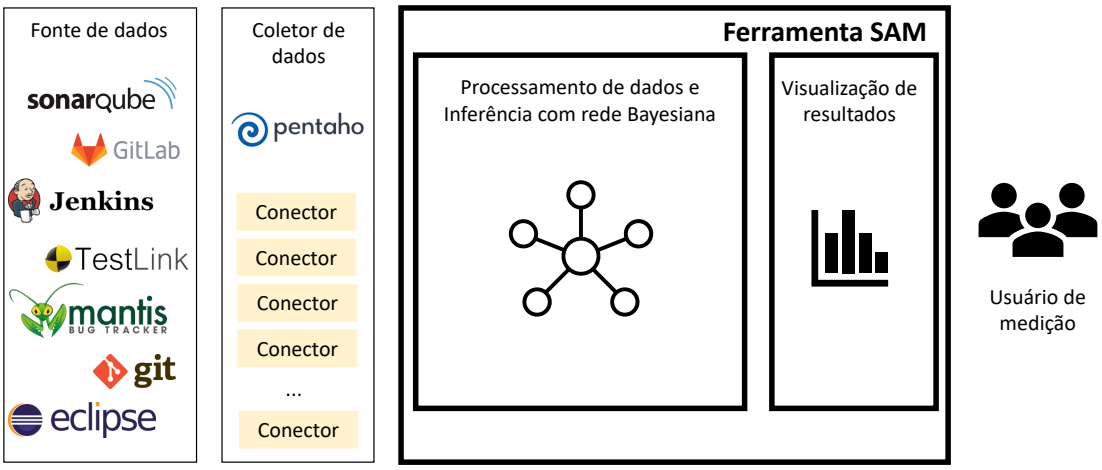


Figura 5.9: Arquitetura da ferramenta SAM.

A ferramenta SAM dá suporte à aplicação do método proposto, incluindo a construção guiada do modelo GQM, a aplicação de critérios de validação de medidas, a construção guiada dos modelos de validação de dados e a configuração de *thresholds*. Para a configu-

[4]http://meanjs.org/
[5]https://www.bayesfusion.com/smile/
[6]https://github.com/isevirtus/KaizenRNM
[7]https://www.hitachivantara.com/en-us/products/data-management-analytics/pentaho-platform.html

ração dos *thresholds*, possibilita-se a conexão de um nó da rede Bayesiana a uma fonte de dados (*e.g.,* ferramenta de gestão de projeto ou de integração contínua). A quantidade de *thresholds* é definida em função do número de estados do nó alvo. Na ferramenta SAM, é possível definir as regras de *thresholds* levando em consideração a escala original do dado a ser coletado. A conexão com a fonte de dados, por meio da ferramenta de *Business Intelligence*, é configurada em cada nó de interesse. Para tal, é necessário que a ferramenta de *Business Intelligence* esteja conectada corretamente com a fonte de dados e exporte os dados via JSON para serem consumidos pela ferramenta SAM. Mais detalhes com relação à ferramenta SAM são apresentados nos Apêndices A e B. O código do SAM está disponível publicamente em repositório hospedado no GitHub[8].

5.4 Grupo de Foco

De acordo com Wallace e Sheetz [144], teorias de aceitação de tecnologia [44, 24, 135, 2, 91, 81, 25, 21, 133, 134, 20, 142, 31], apesar de poderosas no contexto teórico, têm baixa capacidade explicatória. O raciocínio utilizado por Wallace e Sheetz [144] é que, por terem aplicabilidade genérica, ao utilizá-las como propostas originalmente, não é possível identificar as características da tecnologia em questão que influenciam na sua adoção. Desta forma, Wallace e Sheetz [144] propuseram um modelo de aceitação de medidas de software, utilizando como base o TAM [24]. Como discutido previamente, o TAM contém apenas dois construtos exógenos: *Utilidade Percebida* e *Facilidade de Uso*. Em comparação com o TUAUT, a *Utilidade Percebida* do TAM é associada com *Expectativa de Desempenho* e a *Facilidade de Uso* com *Expectativa de Esforço* [142]. Como resultado, Wallace e Sheetz [144] identificaram quatro subdimensões com relação à *Utilidade Percebida* (*i.e., Expectativa de Desempenho, no TUAUT*): *Validade Percebida*, *Prescrição Percebida*, *Independência de Linguagem Percebida* e *Aplicabilidade de Ciclo de Vida Percebida*. Além disso, dado que os instrumentos de coleta de dados para modelos de aceitação de tecnologia tradicionalmente utilizados são questionários, para cada construto, Wallace e Sheetz [144] definiram questões específicas para o contexto de medida de software. Por exemplo, para *Facilidade de Uso* (*i.e., Expectativa de esforço, no TUAUT*), foram definidas cinco questões, dentre elas a questão "Eu

[8]https://github.com/isevirtus/SAM

acredito que a medida é pesada (*i.e.,* tem alto custo) para ser utilizada?".

Para avaliar o potencial de adoção do método proposto no grupo de foco, utiliza-se como base o raciocínio aplicado por Wallace e Sheetz [144], mas, específico para avaliar métodos de medição de software, e não para medida de software. Na ausência de uma teoria de aceitação de tecnologia consolidada e específica para o contexto de interesse (*i.e.,* método de medição de software), utilizou-se como referência para a avaliação da utilidade prática do método proposto o TUAUT e um mapeamento entre as características do método proposto com fatores de sucesso para a implantação efetiva de programas de medição com foco na interpretação de medidas, com base na literatura.

Para viabilizar tal método de avaliação, inicialmente, foram identificados na literatura os fatores de sucesso para programas de medição. Vale salientar que, neste ponto, não foi levado em consideração a relação dos fatores identificados com a interpretação de medidas. Como discutido posteriormente na Seção 5.4.1.3, como parte da metodologia do estudo, praticantes da indústria avaliaram a relevância dos fatores de sucesso identificados na literatura em função da importância para a interpretação de medidas. Desta forma, no pior caso, os fatores de sucesso identificados na literatura para programas de medição que não tem relação com a interpretação de medidas foram descartados posteriormente.

Os fatores de sucesso para programas de medição considerados inicialmente na análise do método proposto foram baseados nos resultados apresentados em Ram *et al.* [107], Hall e Fenton [53] e Staron e Medin [126]. Em Ram *et al.* [107] são apresentados fatores de sucesso para a operacionalização de um programa de medição no contexto ágil. Neste estudo, são identificados onze fatores de sucesso, os quais são comparados com os onze fatores identificados por Hall e Fenton [53], que focou em programas de medição de software em geral, e com os onze fatores identificados por Staron e Medin [126], que focou no uso de longo prazo de programas de medição.

Como discutido em Ram *et al.* [107] há uma interseção entre os fatores identificados pelos três estudos supracitados. Desta forma, para evitar redundância, foram selecionados apenas os fatores identificados por Ram *et al.* [107], por ser o estudo mais recente. Além disso, há fatores identificados por Hall e Fenton [53] e Staron e Medin [126] que são automaticamente satisfeitos ao utilizar um programa baseado em GQM. Como o intuito do estudo é mapear como o método proposto atende aos fatores de sucesso (**QP2**), esses fatores também

foram descartados da análise. Finalmente, um dos fatores identificados por Staron e Medig [126] cita a necessidade do programa de medição ser baseado na ISO/IEC 15939. Como discutido no Capítulo 4, o método proposto é baseado na ISO/IEC 15939, desta forma, não se viu a necessidade de analisar o método proposto com relação a esses fatores, pois, em tese, são automaticamente satisfeitos por definição. Os fatores remanescentes, utilizados no estudo, são apresentados na Tabela 5.1, acompanhados de suas respectivas descrições.

Tabela 5.1: Fatores de sucesso para programas de medição efetivos considerados inicialmente no estudo.

ID	Fator	Descrição	Referência
F1	Disponibilidade dos dados	Os dados necessários para computar as medidas de software do programa estão disponíveis	[107]
F2	Formato dos dados	Os dados estão disponíveis em um formato legível por máquinas permitindo a coleta automática de dados	[107]
F3	Coleta automática dos dados	Os dados são coletados automaticamente	[126, 53]
F4	Alinhamento com o Processo	Os dados são gerados automaticamente durante as atividades do processo de desenvolvimento e as medidas estão alinhadas com o processo de desenvolvimento	[107]
F5	Flexibilidade do Processo	Processo de desenvolvimento flexível o suficiente para acomodar novas práticas para habilitar a operacionalização de medidas com valor	[107]
F6	Operacionalização Incremental	Aplicação de uma abordagem ágil iterativa no programa de medição	[107, 53]

F7	Feedback regular	Ciclos de feedback curtos durante a operacionalização, promovendo aprendizagem e evolução das métricas	[107, 53]
F8	Visão	Uma visão de longo prazo e multi-projeto impulsionando os benefícios técnicos das medidas	[107, 53]
F9	*Champion*	Uma pessoa dedicada para liderar a causa do programa de medição	[107, 53]
F10	Familiaridade	O uso de medidas familiares (*e.g.,* conhecidas) para estabelecer uma confiança inicial	[107, 53]
F11	Fonte de dados da medida	A transparência da ligação de medidas com as correspondentes fontes de dados contendo os dados não tratados	[107, 53]
F12	Valor de fidedignidade	As medidas deve ter uma avaliação quantificando sua fidedignidade	[107, 126, 53]
F13	Medidas base padrão	Medidas base padrão devem ser utilizadas quando aplicável para dar suporte a reuso e benchmarking	[126]
F14	Reuso de medidas base padrão	Os custos da coleta de dados devem ser otimizados e as medidas devem ter como objetivo serem reutilizadas.	[126]
F15	Especificação de medidas e especificação de suas instanciações	As medidas são especificadas em relação ao tipo de entidade mensuradas (*e.g.,* número de arquitetos). Essas medidas são instanciadas para diversos projetos. Essa distinção é importante dado que os sistemas de medição podem ser diferentes por mensurarem projetos distintos (entidades) ou mensurarem projetos e projetos (diferentes tipos de entidades).	[126]

Posteriormente, uma análise foi realizada para mapear os fatores de sucesso apresentados na Tabela 5.1 com os construtos exógenos do TUAUT. Para tal, a pesquisadora realizou o mapeamento e, para reduzir o viés de pesquisador, os resultados foram checados por outro pesquisador. O resultado do mapeamento foi utilizado como o modelo de pesquisa referência inical para avaliar o método proposto. Tal mapeamento é apresentado na Figura 5.10.

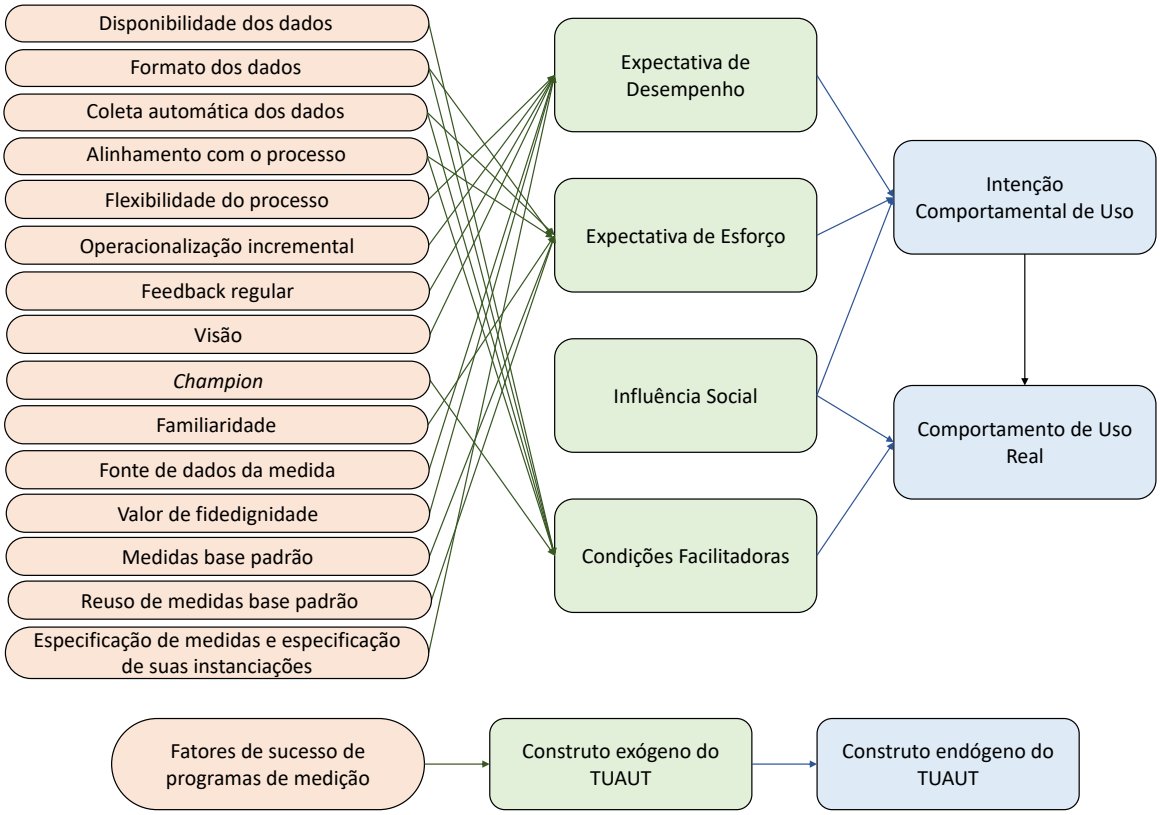


Figura 5.10: Mapeamento dos fatores de sucesso de programas medição e o TUAUT.

Como pode-se perceber, ao analisar a Figura 5.10, nenhum dos fatores de sucesso para programas de medição (ver Tabela 5.1) é relacionado com o construto exógeno *Influência Social*. Além disso, o construto *Expectativa de Desempenho* foi associado com sete fatores de sucesso, *Expectativa de Esforço*, com seis, e *Condições Facilitadoras*, com cinco. Note que há casos nos quais um fator de sucesso foi associado com mais de um construto exógeno do TUAUT. Por exemplo, o fator *Coleta automática dos dados* foi associado com os construtos *Expectativa de Esforço* e *Condições Facilitadoras*, pois a automação da coleta reduz o esforço de execução do programa (*i.e.,* associa-se ao construto *Expectativa de Esforço*) e indica que a organização disponibiliza infraestrutura técnica para a execução do programa

(*i.e.,* associa-se ao construto *Condições Facilitadoras*). Dado isto, para guiar a validação do método proposto, no que diz respeito ao grupo de foco, a questão de pesquisa inicial foi refinada nas seguintes questões de pesquisa:

QP_GF_1 Quais os fatores de sucesso para a implantação efetiva de um programa de medição com foco na interpretação das medidas?

QP_GF_2 O quão bem as características específicas do método proposto satisfazem os fatores de sucesso elencados na **QP_GF_1**?

No que se segue, é apresentado o protocolo do grupo de foco (Seção 5.4.1), os resultados das questões de pesquisa (Seção 5.4.2), a discussão acerca das ameaças à validade (Seção 5.4.3), e as conclusões do estudo (Seção 5.4.4).

5.4.1 Protocolo

O grupo de foco é uma técnica direta e inquisitiva de coleta de dados [122]. Tal técnica é popularmente utilizada para realizar validação estática, com o objetivo de coletar opiniões de profissionais da indústria e construir um relacionamento harmonioso entre pesquisadores e profissionais da indústria [51, 146].

Em um grupo de foco, um grupo de pessoas é reunido com o propósito de focar em um objeto de estudo em particular. É considerado um método de pesquisa muito útil para coletar informações sobre a utilidade de uma solução [122].

Na Engenharia de Software, o grupo de foco é popularmente utilizado como pré-teste de questionários, para garantir a validade do instrumento [68, 48, 90]. Por outro lado, o grupo de foco também é utilizado como um método de pesquisa qualitativo para atender diferentes tipos de questões de pesquisa e objetos de estudo. Por exemplo, em [70], Kontio aplicou o grupo de foco para entender os motivos de organizações buscarem melhorar as práticas de gestão de risco e obter *feedback* com relação ao método Riskit e a ferramenta de software correspondente (eRiskit), com relação ao seu potencial em atender às necessidades das empresas. Para tal, foram executadas três seções de grupo de foco, contendo um total de doze participantes de diversas empresas.

Lehtola, Kaupinnen e Kujala, em [75], aplicaram o grupo de foco para entender os desafios referentes à priorização de requisitos. Para tal, foi executada uma seção de grupo de foco, contendo um total de quatro participantes de duas empresas.

Sunikka, em [131], aplicou o grupo de foco para coletar opiniões de usuários com relação à usabilidade do Website de uma universidade. Para tal, foi executada uma seção de grupo de foco, contendo um total de nove colaboradores da universidade em questão.

Drury, Conboy e Power, em [30], aplicaram o grupo de foco para entender quais as decisões tomadas no contexto de equipes ágeis e os obstáculos, problemas e complexidades inerentes à tomada de decisão neste contexto. Para tal, foi realizado um grupo de foco com 43 participantes.

Em [69, 71], foi apresentado um guia de como aplicar o grupo de foco em Engenharia de Software. Este guia foi utilizado na presente pesquisa para planejar e executar o grupo de foco, sendo este executado em parceria com o VIRTUS, organização parceira que tinha interesse em implantar o método proposto. Como disposto anteriormente, o planejamento inicial seria executar um estudo de caso no VIRTUS, mas, devido ao distanciamento social causado pela pandemia do COVID-19, foi realizado um grupo de foco com profissionais do VIRTUS.

No que se segue, apresentam-se os objetivos do grupo de foco (Seção 5.4.1.1), caracterização da unidade de análise e sujeitos (Seção 5.4.1.2), e procedimentos (Seção 5.4.1.3).

5.4.1.1 Objetivos

Como discutido anteriormente, o objetivo do grupo de foco é coletar a percepção de profissionais da indústria, mais especificamente do VIRTUS, com relação às questões de pesquisa **QP_GF_1** e **QP_GF_2**. Com o intuito de clarificar para os participantes tais objetivos e guiar a seleção dos sujeitos do estudo, os objetivos foram expressados utilizando um *template* GQM. O primeiro objetivo, referente à **QP_GF_1** e apresentado na Tabela 5.2, foi um objetivo de pesquisa descritivo, focado em capturar as principais necessidades da indústria para implantar programas de medição com foco na interpretação de medidas.

O segundo objetivo, referente à **QP_GF_2**, teve como foco discutir o quão importante os participantes consideraram as características principais do método proposto (e o SAM). Este objetivo foi expressado utilizando um *template* GQM, como apresentado na Tabela 5.3

Tabela 5.2: Expressão GQM para o grupo de foco sobre as necessidade da indústria (**QP_GF_1**).

Analisar	A necessidade de programas de medição na indústria
Para	Descrever os principais de fatores de sucesso
Com respeito a	Sua implantação efetiva e com foco na interpetação de medidas
Da perspectiva de	Gerente de qualidade, CTO, e gerente de projeto
No contexto de	Desenvolvimento de software
Porque	Entender a necessidade da indústria é necessária para avaliar se os resultados da pesquisa têm aplicação prática.

Tabela 5.3: Expressão GQM para o grupo de foco sobre a avaliação do método proposto à luz dos fatores de sucesso para implantar sistemas de medição (**QP_GF_2**).

Analisar	O método proposto (e do SAM), especificamente, as quatro etapas que o compõem, a sua natureza iterativa e incremental, a sua operacionalização com rede Bayesiana, e a sua integração com fontes de dados.
Para	Avaliar o valor percebido
Com respeito a	Os benefícios de sua aplicação à luz dos fatores de sucesso para implantação efetiva de um sistema de medição
Da perspectiva de	Gerente de qualidade, CTO, e gerente de projeto
No contexto de	Desenvolvimento de software
Porque	Entender a necessidade da indústria com relação às características do método proposto reflete o seu potencial benefício na prática.

Tabela 5.4: Dados demográficos dos participantes do grupo de foco.

ID	**Papel**	**Gênero**	**Experiência na Indústria de Software**	**Experiência com Sistemas de Medição de Software**
P1	Gerente de projeto	Masculino	14 anos	14 anos
P2	Gerente de qualidade	Masculino	14 anos	10 anos
P3	Gerente de projeto	Masculino	11 anos	4 anos
P4	Gerente de projeto	Masculino	15 anos	5 anos

5.4.1.2 Caracterização da unidade de análise e sujeitos

O grupo de foco foi realizado com quatro profissionais do VIRTUS. Os participantes do grupo de foco foram selecionados em função da sua importância nas tomadas de decisão e experiência (*i.e.,* amostragem intencional), como ilustrado na Tabela 5.4.

Como discutido por Morgan [92], quando o tópico do grupo de estudo é complexo e há o interesse de coletar comentários detalhados, é recomendável ter poucos participantes no grupo de foco. Além disso, para definir o número de participantes, foi levado em consideração o fato de que a sessão seria realizada de forma síncrona e remota utilizando-se de uma ferramenta de comunicação de vídeo pela Internet (*i.e., Google Meet*[9]).

5.4.1.3 Procedimentos

Dado que um grupo de foco só é útil quando se estuda conceitos que podem ser compreendidos por participantes bem informados, em um período limitado, o primeiro passo para a realização dessa dinâmica foi preparar um vídeo apresentando uma visão geral do método proposto e do SAM para ser enviado aos participantes com antecedência. Além disso, uma lista com os fatores de sucesso para programas de medição efetivos, baseada na literatura (ver

[9]https://meet.google.com/

Tabela 5.1), também foi compartilhada previamente com os especialistas. Como a tarefa de compreender o método e discutir **QP_GF_1** e **QP_GF_2** pode ser considerada complexa, reduze-se a complexidade ao ter os participantes iniciando o grupo de foco já com uma visão geral das principais características do método proposto e de fatores de sucesso para programas de medição efetivos. Vale ressaltar que o Termo de Consentimento Livre e Esclarecido (TCLE) relacionado ao estudo foi enviado para os participantes.

Para facilitar a condução do grupo de foco, foi convidado um moderador com dez anos de experiência facilitando tais sessões e com experiência em relacionamentos academia-indústria. Vale salientar que o moderador já tinha um relacionamento profissional tanto com a pesquisadora quanto com os participantes do estudo. Desta forma, a pesquisadora teve como papel, durante o grupo de foco, clarificar questões referentes ao método proposto e ao SAM e tomar notas das discussões.

O grupo de foco foi planejado levando em considerações as diretrizes apresentadas por Kontio, Bragge e Lehtola [69] e McQuarrie e McIntrye [83]. Como premissa para conduzir o grupo de foco, levou-se em consideração que a duração dele deveria ser o suficiente para possibilitar a apresentação aprofundada do método proposto e do SAM, bem como a discussão acerca das questões de pesquisa **QP_GF_1** e **QP_GF_2**. Desta forma, a estrutura dele foi dividida em cinco etapas: (i) abertura (10 minutos), (ii) avaliação da **QP_GF_1** (1 hora), (iii) exposição do método proposto (30 minutos), (iv) avaliação da **QP_GF_2** (1 hora), e (v) fechamento (20 minutos).

Na (i) abertura, os objetivos do grupo de foco (ver Tabelas 5.2 e 5.3) foram apresentados para os participantes. Além disso, a estrutura do evento e as regras de colaboração foram detalhadas. Finalmente, foi reforçado que a informação compartilhada durante a sessão seria gravada e analisada e faria parte dos artefatos de pesquisa. Por outro lado, foi informado que as identidades dos participantes não seriam divulgadas.

Na etapa seguinte (*i.e.,* avaliação da **QP_GF_1**), com o objetivo de reduzir a carga cognitiva nos participantes, o moderador apresentou para eles uma lista de fatores de sucesso para programas de medição de software efetivos (*i.e.,* a mesma lista enviada antes da sessão). Após a análise dos fatores, foi reservado um período para que os participantes colaborassem para adicionar fatores não contidos na lista, mas que eles julgassem relevantes para o contexto do VIRTUS. Posteriormente, utilizou-se a técnica *dot voting*, muito comum em tarefas

de tomada de decisão [29, 63], na qual cada participante teve a oportunidade de votar nos fatores considerados por ele como *must have* para otimizar a interpretação das medidas. Cada participante teve direito a n votos, onde $n = \lfloor 0,4 * (NumeroDeFatoresDeSucesso) \rfloor$. Utilizando-se de uma abordagem *round-robin*, os participantes alternaram-se em realizar a votação e explicar o raciocínio por trás dela. Como resultado, ao final da etapa (ii) avaliação da **QP_GF_1**, tem-se uma lista dos principais fatores de sucesso para um programa de medição efetiva de acordo com a percepção dos participantes, com foco na interpretação de medidas. Os resultados referentes a esta etapa são discutidos na Seção 5.4.2.1.

Posterioremente, foi dado início a etapa (iii), exposição do método proposto, na qual a pesquisadora realizou uma apresentação com detalhes do método proposto e do SAM, com exemplos. Durante a apresentação, foi pedido para os participantes anotarem suas dúvidas, e aguardarem o final dela para realizar as perguntas.

No próximo passo, (iv) avaliação da **QP_GF_2**, inicialmente, foi realizada uma pergunta aberta para todos os participantes: "O que acharam?".

Posteriormente, foi pedido para os participantes, utilizando-se de uma abordagem *round-robin*, classificarem cada uma das principais características do método proposto por meio da técnica *termômetro de humor* [29], na qual cada participante julgou cada característica em função de um espectro de possibilidades indo de *satisfeito* até *insatisfeito*. Na Figura 5.11 ilustra-se o estado inicial do quadro de registro do *termômetro de humor*.

Para possibilitar a coleta de dados por meio do quadro do *termômetro de humor*, um arquivo foi compartilhado utilizando uma ferramenta de colaboração na nuvem, na qual todos os participantes tiveram acesso simultâneo. Desta forma, uma cor foi alocada para cada participante e cada um utilizou a sua para avaliar as características em questão. Ao avaliar cada característica, foi solicitado que os participantes justificassem tal avaliação com o intuito de gerar um entendimento mais aprofundado da sua percepção. Durante as discussões, a pesquisadora interrompeu os participantes para clarificar qualquer característica que não tivesse sido completamente compreendida por eles.

Após todos os participantes terem tido a oportunidade de compartilhar a sua percepção e terem obtido um entendimento mais aprofundado do método proposto e do SAM, passou-se para a última atividade da etapa (iv). Nesta atividade, teve-se como objetivo construir um mapa conceitual relacionando os fatores de sucesso mais importantes definidos na etapa (ii)

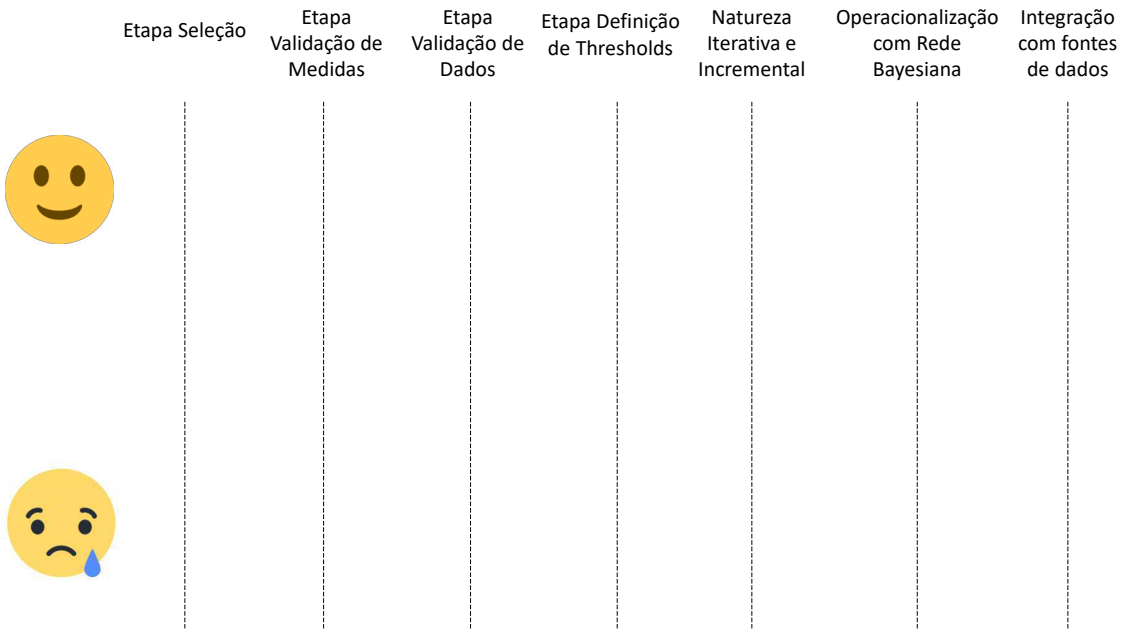


Figura 5.11: Estado inicial do quadro de termômetro de humor utilizado para coleta de dados no grupo de foco.

com as características principais do método proposto e do SAM. O objetivo desta etapa foi ter um entendimento explícito com relação à percepção dos participantes em como o método proposto e o SAM atendem aos fatores de sucesso discutidos na etapa (ii). O resultado desta etapa é discutido na Seção 5.4.2.2.

Finalmente, para encerrar o grupo de foco, na etapa (v), foi solicitado para que cada participante sugerisse alguma oportunidade de melhoria para o método proposto, antes de finalizar o evento. Todos os dados qualitativos registrados durante o estudo foram analisados por meio da análise temática [23].

5.4.2 Resultados

No que segue, são apresentados os resultados coletados durante o estudo de grupo e classificados por questão de pesquisa. Os resultados relacionados à **QP_GF_1** são apresentados na Seção 5.4.2.1 e os relacionados à **QP_GF_2**, na Seção 5.4.2.2.

5.4.2.1 Fatores de sucesso para programas de medição, com foco na interpretação de medidas (QP_GF_1)

Na **QP_GF_1** foi explorado quais os fatores de sucesso para que se tenha um programa de medição efetivo, com foco na interpretação de medidas. Inicialmente, foi discutido com os participantes se a lista de fatores apresentadas na Tabela 5.1 é completa e exaustiva, cobrindo todos os fatores que eles julgam serem relevantes para ter sucesso na aplicação de um programa de medição. Na discussão, foi levantado um novo fator por P2: *Cultura de medição*. Tal fator foi associado com o construto exógeno *Influência Social* do TUAUT, pois a presença de tal cultura tem como consequência que o usuário da medição perceba a importância que os demais membros da organização dão para o programa de medição.

Na Figura 5.12 é apresentado o resultado da votação dos participantes do grupo de foco com relação aos fatores de sucesso mais relevantes para a implantação efetiva de um programa de medição com foco na interpretação de medidas. Na Figura 5.12, cada círculo representa um voto, sendo os círculos de cor laranja referentes ao participante P1; azul, P2; amarelo, P3; e verde, P4.

Na Figura 5.12, nota-se que a quantidade de votos (*i.e.,* círculo) produz um resultado similar a um mapa de calor. Por outro lado, a quantidade de votos em um fator não necessariamente indica a sua importância. Por exemplo, ao justificar os seus votos, P2 argumentou que, para ele, a importância do fator referente ao *Valor de fidedignidade* é óbvia e que ele votou em fatores que, segundo a experiência dele, influenciam o programa a produzir medidas com boa fidedignidade. Desta forma, os resultados apresentados na Figura 5.12 não devem ser utilizados para priorizar os fatores de sucesso.

Como resultado da discussão, notou-se uma maior preocupação de todos os participantes com relação ao esforço em adotar a medida, a necessidade de implantar o programa de medição de forma evolutiva e a fidedignidade das medidas. Por exemplo, com relação ao esforço em adotar a medida (*i.e.,* referente ao construto exógeno *Expectativa de Esforço* do TUAUT), P1 citou que, para definir o programa de medição, é necessário tomar decisões que resultem em menos *overhead*. P2 e P3 citaram a importância de se reduzir o custo operacional do programa de medição. P4 citou a importância de fatores como *Disponibilidade de dados*, *Coleta automática dos dados* e *Alinhamento com o processo* para *"reduzir esforço e*

Figura 5.12: Resultado da votação dos fatores de sucesso para um programa de medição efetivo, com foco na interpretação de medidas. O fator em amarelo significa que ele foi elicitado duranta o grupo de foco.

erros". Com relação à importância do fator *Disponibilidade dos dados*, tal raciocínio está em conformidade com os resultados apresentados em [107], no qual discute-se a importância de iniciar o programa de medição a partir dos dados disponíveis, e só depois pensar em adotar medidas baseadas em novos dados. Complementando tal informação, P4 relatou que o *modus operandi* do VIRTUS com relação às medidas é tentar trabalhar com os dados e medidas existentes, ou seja, eles tentam definir novas medidas derivadas, mas em função do que as ferramentas utilizadas já disponibilizam.

Com relação à necessidade de implantar o programa de medição de forma evolutiva, que foi relacionado com os fatores *Operacionalização incremental* e *Feedback regular*, P1 citou que *"não acredita em revolução"* e P2 que é essencial aplicar o processo de *"implantar-feedback-ouvir"*. Houve um consenso de que é necessária a implantação incremental do programa e da melhoria contínua dele a partir do feedback regular. P3 destacou a necessidade da *"consolidação incremental das medidas"*. Um porém foi levantado por P4, citando que pode ser possível uma implantação mais brusca do programa de medição caso ele seja reaproveitado de um contexto similar (*e.g.,* outra empresa ou projeto com características similares), por já ter sido validado previamente.

Sobre o valor de fidedignidade, foi discutido que ter confiança nas medidas coletadas e entendê-las é fundamental para conseguir interpretá-las e tomar decisões. Esse entendimento, aliado à transparência do processo de medição, promove a melhoria do processo de coleta dos dados. P3 citou a necessidade de *conhecer o porquê de utilizar determinada medida*. Reforçando o argumento de P3, P2 destacou que implantar a *Cultura de medição* é necessária para o sucesso do programa, e que para tal é essencial que as partes interessadas entendam o porquê das medidas e o que elas representam.

Devido à ausência de dados para dar suporte à priorização ou causalidade entre os fatores, como discutido acima, utilizam-se os dados apenas para separar quais os fatores de maior relevância no sucesso de programas de medição efetivos, dos que os participantes julgaram não ser tão importantes. Foram considerados relevantes para este estudo os fatores com pelo menos um voto. Tais fatores foram utilizados como base para analisar a efetividade do método proposto.

***QP_GF_1**: Os fatores de sucesso para programas de medição, com foco na interpretação de medidas são: a disponibilidade dos dados, coleta automática dos dados, alinhamento com o processo, flexibilidade do processo, operacionalização incremental, feedback regular, visão, Champion, familiaridade, fonte de dados da medida, valor de fidedignidade e cultura de medição.*

5.4.2.2 Mapeamento entre as características do método proposto e fatores de sucesso para programas de medição com foco na interpretação de medidas (QP_GF_2)

O mapeamento entre as características do método proposto e fatores de sucesso para programas de medição como foco na interpretação de medidas serviu como informação para inferir a utilidade prática do método proposto. Para tal, após a apresentação do método, foram analisadas as respostas à pergunta *"O que acharam?"*. Inicialmente, P4 julgou o método complexo por achar que, dada a experiência dele, a implantação de um programa de medição já é custosa, e o método proposto adiciona etapas que ele nunca tinha executado explicitamente, como a *Validação de medidas* e *Validação de dados*. Por outro lado, P1, P2, e P3 concordaram no argumento de que, implicitamente, eles já executavam, com menos rigor, as quatro etapas do método proposto, e destacaram a importância da formalização proposta no método. P1 argumentou que nunca tinha considerado executar a etapa de *Validação de medidas* explicitamente, pois ele já escolhia medidas que, de certa forma, ele confiava. Por outro lado, ele não soube explicar o porquê dele confiar nas medidas utilizadas. Além disso, todos os participantes concordaram que um dos principais benefícios do método proposto é formalizar a execução de tais etapas e registrar o conhecimento de especialistas na rede Bayesiana. P1 destacou que tal formalização facilita o processo de *trocar pessoas*, se necessário. Como consequência, P3 argumentou que o método proposto auxilia na *operacionalização* do programa de medição.

Posteriormente, as impressões dos participantes em relação a cada características do método proposto foram registradas no quadro de termômetro de humor. Tal quadro preenchido com os dados registrados pelos participantes é ilustrado na Figura 5.13.

Para interpretar os resultados registrados no quadro de termômetro (Figura 5.13), é necessário levar em consideração os argumentos apresentados pelos participantes ao realizar o preenchimento dele.

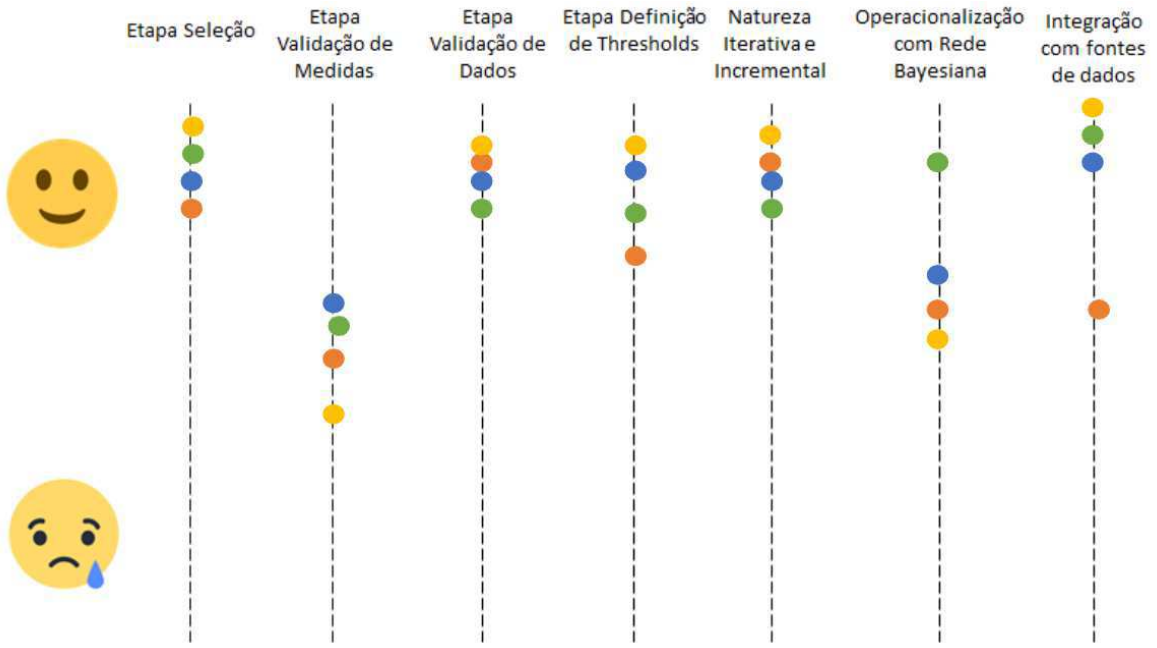


Figura 5.13: Quadro de termômetro de humor preenchido.

Com relação à primeira etapa (*i.e., Seleção*), houve um consenso com relação a concordar que o uso do GQM para realizá-la é positivo. Por outro lado, P2 argumentou que, na opinião dele, o uso das folhas de abstração adiciona uma complexidade desnecessária ao método.

Sobre a etapa *Validação de medidas*, P1 argumentou que acredita ser a etapa de menor importância, pois assume-se que, ao selecionar as medidas, elas são válidas. Por outro lado, P2 argumentou que a etapa é importante por ajudar a formalizar a medida, mas é necessário apresentá-la com uma linguagem mais prática.

Houve um consenso entre os participantes sobre a importância das etapas *Validação de dados*, *Definição de thresholds* e a *Natureza iterativa e incremental* do método. Com relação à *Validação de dados*, houve um consenso de que é a etapa mais impactante do método, pois é essencial que o processo de coleta de dados garanta a *"acurácia da medição"*, como dito por P2. Com relação à *Definição de thresholds*, P1 observou que ele prefere interpretar a medida dinamicamente, ou seja, em função da evolução do dado coletado, e não apenas considerando o valor em um instante de tempo específico. P3 destacou a importância do processo de feedback para esta etapa.

Com relação à operacionalização do método com a rede Bayesiana, P1, P2 e P4 argumentaram que era difícil julgar sem executar na prática o processo de construção e utilização

dos modelos com redes Bayesiana. Por outro lado, P2 comentou que considerou promissora a utilização de redes Bayesianas para que se tenha medidas válidas. P3 complementou destacando que achou o uso de redes Bayesianas *"bastante válido"* para alcançar o objetivo de ter medidas com maior interpretação.

Finalmente, com relação à integração com fontes de dados, apenas P1 não se sentiu confortável em classificar essa característica como primordial, porém comentou que era pelo fato de não ter utilizado a ferramenta SAM. Os demais participantes concordaram que a integração do SAM com as fontes de dados por meio de ferramentas de *Business Intelligence* como sendo uma ótima decisão de arquitetura, viabilizando a adoção do método proposto na prática.

Após o registro das informações no quadro de termômetro de humor, os participantes, de forma independente, relacionaram as principais características do método proposto com os fatores de sucesso chave para um programa de medição efetivo, como foco na interpretação de medidas. Na Figura 5.14 é apresentado o mapa conceitual resultante do consenso entre os participantes. Como pode ser observado na Figura 5.14, de acordo com a perspectiva dos participantes do grupo de foco, o método proposto satisfaz os principais fatores de sucesso para programas de medição efetivos com foco na interpretação de medidas.

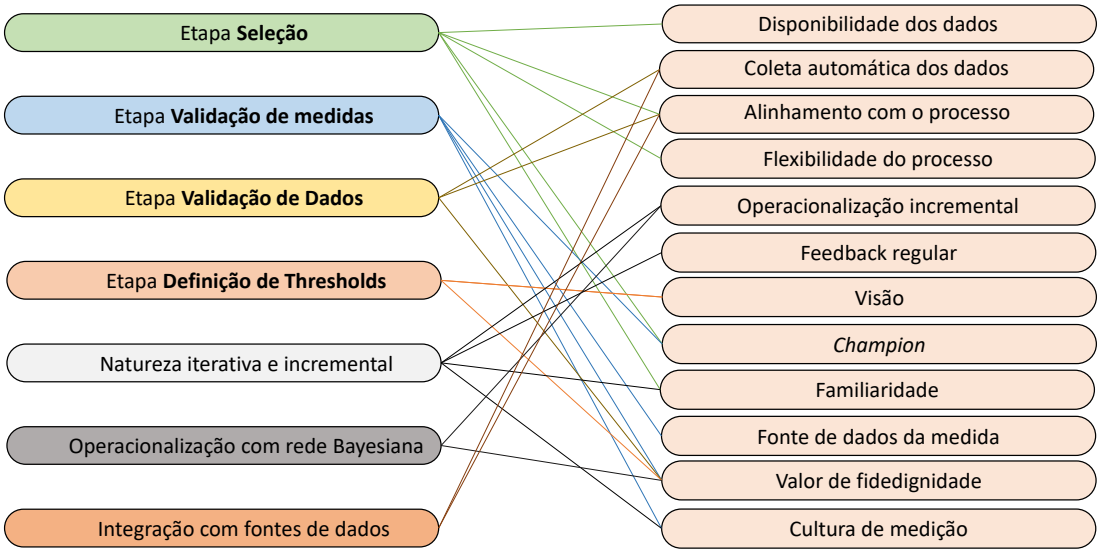


Figura 5.14: Mapa conceitual demostrando quais fatores de sucesso para programas de medição efetivos com foco na interpretação de medidas são satisfeitos pelo método proposto.

***QP_GF_2**: O método proposto satisfaz todos os fatores de sucesso elencados em **QP_GF_1** (i.e., disponibilidade dos dados, coleta automática dos dados, alinhamento com o processo, flexiblidade do processo, operacionalização incremental, feedback regular, visão, Champion, familiaridade, fonte de dados da medida, valor de fidedignidade, e cultura de medição.)*

Ao final do evento, como oportunidade de melhoria do método proposto, P3 sugeriu a definição de catálogos. Tais catálogos seriam reutilizados para reduzir o esforço e minimizar os erros na execução das etapas do método. Este reúso teria um impacto no construto exógeno *Expectativa de esforço* do TUAUT, que, por sua vez, teria um impacto direto na *Intenção Comportamental de Uso* e indireto no *Comportamento de Uso Real*.

5.4.3 Ameaças à validade

As ameaças à validade dos resultados referentes ao grupo de foco são discutidas à luz da classificação apresentada por [149]. Como o grupo de foco é um estudo qualitativo, pode-se considerar as ameaças à validade interna, de construção, externa e de confiabilidade.

Em um grupo de foco, o intenso envolvimento do pesquisador e dos sujeitos podem influenciar a validade interna do estudo. Para minimizar essa ameaça, foi convidado para moderar o grupo de foco um profissional experiente com tal evento. Foi observado durante o evento que ele foi estruturado de forma que todos os participantes tiveram oportunidades iguais de contribuição. Além disso, apesar do evento ter tido uma duração de três horas, devido à dinâmica da sua estrutura, intercalando tipos de atividades distintos, acredita-se que o possível efeito do cansaço dos participantes foi mínimo, não influenciando os resultados. Finalmente, não foi observado desproporcionalidade com relação à sobreposição da opinião de um participante sobre as opiniões dos demais, nem problemas relacionados à timidez ou excesso de cautela no compartilhamento de opiniões.

No que se refere à validade de construção, em um grupo de foco, o pesquisador e os participantes foram os instrumentos do estudo em si. Primeiramente, para minimizar os efeitos da sobrecarga cognitiva durante o grupo de foco, foi disponibilizado para os participantes material em texto e vídeo descrevendo os fatores de sucesso para programas de medição (ver Figura 5.1) e uma visão geral do método proposto. Além disso, foram convidados apenas participantes com larga experiência na indústria (*i.e.,* amostragem intencional), como pode

ser percebido pelos dados apresentados na Tabela 5.4.

Como em um grupo de foco os comentários e interações podem ser consideradas as medidas em si do estudo, há ameaças que são fortemente relacionadas tanto à validade interna como de construção. Neste estudo, a representatividade das informações coletadas durante o grupo de foco é consequência do entendimento dos participantes com relação ao método proposto, os fatores de sucesso para programas de medição efetivos com foco na interpretação de medidas, e as atividades executadas, incluindo: o *dot voting*, o preenchimento do quadro de termômetro de humor e o mapeamento entre as características do método proposto e os fatores de sucesso. Como discutido previamente, para mitigar tais ameaças foram utilizadas três estratégias: a disponibilização prévia de material de apoio, a seleção de participantes experientes com medição de software e a presença de um moderador experiente para conduzir o grupo de foco.

Um potencial viés do estudo está relacionado com a atitude dos participantes. De acordo com Dwivedi [31], a atitude dos usuários com relação à tecnologia em questão influencia a sua intenção de uso e, principalmente, o seu uso de fato. Neste contexto, considera-se o grau de inovação dos participantes como sendo alta, dado que trabalham em um núcleo de pesquisa, desenvolvimento e inovação. Com isto, entende-se que eles têm uma predisposição com relação a adotar produtos inovadores, como discutido em Oliveira *et al.* [99]. Então, acredita-se que essa característica inovadora por parte dos participantes do estudo limita a generalização dos resultados para outros contextos. Nota-se que tal observação seria factível independente de uma validação estática (*i.e.,* como foi realizada) ou dinâmica (*i.e.,* como planejado inicialmente, mas cancelado devido à pandemia do COVID-19).

A validade externa do estudo refere-se a habilidade de generalizar os resultados a partir da amostra da população. Como todo estudo qualitativo, utilizou-se uma amostra pequena e homogênea, o que dificulta a generalização dos resultados. Por outro lado, dado que participaram do estudo profissionais experientes do VIRTUS, pode-se assumir que os resultados são adequados para o VIRTUS e outras organizações com contexto similar. Na Seção 5.4.1.2, foram apresentados detalhes do contexto do VIRTUS para possibilitar a generalização dos resultados do estudo para outras organizações com contexto similar. Com relação à confiabilidade, os códigos na análise temática gerados foram revisados por outro pesquisador.

5.4.4 Conclusão

Ao analisar as respostas às questões de pesquisa **QP_GF_1** e **QP_GF_2**, pode-se concluir que, da perspectiva dos gerentes de projeto e gerente de qualidade do VIRTUS, a aplicação do método é causa para os principais fatores de sucesso para um programa de medição efetivo com foco na interpretação de medidas: disponibilidade dos dados, coleta automática dos dados, alinhamento com o processo, flexibilidade do processo, operacionalização incremental, *feedback* regular, visão, *Champion*, familiaridade, fonte de dados da medida, valor de fidedignidade, e cultura de medição.

Analisando o mapa conceitual apresentado na Figura 5.14, o mapeamento entre os fatores de sucesso com o construto do TUAUT (ver Figura 5.10) e, considerando que o fator *Cultura de medição* é mapeado com o construto exógeno *Influência Social* do TUAUT, pode-se concluir que, como consequência lógica, o método proposto tem alto potencial de sucesso em sua adoção na prática.

Além disso, como reportado na Seção 5.4.2.2, os resultados do grupo de foco indicam que um dos benefícios da utilização do método proposto é de formalizar o conhecimento do especialista, ou seja, os modelos (*i.e.,* redes Bayesianas) criados servem como um sistema especialista ou um sistema baseado em conhecimento, dando transparência e auxiliando no entendimento das medidas por todas as partes interessadas. Tal formalização auxilia na gestão do conhecimento, dando mais transparência ao programa. Com isto, pode-se afirmar que os participantes do grupo de foco concluíram que o método proposto tem como benefício a *Externalização*.

Finalmente, pode-se relacionar tal benefício com a melhoria proposta por P3 relacionada à definição de catálogos. Como a utilização do método proporciona a *Externalização*, o conhecimento torna-se explícito por meio das redes Bayesianas construídas. Tais redes podem ser reutilizadas em nível organizacional, servindo como catálogo, e assim reduzir o esforço e minimizar os erros na adoção do método. Além disso, vale salientar que a utilização de redes Bayesianas na Engenharia de Software é ampla [89], tendo aplicações no gerenciamento de requisitos [28], gerenciamento de qualidade [139] e estimativa de esforço [85]. Assim, ao adotar o método, pode-se reutilizar uma gama de redes Bayesianas já disponíveis para adaptá-las ao contexto da organização, seguindo os passos descritos nele.

Então, em função do construto de utilidade do TUAUT, pode-se concluir que há chances de haver uma alta intenção de adoção do método proposto. Por outro lado, deve-se levar em consideração as ameaças à validade elencadas na Seção 5.4.3 que limitam a capacidade de generalização do resultado, dentre outras limitações. Além disso, deve-se levar em consideração que, devido à pandemia do COVID-19, apenas uma validação estática do método foi realizada, o que acentua a incerteza relacionada às conclusões do grupo de foco.

5.5 Discussão dos Resultados

Nesta seção, procura-se responder à questão de pesquisa **QF** em função dos resultados obtidos no estudo de caso e no grupo de foco. Ambos os estudos executados são qualitativos e tiveram objetivo similar, voltado à avaliar a utilidade prática do método proposto. Para tal, o estudo de caso utilizou um construto de utilidade mais simples, em comparação ao TUAUT, o TAM. É importante ressaltar que o estudo de caso foi realizado validando apenas as etapas de *Seleção de Medidas* e *Validação de Dados*, pois as demais etapas ainda não estavam definidas. Como resultado foram identificados os benefícios de *Externalização*, *Apoio ao diagnóstico*, *Melhoria na interpretação das medidas* e *Melhoria na tomada de decisões*. Além disso, foi observado um esforço inicial considerável para aplicar o método proposto, o que é natural para a implantação de qualquer programa de medição. Por outro lado, foi observado que o esforço de utilização dos modelos construídos foi baixo, mesmo no caso de coleta manual dos dados. Vale ressaltar que tal custo, principalmente no contexto de haver coleta manual dos dados, aumenta proporcionalmente à quantidade de dados necessários a serem coletados manualmente.

No grupo de foco, foi observado que o método proposto influencia positivamente nos quatro construtos exógenos do TUAUT: *Expectativa de desempenho*, *Expectativa de esforço*, *Influência social* e *Condições facilitadoras*. Tal fenômeno decorreu do processo de *Externalização* que deixa explícito para os usuários da medição os fatores relacionados à incerteza na tomada de decisão por meio da formalização em uma rede Bayesiana, e sua natureza iterativa e incremental.

Destaca-se o fato do grupo de foco confirmar resultados do estudo de caso no que tange o método proposto proporcionar os benefícios da *Externalização*, *Melhoria na interpretação*

das medidas e *Melhoria na tomada de decisões*. Além disso, é importante observar que o grupo de foco não confirmou o benefício de *Apoio ao diagnóstico*, observado no estudo de caso. Por outro lado, vale salientar que os participantes do estudo de caso não tiveram a oportunidade de utilizar o método na prática em seus projetos, construindo e interagindo com os modelos construídos. Além disso, o apoio ao diagnóstico é uma característica de modelos causais, em especial, de redes Bayesianas [40, 5].

Desta forma, apesar das limitações de ter sido executado um estudo de caso com apenas uma unidade de análise (*i.e.,* um sujeito) e um grupo de foco, e em função do construto do TUAUT, os resultados de ambos estudos indicam que há altas chances do método proposto ser adotado com sucesso no VIRTUS e em ambientes com contexto similar. Adiciona-se a isso o fato do desenvolvimento da ferramenta SAM demonstrar a viabilidade da operacionalização dos modelos conceituais inerentes ao método proposto.

Capítulo 6

Considerações Finais

Neste trabalho de tese, apresentou-se um método iterativo e incremental para a medição de software com foco na interpretação de medidas. O objetivo de tal método é possibilitar aos usuários de medição uma assertividade maior na tomada de decisão a partir de uma interpretação mais correta das medidas utilizadas. Como primeiro passo para o alcance deste objetivo, foi realizada uma pesquisa na ISO/IEC 15939:2007, que refere-se ao processo de medição de Engenharia de Software. A pesquisa teve como intuito a identificação de atividades e tarefas que têm um impacto maior na interpretação das medidas. Como resultado, foram definidas as seguintes etapas: seleção de medidas, validação de medidas, validação de dados e definição de *thresholds*. Com isto, respondeu-se a questão de pesquisa **QP1**.

Uma vez identificadas as etapas que compõem o método, realizaram-se várias pesquisas na literatura a fim de levantar o estado da arte de cada uma das etapas descritas. Com isto, foram extraídos trabalhos relevantes de cada área, o que possibilitou um entendimento maior sobre como operacionalizar cada etapa do método proposto. Neste processo, confirmou-se a hipótese de utilização de redes Bayesianas como uma forma de integrar as etapas, modelando a incerteza envolvida em cada uma delas. Assim, respondeu-se as questões de pesquisa **QP2** e **QP3**.

O método proposto foi validado mediante a realização de um estudo de caso e de um grupo de foco com praticantes da indústria. O estudo de caso foi realizado em parceria com um aluno de mestrado do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande. Dado isto, uma vez que as etapas de validação de medidas e definição de *thresholds* ainda estavam sendo definidas, o estudo de caso contem-

plou uma versão parcial do método proposto, contendo apenas as etapas *Seleção de Medidas* e *Validação de Dados*.

O referente estudo de caso contemplou uma unidade de análise, que correspondeu a um projeto de desenvolvimento de software, com 12 membros, sendo 2 líderes de projetos, 8 desenvolvedores e 2 testadores. O grupo de foco, por sua vez, foi realizado com a participação de um moderador e quatro profissionais do VIRTUS. Este estudo seguiu uma metodologia baseada no arcabouço da Teoria Unificada de Aceitação e Uso de Tecnologia (TUAUT) [142]. Mais especificamente, utilizou-se como base para avaliação da utilidade prática do método proposto o TUAUT e um mapeamento entre as características do referido método com fatores de sucesso para implantação efetiva de método de medição apresentados na literatura.

Os resultados de ambos os estudos indicam que o método apresentado neste trabalho de tese é promissor no sentido de auxiliar usuários de medição na tomada de decisão, uma vez que ele proporciona um melhor entendimento acerca das medidas utilizadas, dos fatores que influenciam estas medidas e, consequentemente, uma melhor interpretação dessas medidas, o que possibilita uma tomada de decisão mais assertiva. Adiciona-se a isto o fato do desenvolvimento da ferramenta SAM demonstrar a viabilidade de operacionalização dos modelos conceituais inerentes ao referido método.

Contudo, é importante considerar alguns fatores limitantes da pesquisa, como: a necessidade de um analista e/ou de um guru de medição, sendo este último necessário principalmente no que se refere à etapa de validação de medidas; a necessidade de um conhecimento prévio de redes Bayesianas, uma vez que esta foi a técnica utilizada para viabilizar a execução do método; a necessidade de um conhecimento prévio de lógica *Fuzzy*, dado que a etapa de definição de *thresholds* utiliza-se desta lógica; e, por fim, tem-se a limitação referente à validação do método proposto, uma vez que o estudo de caso realizado contemplou apenas uma unidade de análise e que o grupo de foco consistiu em uma única sessão com quatro participantes e um moderador.

Esta pesquisa possibilitou a identificação de vários trabalhos futuros, como: a necessidade de uma investigação sobre a prioridade e causalidade entre os fatores de sucesso de um programa efetivo de medição, tendo como foco a interpretabilidade de medidas; a validação de uma taxonomia para classificação de *thresholds*; uma investigação sobre *thresholds* dinâ-

micos; a validação da Teoria Unificada de Aceitação e Uso de Tecnologia (TUAUT) [142] como guia para o processo de validação de um método (ou programa) de medição; validação dinâmica de longo prazo com o método proposto, utilizando o protótipo[1]; uma documentação mais prática a respeito da validação de medidas; elaboração de um catálogo com a formalização das etapas, bem como com um descritivo da aplicação do método considerando um projeto real do VIRTUS.

[1] https://github.com/isevirtus/SAM

Bibliografia

[1] Moataz A Ahmed and Zeeshan Muzaffar. Handling imprecision and uncertainty in software development effort prediction: A type-2 fuzzy logic based framework. *Information and Software Technology*, 51(3):640–654, 2009.

[2] Icek Ajzen et al. The theory of planned behavior. *Organizational behavior and human decision processes*, 50(2):179–211, 1991.

[3] Tiago L Alves, Christiaan Ypma, and Joost Visser. Deriving metric thresholds from benchmark data. In *Software Maintenance (ICSM), 2010 IEEE International Conference on*, pages 1–10. IEEE, 2010.

[4] Scott Ambler. *Agile modeling: effective practices for extreme programming and the unified process*. John Wiley & Sons, 2002.

[5] Ronald D Anderson and Gyula Vastag. Causal modeling alternatives in operations research: Overview and application. *European Journal of Operational Research*, 156(1):92–109, 2004.

[6] Vard Antinyan, Miroslaw Staron, Anna Sandberg, and Jörgen Hansson. Validating software measures using action research a method and industrial experiences. In *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*, pages 1–10, 2016.

[7] L Bruce Archer. *The structure of design processes*. PhD thesis, Royal College of Art, 1968.

[8] Victor R Basili. Software modeling and measurement: the goal/question/metric paradigm. 1992.

[9] Victor R Basili, Lionel C. Briand, and Walcélio L Melo. A validation of object-oriented design metrics as quality indicators. *IEEE Transactions on software engineering*, 22(10):751–761, 1996.

[10] V.R. Basili and D.M. Weiss. A methodology for collecting valid software engineering data. *Software Engineering, IEEE Transactions on*, SE-10(6):728–738, Nov 1984.

[11] Barry Boehm. Software engineering is a value-based contact sport. *IEEE Software*, 19(5):95–96, 2002.

[12] Alexandre Boucher and Mourad Badri. Software metrics thresholds calculation techniques to predict fault-proneness: An empirical comparison. *Information and Software Technology*, 96:38–67, 2018.

[13] Eric Bouwers, Arie van Deursen, and Joost Visser. Towards a catalog format for software metrics. In *Proceedings of the 5th International Workshop on Emerging Trends in Software Metrics*, pages 44–47. ACM, 2014.

[14] Lionel C Briand, John W Daly, and Jürgen K Wüst. A unified framework for coupling measurement in object-oriented systems. *Software Engineering, IEEE Transactions on*, 25(1):91–121, 1999.

[15] Lionel C Briand, Christiane M Differding, and H Dieter Rombach. Practical guidelines for measurement-based process improvement. *Software Process Improvement and Practice*, 2(4):253–280, 1996.

[16] Lionel C. Briand, Sandro Morasca, and Victor R. Basili. An operational process for goal-driven definition of measures. *IEEE transactions on software engineering*, 28(12):1106–1125, 2002.

[17] Zubaidah Bukhari, Jamaiah Yahaya, and Aziz Deraman. Software metric selection methods: A review. In *Electrical Engineering and Informatics (ICEEI), 2015 International Conference on*, pages 433–438. IEEE, 2015.

[18] Coral Calero, Julián Ruiz, and Mario Piattini. Classifying web metrics using the web quality model. *Online Information Review*, 29(3):227–248, 2005.

[19] Shyam R Chidamber and Chris F Kemerer. A metrics suite for object oriented design. *IEEE Transactions on software engineering*, 20(6):476–493, 1994.

[20] Deborah Compeau, Christopher A Higgins, and Sid Huff. Social cognitive theory and individual reactions to computing technology: A longitudinal study. *MIS quarterly*, pages 145–158, 1999.

[21] Deborah R Compeau and Christopher A Higgins. Computer self-efficacy: Development of a measure and initial test. *MIS quarterly*, pages 189–211, 1995.

[22] Matteo Convertino, Kelsie Baker, Connie Lu, John T Vogel, Kyle McKay, and Igor Linkov. Metric selection for ecosystem restoration. Technical report, DTIC Document, 2013.

[23] Daniela S Cruzes and Tore Dyba. Recommended steps for thematic synthesis in software engineering. In *2011 international symposium on empirical software engineering and measurement*, pages 275–284. IEEE, 2011.

[24] Fred D Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, pages 319–340, 1989.

[25] Fred D Davis, Richard P Bagozzi, and Paul R Warshaw. Extrinsic and intrinsic motivation to use computers in the workplace 1. *Journal of applied social psychology*, 22(14):1111–1132, 1992.

[26] Mário André de Freitas Farias, Manoel Gomes de Mendonça Neto, Marcos Kalinowski, and Rodrigo Oliveira Spínola. Identifying self-admitted technical debt through code comment analysis with a contextualized vocabulary. *Information and Software Technology*, 121:106270, 2020.

[27] Josyleuda MM de Oliveira, Karlson B De Oliveira, and Arnaldo Dias Belchior. Measurement process: A mapping among cmmi-sw, iso/iec 15939, ieee std 1061, six sigma and psm. In *Service Systems and Service Management, 2006 International Conference on*, volume 1, pages 810–815. IEEE, 2006.

[28] Isabel M Del Águila and José Del Sagrado. Bayesian networks for enhancement of requirements engineering: a literature review. *Requirements engineering*, 21(4):461–480, 2016.

[29] Esther Derby, Diana Larsen, and Ken Schwaber. *Agile retrospectives: Making good teams great*. Pragmatic Bookshelf, 2006.

[30] Meghann Drury, Kieran Conboy, and Ken Power. Obstacles to decision making in agile software development teams. *Journal of Systems and Software*, 85(6):1239–1254, 2012.

[31] Yogesh K Dwivedi, Nripendra P Rana, Anand Jeyaraj, Marc Clement, and Michael D Williams. Re-examining the unified theory of acceptance and use of technology (utaut): Towards a revised theoretical model. *Information Systems Frontiers*, 21(3):719–734, 2019.

[32] Chin-Feng Fan and Yuan-Chang Yu. Bbn-based software project risk management. *Journal of Systems and Software*, 73(2):193–203, 2004.

[33] Sheikh Umar Farooq and SMK Quadri. Effectiveness of software testing techniques on a measurement scale. *Oriental Journal of Computer Science & Technology*, 3(1):109–113, 2010.

[34] Mostafa Farshchi, Jean-Guy Schneider, Ingo Weber, and John Grundy. Metric selection and anomaly detection for cloud operations using log and metric correlation analysis. *Journal of Systems and Software*, 137:531–549, 2018.

[35] Norman Fenton. Software measurement: A necessary scientific basis. *Software Engineering, IEEE Transactions on*, 20(3):199–206, 1994.

[36] Norman Fenton and James Bieman. *Software metrics: a rigorous and practical approach*. CRC Press, 2014.

[37] Norman Fenton, Paul Krause, and Martin Neil. Software measurement: Uncertainty and causal modeling. *IEEE software*, 19(4):116–122, 2002.

[38] Norman Fenton and Martin Neil. Software metrics and risk. In *Proc 2nd European Software Measurement Conference (FESMA'99), TI-KVIV, Amsterdam, ISBN*, pages 90–76019, 1999.

[39] Norman Fenton and Martin Neil. *Risk assessment and decision analysis with Bayesian networks*. CRC Press, 2012.

[40] Norman E Fenton and Martin Neil. Software metrics: roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 357–370. ACM, 2000.

[41] Norman E Fenton, Martin Neil, and Jose Galan Caballero. Using ranked nodes to model qualitative judgments in bayesian networks. *Knowledge and Data Engineering, IEEE Transactions on*, 19(10):1420–1432, 2007.

[42] Kecia AM Ferreira, Mariza AS Bigonha, Roberto S Bigonha, Luiz FO Mendes, and Heitor C Almeida. Identifying thresholds for object-oriented software metrics. *Journal of Systems and Software*, 85(2):244–257, 2012.

[43] Ludwik Finkelstein and MS Leaning. A review of the fundamental concepts of measurement. *Measurement*, 2(1):25–34, 1984.

[44] Martin Fishbein and Icek Ajzen. Belief, attitude, intention, and behavior: An introduction to theory and research. 1977.

[45] Matthieu Foucault, Marc Palyart, Jean-Rémy Falleri, and Xavier Blanc. Computing contextual metric thresholds. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, pages 1120–1125. ACM, 2014.

[46] Francisco García, M Serrano, J Cruz-Lemus, Francisco Ruiz, Mario Piattini, ALARCOS Research Group, et al. Managing software process measurement: A metamodel-based approach. *Information sciences*, 177(12):2570–2586, 2007.

[47] Cigdem Gencel, Kai Petersen, Aftab Ahmad Mughal, and Muhammad Imran Iqbal. A decision support framework for metrics selection in goal-based measurement programs: Gqm-dsfms. *Journal of Systems and Software*, 86(12):3091–3108, 2013.

[48] Ahmad Nauman Ghazi, Kai Petersen, Sri Sai Vijay Raj Reddy, and Harini Nekkanti. Survey research in software engineering: Problems and mitigation strategies. *IEEE Access*, 7:24703–24718, 2018.

[49] Oswaldo Gómez, Hanna Oktaba, Mario Piattini, and Félix García. A systematic review measurement in software engineering: state-of-the-art in measures. In *Software and Data Technologies*, pages 165–176. Springer, 2006.

[50] Anandasivam Gopal, Mayuram S. Krishnan, Tridas Mukhopadhyay, and Dennis R Goldenson. Measurement programs in software development: determinants of success. *IEEE Transactions on software engineering*, 28(9):863–875, 2002.

[51] Tony Gorschek, Per Garre, Stig Larsson, and Claes Wohlin. A model for technology transfer in practice. *IEEE software*, 23(6):88–95, 2006.

[52] Gregor Grambow, Roy Oberhauser, and Manfred Reichert. Employing semantically driven adaptation for amalgamating software quality assurance with process management. 2010.

[53] Tracy Hall and Norman Fenton. Implementing effective software metrics programs. *IEEE software*, 14(2):55–65, 1997.

[54] Nils Christian Haugen. An empirical study of using planning poker for user story estimation. In *AGILE 2006 (AGILE'06)*, pages 9–pp. IEEE, 2006.

[55] Steffen Herbold, Jens Grabowski, and Stephan Waack. Calculation and optimization of thresholds for sets of software metrics. *Empirical Software Engineering*, 16(6):812–841, 2011.

[56] Yong Hu, XZ Mo, XZ Zhang, YR Zeng, JF Du, and Kang Xie. Intelligent analysis model for outsourced software project risk using constraint-based bayesian network. *Journal of software*, 7(2):440–449, 2012.

[57] Shamsul Huda, Sultan Alyahya, Md Mohsin Ali, Shafiq Ahmad, Jemal Abawajy, Hmood Al-Dossari, and John Yearwood. A framework for software defect prediction and metric selection. *IEEE access*, 6:2844–2858, 2017.

[58] Watts S Humphrey. Statistically managing the software process. *CHANCE*, 2(2):30–39, 1989.

[59] Jakob Iversen and Lars Mathiassen. Lessons from implementing a software metrics program. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, pages 11–pp. IEEE, 2000.

[60] Jakob Iversen and Lars Mathiassen. Cultivation and engineering of a software metrics program. *Information Systems Journal*, 13(1):3–19, 2003.

[61] Kawal Jeet, Nitin Bhatia, and Rajinder Singh Minhas. A bayesian network based approach for software defects prediction. *ACM SIGSOFT Software Engineering Notes*, 36(4):1–5, 2011.

[62] Kawal Jeet, Nitin Bhatia, and Rajinder Singh Minhas. A model for estimating the impact of low productivity on the schedule of a software development project. *ACM SIGSOFT Software Engineering Notes*, 36(4):1–6, 2011.

[63] Eva-Sophie Katterfeldt, Anja Zeising, and Heidi Schelhowe. Designing digital media for teen-aged apprentices: a participatory approach. In *Proceedings of the 11th International Conference on Interaction Design and Children*, pages 196–199, 2012.

[64] Foutse Khomh, Stephane Vaucher, Yann-Gaël Guéhéneuc, and Houari Sahraoui. Bdtex: A gqm-based bayesian approach for the detection of antipatterns. *Journal of Systems and Software*, 84(4):559–572, 2011.

[65] Barbara Kitchenham. What's up with software metrics?–a preliminary mapping study. *Journal of systems and software*, 83(1):37–51, 2010.

[66] Barbara Kitchenham, Shari Lawrence Pfleeger, and Norman Fenton. Towards a framework for software measurement validation. *Software Engineering, IEEE Transactions on*, 21(12):929–944, 1995.

[67] Barbara Kitchenham, Lesley Pickard, and Shari Lawrence Pfleeger. Case studies for method and tool evaluation. *IEEE Softw.*, 12(4):52–62, July 1995.

[68] Barbara A Kitchenham and Shari L Pfleeger. Personal opinion surveys. In *Guide to advanced empirical software engineering*, pages 63–92. Springer, 2008.

[69] Jyrki Kontio, Johanna Bragge, and Laura Lehtola. The focus group method as an empirical tool in software engineering. In *Guide to advanced empirical software engineering*, pages 93–116. Springer, 2008.

[70] Jyrki Kontio et al. *Software engineering risk management: a method, improvement framework, and empirical evaluation*. Helsinki University of Technology, 2001.

[71] Jyrki Kontio, Laura Lehtola, and Johanna Bragge. Using the focus group method in software engineering: obtaining practitioner and user experiences. In *Proceedings. 2004 International Symposium on Empirical Software Engineering, 2004. ISESE'04.*, pages 271–280. IEEE, 2004.

[72] Philippe Kruchten, Robert L Nord, and Ipek Ozkaya. Technical debt: From metaphor to theory and practice. *Ieee software*, 29(6):18–21, 2012.

[73] Eetu Kupiainen, Mika V Mäntylä, and Juha Itkonen. Using metrics in agile and lean software development–a systematic literature review of industrial studies. *Information and Software Technology*, 62:143–163, 2015.

[74] Linda M Laird and M Carol Brennan. *Software measurement and estimation: a practical approach*, volume 2. John Wiley & Sons, 2006.

[75] Laura Lehtola, Marjo Kauppinen, and Sari Kujala. Requirements prioritization challenges in practice. In *International Conference on Product Focused Software Process Improvement*, pages 497–508. Springer, 2004.

[76] Igor Linkov, Drew Loney, Susan Cormier, F Kyle Satterstrom, and Todd Bridges. Weight-of-evidence evaluation in environmental assessment: Review of qualitative and quantitative approaches. *Science of the Total Environment*, 407(19):5199–5205, 2009.

[77] Paul Longley. *Geographic information systems and science*. John Wiley & Sons, 2005.

[78] Martí Manzano, Emilia Mendes, Cristina Gómez, Claudia Ayala, and Xavier Franch. Using bayesian networks to estimate strategic indicators in the context of rapid software development. In *Proceedings of the 14th International Conference on Predictive Models and Data Analytics in Software Engineering*, pages 52–55, 2018.

[79] Radu Marinescu. Detection strategies: Metrics-based rules for detecting design flaws. In *20th IEEE International Conference on Software Maintenance, 2004. Proceedings.*, pages 350–359. IEEE, 2004.

[80] Karl S Mathias, James H Cross II, T Dean Hendrix, and Larry A Barowski. The role of software measures and metrics in studies of program comprehension. In *Proceedings of the 37th annual Southeast regional conference (CD-ROM)*, page 13. ACM, 1999.

[81] Kieran Mathieson. Predicting user intentions: comparing the technology acceptance model with the theory of planned behavior. *Information systems research*, 2(3):173–191, 1991.

[82] John McGarry. *Practical software measurement: objective information for decision makers*. Addison-Wesley Professional, 2002.

[83] Edward F McQuarrie and Shelby H McIntyre. Focus groups and the development of new products by technologically driven companies: some guidelines. *Journal of Product Innovation Management*, 3(1):40–47, 1986.

[84] Wilhelm Meding and Miroslaw Staron. Making software measurement standards understandable. In *Proceedings of the Evaluation and Assessment in Software Engineering*, pages 368–370. 2020.

[85] Emilia Mendes. The use of a bayesian network for web effort estimation. In *International Conference on Web Engineering*, pages 90–104. Springer, 2007.

[86] Emilia Mendes. *Practitioner's knowledge representation: a pathway to improve software effort estimation*. Springer Science & Business, 2014.

[87] Manoel G Mendonça and Victor R Basili. Validation of an approach for improving existing measurement frameworks. *IEEE transactions on software engineering*, 26(6):484–499, 2000.

[88] Andrew Meneely, Ben Smith, and Laurie Williams. Validating software metrics: A spectrum of philosophies. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 21(4):24, 2012.

[89] A. T. Misirli and A. B. Bener. Bayesian networks for evidence-based decision-making in software engineering. *IEEE Transactions on Software Engineering*, 40(6):533–554, June 2014.

[90] Jefferson Seide Molléri, Kai Petersen, and Emilia Mendes. An empirically evaluated checklist for surveys in software engineering. *Information and Software Technology*, 119:106240, 2020.

[91] Gary C Moore and Izak Benbasat. Development of an instrument to measure the perceptions of adopting an information technology innovation. *Information systems research*, 2(3):192–222, 1991.

[92] David L Morgan. *Focus groups as qualitative research*, volume 16. Sage publications, 1996.

[93] Martin Neil, Norman Fenton, and Lars Nielson. Building large-scale bayesian networks. *The Knowledge Engineering Review*, 15(03):257–284, 2000.

[94] Thomas Dyhre Nielsen and Finn Verner Jensen. *Bayesian networks and decision graphs*. Springer Science & Business Media, 2009.

[95] Frank Niessink and Hans Van Vliet. Measurements should generate value, rather than data [software metrics]. In *Proceedings Sixth International Software Metrics Symposium (Cat. No. PR00403)*, pages 31–38. IEEE, 1999.

[96] Paloma Oliveira, Fernando P Lima, Marco Tulio Valente, and Alexander Serebrenik. Rttool: A tool for extracting relative thresholds for source code metrics. In *2014 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 629–632. IEEE, 2014.

[97] Paloma Oliveira, Marco Tulio Valente, Alexandre Bergel, and Alexander Serebrenik. Validating metric thresholds with developers: An early result. In *Software Mainte-*

nance and Evolution (ICSME), 2015 IEEE International Conference on, pages 546–550. IEEE, 2015.

[98] Paloma Oliveira, Marco Tulio Valente, and Fernando Paim Lima. Extracting relative thresholds for source code metrics. In *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week-IEEE Conference on*, pages 254–263. IEEE, 2014.

[99] Tiago Oliveira, Manoj Thomas, Goncalo Baptista, and Filipe Campos. Mobile payment: Understanding the determinants of customer adoption and intention to recommend the technology. *Computers in Human Behavior*, 61:404–414, 2016.

[100] Mauricio J Ordonez and Hisham M Haddad. The state of metrics in software industry. In *Information Technology: New Generations, 2008. ITNG 2008. Fifth International Conference on*, pages 453–458. IEEE, 2008.

[101] Judea Pearl and Stuart Russell. *Bayesian networks*. Computer Science Department, University of California, 1998.

[102] Mirko Perkusich, Amaury Medeiros, Kyller Costa Gorgônio, Hyggo Oliveira de Almeida, Angelo Perkusich, et al. A bayesian network approach to assist on the interpretation of software metrics. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 1498–1503. ACM, 2015.

[103] Mirko Perkusich, Angelo Perkusich, and Hyggo Oliveira de Almeida. Using survey and weighted functions to generate node probability tables for bayesian networks. In *Computational Intelligence and 11th Brazilian Congress on Computational Intelligence (BRICS-CCI & CBIC), 2013 BRICS Congress on*, pages 183–188. IEEE, 2013.

[104] Mirko Perkusich, Gustavo Soares, Hyggo Almeida, and Angelo Perkusich. A procedure to detect problems of processes in software development projects using bayesian networks. *Expert Systems with Applications*, 42(1):437–450, 2015.

[105] Kai Petersen and Claes Wohlin. Context in industrial software engineering research. In *2009 3rd International Symposium on Empirical Software Engineering and Measurement*, pages 401–404. IEEE, 2009.

[106] Roger S Pressman. *Software engineering: a practitioner's approach*. Palgrave Macmillan, 2005.

[107] Prabhat Ram, Pilar Rodriguez, Markku Oivo, and Silverio Martínez-Fernández. Success factors for effective process metrics operationalization in agile software development: A multiple case study. In *2019 IEEE/ACM International Conference on Software and System Processes (ICSSP)*, pages 14–23. IEEE, 2019.

[108] Silja Renooij and Cilia Witteman. Talking probabilities: communicating probabilistic information with words and numbers. *International Journal of Approximate Reasoning*, 22(3):169 – 194, 1999.

[109] Luis Reynoso, Marcela Genero, and Mario Piattini. Refinement and extension of smdm, a method for defining valid measures. *Journal of Universal Computer Science*, 16(21):3210–3244, 2010.

[110] C.K. Riemenschneider, B.C. Hardgrave, and F.D. Davis. Explaining software developer acceptance of methodologies: a comparison of five theoretical models. *Software Engineering, IEEE Transactions on*, 28(12):1135–1145, Dec 2002.

[111] Yin Robert et al. Case study research: design and methods. *Sage Publications, Inc ISBN 0*, 761(92553):8, 2003.

[112] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14(2):131–164, 2009.

[113] Laura Sánchez-González, Félix García, Francisco Ruiz, and Jan Mendling. A study of the effectiveness of two threshold definition techniques. 2012.

[114] Leonardo Da Costa Santos, Renata M Saraiva, Mirko Perkusich, Hyggo O Almeida, and Angelo Perkusich. An empirical study on the influence of context in computing thresholds for chidamber and kemerer metrics. In *28th International Conference on Software Engineering and Knowledge Engineering*, pages 357–362, 2017.

[115] Renata Saraiva, Amaury Medeiros, Mirko Perkusich, Dalton Valadares, Kyller Gorgônio, Hyggo Almeida, and Angelo Perkusich. A bayesian networks-based method to analyze the validity of the data of software measurement programs. *International Journal of Software Engineering and Knowledge Engineering*, Under Review.

[116] Renata M Saraiva, Mirko Perkusich, Hyggo O Almeida, and Angelo Perkusich. A process to calculate the uncertainty of software metrics-based models using bayesian networks. In *28th International Conference on Software Engineering and Knowledge Engineering*, pages 467–472, 2017.

[117] Renata M Saraiva, Mirko Perkusich, Hyggo Oliveira de Almeida, and Angelo Perkusich. A systematic process to define expert-driven software metrics thresholds (s). In *30th International Conference on Software Engineering and Knowledge Engineering*, pages 171–226, 2019.

[118] Norman F Schneidewind. Methodology for validating software metrics. *Software Engineering, IEEE Transactions on*, 18(5):410–422, 1992.

[119] Tushar Sharma and Diomidis Spinellis. A survey on software smells. *Journal of Systems and Software*, 138:158–173, 2018.

[120] Raed Shatnawi. Deriving metrics thresholds using log transformation. *Journal of Software: Evolution and Process*, 27(2):95–113, 2015.

[121] Raed Shatnawi, Wei Li, James Swain, and Tim Newman. Finding software metrics threshold values using roc curves. *Journal of software maintenance and evolution: Research and practice*, 22(1):1–16, 2010.

[122] Forrest Shull, Janice Singer, and Dag IK Sjøberg. *Guide to advanced empirical software engineering*. Springer, 2007.

[123] IEEE Computer Society. IEEE Standard Adoption of ISO/IEC 15939:2007—Systems and Software Engineering— measurement process, 2009.

[124] KP Srinivasan and T Devi. Software metrics validation methodologies in software engineering. *International Journal of Software Engineering & Applications*, 5(6):87, 2014.

[125] Fernanda Staniscuaski, Fernanda Reichert, Fernanda P Werneck, Letícia de Oliveira, Pâmela Billig Mello-Carpes, Rossana Colla Soletti, Camila Infanger Almeida, Eugenia Zandona, Felipe Klein Ricachenevsky, Adriana Neumann, et al. Impact of covid-19 on academic mothers. *Volume 368, Número 6492, Pags. 724-*, 2020.

[126] Miroslaw Staron and Wilhelm Medig. Factors determining long-term success of a measurement program: an industrial case study. *e-Informatica Software Engineering Journal*, 5(1), 2011.

[127] Miroslaw Staron and Wilhelm Meding. *Software Development Measurement Programs: Development, Management and Evolution*. Springer, 2018.

[128] Miroslaw Staron, Wilhelm Meding, Göran Karlsson, and Christer Nilsson. Developing measurement systems: an industrial case study. *Journal of Software Maintenance and Evolution: Research and Practice*, 23(2):89–107, 2011.

[129] Antonia Stefani and Michalis Xenos. Meta-metric evaluation of e-commerce-related metrics. *Electronic Notes in Theoretical Computer Science*, 233:59–72, 2009.

[130] Anselm Strauss and Juliet Corbin. *Basics of qualitative research*. Sage publications, 1990.

[131] Anne Sunikka et al. Usability evaluation of the helsinki school of economics website. Master's thesis, 2004.

[132] Touseef Tahir, Ghulam Rasool, and Cigdem Gencel. A systematic literature review on software measurement programs. *Information and Software Technology*, 73:101–121, 2014.

[133] Shirley Taylor and Peter Todd. Assessing it usage: The role of prior experience. *MIS quarterly*, pages 561–570, 1995.

[134] Shirley Taylor and Peter A Todd. Understanding information technology usage: A test of competing models. *Information systems research*, 6(2):144–176, 1995.

[135] Ronald L Thompson, Christopher A Higgins, and Jane M Howell. Personal computing: toward a conceptual model of utilization. *MIS quarterly*, pages 125–143, 1991.

[136] Steven K Thompson. Simple random sampling. *Sampling, Third Edition*, pages 9–37, 2012.

[137] Maarit Tihinen, Päivi Parviainen, Rob Kommeren, and Jim Rotherham. Metrics and measurements in global software development. *International Journal on Advances in Software*, 5(3), 2012.

[138] Edith Tom, AybüKe Aurum, and Richard Vidgen. An exploration of technical debt. *Journal of Systems and Software*, 86(6):1498–1516, 2013.

[139] Ayse Tosun, Ayse Basar Bener, and Shirin Akbarinasaji. A systematic literature review on the applications of bayesian networks to predict software quality. *Software Quality Journal*, 25(1):273–305, 2017.

[140] Laura Uusitalo. Advantages and challenges of bayesian networks in environmental modelling. *Ecological modelling*, 203(3):312–318, 2007.

[141] Rini Van Solingen, Vic Basili, Gianluigi Caldiera, and H Dieter Rombach. Goal question metric (gqm) approach. *Encyclopedia of software engineering*, 2002.

[142] Viswanath Venkatesh, Michael G Morris, Gordon B Davis, and Fred D Davis. User acceptance of information technology: Toward a unified view. *MIS quarterly*, pages 425–478, 2003.

[143] Stefan Wagner. A bayesian network approach to assess and predict software quality using activity-based quality models. *Information and Software Technology*, 52(11):1230–1241, 2010.

[144] Linda G Wallace and Steven D Sheetz. The adoption of software measures: A technology acceptance model (tam) perspective. *Information & Management*, 51(2):249–259, 2014.

[145] Roel Wieringa, Neil Maiden, Nancy Mead, and Colette Rolland. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requirements engineering*, 11(1):102–107, 2006.

[146] Roel J Wieringa. *Design science methodology for information systems and software engineering*. Springer, 2014.

[147] Dumidu Wijayasekara and Milos Manic. Data driven fuzzy membership function generation for increased understandability. In *2014 IEEE international conference on fuzzy systems (FUZZ-IEEE)*, pages 133–140. IEEE, 2014.

[148] TW Williams, MR Mercer, JP Mucha, and R Kapur. Code coverage, what does it mean in terms of quality? In *Annual Reliability and Maintainability Symposium. 2001 Proceedings. International Symposium on Product Quality and Integrity (Cat. No. 01CH37179)*, pages 420–424. IEEE, 2001.

[149] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[150] Feng Zhang, Audris Mockus, Ying Zou, Foutse Khomh, and Ahmed E Hassan. How does context affect the distribution of software maintainability metrics? In *2013 IEEE International Conference on Software Maintenance*, pages 350–359. IEEE, 2013.

[151] Haopeng Zhang, Fengdi Shu, Ye Yang, Xu Wang, and Qing Wang. A fuzzy-based method for evaluating the trustworthiness of software processes. In *International Conference on Software Process*, pages 297–308. Springer, 2010.

[152] Hadar Ziv and Debra J Richardson. Constructing bayesian-network models of software testing and maintenance uncertainties. In *Software Maintenance, 1997. Proceedings., International Conference on*, pages 100–109. IEEE, 1997.

Apêndice A

Manual do Usuário da Ferramenta SAM

O manual do usuário completo da ferramenta SAM é apresentado nas páginas que se seguem.

SAM

MANUAL DO USUÁRIO

Manuel Severino da Silva Neto
Caio Felipe de Alcântara Melo
Rerisson Daniel Costa SilvaMatos
Mendelssohn Dantas de Sá
Renata Mendonça Saraiva

Sumário

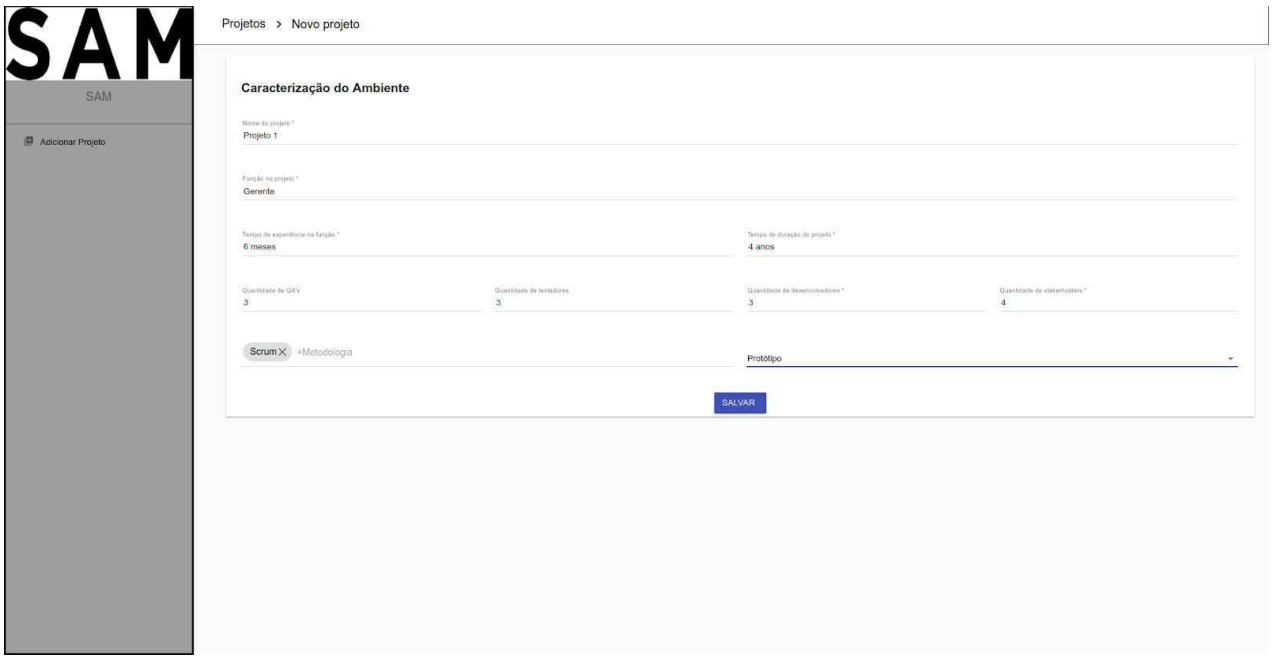
1. Visão geral do aplicativo
2. Aplicativo web
    2.1 Projeto
    2.2 Objetivo
    2.3 Questão
    2.4 Métrica
    2.5 Rede
    2.6 Validação
    2.7 Thresholds
    2.8 Definir Evidência
    2.9 Modelo de confiabilidade

1. Visão geral do aplicativo

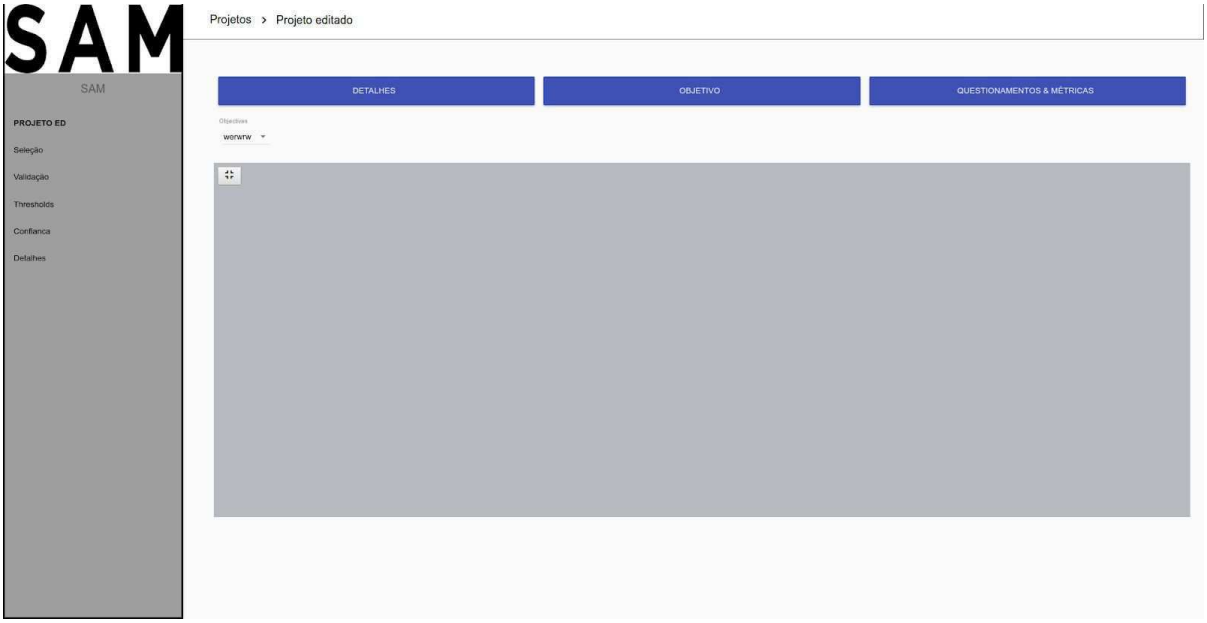
O SAM (Software Attribute Measurement) é uma ferramenta para auxiliar líderes de projeto na definição de planos de medição de software baseados na abordagem GQM. A partir do sistema será possível realizar: cadastro de projetos, adição de objetivos a um projeto, montagem e visualização de redes bayesianas de cada objetivo e validação das métricas de um objetivo, a partir de critérios de validação selecionados pelo líder de projetos.

2.1 Projeto

Ao selecionar a opção “Adicionar Projeto” o usuário será redirecionado para a página de criação de projetos(Fig. x). Nesta página se encontra o formulário para cadastro do projeto e de seu respectivo objetivo. Para criar um projeto é necessário o preenchimento de todos os campos.

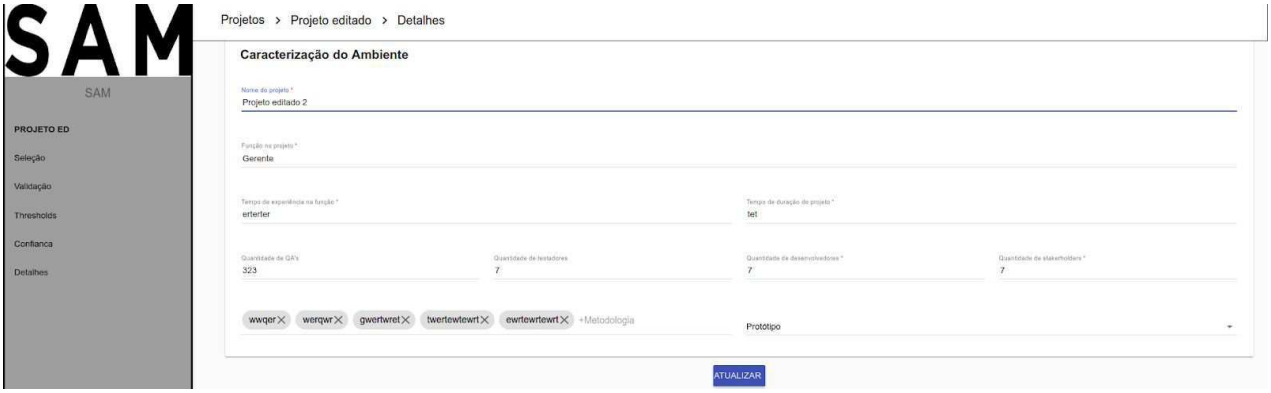


Ao preencher os campos corretamente e clicar no botão de salvar, o usuário é redirecionado para a página principal do projeto(Fig x). Na parte superior da tela temos os Tabs “Detalhes”, “Objetivos” e “Questões & Métrica”. Na parte inferior, retângulo cinza, ficará exibida a rede após sua criação.



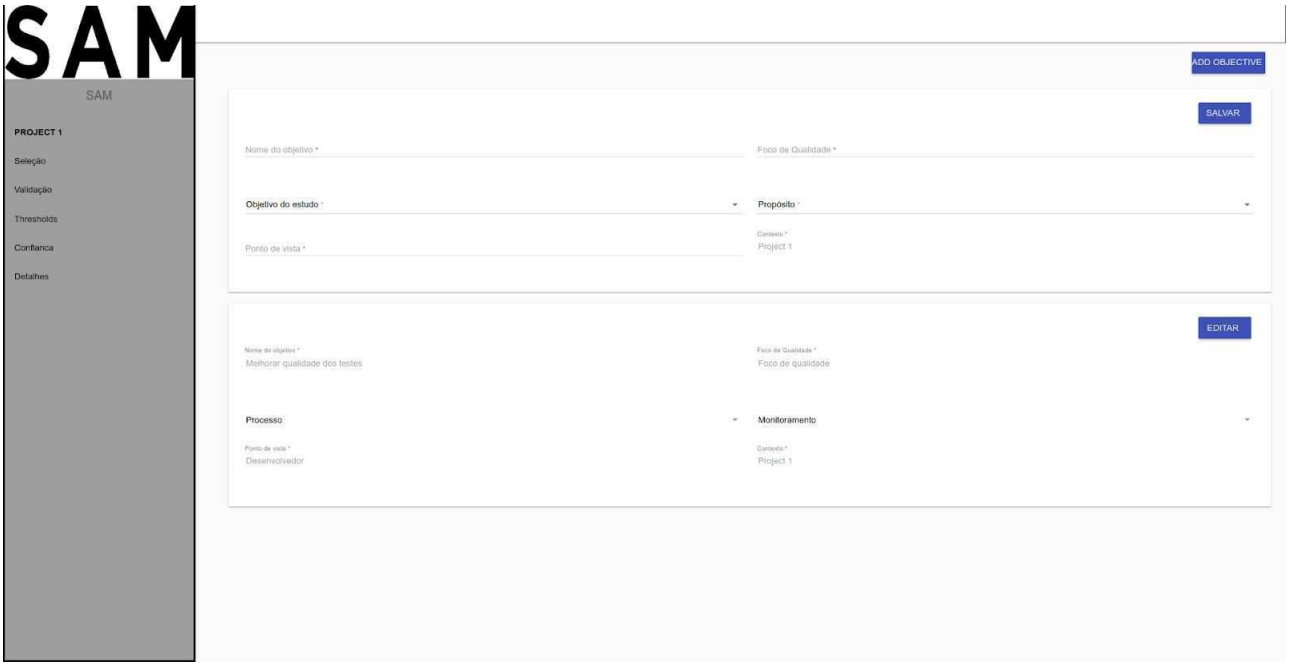
Ao clicar em “Detalhes”, o usuário é redirecionado para uma página(Fig. x) onde são exibidos os dados do projeto cadastrado. O usuário tem a opção de editar os dados ao clicar no botão “editar”.

Ao clicar no botão “editar”, o usuário os campos são liberados para edição pelo usuário.



2.2 Objetivo

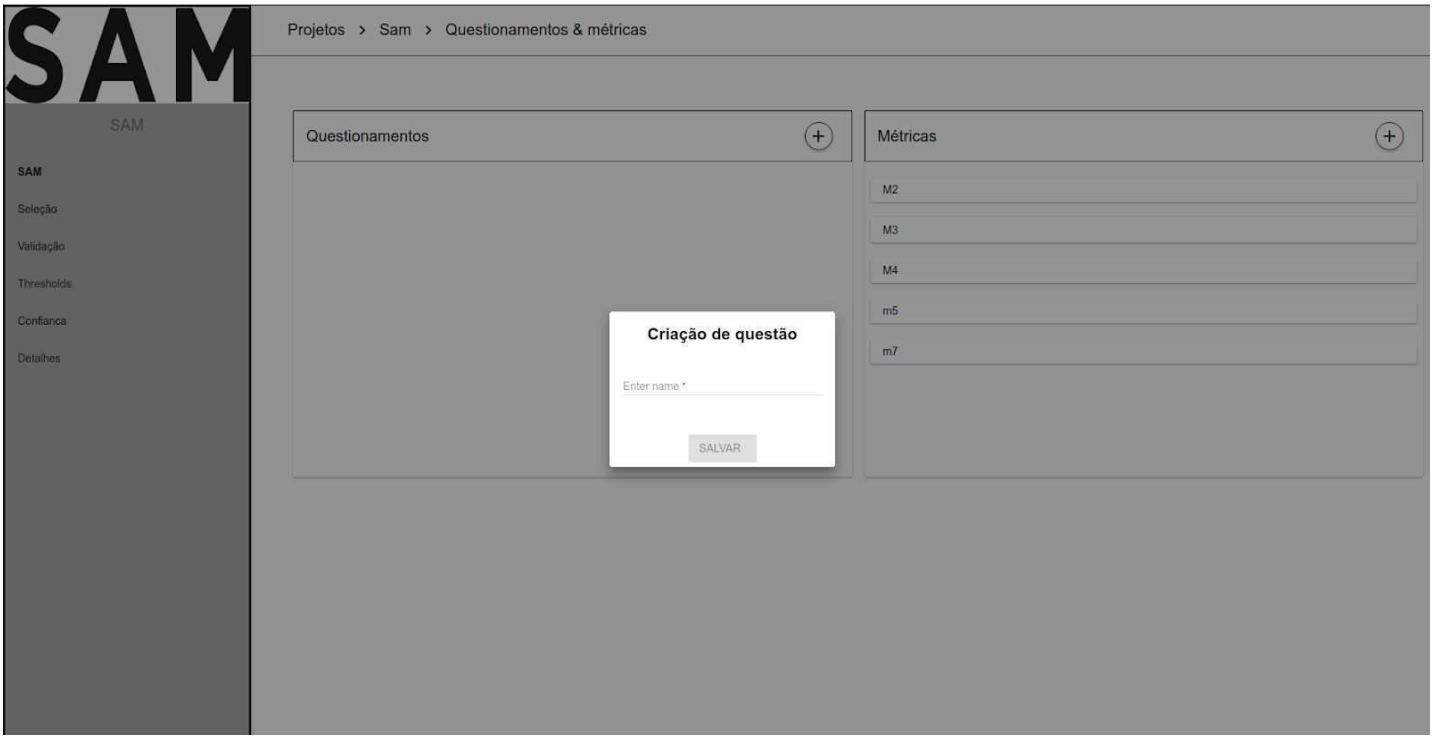
Caso o usuário escolha pela TAB de “Objetivo”, ele será redirecionado para uma página(Fig. x) onde são exibidos os dados dos objetivos cadastrados e um formulário para o cadastro de novos objetivos. O usuário tem a opção de editar os dados ao clicar no botão “editar”.



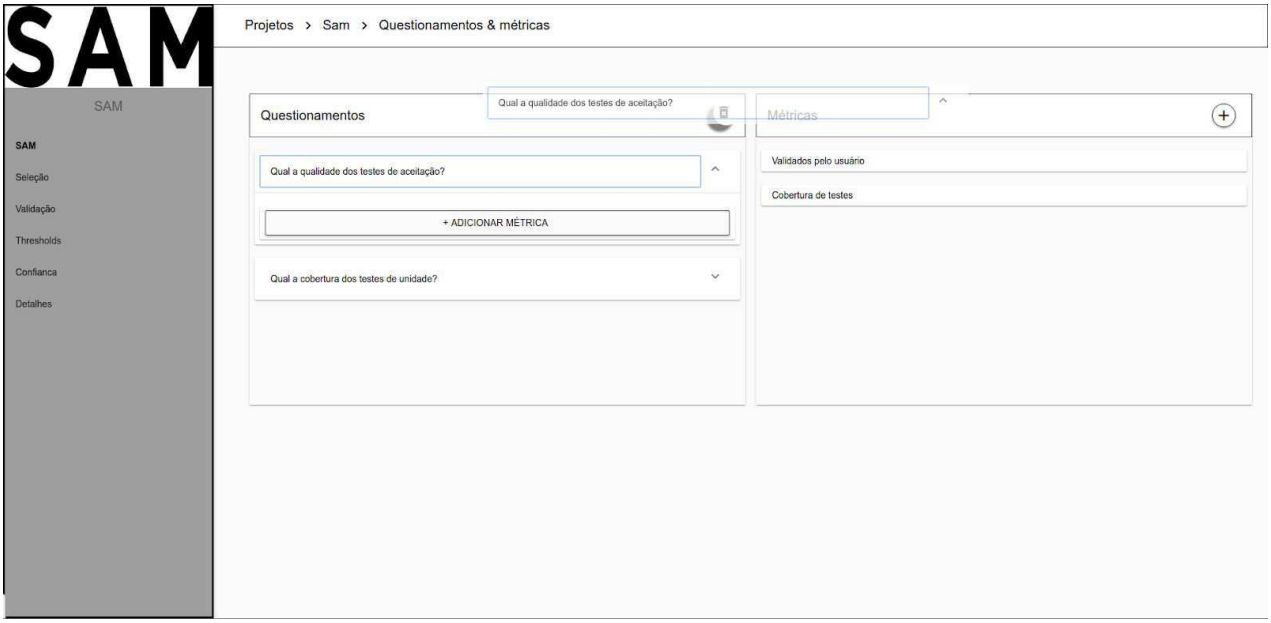
2.3 Questões e Métricas

Por fim, caso o usuário escolha por clicar na botão “Questões & Métricas”, ele será redirecionado para a página(Fig. X) de cadastro e relacionamento de questões e métricas.

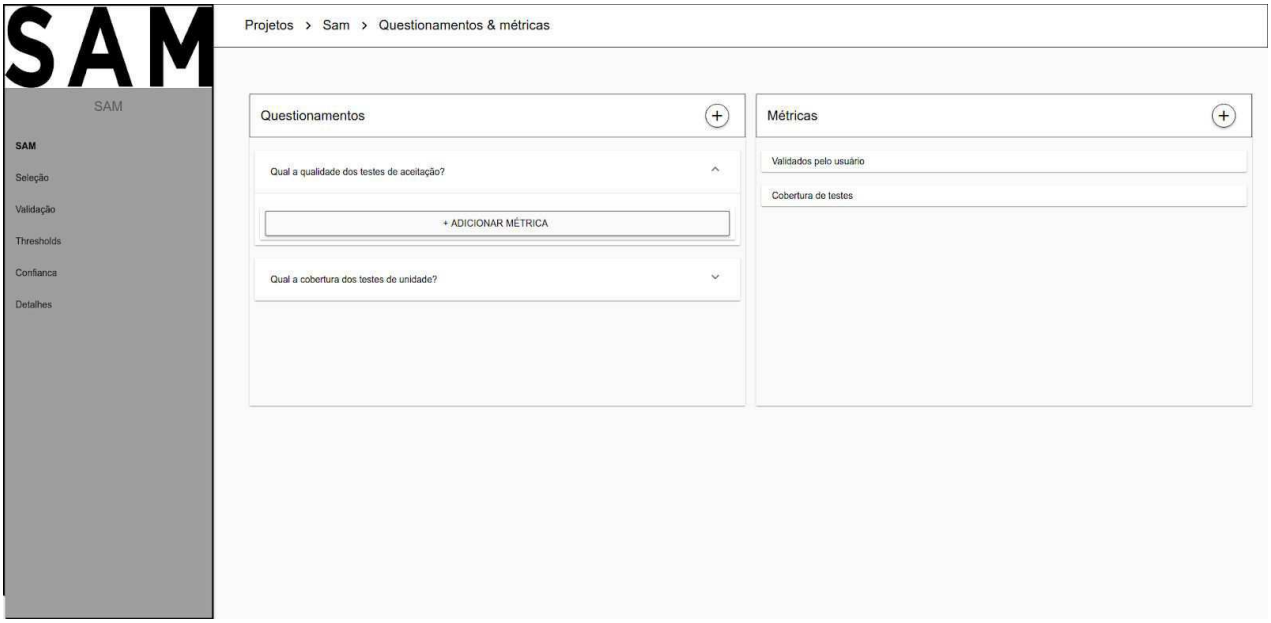
Para adicionar uma questão, basta clicar no botão “+” ao lado de “Questionamentos” e preencher o formulário(Fig. x) que será disponibilizado.



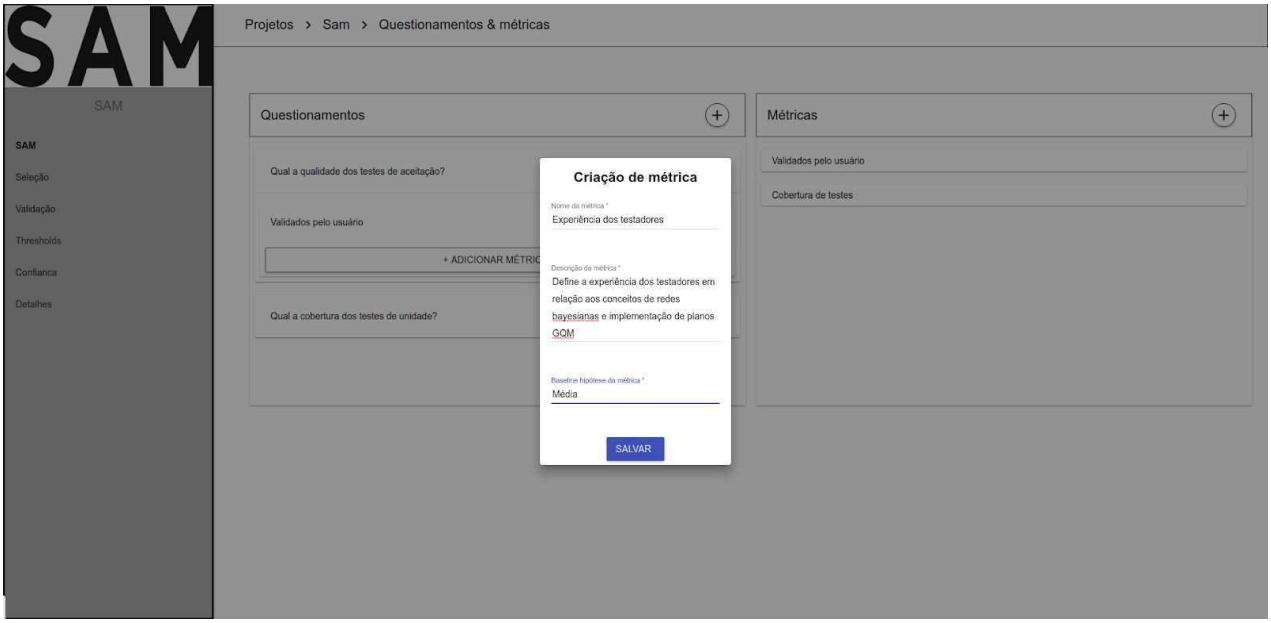
Para remover uma questão, basta arrastá-la para a lixeira.



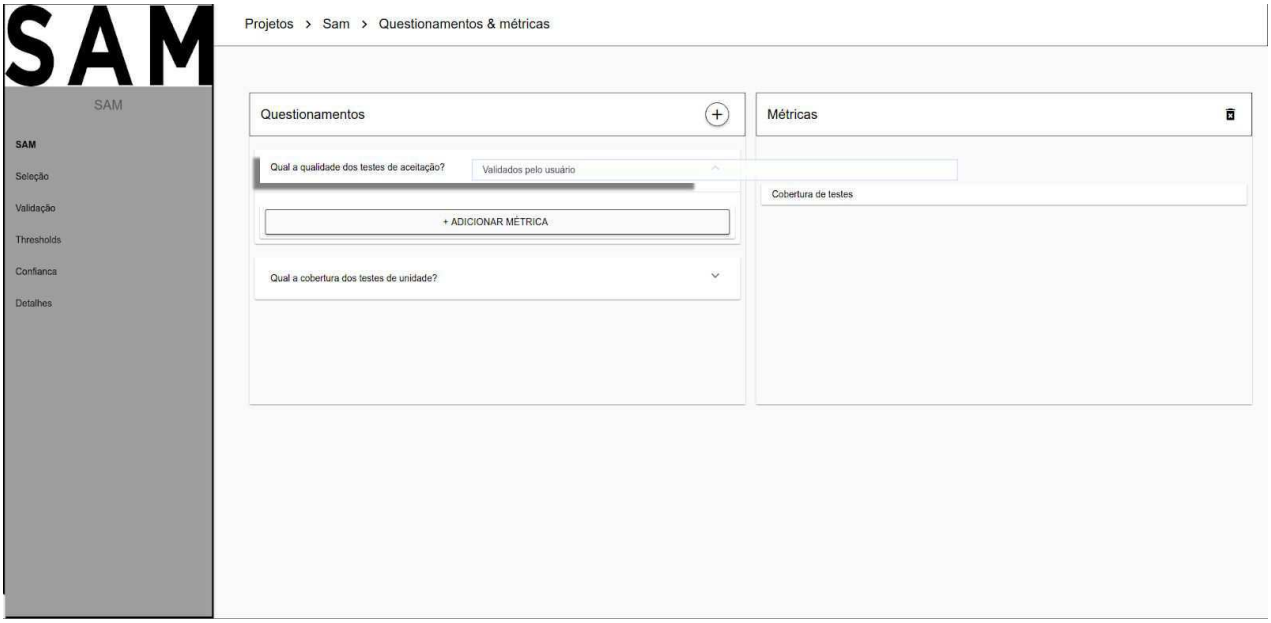
Existem duas maneiras para adicionar uma métrica, a primeira basta clicar no botão “Adicionar Métrica” que aparece abaixo de todo questionamento. A segunda maneira é clicando no botão “+” na aba de métricas.



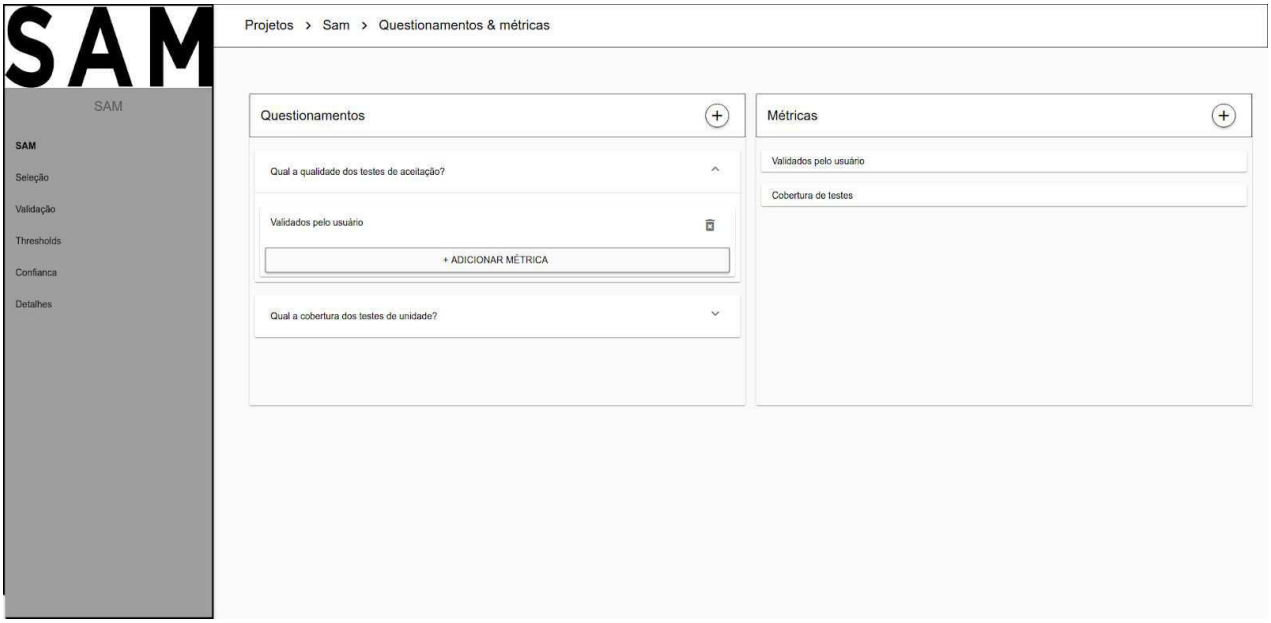
Após clicar em um dos botões de adicionar métrica, será exibido um formulário que deverá ser preenchido corretamente para que a métrica seja salva no sistema ao clicar no botão “salvar”.



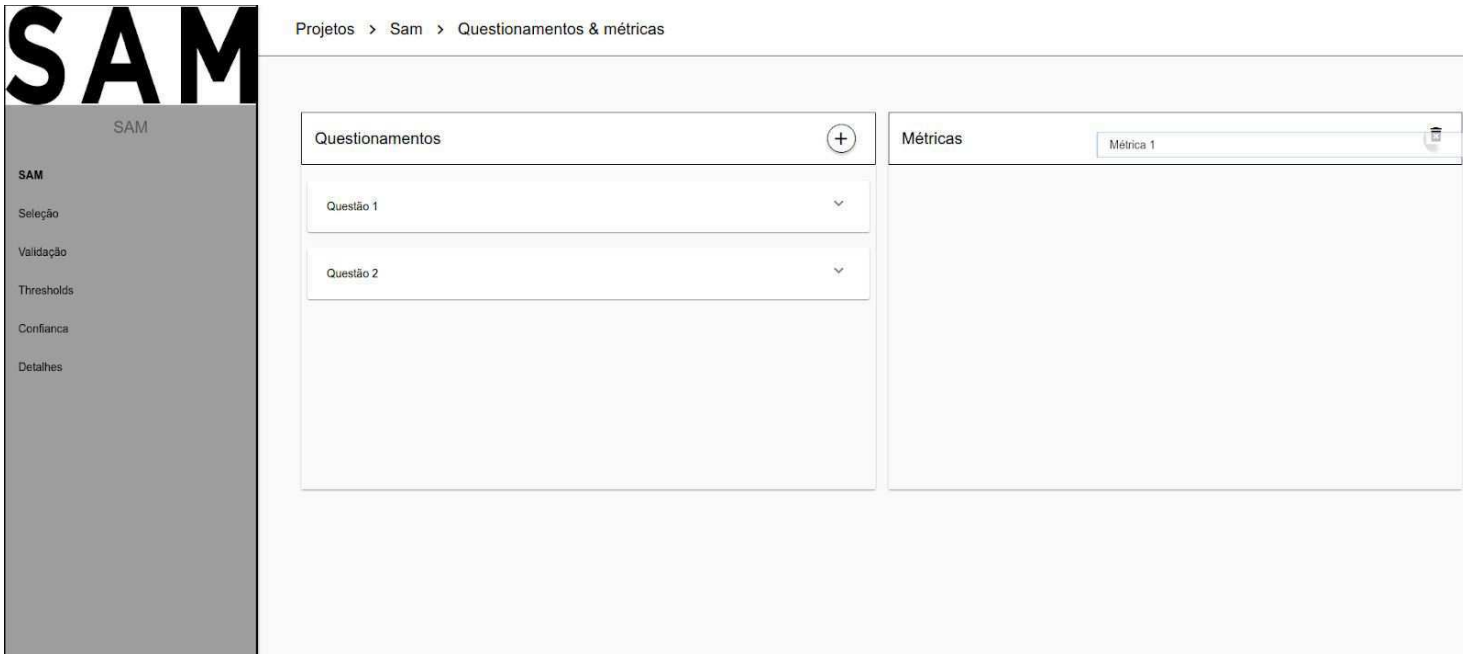
Para relacionar uma métrica com um questão basta apenas arrastar a métrica para cima questão.

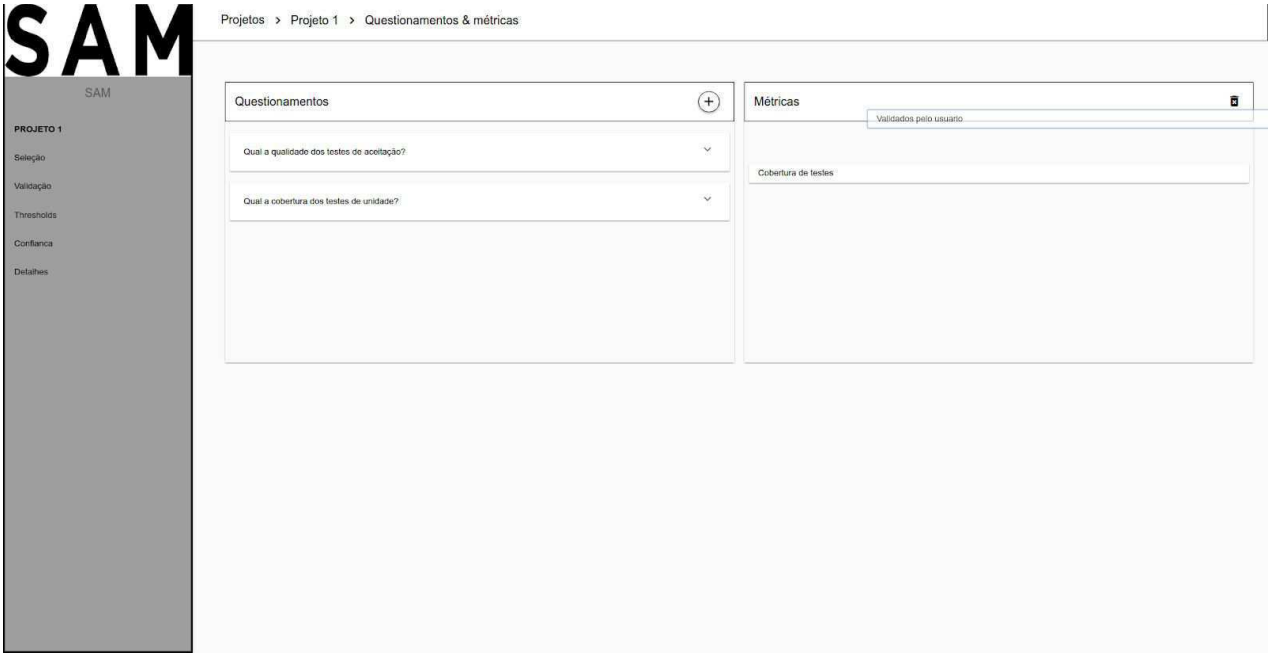


Para desfazer o relacionamento entre uma métrica e uma questão basta clicar na lixeira que fica ao lado do nome da métrica que se deseja remover.



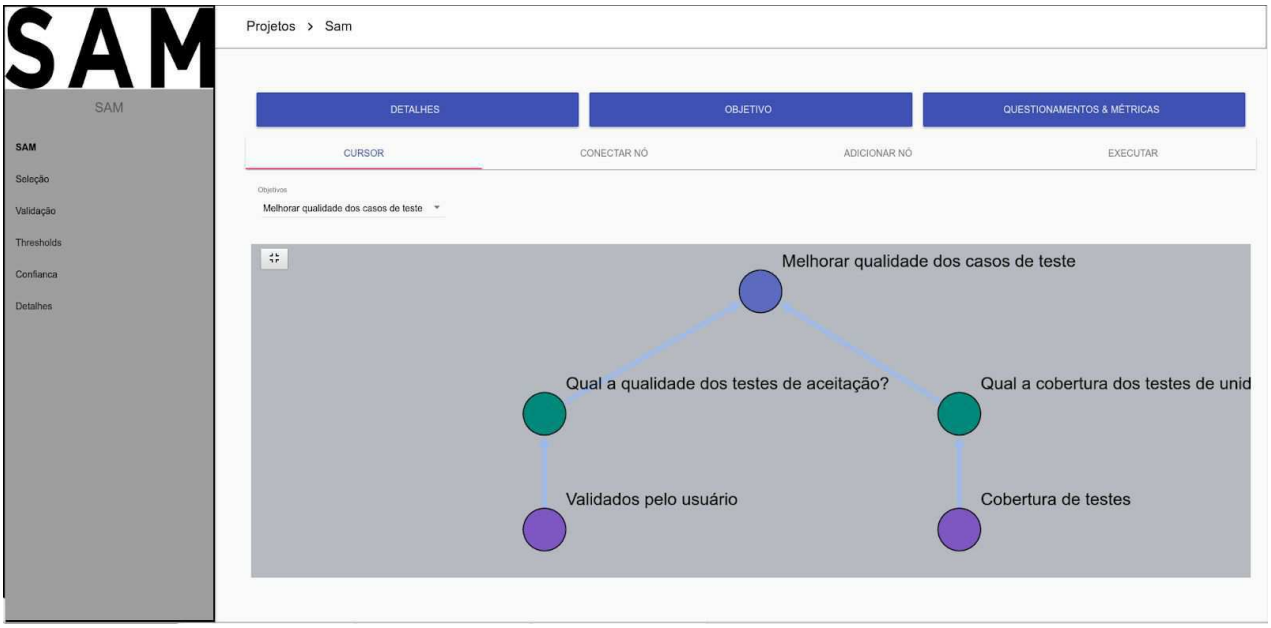
Para deletar uma métrica é necessário que ela não esteja associada a algum questionamento, após isso é só arrastá-la para a lixeira.





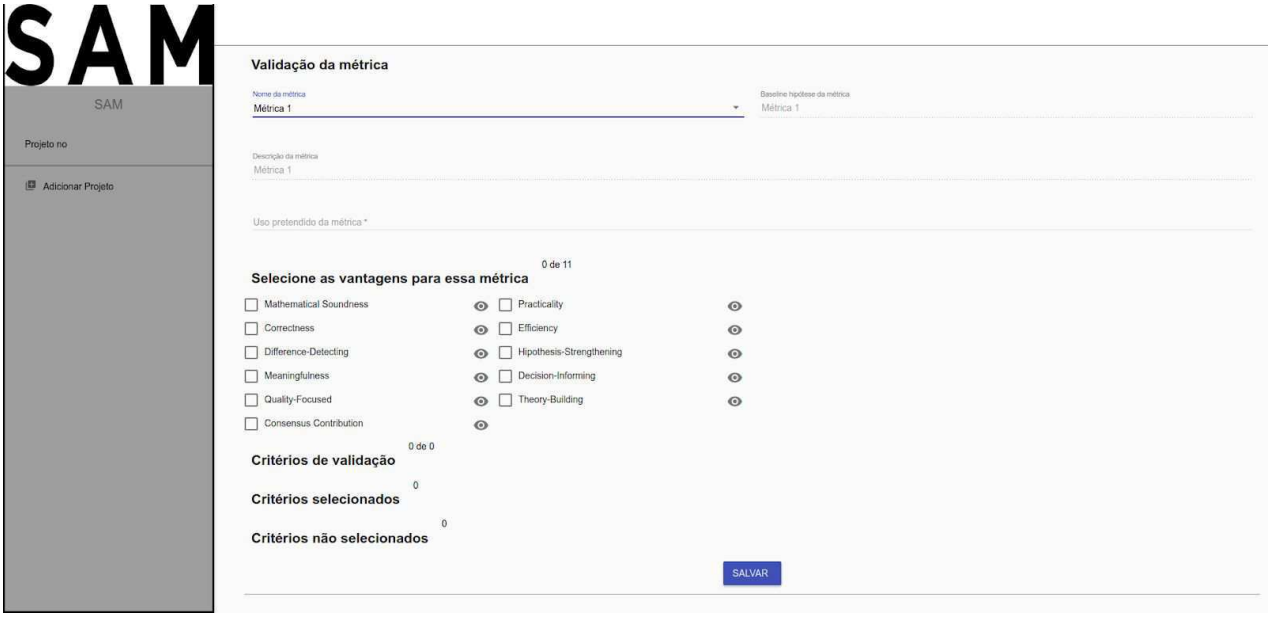
2.5 Rede

Após cadastrar as questões e métricas, relacionando-as, a página(Fig. x) principal do projeto irá exibir a rede.

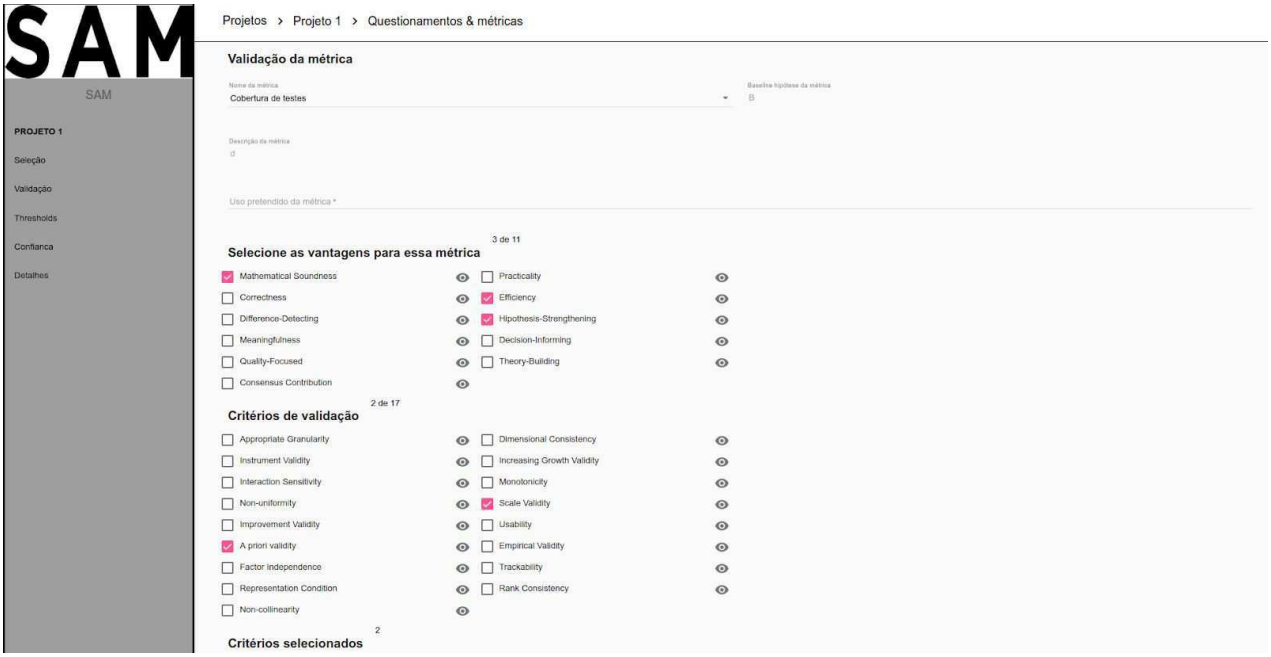


2.6 Validação

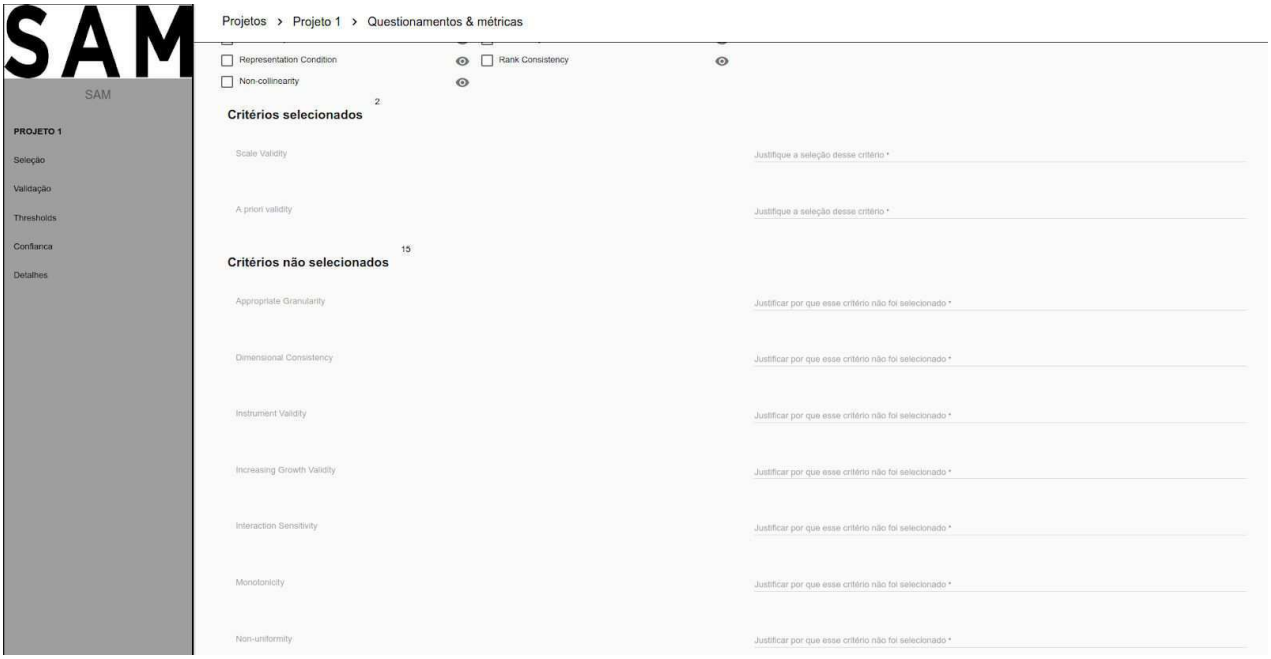
Ao acessar a opção “Validação”, será exibido um formulário para a validação de métrica. O usuário selecionará uma métrica cadastrada no sistema e preencherá as demais informações necessárias para a validação.



O usuário tem a opção de selecionar vantagens pré estabelecidas para a métrica. A descrição de cada vantagem pode ser vista ao passar o mouse sobre o ícone de visualização. Cada vantagem possui seus critérios, e ao selecionar uma vantagem uma lista com os critérios referentes a cada vantagem selecionada será exibida.

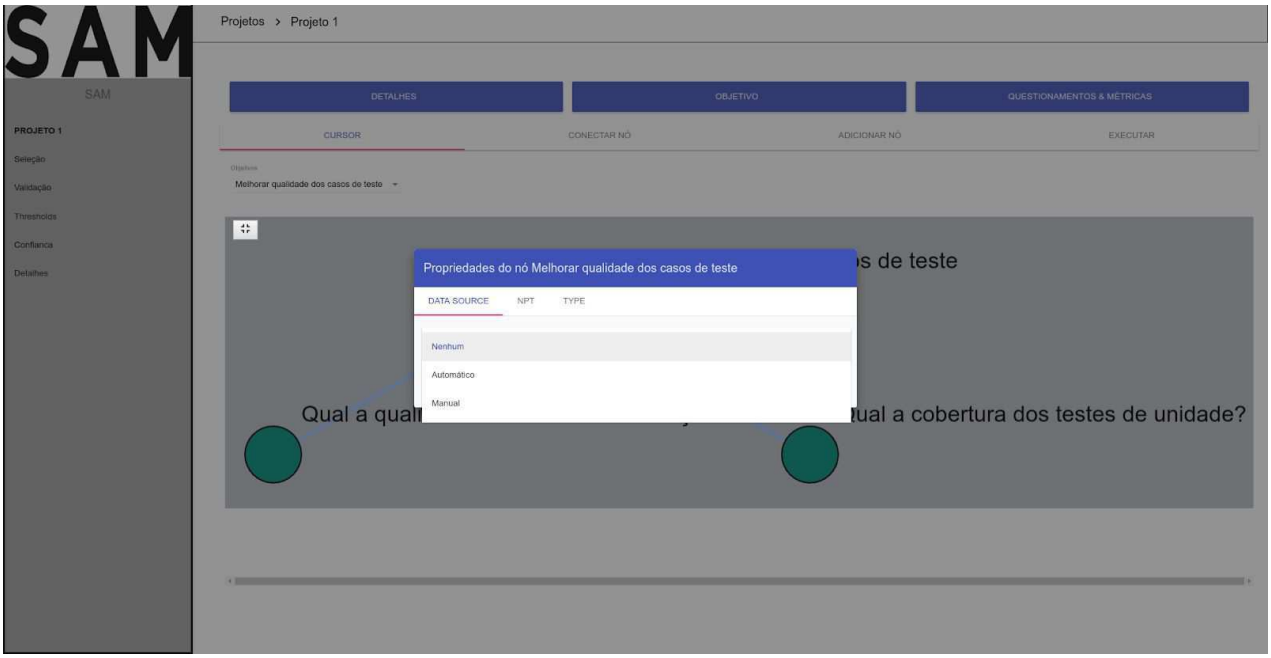


Ao selecionar algum critério, será necessário preencher um formulário para cada critério justificando o motivo da sua seleção. Os critérios que não foram selecionados também devem ter sua não-seleção justificadas.

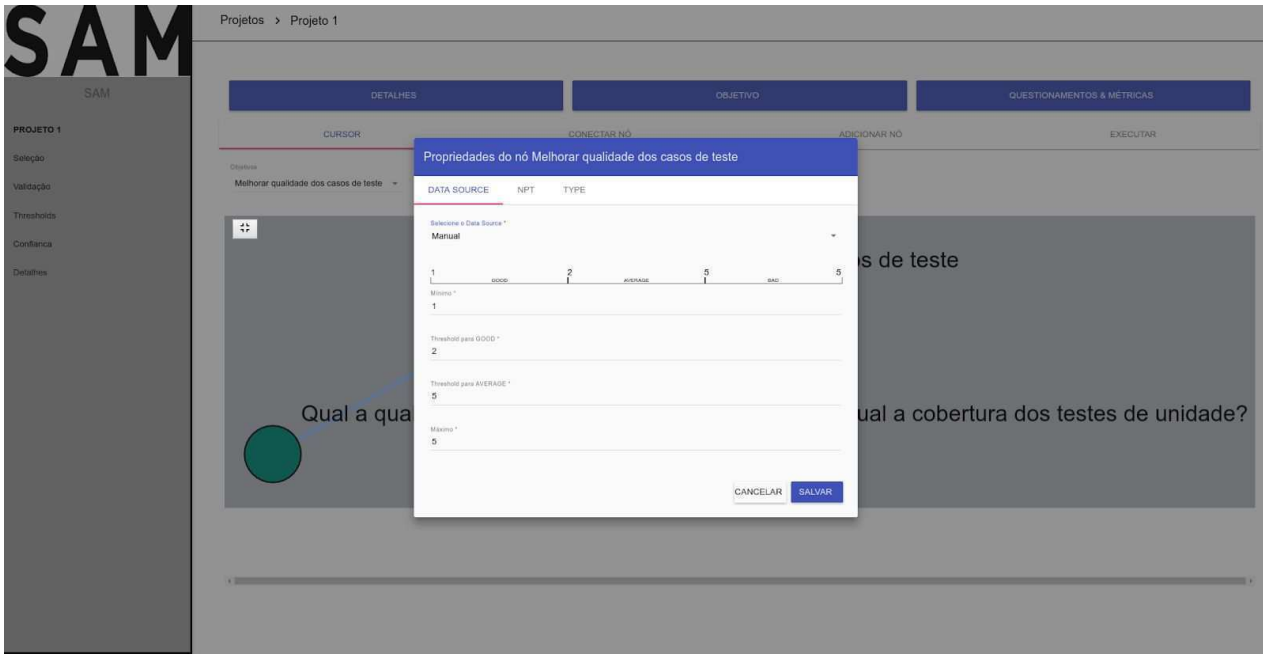


2.7 Thresholds

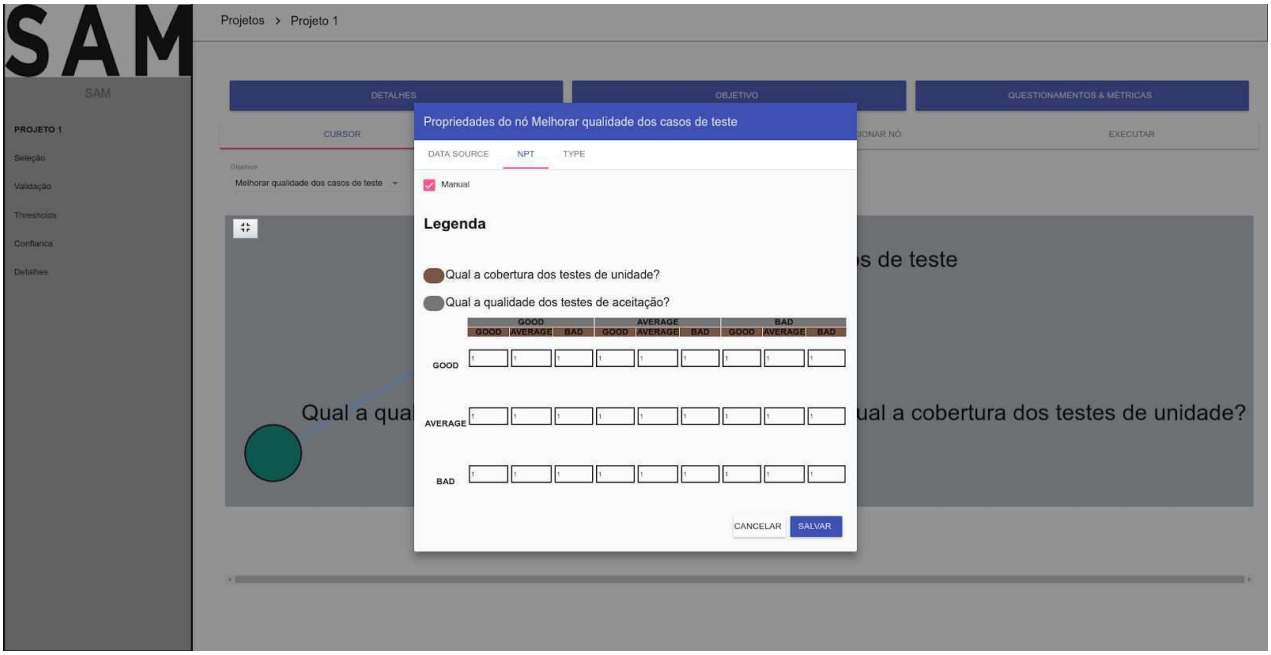
Para definir os *Thresholds*, é necessário que o usuário escolha a opção propriedades no menu do nó, em seguida a opção Manual na *janela*.



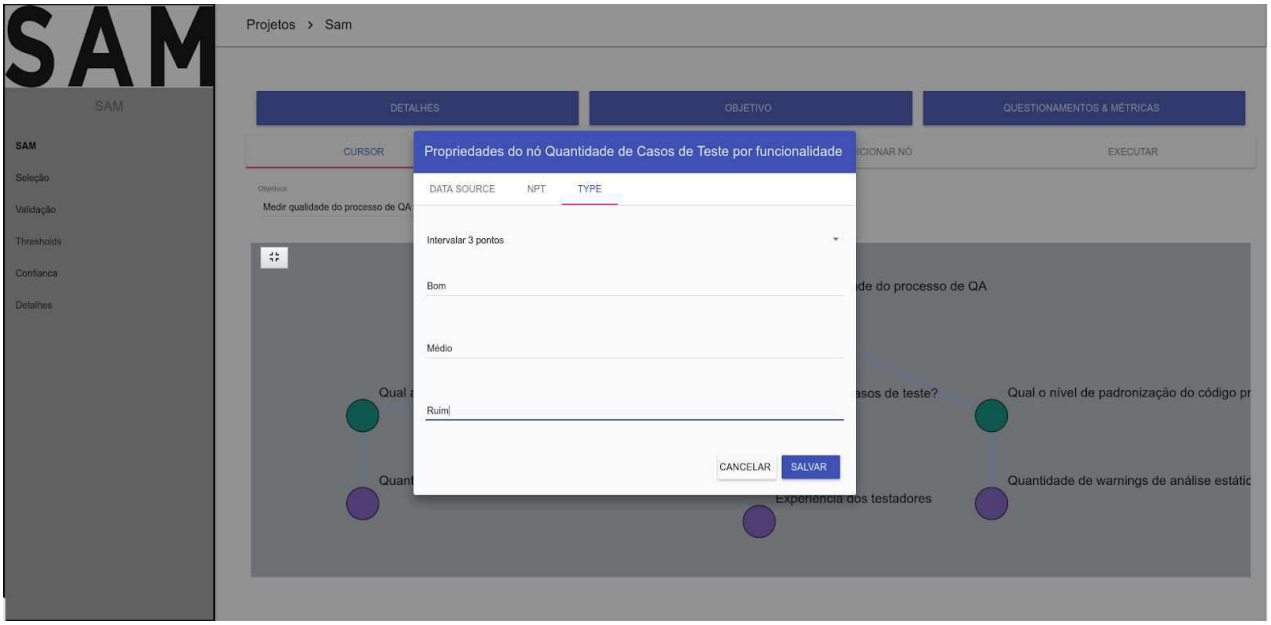
Ao selecionar a opção manual, na aba *data source*, o usuário irá preencher o limite mínimo e máximo que a threshold pode assumir, assim como os limiares, dependendo da quantidade de estados escolhida.



Na aba NPT (*Node Probability Table*), o usuário poderá definir os valores da NPT gerada de acordo com os estados do nó e de seus pais.



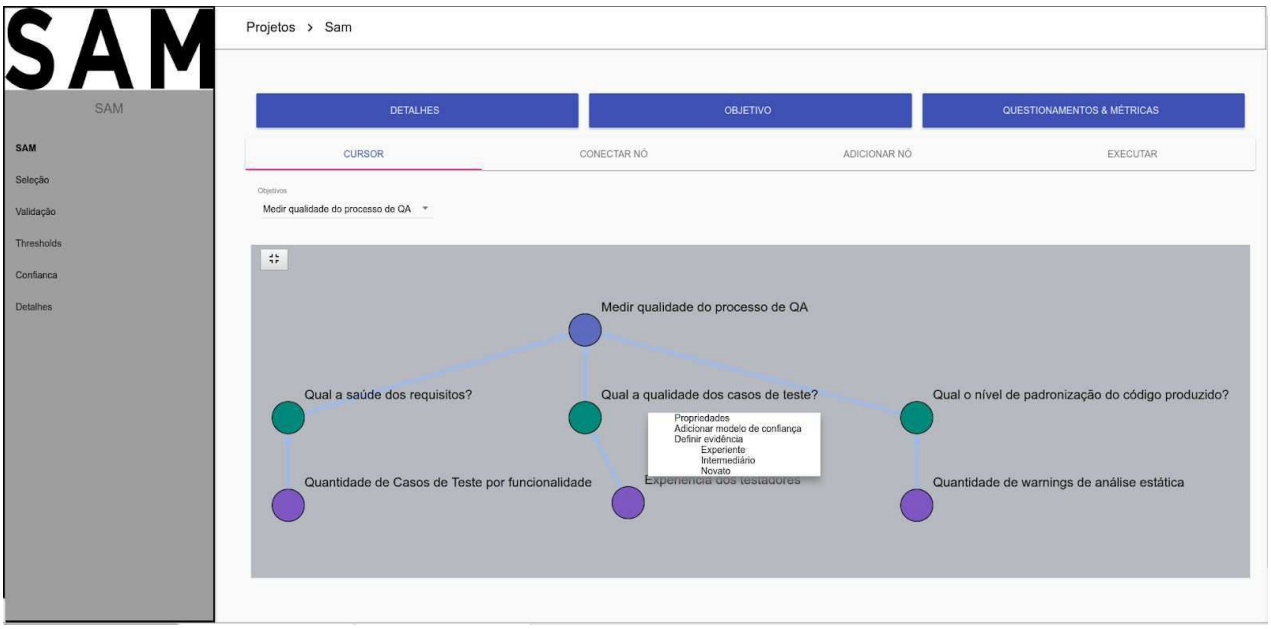
Na aba *TYPE*, o usuário selecionará a escala ele irá trabalhar, no caso ele poderá escolher a escala de 3, 5 ou 7 pontos, e assim ele preencherá o nome dos estados de acordo com a escala selecionada.



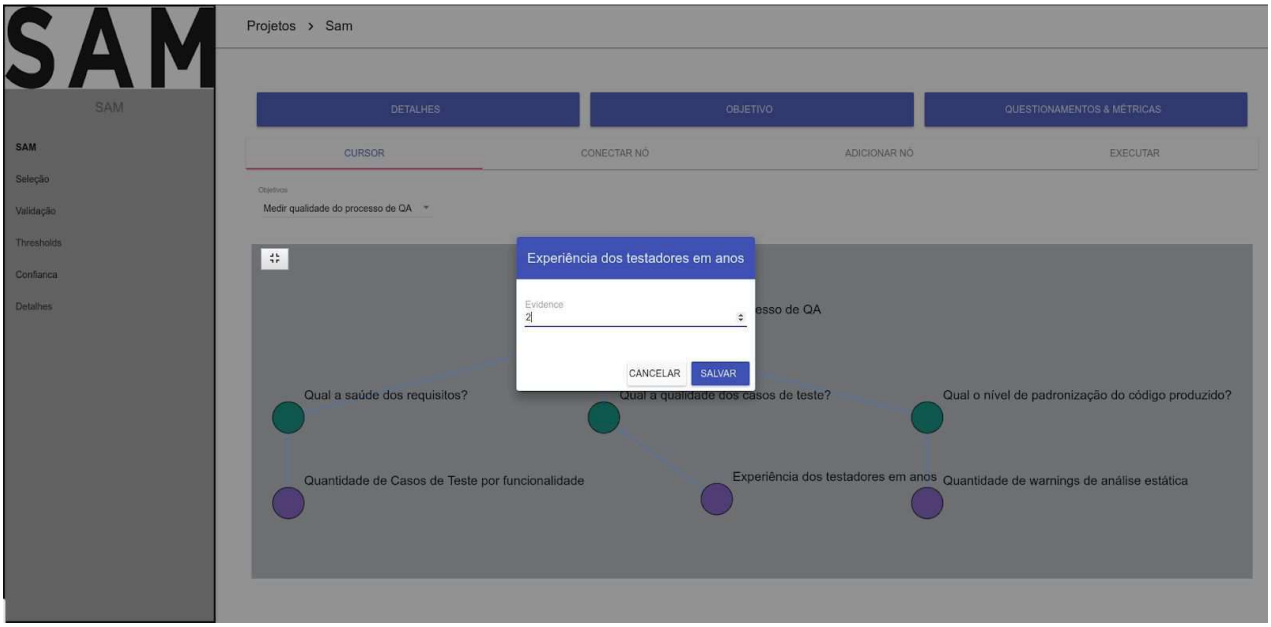
2.7 Definir evidência

Para definir a evidência de um nó, é necessário que o usuário escolha a opção definir evidência no sub-menu dele.

Caso não haja *data source* definido para o nó, aparecerão os possíveis estados para aquele nó. O usuário deve selecionar um dos estados para ser definido como evidência.



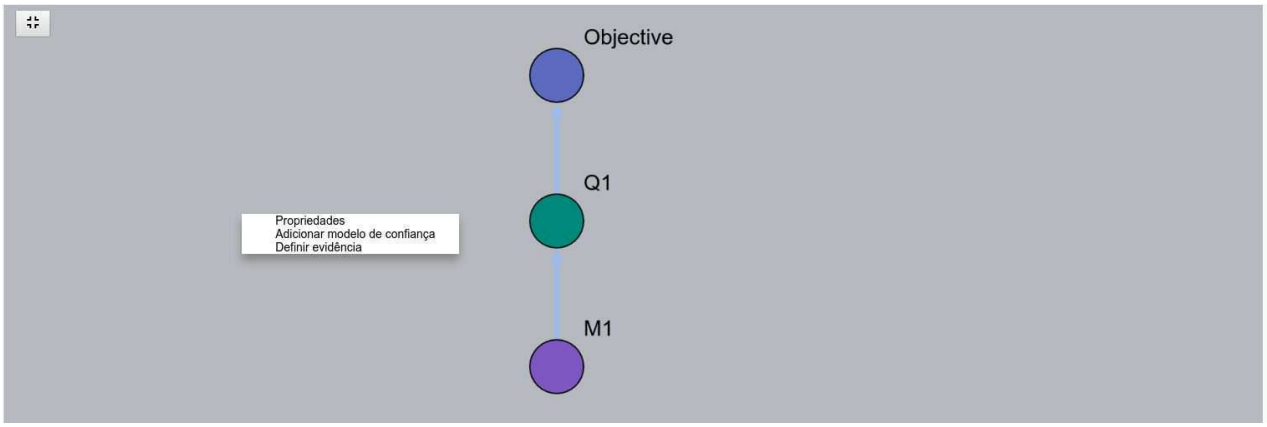
Caso o *data source* seja manual, será mostrado um campo onde deve ser inserido o valor da evidência daquele nó. A partir desse valor e dos *thresholds* definidos para o nó, a evidência será configurada.



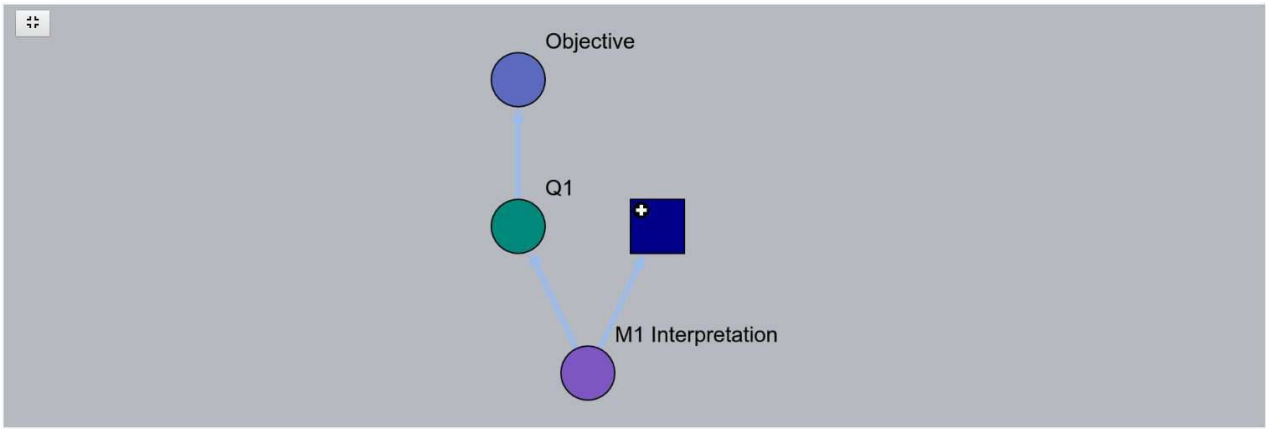
Por fim, caso o *data source* tenha sido definido como automático, será realizada uma requisição para atualização do valor da evidência do nó.

2.8 Modelo de confiabilidade

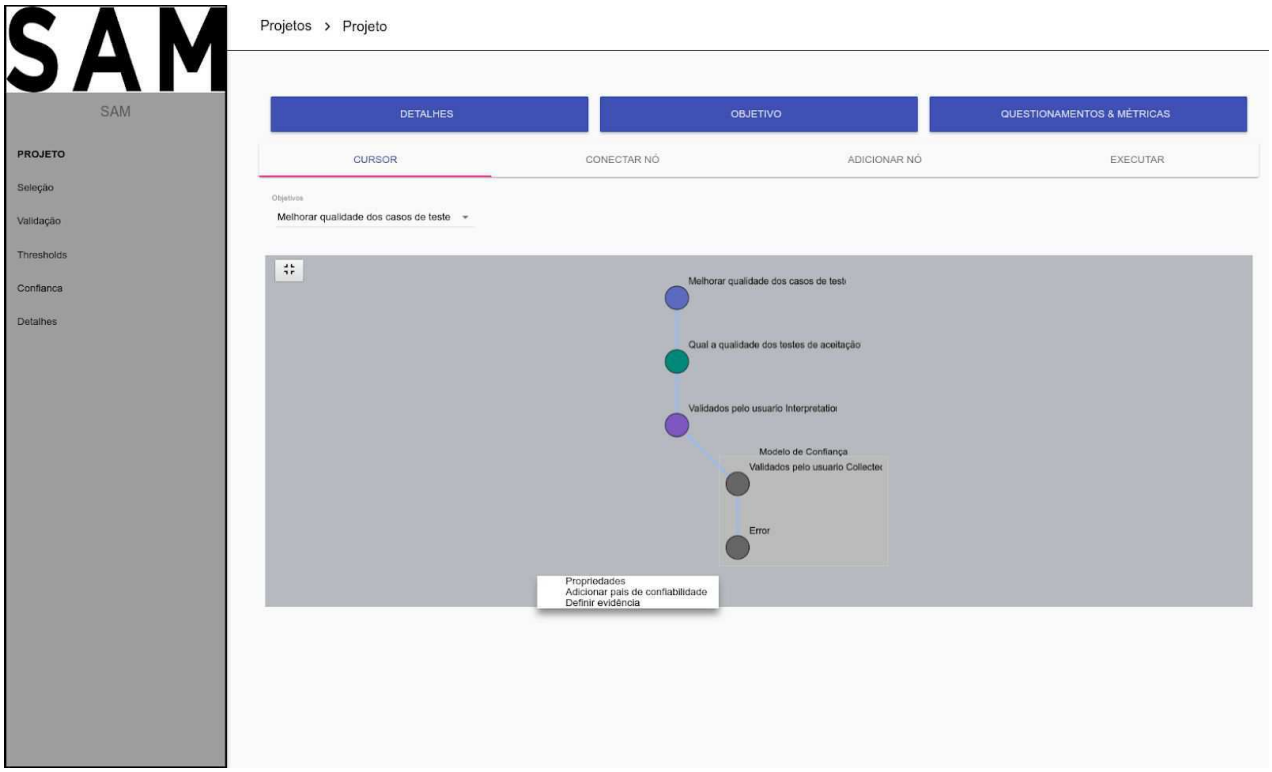
Para adicionar o modelo de confiabilidade à métrica é necessário clicar com o botão direito, abrindo o submenu de opções. O usuário deve escolher a opção “Adicionar modelo de confiança”.

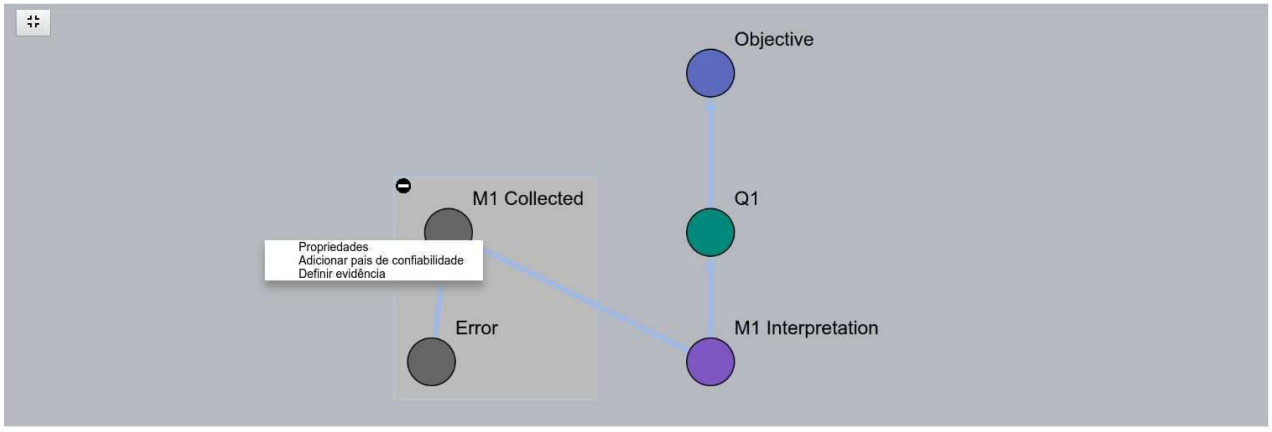


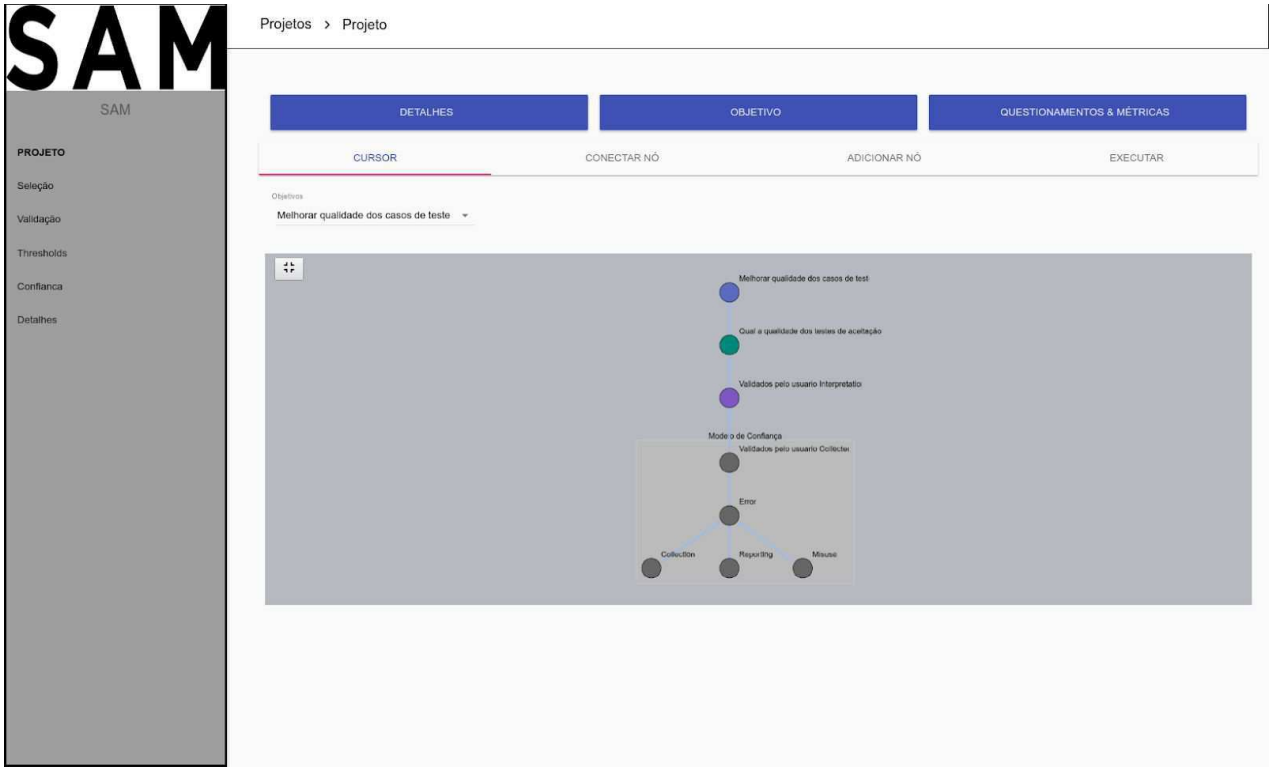
Ao clicar em “Adicionar modelo de confiança”, o modelo de confiança é adicionado a rede, sendo composto por um nó collected e um de error.

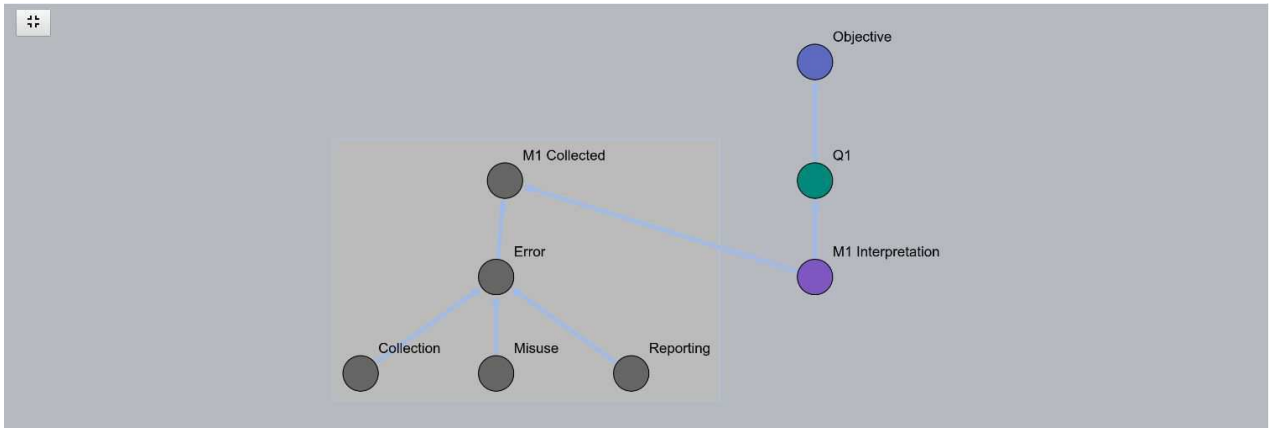


É Possível adicionar pais de confiabilidade. Para fazer isso, o usuário deve clicar nesta opção presente no submenu de opções.






SAM
Projetos › Projeto
SAM
PROJETO
Seleção
Validação
Thresholds
Confianca
Detalhes
DETALHES
OBJETIVO
QUESTIONAMENTOS & MÉTRICAS
CURSOR
CONECTAR NÓ
ADICIONAR NÓ
EXECUTAR
Objetivos
Melhorar qualidade dos casos de teste
Melhorar qualidade dos casos de testi
Qual a qualidade dos testes de aceitação
Validados pelo usuario Interpretatio
Mode o de Confiança
Validados pelo usuario Collecter
Error
Collection
Reporting
Misuse


Objective
Q1
M1 Interpretation
M1 Collected
Error
Collection
Misuse
Reporting

Apêndice B

Manual do Desenvolvedor da Ferramenta SAM

O manual do desenvolvedor completo da ferramenta SAM é apresentado nas páginas que se seguem.

SAM

MANUAL DO DESENVOLVEDOR

Manuel Severino da Silva Neto
Caio Felipe de Alcântara Melo
Rerisson Daniel Costa Silva Matos
Mendelssohn Dantas de Sá
Renata Mendonça Saraiva

Sumário

Introdução

Front-end

Visão Geral

Dependências

Configuração

Estrutura

Componentes

2.5.1 Projeto

2.5.1.1 Criar projeto

2.5.1.2 Editar projeto

2.5.1.3 Remover projeto

2.5.1.4 Rede

2.5.1.4.1 Criar rede

2.5.1.4.2 Criar Nó

2.5.1.4.3 Remover Nó

2.5.1.4.4 Conectar Nós

2.5.1.4.5 Desconectar Nós

2.5.1.5 Propriedades de um nó

2.5.1.5.1 Fonte de Dados

2.5.1.5.2 Tabela da NPT

2.5.1.5.3 Tipos de Intervalos

2.5.1.6 Definir Evidência de um nó

2.5.1.7 Modelo de Confiabilidade

2.5.2 Objetivo

2.5.2.1 Criar objetivo

2.5.2.2 Editar objetivo

2.5.2.3 Remover objetivo

2.5.3 Questão-Métrica

2.5.3.1 Criar questão

2.5.3.2 Editar questão

2.5.3.3 Remover questão

2.5.3.4 Listagem de questões

2.5.3.5 Criar métrica

2.5.3.6 Editar métrica

2.5.3.7 Remover métrica

2.5.3.8 Relacionar métrica com questão

2.5.3.9 Listagem de métricas

2.5.3.10 Validação

2.5.3.11 Thresholds

2.5.3.12 Definir Evidência

2.5.4 SideBar

2.5.5 NavigationBar

□□ Back□end

□□□ Vis o geral

□□□ Depend ncias

□□□ Banco de Dados

3.3.1 Project

3.3.2 Objective

3.3.3 Question

3.3.4 Metric

3.3.5 VariationFactor

3.3.6 Advantage

3.3.7 Criteria

□□□ Controllers

3.4.1 ProjectCtrl / ObjectiveCtrl / QuestionCtrl / MetricCtrl

3.4.2 NetworkCtrl

3.4.3 AdvantagesCtrl

3.4.4 ValidationCtrl

Introduç o

O SAM (Software Attribute Measurement) uma ferramenta para auxiliar l deres de projeto na defini o de planos de medi o de software baseados na abordagem GQM. A partir do sistema ser poss vel reali ar: cadastro de projetos, adi o de objetivos a um projeto, montagem e visuali a o de redes bayesianas de cada objetivo e valida o das m tricas de um objetivo, por meio de crit rios de valida o selecionados pelo l der de projetos.

Front end

Vis o Geral

No Front-end web encontra-se o necessário para apresentação da aplicação no navegador. A principal tecnologia utilizada para o desenvolvimento desta aplicação é o framework AngularJS. Nesta seção encontram-se as dependências utilizadas no desenvolvimento do aplicativo, instruções para instalação e descrições dos componentes desenvolvidos.

Depend ncias

O aplicativo possui as seguintes dependências:

Depend ncia	Vers o	Descriç o
Bower	-	Gerenciador de pacotes necessário para a instalação de dependências.
AngularJS	^1.6.3	Framework Javascript

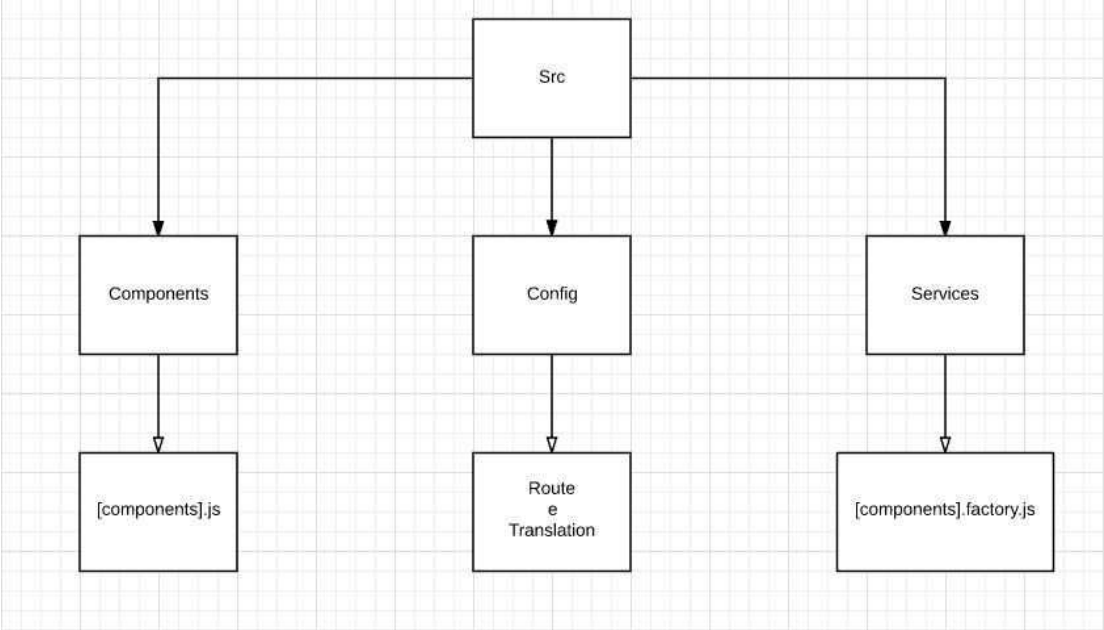
		utilizado na aplicação.
Angular Material	1.1.4	UI Framework.
Angular-ui-router	0.4.2	Módulo necessário para a configuração de rotas da aplicação.
ngstorage	0.3.11	Módulo necessário para armazenamento de dados no navegador.
angular-sanitize	1.2.16	Utilizado para garantir o parse correto do HTML.
angular-messages	^1.6.3	Utilizado para exibir mensagens de validação de formulários.
angular-translate	2.15.2	Utilizado para a internacionalização da aplicação.
angular-translate-loader-static-files	^2.15.2	Permite a definição de arquivos estáticos de internacionalização.
angular-aria	^1.6.3	Aumentar a acessibilidade da aplicação
angular-animate	^1.6.3	Permite a criação de animações. É usado pelo módulo angular-material.
angular-filter	^0.5.16	Provê diferentes filtros utilitários para aplicações Angular.

Também conta com as dependências de desenvolvimento:

Dependência	Versão	Descrição
grunt	0.4.2	
grunt-contrib-concat	0.3.0	
grunt-contrib-copy	0.5.0	
grunt-contrib-jshint	0.8.0	
grunt-contrib-nodeunit	0.3.0	
grunt-contrib-uglify	0.2.2	
grunt-contrib-watch	0.5.3	
grunt-jsdoc	0.5.4	
grunt-exec	0.4.5	

Estrutura

Para melhor organizar o front end, foi implementada uma estrutura que visa fazer uma separação em componentes, agrupados por funcionalidade. Cada componente representa uma página (como login, por exemplo) ou algum elemento da aplicação (como o menu lateral), podendo conter *views* em *html*, *controllers*, *services*, *directives* e folhas de estilo. Os componentes são organizados da seguinte forma:



□ □□ Componentes

□ □□□ □□ Projeto

□ □□□ □□ □□ Criar projeto

Na pasta components/project/create-edit-project temos o arquivo □ □□ □ □□ □□ □ □□□□ □□ □□ □ □□□. □ Neste □□□□□□ arquivo está localizada a função createProject, que ao ser executada, chamará a função *createProject*, presente no project.factory.js. Essa por sua vez, fará uma requisição *POST* e criará o projeto, de acordo com os parâmetros passados no corpo da requisição.

```
/**
 * create project
 * @param project
 */

$scope.createProject = function (project) {
    project.objectives.forEach(function (objective) {
        objective.context = project.name;
    });
    ProjectFactory.createProject(project).then(function (result) {
        $scope.showSuccessMessage('PROJECT_CREATED_SUCCESSFUL');
        $scope.setActualProject(project);
        $state.go('principal-project', {idProject: result.data.id});
    }).catch(function (result) {
        $scope.showNetworkError('NETWORK_ERROR');
    });
};
```

Editar projeto

O *controller* usado para a criação de projetos é o mesmo usado para a edição dos mesmos. Nele existe a função *editProject* que recebe um projeto como parâmetro. O projeto então é atualizado, através da delegação para o serviço de projetos (project.factory.js).

```
/**
 * edit project
 * @param project
 */

$scope.editProject = function (project) {
    ProjectFactory.editProject(project).then(function () {
        $scope.showSuccessMessage('PROJECT_UPDATE_SUCCESS');
        $state.go('principal-project', {idProject: project.id});
    }, function (result) {
        $scope.showNetworkError('NETWORK_ERROR');
    });
};
```

Deletar projeto

Não implementado no protótipo.

Rede

Criar Rede

A criação de uma rede acontece em network.controller.js, localizado em components/project/network através da função *createNetwork*, que recebe o id do objetivo cuja rede será criada. Essa função irá realizar a criação da rede através da chamada a *queryNetwork* do network.factory.js, localizado na pasta components/services, para executar a rede do objetivo cujo id foi passado como parâmetro. Após a requisição à rede processada ser finalizada, o objeto de retorno é convertido para um grafo adequado para exibição pela API cytoscape. Após a conversão, o grafo cytoscape é criado na UI. O retorno da requisição é a rede bayesiana do objetivo processada, com os valores calculados de acordo com os dados definidos para cada nó.

```
$scope.createNetwork = function (idObjective) {
        NetworkFactory.queryNetwork(idObjective).then(function (result) {
                var elements = convertToCyGraph(result.data.nodes);
                createCytosCapeGraph(elements);
        }).catch(function (result) {
                console.error('Erro de requisição não tratado!');
                //TODO ver como tratar os erros das requisições.
        });
};
```

Criar Nó

A criação de um nó acontece em node.controller.js, sendo delegada a partir da função *addNode*, presente em network.controller.js. Após inserir o nome do nó no campo presente na janela de criação de nó e clicar em salvar, a função *addNode* do controller é chamada. Em seguida a função *createNode* do node.factory.js, localizada na pasta *components/services*, é utilizada para realizar a requisição de criação do nó, passando o objeto *node* como parâmetro. Ao final da requisição, é mostrada uma mensagem de adição do nó e a rede bayesiana é recarregada.

```
$scope.addNode = function(node){
    $scope.disableAddBtn = true;
    node.objectiveId = idObjective;
    NodeFactory.createNode(node).then(function () {
        showSuccessMessage('NODE_ADDED');
        createNetwork(idObjective);
        $mdDialog.hide()
    }).catch(function () {
        showNetworkError('NETWORK_ERROR');
    });
}
```

Remover Nó

O nó é removido pela função *$scope.removeNode*, que recebe dois parâmetros, o nó a ser removido e o objetivo. Ao confirmar a remoção do nó, a função *deleteNode* do *network.factory* é chamada, passando os identificadores de objetivo e nó. Após a remoção do nó, a rede é recriada pela função *$scope.createNetwork.*

```
$scope.removeNode = function (node, objective) {
    self.hideMenu();
    NetworkFactory.deleteNode(objective.id, node.id).then(function (result) {
        $scope.showSuccessMessage('NODE_REMOVAL_SUCCESS');
        $scope.createNetwork($scope.selectedObjective.id);
        $scope.createNetwork(objective.id);
    }).catch(function (result) {
        console.error(result);
        $scope.showNetworkError('NETWORK_ERROR');
    });
};
```

Conectar Nós

A conexão entre dois nós acontece no network.controller, através do evento *cy.on('tap',)*do CytoscapeJS, ao clicar em um nó, esse é verificado se realmente é um nó. Se o primeiro nó for clicado, é adicionado uma classe CSS ao nó para marcá-lo como selecionado através do método *self.selectParent*, caso clique novamente no mesmo nó, esse é desmarcado e a classe é removida pela função *self.clearParentNode*. Quando o primeiro nó estiver selecionado e o usuário clicar no segundo nó, a função *self.selectChild* é

chamada para marcá-lo. Em seguida os nós são verificados, caso o segundo nó seja uma questão e o primeiro uma métrica, então é chamada a função *associateToQuestion* presente no *question-metric.factory* que se encontra na pasta *components/services*, passando como parâmetros . Caso contrário a função *connectNodes* do *network.factory* é chamada para realizar a conexão dos nós. Por fim a função *self.clearConnection* é chamada para retirar a classe CSS dos nós clicados anteriormente.

```
cy.on('tap', function (evt) {
    self.hideMenu();
    $scope.hideAddDataSource = true;

    $scope.$apply();

    if (isConnectNode) {
        if (evt.cyTarget.data().name && evt.cyTarget.group() === 'nodes') {
            if (parentNode === evt.cyTarget) {
                self.clearParentNode();
            } else {
                //Parent is not selected, so we select the parent
                if (!parentNode) {
                    self.selectParent(evt);
                //Parent is selected, but child is not, so we connect them.
                } else if (parentNode && !childNode) {
                    self.selectChild(evt);

                    var child = childNode.data();
                    var parent = parentNode.data();

                    child.node_id = child.id;
                    parent.node_id = parent.id;

                    if(child.nodeSource == "question" && parent.nodeSource == "metric" ) {
                        QuestionMetricFactory
                            .associateToQuestion(parent.nodeSourceId, child.nodeSourceId, $scope.selectedObjective.id)
                            .then(function () {
                                $scope.createNetwork($scope.selectedObjective.id);
                                $scope.showSuccessMessage('METRIC_QUESTION_RELATION_ADDED');
                            }).catch(function (result) {
                                $scope.showNetworkError(result.data.message);
                            })
                    } else {
                        NetworkFactory.connectNodes(parent.node_id, child.node_id).then(function () {
                            $scope.createNetwork($scope.selectedObjective.id);
                            $scope.showSuccessMessage('NODE_UPDATED');
                        }).catch(function (result) {
                            $scope.showNetworkError(result.data.message);
                        });
                    }
                    self.clearConnection();
                }
            }
        } else {
            self.clearConnection();
        }
    }
});
```

[illegible] Desconectar Nós

Um nó é desconectado de outro pela função *$scope.removeEdge* que tem como parâmetro a aresta que liga dois nós. Primeiro é verificado se a conexão é entre uma questão e uma métrica, caso for, a função *disassociate* do *question-metric.factory* é chamada, desassociando os dois nós e removendo a métrica da questão à qual ela estava associada. Caso contrário a função *disconnectNodes* do *network.factory* é chamada.

```
$scope.removeEdge = function (edge) {
    var parent = edge.source().data();
    var child = edge.target().data();
    self.hideMenu();
    if(child.nodeSource === "question" && parent.nodeSource === "metric" ) {
        QuestionMetricFactory.disassociate($scope.selectedObjective.id,child.nodeSourceId, parent.nodeSourceId)
            .then(function () {
                $scope.createNetwork($scope.selectedObjective.id);
                $scope.showSuccessMessage('METRIC_REMOVAL_SUCCESS');
        }).catch(function (result) {
            console.error(result);
            $scope.showNetworkError('NETWORK_ERROR');
        })
    }else {
        NetworkFactory.disconnectNodes(parent.id, child.id).then(function (result) {
            $scope.createNetwork($scope.selectedObjective.id);
        }).catch(function (result) {
            console.error(result);
            $scope.showNetworkError('NETWORK_ERROR');
        });
    }
};
```

□ □□ □□ □□ Propriedades de um nó

□ □□ □□ □□ □□ Fonte de Dados

Um nó é atualizado com fonte de dados a partir da função *updateNode* do *node.controller*. O *$scope.isUpdating* serve para desativar o botão de confirmação do formulário de dados enquanto o nó estiver sendo atualizado, impedindo que o usuário clique várias vezes no botão e consequentemente ocasionando possíveis erros no servidor.

```
$scope.updateNode = function (node) {
    node.node_id = node.id;
    $scope.isUpdating = true;

    if (!$scope.editManual) {
        node.function = "manual";
    }

    NodeFactory.updateNode(node).then(function () {
        showSuccessMessage('NODE_UPDATED');
        createNetwork(idObjective);
        $scope.isUpdating = false;
        $mdDialog.hide();
    }).catch(function () {
        showNetworkError('NETWORK_ERROR');
        $scope.isUpdating = false;
        $mdDialog.hide();
    });
};
```

□ □□ □□ □□ □□ Tabela da NPT

Para a criação da tabela NPT, é necessário usar a função *$scope.getStates*, tendo como parâmetros: O atual índice da lista *$scope.numberParents* que é nada mais que a

quantidade de pais que um determinado nó possui, isso servirá para apresentar as linhas de cada coluna.

```
$scope.nParents = function () {
    var parents = [];

    for (var index = 0; index < node.parents.length; index++) {
        parents.push(Math.pow(3, index));
        $scope.parentsDescription[Math.pow(3, index)] = node.parents[index].name;
    }

    return parents;
};
```

O *$scope.numberColumn* serve para determinar a quantidade de caixas de textos que devem ser colocadas na tabela. O *$scope.numberColumn* é determinado a partir da função *$scope.nColumn*, que a partir de cada pai do atual nó, realiza o cálculo para determinar a quantidade de caixas de textos. Em seguida um array contendo os índices das caixas é preenchido e retornado pela função.

```
$scope.nColumn = function () {
    var column = 1;
    node.parents.forEach(function (parent) {
        column = parent.nodeStates.length * column;
    });
    var columnsArray = [];
    for (var index = 0; index < column; index++) {
        columnsArray.push(index);
    }
    return columnsArray;
};
```

□ □□ □□ □□ □□ Tipos de Intervalos

Sempre que um intervalo for selecionado na caixa de seleção, a função *$scope.onPointSelected*, que tem como parâmetros o atual nó e o intervalo selecionado, é chamada. Essa função garante que quantidade de estados do atual nó esteja correta.

```
$scope.onPointSelected = function (node, selectedPoint) {
    var i;
    var originalStatesLength = node.statesName.length;
    if (originalStatesLength < selectedPoint) {
        for (i = originalStatesLength; i < selectedPoint; i++) {
            node.statesName.push('');
            node.states.push(0);
        }
    } else if (originalStatesLength > selectedPoint) {
        for (i = originalStatesLength; i > selectedPoint; i--) {
            node.statesName.splice(i - 1, 1);
            node.states.splice(i - 1, 1);
        }
    }
    if (originalStatesLength < selectedPoint) {
        for (i = node.threshold.limits.length; i < selectedPoint - 1; i++) {
            node.threshold.limits.push(0);
        }
    } else if (originalStatesLength > selectedPoint) {
        for (i = node.threshold.limits.length; i > selectedPoint - 1; i--) {
            node.threshold.limits.splice(i - 1, 1);
        }
    }
    updateNode();
};
```

□ □□ □□ □□ Definir Evid ncia em um Nó

A evidência de um nó acontece no *node.controller.js*, através da função *$scope.updateEvidenceNode,* que recebe o nó selecionado como parâmetro. Na função, é utilizada a função *setEvidenceVal*, encontrada em *node.factory.js*, que utiliza o identificador e o valor da evidência como parâmetros.

```
$scope.updateEvidenceNode = function (node) {
    NodeFactory.setEvidenceVal(node.id, node.evidenceVal).then(function (result) {
        showSuccessMessage('NODE_UPDATED');
        $mdDialog.hide();
        createNetwork(idObjective);
    }).catch(function (result) {
        showNetworkError('NETWORK_ERROR');
        $mdDialog.hide();
    });
};
```

□ □□ □□ □□ Modelo de Confiabilidade

O modelo de confiabilidade

□ □□ □□ Objetivo

□ □□ □□ □□ Criar objetivo

A criação de um objetivo acontece no create-edit-objective.controller.js , através da função createObjective, que recebe um objetivo e o id do projeto como parâmetro. Esta função irá utilizar a função de createObjective do objective.factory.js, localizado na pasta components/services , para criar o objetivo passado como parâmetro.

```
/**
 * create objective
 * @param objective
 */
$scope.createObjective = function (objective) {
        ObjectiveFactory.createObjective(objective, $stateParams.idProject).then(function () {
                $state.go('list-projects');
        }, function (result) {
                //TODO ver como os erros serão tratados.
        });
};
```

□ □ □ □ □ □ Editar objetivo

A edição de um objetivo é bem similar a criação. Ela acontece no create-edit-objective.controller.js , através da função editObjective, que recebe um objetivo como parâmetro. O editObjective chamará a função editObjective do objective.factory.js para realizar a edição.

```
/**
 * Edit objective
 * @param objective
 */
$scope.editObjective = function (objective) {
        ObjectiveFactory.editObjective(objective).then(function () {
                $state.go('list-projects');
        }, function (result) {
                //TODO ver como os erros serão tratados.
        });
};
```

□ □ □ □ □ Quest o□ Métrica

□ □ □ □ □ □ Criar quest o

Na pasta components/question-metric temos o arquivo □ □ □ □ □□□ □ □□ □ □□□□ □□ □. Neste arquivo está localizada a função saveQuestion, que recebe uma questão como parâmetro. Esta função irá adicionar essa questão na lista de questões do objetivo corrente por meio da chamada da função editObjective presente no objective.factory.js.

```
$scope.saveQuestion = function (question) {
    ObjectiveFactory.getObjective(idObjective).then(function (result) {
        result.data.questions.push(question);
        ObjectiveFactory.editObjective(result.data).then(function (result) {
            $scope.closeDialog();
            $scope.currentQuestion = {};
            $rootScope.initQuestions();
            $scope.showSuccessMessage('QUESTION_ADDED_SUCCESS');
            $state.go("view-question-metric");
        }).catch(function () {
            $scope.showNetworkError('NETWORK_ERROR');
        });
    }).catch(function () {
        $scope.showNetworkError('NETWORK_ERROR');
    });


};
```

Editar questão

Usamos o mesmo controller da criação para editar. Nele existe o método updateQuestion que recebe uma questão como parâmetro. O updateQuestion chamará a função updateQuestion do question.factory.js para realizar a edição.

```
$scope.updateQuestion = function (question) {
    QuestionFactory.updateQuestion(question).then(function (result) {
        $scope.showSuccessMessage('QUESTION_UPDATE_SUCCESS');
        $scope.closeDialog();
    }).catch(function () {
        $scope.showNetworkError('NETWORK_ERROR');
    });
}
```

Remover questão

A remoção acontece através de uma diretiva chamada removeDroppable, localizada em components/question-metric/directives. Ao executar o evento 'drop' presente nessa diretiva, o deleteQuestion é executado recebendo o id da questão e do objetivo corrente.

```
el.addEventListener('drop', function (e) {
    if (e.stopPropagation) e.stopPropagation(); // Stops some browsers from redirecting.
    this.classList.remove('over');

    var idMetric = $rootScope.metricId;
    var idQuestion = $rootScope.questionId;

    if (idMetric) {
        MetricFactory.deleteMetric(idMetric).then(function (metricResult) {
            $rootScope.showSuccessMessage("METRIC_REMOVED_SUCCESS");
            $rootScope.initMetrics();
        }).catch(function (result) {
            if (result.status === 406) {
                $rootScope.showNetworkError('METRIC_ASSOCIATED_ERROR');
            } else {
                $rootScope.showNetworkError('NETWORK_ERROR');
            }
        });
    } else {
        QuestionFactory.deleteQuestion(idQuestion, scope.objectiveId).then(function (questionResult) {
            $rootScope.showSuccessMessage("QUESTION_REMOVED_SUCCESS");
            $rootScope.initQuestions();
        }).catch(function (metricResult) {
            $rootScope.showNetworkError('NETWORK_ERROR');
        });
    }

    return false
}, false);
```

Listar questões

A listagem de questões acontece no controller question-metric.controller.js, atraves da função initQuestions. A função realiza uma requisição que tem como resposta um JSON contendo todas as questões referentes ao objetivo corrente.

```
/**
 * Get all questions
 */
$rootScope.initQuestions = function () {
    ObjectiveFactory.getObjective(idObjective).then(function (result) {
        $scope.questions = result.data.questions;
    }).catch(function (result) {
        $scope.showNetworkError('NETWORK_ERROR');
    });
};
```

Criar métrica

A criação de uma métrica acontece no question-metric.controller.js, dentro do controller DialogController através da função saveMetric, que recebe um metrica como parâmetro. Esta função irá utilizar a função createMetric do metric.factory.js,

localizado na pasta components/services , para criar o objetivo passado como parâmetro.

```
$scope.saveMetric = function (metric) {
    MetricFactory.createMetric(metric).then(function (result) {
        $scope.closeDialog();

        //If Question exists
        if (!edit) {
            $scope.saveMetricOnQuestion(result.data, currentQuestion);
        }

        $scope.currentMetric = {};
        $rootScope.initMetrics();
        $rootScope.initQuestions();
        $state.go("view-question-metric");
    }).catch(function () {
        $scope.showNetworkError('NETWORK_ERROR');
    });
};
```

Editar métrica

O processo da edição de uma métrica é bastante parecida com a da criação. Tal processo utiliza os mesmo métodos, apenas existe uma verificação na função saveMetric pra saber se a métrica passada possui id. Se não tiver cria, se não edita.

Remover métrica

Também no question-metric.controller.js é possível remover uma métrica através da função removeMetric, que recebe a questão a qual essa métrica está associada e a posição na qual ela se encontra na lista das métricas relacionada à questão. Ao executar removeMetric, a função removeMetric, presente no metric.factory.js, fará uma requisição de DELETE e removerá a métrica.

```
$scope.removeMetric = function (question, index) {
    var metric = question.metrics[index];
    QuestionFactory.removeMetric(idObjective, question.id, metric.id).then(function () {
        $scope.showSuccessMessage('METRIC_REMOVAL_SUCCESS');
        question.metrics.splice(index, 1);
    }).catch(function () {
        $scope.showNetworkError('NETWORK_ERROR');
    });
};
```

Relacionar métrica com quest o

A relação é feita através do método saveMetricOnQuestion, presente no components/question-metric/question-metric.controller.js. Esse método irá executar o updateQuestion para atualizar a lista de métricas associadas a questão.

```
$scope.saveMetricOnQuestion = function (metric, question) {
    question.metrics.push(metric);
    QuestionFactory.updateQuestion(question).then(function (questionResult) {
        $rootScope.initQuestions();
        $scope.showSuccessMessage("METRIC_QUESTION_RELATION_ADDED");
    }).catch(function () {
        $scope.showNetworkError('NETWORK_ERROR');
    });

};
```

□ □□ □□ □□ Listar métricas

A listagem de métricas acontece no controller question-metric.controller.js, atraves da função initMetrics. A função realiza uma requisição que tem como resposta um JSON contendo todas as métricas do sistema.

```
/**
 * Get all metrics
 */
$rootScope.initMetrics = function () {
    MetricFactory.getMetrics().then(function (result) {
        $rootScope.metrics = result.data;
    }).catch(function (result) {
        $scope.showNetworkError('NETWORK_ERROR');
    });
};
```

□ □□ □□ □□ □ Thresholds

Os thresholds são definidos no m etricS caleP opU □ . através da função updateE videnceN od que recebe uma métrica como parâmetro. Essa metrica possui uma lista de limites, 2 limites para escala intervalar de 3 pontos ou 4 limites para a escala intervalar de 5 pontos, e estes limites serão armazenados no nó como seus thresholds.

```
$scope.updateEvidenceNode = function (metric) {
    if($scope.node.threshold.scale === '3-POINT_INTERVAL'){

        $scope.node.threshold.limits[2] = null;
        $scope.node.threshold.limits[3] = null;

        limit1 = $scope.node.threshold.limits[0];
        limit2 = $scope.node.threshold.limits[1];
        if (metric.evidenceVal > limit2) {
            node.evidence = 0;
        } else if (metric.evidenceVal < limit1) {
            node.evidence = 2;
        } else {
            node.evidence = 1;
        }

    }else if($scope.node.threshold.scale === '5-POINT_INTERVAL'){

        limit1 = $scope.node.threshold.limits[0];
        limit2 = $scope.node.threshold.limits[1];
        limit3 = $scope.node.threshold.limits[2];
        limit4 =$scope.node.threshold.limits[3];

        if (metric.evidenceVal > limit4) {
            //Muito bom
            node.evidence = 0;
        } else if (metric.evidenceVal >= limit3 && metric.evidence < limit4) {
            //Bom
            node.evidence = 0;
        } else if (metric.evidenceVal >= limit2 && metric.evidence < limit3) {
            //Medio
            node.evidence = 1;
        }else if (metric.evidenceVal >= limit1 && metric.evidence < limit2) {
            //Ruim
            node.evidence = 2;
        } else if (metric.evidenceVal < limit1) {
            //Muito ruim
            node.evidence = 2;
        }
    }
```

□ □ □ □ □ □ □ Definir evid ncia

Existem duas formas de definir a evidência de um nó. A primeira seria definir através da escala ordinal, que usa o □ □ □□ □ □□ □□ □ □, em específico a função □ □ □ □ □ □□ □□□□ □ . □ A evidência será definida de acordo com o parâmetro passado. Caso itemName seja AVEREGE o node.evidence será 1, já para BAD ou GOOD, o node.evidence será 1 ou 0, respectivamente.

```
//Setando o valor da evidência do nó
if (itemName === 'AVERAGE') {
    $scope.currentNode.evidence = 1;
} else if (itemName === 'GOOD') {
    $scope.currentNode.evidence = 0;
} else if (itemName === 'BAD') {
    $scope.currentNode.evidence = 2;
}
```

A segunda forma de definir a evidência é usando escalas intervalar, seja ela de 3 ou 5 pontos. Tal operação é realizada pela função updateE videnceN od , no m etric$ caleP opU P .controller.

```
$scope.updateEvidenceNode = function (metric) {
    if($scope.node.threshold.scale === '3-POINT_INTERVAL'){

        $scope.node.threshold.limits[2] = null;
        $scope.node.threshold.limits[3] = null;

        limit1 = $scope.node.threshold.limits[0];
        limit2 = $scope.node.threshold.limits[1];
        if (metric.evidenceVal > limit2) {
            node.evidence = 0;
        } else if (metric.evidenceVal < limit1) {
            node.evidence = 2;
        } else {
            node.evidence = 1;
        }

    }else if($scope.node.threshold.scale === '5-POINT_INTERVAL'){

        limit1 = $scope.node.threshold.limits[0];
        limit2 = $scope.node.threshold.limits[1];
        limit3 = $scope.node.threshold.limits[2];
        limit4 =$scope.node.threshold.limits[3];

        if (metric.evidenceVal > limit4) {
            //Muito bom
            node.evidence = 0;
        } else if (metric.evidenceVal >= limit3 && metric.evidence < limit4) {
            //Bom
            node.evidence = 0;
        } else if (metric.evidenceVal >= limit2 && metric.evidence < limit3) {
            //Medio
            node.evidence = 1;
        }else if (metric.evidenceVal >= limit1 && metric.evidence < limit2) {
            //Ruim
            node.evidence = 2;
        } else if (metric.evidenceVal < limit1) {
            //Muito ruim
            node.evidence = 2;
        }
    }
```

SideBar

Na pasta components/sidenav estão presentes os arquivos necessários para o funcionamento do menu lateral. No arquivo sidenav.html está presente uma lista de itens, em que cada item representa uma funcionalidade da aplicação.

No controller, também está presente a função getProjects , que pega todos os projetos do sistema , para exibir no html da sidebar.

```
/**
 * get all projects
 */
getProjects = function () {
    ProjectFactory.getProjects().then(function (result) {
        $scope.projects = result.data;
    }, function () {
        $scope.showNetworkError('NETWORK_ERROR');
    });
};
```

NavegationBar

Na pasta components/navegationBar estão presentes os arquivos necessários para o funcionamento do menu lateral. No arquivo sidenav.html está presente uma lista(navigationElements) de itens, em que cada item representa um estado anterior ou atual da navegação.

```
<md-toolbar layout="row" class="md-hue-2 navigation-bar" id="navbar">
    <div class="md-toolbar-tools">
        <md-list>
            <md-list-item ng-repeat="element in navigationElements" class="navigation-list-item">
                <md-icon class="navigation-element">keyboard_arrow_right</md-icon>
                <a class="navigation-element" ng-if="element.stateName"
                   ui-sref="{{element.stateName}}({{element.params}})">{{element.viewName | translate | capitalize}}</a>
                <p class="navigation-element" ng-if="!element.stateName">{{element.viewName | translate | capitalize}}</p>
            </md-list-item>
        </md-list>
    </div>
</md-toolbar>
```

Back end

Vis o Geral

O Back-end trata-se de uma API feita para se comunicar com o front-end da aplicação SAM. Utilizando o SpringBoot foram criados os controllers, modelos, serviços e repositórios para tratar os dados de projetos, objetivos, métricas, da rede, dentre outros objetos.

Depend ncias

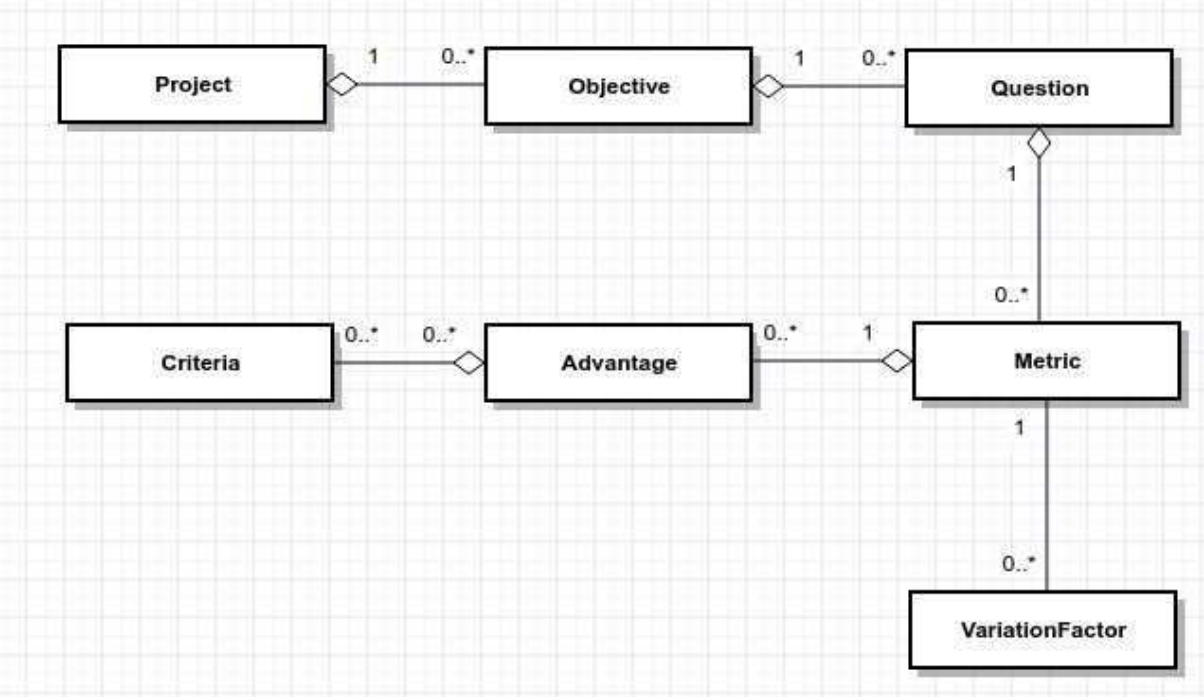
O aplicativo possui as seguintes dependências:

Depend ncia	Vers o	Descriç o
Spring Boot	1.5.3.RELEASE	Framework que auxilia na configuração do projeto Java.
MySql	----	Sistema de gerenciamento de banco de daos que utiliza a linguagem SQL.
Junit	4.12	Um framework simples de criação de testes de unidade.
Mockito	1.10.19	Biblioteca que ajuda na criação de mock para testes de unidade.
hamcrest	1.3	Auxilia na criação de testes de unidade.
Java Servlet API	3.1.0	Auxilia na criação de servidores para responder as requisições.
JSONPath	2.2.0	
SpringFox	2.4.0	
JSON	20140107	

Banco de Dados

Para o desenvolvimento foi escolhido o banco de dados mysql, que é escrito em linguagem SQL. Para utilizá-lo juntamente com o SpringBoot foi utilizado o hibernate e o jpa para facilitar a manipulação dos objetos.

Os principais modelos do sistema são: Project, Objective, Question, Metric, VariationFactor, Advantage e Criteria. A seguir, detalhes da relação desses modelos:



A partir das informações dos modelos acima é possível gerar a rede bayesiana, a qual tem como modelos: Network, Node e ParentNode.

Project

O modelo Project (Projeto) define o esquema de um projeto no Banco de Dados, sendo assim, leva-se em consideração que um projeto, pode possuir os atributos listados abaixo.

- name: Nome do projeto.
- function: Função da pessoa que está cadastrando o projeto.
- experienceTime: Tempo de experiência do usuário criando o projeto em meses.
- duration: Duração do projeto em meses.
- amountOfPeople: Quantidade de pessoas no projeto.
- methodologies: Metodologias utilizadas no projeto.

- projectType: Tipo do projeto, deve ser protótipo ou produto.
- objectives: Lista de objetivos do projeto.

Objective

O modelo Objective (Objetivo) define o esquema de um objetivo no Banco de Dados, sendo assim, leva-se em consideração que um objetivo, pode possuir os atributos listados abaixo.

- name: Nome do objetivo.
- objectOfStudy: Objeto de estudo do objetivo.
- purpose: Propósito do objetivo.
- viewPoint: Ponto de visão do objetivo.
- qualityFocus: Foco de qualidade do objetivo.
- context: Contexto do objetivo, nome do projeto o qual o objetivo está inserido.
- questions: Lista de questões do objetivo.

Question

O modelo Question (Questionamento) define o esquema de um Questionamento no Banco de Dados, sendo assim, leva-se em consideração que um Questionamento, pode possuir os atributos listados abaixo.

- question: Questionamento.
- metrics: Lista de métricas do questionamento.

Metric

O modelo Metric (Métrica) define o esquema de uma métrica no Banco de Dados, sendo assim, leva-se em consideração que uma métrica, pode possuir os atributos listados abaixo.

- name: Nome da métrica.
- description: Descrição da métrica.
- variationFactors: Lista de fatores de variação da métrica.
- advantages: Vantagens relacionadas a métrica.
- baselineHypothesis: Valor estimado da métrica.
- intendedUse: Uso pretendido da métrica.

VariationFactor

O modelo VariationFactor (Fator de variação) define o esquema de um fator de variação no Banco de Dados, sendo assim, leva-se em consideração que um fator de variação, pode possuir os atributos listados abaixo.

- description: Descrição do fator de variação.
- variationMetric: Métrica a qual o fator de variação pertence.
- variationHypothesis: Hipótese de variação.

Advantage

O modelo Advantage (Vantagem) define o esquema de uma vantagem no Banco de Dados, sendo assim, leva-se em consideração que uma vantagem, pode possuir os atributos listados abaixo.

- name: Nome da vantagem.
- criteria: Lista de critérios pertencente a vantagem.

Criteria

O modelo Criteria (Critério) define o esquema de um critério no Banco de Dados, sendo assim, leva-se em consideração que um critério, pode possuir os atributos listados abaixo.

- name: Nome do critério.
- selected: Boolean que representa se o critério está selecionado.
- justification: Justificação da seleção ou não seleção do critério.

Network

O modelo Network (Rede) define o esquema de uma rede bayesiana no Banco de Dados, sendo assim, leva-se em consideração que uma rede, pode possuir os atributos listados abaixo.

- status: Resposta da API ao solicitar requisição da rede.
- objective: Objetivo da rede.
- nodes: Lista de nós pertencentes a rede.

Node

O modelo Node (Nó) define o esquema de um nó da rede bayesiana no Banco de Dados, sendo assim, leva-se em consideração que um nó pode possuir os atributos listados abaixo.

- DEFAULT_STATES: Valores default dos estados do nó.
- DEFAULT_STATES_NAMES: Nomes default dos estados do nó.
- MANUAL_FUNCTION: Função manual.
- DEFAULT_MIN_VARIANCE: Valor default da variância.
- variance: Variância do nó.
- objectiveId: Id do objetivo que o nó pertence.
- nodeSource:
- nodeSourceId:
- dataSource:
- min:
- max:
- hasReabilityModel:
- states: Estado do nó.
- evidence: Evidência do nó.
- evidenceVal:
- name:
- statesName:
- npt:
- function:
- id:
- threshold:
- parents: Lista de pais do nó.

ParentNode

O modelo ParentNode (Nó pai) define o esquema de um nó pai da rede bayesiana no Banco de Dados, sendo assim, leva-se em consideração que um nó pai pode possuir os atributos listados abaixo.

- son_id: Id do filho do nó pai.
- node_id: Id próprio nó pai.

- weight: Peso do nó pai.
- name: Nome do nó pai.
- node_states: Lista de estados do nó pai.

Threshold

O modelo Threshold define o esquema de uma threshold no Banco de Dados, sendo assim, leva-se em consideração que uma threshold pode possuir os atributos listados abaixo.

- min: Limite inferior da threshold.
- max: Limite superior da threshold
- limits: Limiares da threshold.

Role

O enumeration Role define os possíveis papéis que um uma pessoa pode estar envolvida no projeto, que podem ser possuir os seguintes valores de descrição:

- DEVELOPER
- STAKEHOLDER
- TESTER
- QA

Controllers

ProjectCtrl ObjectiveCtrl QuestionCtrl MetricCtrl

Os controllers de Project, Objective, Question e Metric, têm a funcionalidade de obter todos objetos, de determinado tipo, cadastrados a partir da rota pré-definida do objeto + “/list”, além disso, todos esses controllers têm funcionalidade de POST, DELETE, PUT, e GET por id específico.

NetworkCtrl

O controller Network (Rede) é responsável pelas requisições referentes a rede bayesiana.

AdvantagesCtrl

O controller de Advantages (Vantagens) é responsável por retornar as vantagens disponíveis na API, através de sua única rota “advantages/get”.

ValidationCtrl

O controller de Validation (Validação) é responsável por validar as requisições no formato Json e emitir exceções caso necessário.

NodeCtrl

O NodeCtrl é responsável pelas requisições referentes aos nós da rede bayesiana, com operações de get, post, put e delete.