Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Programa de Pós-Graduação em Ciência da Computação

Uma Abordagem para a Definição de Valores de Referência de Métricas de Software Baseada em Contexto usando Redes Bayesianas

Leonardo da Costa Santos

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

Área de Concentração: Ciência da Computação

Linha de Pesquisa: Engenharia de Software

Hyggo Oliveira de Almeida

(Orientador)

Campina Grande, Paraíba, Brasil

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

S237a Santos, Leonardo da Costa.

Uma abordagem para a definição de valores de referência de métricas de software baseada em contexto usando Redes Bayesianas / Leonardo da Costa Santos. – Campina Grande, 2017.

67 f. : il. color.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2017.

"Orientação: Prof. Dr. Hyggo Oliveira de Almeida".

Referências.

1. Engenharia de Software. 2. Valores de Referência de Métricas de Software. I. Almeida, Hyggo Oliveira de. II. Título.

CDU 004.41(043)

Resumo

Métricas possuem um papel fundamental no processo de gerenciamento da qualidade de software. Apesar dos seus potenciais benefícios, elas geralmente são utilizadas apenas para quantificação, não oferecendo suporte adequado à tomada de decisão durante o ciclo de vida do software. Para potencializar a utilização das métricas, é essencial definir valores de referência significativos (i.e., *thresholds*), atribuindo, assim, significado para os números coletados. O objetivo deste trabalho é apresentar uma abordagem para definição de valores de referência de métricas de software de acordo com o contexto do projeto. A abordagem consiste em definir os fatores de contexto que influenciam os valores de referência da métrica em questão a partir de conhecimento elicitado de especialistas. Essas informações são utilizadas para construir uma rede Bayesiana que pode ser utilizada para auxiliar na tomada de decisão. A solução proposta foi avaliada por meio de um estudo piloto realizado com três gerentes de projetos reais de desenvolvimento de software. Os dados foram coletados com os profissionais para construir redes Bayesianas, para identificar e validar os valores de referência para as métricas *Número de Bugs Minor*, *Número de Alertas de Análise Estática* e *Porcentagem de Cobertura de Código*. A abordagem proposta mostrou-se promissora para auxiliar os profissionais a identificar valores de referência representativos, potencializando tomadas de decisões mais assertivas no processo de gerenciamento de projetos de software.

Abstract

Metrics play a key role in the software quality management process. Despite their potential benefits, they are generally only used for quantification, not providing adequate support to the decision-making process during the software's life cycle. To enhance the use of metrics, it is essential to define meaningful reference values (i.e., thresholds), thus giving meaning to the data collected. This work aims to propose an approach to define the software metrics' reference values according to the project's context. The approach consists of using the specialists' elicited knowledge to define context factors that influence the metric's reference values. This information is used to build a Bayesian network that can be used to aid in the decision-making process. The proposed solution was evaluated through a pilot study conducted with three managers of real software development projecst. Data were collected from the software project managers in order to build Bayesian networks to identify and validate reference values for the *Number of Minor Bugs*, the *Number of Static Analysis Alerts*, and the *Code Coverage Percentage* metrics. Each metric was validated in three scenarios. The proposed approach has shown to be promising in helping professionals to identify representative reference values, promoting a more assertive decision making when it comes to the software project management process.

Agradecimentos

Agradeço primeiramente a Deus e a minha família por terem me educado de forma que eu me tornasse um homem que está sempre em busca de expandir o seu conhecimento.

Agradeço à minha esposa e amigos que me deram força e me ajudaram a manter o foco para que eu continuasse me esforçando ao máximo para concluir este trabalho.

Finalmente, agradeço ao professor Hyggo Almeida pela orientação e por ajudar a me tornar um melhor pesquisador, e a Mirko Perkusich e a Renata Saraiva, que me ajudaram bastante na concepção da ideia central da pesquisa e na coleta dos dados.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Em linhas gerais, medições possuem um papel fundamental no processo de gerenciamento da qualidade de software, contribuindo para a avaliação da eficácia dos métodos, ferramentas e processos de desenvolvimento [39]. Existem diversas razões para realizar medições de processo, produto e recursos de software, tais como: (i) auxiliar no planejamento do projeto; (ii) determinar os pontos fortes e fracos do processo e do produto; e (iii) avaliar o impacto de determinada técnica utilizada [2]. Dessa forma, os dados de métricas coletados a partir da medição podem auxiliar no monitoramento e no controle do projeto, uma vez que permitem o acompanhamento do seu progresso.

Apesar dos potenciais benefícios das métricas, elas são geralmente utilizadas apenas para quantificação, não oferecendo suporte à tomada de decisão [1]. Esse é um dos principais motivos pelos quais métricas não são efetivamente utilizadas na indústria [15].

Para potencializar a utilização das métricas, é essencial que se definam valores de referência (i.e., *thresholds*) significativos. A partir desses valores, os engenheiros de software poderão ter uma melhor interpretação das métricas coletadas e, consequentemente, poderão tomar decisões mais assertivas durante o projeto.

Além disso, os valores definidos devem ser representativos, ou seja, devem condizer com as características do projeto em questão. Valores não representativos potencializam decisões incorretas. Se o valor de referência for subestimado (e.g., definir que a porcentagem de cobertura dos testes é 60%, quando esse valor de referência é insuficiente), pode-se entregar software com baixa qualidade. Por outro lado, se ele for superestimado (e.g., definir que a porcentagem de cobertura dos testes é 80%, quando não há necessidade real de implementar

tantos testes), pode-se entregar um software com custo maior que o necessário.

1.1 Problemática

Na literatura, várias abordagens têm sido propostas para a identificação de valores de referência, como nos trabalhos de Alves *et al.* [1], Shatnawi *et al.* [45], [44], Herbold *et al.* [22], Ferreira *et al.* [15], Zhang *et al.* [53], Oliveira *et al.* [36], [34], [35] e Foucault *et al.* [17]. Apesar de apresentarem resultados promissores, esses trabalhos possuem uma ou mais limitações, incluindo a dependência de ter uma base de dados históricos e a desconsideração do contexto (i.e., características de cada projeto, tais como: complexidade do produto, qualidade da equipe, etc.) no qual a métrica foi coletada.

Por exemplo, Ferreira *et al.* [15] usaram a ferramenta *EasyFit* [29] para encontrar a distribuição que mais se assemelha à distribuição da métrica, definindo, por meio de cortes dos quartis, a categorização dos valores de referência em: bom, moderado e ruim. Foucault *et al.* [17] utilizaram análise estatística (i.e., *bootstrap* [11]) para estimar o intervalo de confiança dos quartis, derivando o valor de referência a partir destes.

Nesse cenário, os trabalhos supracitados foram usados apenas para derivar valores de referência para métricas de código-fonte. Além disto, por usarem métodos estatísticos, tais trabalhos necessitam de dados históricos.

Shatnawi *et al.* [45] utilizaram curvas ROC (do inglês, *Receiver Operating Characteristic*) para realizar a associação da métrica aos erros e encontrar o valor de referência que melhor fornecesse justificativa para esses erros. Posteriormente, Shatnawi *et al.* [44] propuseram uma abordagem que realiza uma transformação logarítmica nos dados para deixá-los com maior simetria, o que segundo os autores diminui a perda de informação. Nessa abordagem, extraiu-se o valor de referência a partir da média e do desvio padrão da distribuição relacionada à métrica.

Não obstante, as abordagens propostas por Shatnawi *et al.* foram usadas apenas para métricas de código-fonte e necessitam de dados históricos. Além disso, elas não levam em consideração as características de cada projeto (i.e., contexto), podendo derivar valores de referência não representativos e, consequentemente, potencializando decisões incorretas, uma vez que em Zhang *et al.* [53] foi demonstrado que as distribuições dos valores das

métricas sofrem influência do contexto, dando indícios de que seus valores de referência também sofrem.

Portanto, com base nas limitações dos trabalhos supracitados, o problema técnico definido no presente documento é: como definir valores de referência representativos para métricas de software, considerando o contexto do projeto e a ausência de dados históricos?

1.2 Objetivos

Neste trabalho, tem-se como objetivo descrever uma abordagem para definir valores de referência para métricas de software, de forma que:

(i) possa ser usada para métricas de processo ou projeto;

(ii) não necessite de dados históricos;

(iii) leve em consideração as características de cada projeto, obtendo valores representativos do contexto;

(iv) viabilize um processo sistemático de definição de valores de referência de métricas de software.

Dessa forma, a abordagem deve definir os fatores de contexto que influenciam os valores de referência de uma métrica a fim de garantir que sejam obtidos valores representativos. Além disso, os fatores devem ser elicitados a partir de conhecimento de especialistas, uma vez que a abordagem não necessita de dados históricos.

O problema de relacionar os fatores de contexto e o valor de referência da métrica em questão exige raciocínio causal e incerteza aleatória (i.e., variabilidade intrínseca), que pode ser reduzida coletando mais dados ou elicitando conhecimento de especialistas do domínio. Dado que o propósito é que a solução seja utilizada por profissionais, as inferências precisam ser claras. Além disso, os modelos devem ser adaptáveis e devem servir de auxílio à tomada de decisão. De acordo com Verbert *et al.* [52], o uso de redes Bayesianas é apropriado nesse contexto.

O objetivo principal da pesquisa pode ser dividido nos seguintes objetivos específicos:

(i) propor uma abordagem baseada em redes Bayesianas para definir valores de referência para métricas de software;

(ii) proporcionar aos gerentes de projetos de desenvolvimento de software um processo sistemático que leve em consideração as características de cada projeto, produzindo valores de referência significativos;

(iii) aplicar a abordagem em projetos de desenvolvimento de software para observar sua utilidade em projetos reais.

1.3 Relevância

Valores de referência não são conhecidos para várias métricas propostas na literatura. Sem esses valores, o uso de métricas fica restrito apenas à quantificação, impossibilitando seu uso para tomada de decisão, uma vez que não há diferenciação adequada de um bom ou mau projeto. Com a identificação e utilização adequada desses valores, possíveis problemas de software poderiam ser identificados durante seu desenvolvimento, resultando em um projeto com maior qualidade e, consequentemente, com menor custo de manutenção.

Para resolver esse problema, várias abordagens foram propostas [1; 36; 34; 35; 45; 44; 15]. Apesar dos esforços, todos esses trabalhos possuem uma ou mais limitações, incluindo a dependência de ter uma base de dados históricos e a desconsideração do contexto no qual a métrica foi coletada.

1.4 Metodologia

Para que fosse possível atender os objetivos do presente trabalho, inicialmente foram investigadas na literatura tecnologias que auxiliassem a representação do conhecimento de especialistas, uma vez que um dos objetivos seria a não necessidade de dados históricos.

Definido com base em Verbert *et al.* [52], que redes Bayseianas é apropriada para esse cenário, passou-se a investigar se o contexto influencia os valores de referência, um vez que em Zhang *et al.* [53] foi demonstrado que as distribuições dos valores das métricas sofrem influência do contexto, dando indícios de que seus valores de referência também sofrem.

Para avaliar se o contexto influencia os valores de referência, realizou-se um estudo empírico composto por dois quase-experimentos. Para cada quase-experimento usou-se uma abordagem presente na literatura para derivar os valores de referência para as seis métricas de Chidamber e Kemerer (CK) [7]. No estudo, os valores de referência definidos são a variável dependente. Como fator, usou-se uma variável com dois possíveis tratamentos: considerar o contexto ou não.

O estudo apresentou evidências de que, no domínio das métricas de CK, é relevante considerar o contexto para definir valores de referência. Esse estudo foi publicado no artigo *"An empirical study on the influence of context in computing thresholds for Chidamber and Kemerer metrics"* [41], no *The 29th International Conference on Software Engineering & Knowledge Engineering*, realizado no ano de 2017 em *Wyndham Pittsburgh University Center, Pittsburgh, USA*.

Uma vez definida a tecnologia que seria usada para representação do conhecimento de especialistas e sabendo que há fortes evidências de que o contexto influencia os valores de referência, iniciou-se a definição da abordagem foco deste trabalho. Trata-se de um processo sistemático, que foi dividido em etapas e subetapas que vão desde a seleção das métricas usadas no projeto até a construção da rede Bayesiana que será usada para definir os valores de referência.

A saída da rede Bayesiana é o valor de referência da métrica representada pela rede e, como entrada, têm-se todos os fatores de contexto definidos pelo especialista que usará a abordagem, representando o cenário em que o projeto está sendo desenvolvido.

Entende-se por contexto qualquer fator que possa modificar o valor de referência da métrica, por exemplo, a "Criticidade do projeto" é um dos fatores de contexto para a métrica "Número de Bugs", uma vez que quanto mais crítico for o projeto, menor será o "Número de Bugs" aceitos antes da entrega do produto e, consequentemente, menor será o valor de referência.

Um dos problemas encontrados na criação da rede Bayesiana é a complexidade na definição da Tabela de Probabilidade de Nós (TPN), uma vez que estas crescem exponencialmente de acordo com o número de nós-pai [25]. Para diminuir a complexidade da rede e facilitar a sua construção, definiu-se que precisariam ser criados nós intermediários utilizando a técnica de Fenton *et al.* [13], de forma que nenhum nó fique com mais de três pais.

Para validação da abordagem, um estudo piloto foi realizado com três especialistas envolvidos diretamente em um projeto de desenvolvimento de software. Neste estudo, foram identificadas as métricas usadas no projeto, construídas as redes Bayesianas para a definição dos valores de referência destas e, por meio de cenários, foi comparado o resultado esperado pelo especialista com o resultado gerado pela rede.

1.5 Contribuição e Resultados

A abordagem proposta neste trabalho viabiliza um processo sistemático dividido em etapas que englobam desde a seleção de métricas coletadas no projeto e identificação de fatores de contexto, até a construção das redes Bayesianas que modelam tais métricas.

Com a construção desses modelos, é possível identificar, de forma probabilística, os valores de referência baseados nos fatores de contexto, sem a necessidade de dados históricos. Ao aplicar a abordagem em determinado projeto, as saídas dos modelos podem ser observadas por seu gerente. Tais saídas irão indicar, em um intervalo pré-definido na construção da rede, qual valor de referência é representativo do projeto em questão, ou seja, qual valor é mais adequado às suas características, de forma que proporcione tomadas de decisão mais assertivas durante todo seu desenvolvimento.

A abordagem foi avaliada a partir de um estudo piloto realizado com três especialistas de projetos reais de desenvolvimento de software. Tal estudo consistiu na criação de uma rede Bayesiana para cada métrica usada no projeto: **Análise Estática**, **Número de Bugs Minor** e **Cobertura de Código**. Para cada métrica, um cenário foi criado e o resultado gerado a partir da execução da rede foi comparado ao resultado esperado pelo especialista. Em todos os cenários, exceto por um, relacionado à métrica **Número de Bugs Minor**, o resultado gerado foi igual ao esperado.

1.6 Organização da Dissertação

O restante deste documento está organizado da seguinte forma:

- no Capítulo 2, são apresentados conceitos fundamentais para o entendimento do restante do trabalho. Mais especificamente, na Seção 2.1.3 são apresentados trabalhos

relacionados que compõem o estado da arte, assim como suas limitações;

- no Capítulo 3, a abordagem proposta é apresentada;
- a validação da abordagem é feita no Capítulo 4, no qual se descreve o estudo piloto realizado, assim como uma breve discussão dos resultados obtidos e suas ameaças à validade;
- por fim, as considerações finais e propostas de trabalhos futuros são apresentadas no Capítulo 5.

Capítulo 2

Fundamentação Teórica e Estado da Arte

Na Seção 2.1 deste capítulo são descritos os conceitos fundamentais acerca do processo de medição de software, apresentando as três principais atividades desse processo: seleção de métricas, validação de métricas e definição de valores de referência. Em seguida, na Seção 2.2, aborda-se a temática de redes Bayesianas.

2.1 Processo de Medição

Medição é a representação objetiva de um conhecimento empírico acerca de uma entidade do mundo real [16]. 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, como linhas de código, *bugs*, manutenibilidade e eficiência. Quando o atributo é quantificável, ou seja, quando se atribui um valor a ele, tem-se uma medida. Ao combinar essa medida com uma informação útil, tem-se uma métrica (e.g., média de números de *bugs* por módulo) [30].

O processo de medição tem como foco a avaliação da eficácia de métodos, ferramentas e processos de desenvolvimento [46]. Uma das primeiras atividades desse processo é identificar qual desses usos deseja-se atribuir ao processo de medição. Em seguida, 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: (i) processos, que correspondem a quaisquer atividades relacionadas ao software, que ocorrem ao longo do tempo; (ii) produ-

tos, que correspondem a quaisquer artefatos que resultam dos processos; e (iii) recursos, que consistem em itens que são as entradas para os processos [12].

Assim como as entidades, os atributos também são classificáveis. Em linhas gerais, um atributo interno é aquele que pode ser medido puramente em termo de sua entidade, por exemplo, tamanho é um atributo interno de qualquer documento de software. A medição de um atributo interno é denominada "medição direta".

A medição indireta é aquela que mede o atributo externo, ou seja, o atributo que só pode ser medido em relação a como sua entidade interage com outras entidades do ambiente. Por exemplo, a confiabilidade de um programa é dependente não somente do programa em si, mas também do compilador, da máquina e do usuário. Nesse sentido, para medir esses atributos, é feita uma medição do software e assume-se que os atributos estão relacionados com as características de qualidade que são de interesse ao longo do processo de desenvolvimento de software [12].

Uma vez que não há métricas definitivas que possam ser prescritas para cada objetivo aplicado em qualquer área, é preciso selecionar as métricas que melhor representem o que se deseja medir (Seção 2.1.1). Além disto, embora muitas métricas tenham sido propostas ao longo dos anos, há evidências de que muitas delas são incompletas, ambíguas e abertas a diferentes interpretações [4], o que torna essa etapa bastante discutida entre os pesquisadores da área.

Após selecionar as métricas que serão usadas, sua validade deve ser verificada a fim de determinar se elas de fato medem o que se propõem a medir [42]. Os principais processos de validação de métricas são discutidos na Seção 2.1.2.

Por fim, como foco principal deste trabalho, é preciso definir valores de referência que possibilitem uma melhor interpretação dos valores coletados. Na Seção 2.1.3 são apresentadas as principais abordagens para definição de valores de referência presentes na literatura, assim como suas principais características e limitações.

2.1.1 Seleção de Métricas

Com o objetivo de selecionar quais métricas melhor representam o que se deseja medir, a etapa de seleção de métricas vem sendo discutida por diversos pesquisadores [2; 5; 27; 8; 6; 48; 28; 19].

Essa é uma etapa importante, uma vez que programas de medição bem estabelecidos são bastante influenciados pela seleção de métricas de software [19] e que, apesar de existirem centenas de métricas propostas na literatura, na maioria dos casos elas foram propostas em situações *ad hoc*, não havendo um guia de como medi-las e utilizá-las [5].

Bukhari *et al.* [5] apresentam uma discussão sobre os métodos de seleção de métricas de software disponíveis a fim de auxiliar os profissionais e pesquisadores a entender os pontos fortes e fracos desses métodos:

- **Best Professional Judgement(BPJ):** consiste em selecionar métricas a partir da combinação de vários fatores, como conhecimento do especialista, critério de decisão e objetivos do projeto [27];
- **Historical Precedence (HP):** consiste em montar uma lista de métricas utilizadas anteriormente em projetos equivalentes, avaliá-las e especificá-las para o novo projeto [8];
- **Web Quality Model (WQM):** consiste na classificação de métricas *web* a partir de características *web*, de características de qualidade e do processo do ciclo de vida do projeto [6];
- **Meta-metrics:** indica que as métricas que deveriam ser usadas para avaliar o sistema deveriam ser baseadas em suas próprias características específicas [48];
- **Multi-Criteria Decision Analysis (MCDA):** consiste em uma abordagem estruturada para a tomada de decisão que avalia quantitativamente as alternativas, baseando-se em: critério do projeto definido, opiniões de especialistas e preferências de *stakeholders* [28];
- **GQM-Decision Support Framework for Metric Selection (GQM-DSFMS):** consiste em um processo de seleção de métricas iterativo baseado em objetivos, incluindo um mecanismo de tomada de decisão em seleção de métricas, um repositório de métricas e atributos predefinidos e um modelo rastreável entre elementos GQM [19].

Segundo Bukhari et al. [5], apesar dos seis métodos apresentarem tanto pontos fortes quanto limitações, o método GQM-DSFMS é mais flexível e estruturado, uma vez que se

baseia nos objetivos principais da empresa. Por ser considerado uma evolução da abordagem tradicional, que usa a opinião de especialistas, sua estrutura ajuda os tomadores de decisão a selecionar apenas as métricas importantes, deixando de lado as métricas sem importância.

Levando em conta que a medição orientada a objetivo é considerada um fator de sucesso para a implementação de programas de medição e 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 [49], GQM vem sendo amplamente adotado e recomendado na literatura [49; 2].

2.1.2 Validação de Métricas

Parece claro afirmar que de nada adianta selecionar métricas se elas não forem representativas do atributo quantificado. Em outras palavras, métricas devem ser validadas para determinar se elas efetivamente medem o que se propõem a medir [42].

Por quase meio século, pesquisadores têm debatido sobre o que constitui uma métrica válida [33], suscitando inúmeras pesquisas que abordam a questão da validação de métricas [42; 12; 33; 23; 24; 47]. Uma vez que esse debate tem como ponto central critérios de validação de métricas de software, Meneely *et al.* [33] realizaram uma revisão sistemática acerca dos critérios de validação encontrados na literatura acadêmica. De acordo com os autores, os 47 critérios de validação encontrados, ilustrados no Quadro 2.1, representam uma visão diversa do que constitui uma métrica válida.

De forma geral, uma métrica que é válida para um determinado objetivo pode não ser válida para outro. Definir o objetivo de utilização de uma métrica é uma etapa crítica para validá-la. Ao tomar essa decisão, o líder de projeto pode especificar propriedades da métrica que são mais apropriadas ao seu uso. Meneely *et al.* [33] chamam isso de vantagem. Mostrar que uma métrica é uma representação significante de determinado atributo é uma vantagem e demonstrar que ela pode ser aplicada a um processo de desenvolvimento é outra vantagem.

Nesse contexto, Meneely *et al.* [33] observaram 11 vantagens comuns (Tabela 2.2) a partir dos 47 critérios encontrados. Cada vantagem é uma propriedade abstrata de uma métrica que um líder de projeto pode demonstrar usando seus critérios de validação associados.

A partir do mapeamento entre vantagens e critérios de validação, ilustrado na Tabela 2.3, Meneely *et al.* [33] apresentam um processo de aplicação de critérios capaz de auxiliar

Tabela 2.1: Lista dos 47 Critérios de Validação

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

Fonte: Adaptado de Meneely *et al.*, 2012 [33].

líderes de projetos na escolha dos critérios apropriados para uma métrica específica. Por exemplo, suponha que um líder de projeto propõe usar a métrica *YearsExperience*, referente ao somatório dos anos de experiência de cada integrante do time, para avaliar a habilidade do time em desenvolver software de alta qualidade.

(i) O líder determina a métrica *YearsExperience* para avaliar a capacidade da equipe em desenvolver software de alta qualidade;

(ii) O líder acredita que essa métrica vai permitir economizar tempo e dinheiro, quando aplicada a um processo (i.e., ela possui a vantagem *Efficiency*);

(iii) Ao verificar a Tabela 2.3, percebe-se que a vantagem *Efficiency* mapeia para os critérios de validação *Improvement Validity* e *Usability*;

Tabela 2.2: Vantagens de Métricas

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

Fonte: Adaptado de Meneely *et al.*, 2012 [33].

(iv) Uma métrica tem *Improvement Validity* se pode ser considerada uma melhoria das métricas existentes. *Improvement Validity* pode se referir à facilidade de medição, uma forte associação com um fator de qualidade ou uma representação mais próxima do atributo a ser medido. Por outro lado, métrica tem *Usability* se ela pode ser implementada de forma rentável em um programa de garantia de qualidade. Uma métrica que requer vários meses de computação, por exemplo, pode não ser considerada usável;

(v) Uma vez que o líder de projeto demonstra que a referida métrica segue ambos os critérios de validação, pode-se afirmar que ela é válida para o propósito pretendido.

Tabela 2.3: Mapeamento entre Critérios de Validação e Vantagens

#	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		

Continua na próxima página

#	Critérios	*Mathematical Soundness*	*Practicality*	*Correctness*	*Efficiency*	*Hypothesis-Strengthening*	*Meaningfulness*	*Decision-Informing*	*Quality-Focused*	*Theory-Building*	*Consensus Contribution*	*Difference-Detecting*
6	*Attribute Validity*						X				X	
7	*Causal Model Validity*		X					X		X		
8	*Causal Relationship Validity*		X					X		X		
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

Continua na próxima página

#	Critérios	*Mathematical Soundness*	*Practicality*	*Correctness*	*Efficiency*	*Hypothesis-Strengthening*	*Meaningfulness*	*Decision-Informing*	*Quality-Focused*	*Theory-Building*	*Consensus Contribution*	*Difference-Detecting*
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*					X				X		
39	*Representation Condition*	X					X					
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			

Fonte: Adaptado de Meneely *et al.*, 2012 [33].

2.1.3 Definição de Valores de Referência (Trabalhos Relacionados)

Dentre as métricas de software existentes na literatura, poucas possuem valores de referência conhecidos e determinados. Com base nisso, várias abordagens para identificar valores de referência para métricas têm sido propostas [1; 15; 36; 34; 35; 17; 45; 44].

Alves *et al.* [1] desenvolveram uma abordagem baseada em dados de medição obtidos a partir de um conjunto representativo de sistemas.

Essa abordagem possui diversas limitações: (i) seu uso é limitado a métricas de código-fonte; (ii) não pode ser usada em situações que não há dados históricos; (iii) não leva em consideração as características de cada projeto para a definição dos valores de referência, uma vez que em Zhang *et al.* [53] foi demonstrado que as distribuições dos valores das métricas sofrem influência do contexto, dando indícios de que seus valores de referência também sofrem; (iv) o processo de validação da abordagem verificou apenas se os valores de referência calculados eram representativos dos percentis escolhidos.

Ferreira *et al.* [15], por sua vez, usaram a ferramenta *EasyFit* para encontrar a distribuição que mais se assemelha à distribuição da métrica de forma que, se a distribuição encontrada tem um valor médio representativo, esse valor é usado como valor de referência; caso contrário, ocorre a quantificação da distribuição em boa, moderada ou ruim. Para essa análise, foram usados quarenta software *open-source* desenvolvidos em Java, com diferentes tamanhos, domínios e tipos. A partir desse conjunto de software foram derivados valores de referência para seis métricas: LCOM (*Lack of Cohesion of Methods*), DIT (*Depth in Tree*), COF (*Coupling Factor*), número de conexões aferentes, número de métodos públicos e número de atributos públicos.

A validação dessa abordagem ocorreu por meio de dois experimentos: no primeiro, foi avaliado se os valores de referência identificavam classes problemáticas e, no segundo, se esses valores identificavam classes bem projetadas. A comparação foi feita entre o resultado do valor de referência e uma inspeção manual da classe estudada.

A abordagem proposta por Ferreira *et al.* [15] foi usada apenas para métricas de código-fonte e não pode ser usada em situações que não há dados históricos.

Oliveira *et al.* [36; 34; 35], por outro lado, propuseram o conceito de valores de referência relativos. Tais valores são representados por um par (p, k), em que $p\%$ de classes de um programa deve ter $M \leq k$. M é o valor da métrica e p é o percentual mínimo, que deve estar abaixo do limite k. Os dados de setenta e nove aplicações foram utilizados para a derivação de valores de referência relativos para quatro métricas.

Na validação foram comparados os resultados das métricas com a avaliação de especialista quanto à qualidade da escrita da classe estudada. Apesar do seu potencial, a abordagem

foi usada apenas para métricas de código-fonte e necessita de dados históricos. Além disso, não leva em consideração as características de cada projeto (i.e., contexto).

Foucault *et al.* [17] definiram uma abordagem que calcula os valores de referência com base no contexto de cada projeto. Essa abordagem foi definida por meio da aplicação de dois métodos estatísticos: (i) *Double sampling* [50], que seleciona aleatoriamente amostras de projetos; e (ii) *Bootstrap*, que estima os valores de referência baseados em quartis. O principal ponto negativo desse trabalho foi o processo de validação, que foi apenas um teste para identificar a melhor configuração para a abordagem, uma vez que, segundo os autores, os dois métodos estatísticos são amplamente utilizados.

Por fim, em Shatnawi *et al.* [45] foi utilizado o método de precisão diagnóstica ROC (*Receiver Operating Characteristic*). Esse método contribuiu para a investigação relacionada à associação do valor da métrica aos erros encontrados, seja pela presença ou ausência de erros ou pelo nível do erro identificado, qual seja, baixo, médio ou alto. Além disso, como amostra do estudo, foram utilizadas três versões do software Eclipse (2.0, 2.1 e 3.0). A análise foi feita a partir da definição de uma faixa de possíveis valores de referência para cada métrica. Para cada um desses valores, foi gerada uma tabela de classificação denominada Matriz de Confusão (i.e., *confusion matrix*), que determina, dentro de um conjunto de classes da amostra, o número de acertos e erros obtidos. Desse modo, cada Matriz de Confusão gerou um ponto na curva ROC, que representou a classificação do valor de referência a ser testado. O valor de referência a ser considerado foi aquele que apresentou um menor número de erros.

Em um trabalho mais recente, Shatnawi *et al.* [44] propuseram uma abordagem que inicialmente transformava a distribuição das métricas em uma distribuição logarítmica, deixando-a o mais próxima possível da distribuição normal. Em seguida, foi coletado o valor de referência temporário (T') utilizando a média (M) e o desvio padrão (SD), de forma que $T' = M + SD$ ou $T' = M - SD$. Por fim, o T' foi convertido à distribuição original por meio de exponenciação do T', gerando, assim, o valor de referência final. O processo para encontrar valores de referência para as métricas de CK foi replicado em 36 software *open-source*.

Nas duas abordagens propostas por Shatnawi *et al.* [45; 44], algumas limitações podem ser encontradas: (i) foram usadas apenas para métricas de código-fonte; (ii) necessitam de

dados históricos; (iii) não levam em consideração as características de cada projeto.

Na Tabela 2.4, as principais características e limitações dos trabalhos descritos anteriormente são apresentadas. Como pode ser observado, todas as soluções propostas na literatura dependem de dados históricos. Além disso, grande parte delas foca especificamente em métricas de código-fonte, em especial, em métricas de Orientação a Objeto.

Tabela 2.4: Principais Características e Limitações dos Trabalhos Relacionados

Abordagem	**Características**	**Limitações**
Alves *et al.* [1]	• Usa métodos estatísticos; • Classifica os valores de referência de acordo com o risco de falha em: *low*, *moderate*, *high* e *very-high*.	• Foi usado apenas para métricas de código-fonte; • Não pode ser usada em situações que não há dados históricos; • Não leva em consideração as características de cada software.
Oliveira *et al.* [36; 34; 35]	• Usa métodos estatísticos; • Propõe o conceito de valores de referência relativos.	
Shatnawi *et al.* [45]	• Utiliza o método de precisão diagnóstica ROC para investigar a relação entre o valor da métrica e os erros encontrados.	
Shatnawi *et al.* [44]	• Usa métodos estatísticos; • Transforma a distribuição das métricas em uma distribuição logarítmica, a fim de deixar os dados mais simétricos.	

Continua na próxima página

Abordagem	**Características**	**Limitações**
Ferreira *et al.* [15]	• Usou-se a ferramenta *EasyFit* para encontrar a distribuição que mais se assemelha à distribuição da métrica; • Estima os valores de referência baseados em quartis; • Leva em consideração o contexto, com base em Zhang *et al.* [53].	• Foi usado apenas para métricas de código-fonte; • Não pode ser usada em situações que não há dados históricos.
Foucault *et al.* [17]	• Definiu uma solução baseada em dois métodos estatísticos: *Double Sampling* e *Bootstrap*; • Estima os valores de referência baseados em quartis; • Leva em consideração o contexto, com base em Zhang *et al.* [53].	

Fonte: Do autor.

2.2 Redes Bayesianas

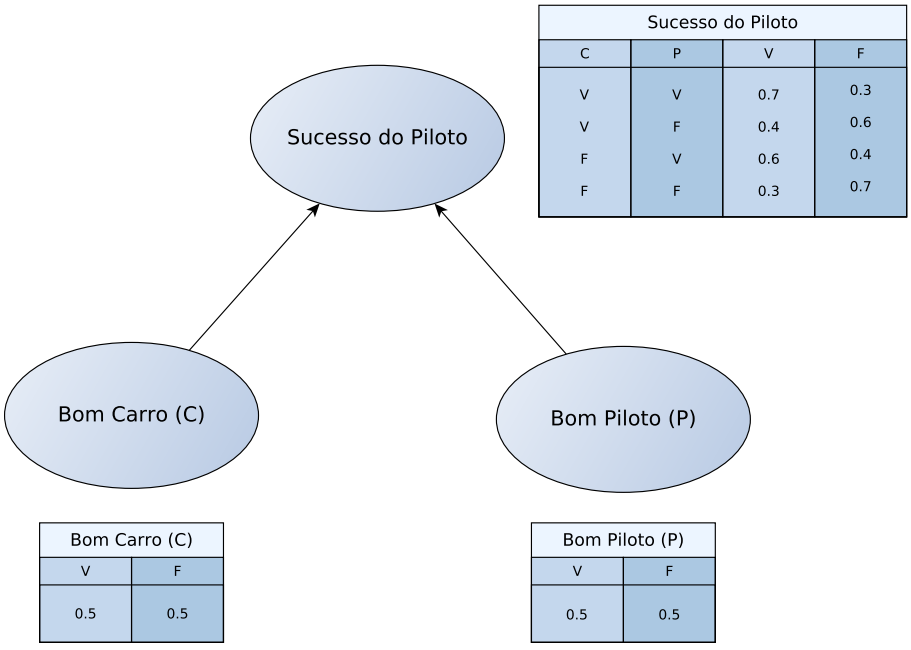
Redes Bayesianas são modelos gráficos probabilísticos usados para representar o conhecimento de um domínio incerto [40]. Uma rede Bayesiana B é um grafo acíclico direcionado (GAD) que representa a distribuição de probabilidade conjunta de variáveis aleatórias V. A rede é definida pelo par $B = \{G, \Theta\}$, onde G é o grafo acíclico no qual os nós $X_1, \ldots, X_{|V|}$ representam variáveis aleatórias e os arcos representam as dependências diretas entre essas variáveis [18]. O componente Θ representa o conjunto de parâmetros que quantificam a rede. Para cada possível valor x_i de cada variável aleatória $X_i \in V$, Θ existe um parâmetro $\Theta_{x_i|\Pi_{x_i}} = P_B(x_i|\Pi_{x_i})$, sendo que Π_{X_i} representa o conjunto de pais de X_i no grafo G.

Além disso, os nós são associados a funções de probabilidade sobre o conjunto V (Equação 2.1), que têm como entrada os conjuntos de valores dos nós-pais e calculam a probabilidade da variável representada pelo nó, geralmente apresentada uma tabela de probabilidade condicional, também chamada de tabela de probabilidade de nó (TPN).

$$P_B(X_1, \ldots, X_{|V|}) = \prod_{i=1}^{|V|} P_B(X_i|\Pi_{X_i}) = \prod_{i=1}^{|V|} \Theta_{X_i|\Pi_{X_i}} \tag{2.1}$$

Na Figura 2.1, ilustra-se um exemplo clássico de uma rede Bayesiana [31]. Nota-se que o valor do nó "Sucesso do Piloto" depende do valor dos seus pais. Apesar da rede ser direcionada, ou seja, os arcos representarem a direção da conexão causal entre as variáveis, informações podem propagar em qualquer direção no grafo [37]. Por exemplo, na Figura 2.1, caso o piloto tenha obtido sucesso e a qualidade do carro for ruim, é provável que o piloto seja bom.

Figura 2.1: Exemplo de Rede Bayesiana



Fonte: Adaptado de Medeiros, 2015 [31].

As redes Bayesianas oferecem um modelo adequado para tratar as incertezas em um produto e processo de desenvolvimento de software, uma vez que, segundo Ziv e Richardson [55], as incertezas são inerentes e inevitáveis nesse cenário, e devem ser modeladas e gerenciadas explicitamente por meio de técnicas de modelagem de incerteza.

Além disso, a estrutura gráfica das redes Bayesianas facilita a comunicação, a análise e as modificações entre os profissionais de software. Por fim, elas também se destacam pela adequação a situações onde não há dados históricos, possibilitando a criação destas por meio do conhecimento de especialistas, cenário comum nas fases iniciais do processo de desenvolvimento de software [51].

No entanto, a construção das Tabelas de Probabilidade dos Nós (TPN) pode fazer com que o uso de redes Bayesianas tenha uma complexidade maior principalmente para profissionais de software sem um conhecimento de probabilidade.

Existem duas formas de se coletar dados e definir as TPN de uma rede Bayesiana: utilizando base de dados por meio de um processo chamado aprendizado em lotes [21] ou utilizando opinião de especialistas [38]. Como geralmente não há dados históricos nas fases iniciais do processo de desenvolvimento de software, será utilizado neste trabalho a construção das TPN com base na opinião de especialistas.

Fenton *et al.* [14] mostraram que vários tipos de inconsistências podem ocorrer se especialistas tentarem definir as TPN de uma rede com nós com um número muito alto (por exemplo, 125) de estados. Nesse caso, esse método de definição de TPN pode ser inviável.

Felizmente, existem algumas abordagens para simplificar a definição das TPN, como a utilização de nós ranqueados [14], assim como se faz neste trabalho. Essa abordagem é baseada numa distribuição normal duplamente truncada.

Capítulo 3

Abordagem Proposta

A abordagem proposta consiste em um processo sistemático para construção de uma rede Bayesiana a ser usada para, de acordo com fatores de contexto e sem a necessidade de dados históricos, identificar o valor de referência de uma métrica de software.

Os fatores de contexto e a quantificação de sua influência no valor de referência da métrica em questão são definidos por meio da elicitação do conhecimento de especialistas. Por exemplo, de acordo com o conhecimento do Gerente de Qualidade, define-se quais fatores influenciam no valor de referência da métrica *Cobertura de testes*. Em seguida, ao coletar os dados referentes a esses fatores em um projeto, calcula-se uma faixa de valor de referência adequada para ele (e.g., entre 70% e 80%). Por fim, para cada métrica que será usada para auxiliar na tomada de decisão, uma rede Bayesiana deve ser construída.

Em linhas gerais, redes Bayesianas são modelos gráficos probabilísticos usados para representar o conhecimento de um domínio incerto. A rede Bayesiana B é um grafo acíclico direcionado que representa a distribuição de probabilidade conjunta de variáveis aleatórias V. A rede é definida pelo par $B = \{G, \Theta\}$, em que G é o grafo acíclico no qual os nós $X_1, \ldots, X_n$ representam variáveis aleatórias e os arcos representam as dependências diretas entre essas variáveis.

Na solução proposta, os nós $X_1, \ldots, X_n$ e as dependências entre eles são definidos de acordo com o conhecimento elicitado de especialistas do domínio utilizando uma abordagem semi-automática: o método de nós ranqueados [14]. Por outro lado, destaca-se que também é possível elicitar todas as probabilidades condicionais manualmente ou utilizar outra abordagem semi-automática (e.g., algoritmo de soma ponderada [9]), se desejado.

Dessa forma, a abordagem proposta não depende de dados históricos para ser utilizada. Em outras palavras, é possível definir valores de referência durante a fase inicial dos projetos ou quando seus dados não são facilmente coletados utilizando um processo sistemático.

Por outro lado, assim como em Srdjana *et al.* [10] e Mendes *et al.* [32], é possível, a partir dos fatores identificados e das métricas selecionadas, coletar dados para criar um histórico e utilizá-los para definir automaticamente as probabilidades condicionais.

A abordagem proposta é composta por quatro principais etapas:

- **Etapa 1 - Seleção da métrica:** Uma métrica m deverá ser selecionada a fim de se identificar seu respectivo valor de referência;
- **Etapa 2 - Definição da escala:** Uma escala intervalar deve ser definida para a métrica m selecionada na fase anterior, na qual faixas de valores são definidas para m (e.g., $[0, 5], [5, 10], [10, \infty]$);
- **Etapa 3 - Discretização da escala:** A escala intervalar é discretizada para uma escala ordinal, assim como em [25] (e.g., o intervalo [5, 10] é mapeado para o estado *Médio*);
- **Etapa 4 - Construção da rede Bayesiana:** A partir das informações definidas nas etapas anteriores, a rede Bayesiana da métrica m deve ser construída.

São ilustradas na Figura 3.1 as quatro subetapas necessárias à construção da rede Bayesiana (Etapa 4). Elas serão melhor detalhadas na seção seguinte.

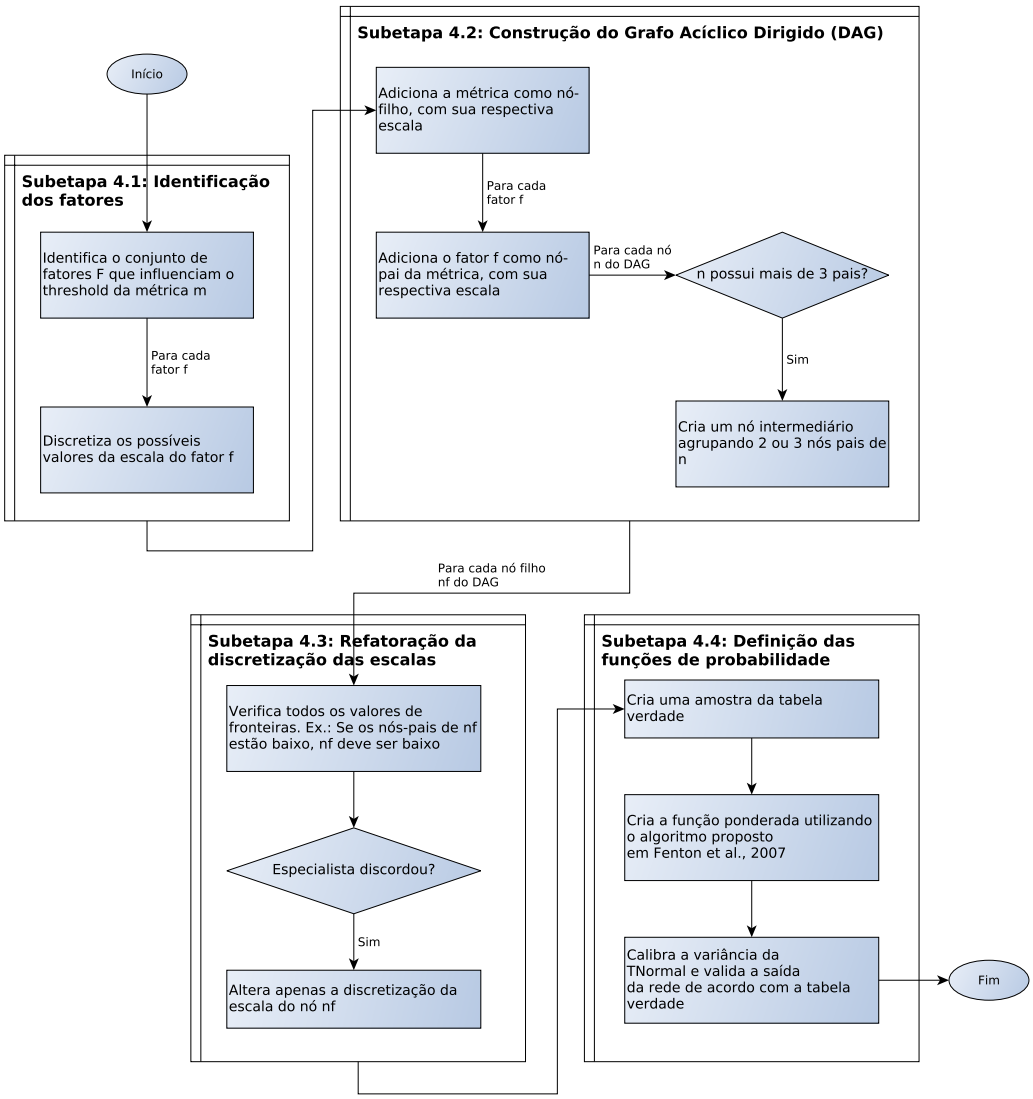
3.1 Construção da Rede Bayesiana

3.1.1 Identificação dos Fatores

Na **Subetapa 4.1** deve ser identificado o conjunto de fatores F (i.e., fatores de contexto) que podem de alguma forma, seja positivamente ou negativamente, influenciar o valor de referência da métrica m.

É importante salientar que os fatores de contexto são aqueles que influenciam o valor de referência da métrica, e não o valor da métrica em si. Por exemplo, é possível pensar que a "volatilidade dos requisitos" seja um fator plausível para métrica "Número de Bugs",

Figura 3.1: Etapa 4 - Construção da Rede



Fonte: Do autor.

uma vez que quanto mais voláteis forem os requisitos, menor será o tempo para testes e, consequentemente, menor será o "Número de Bugs" encontrados. No entanto, apesar do "Número de Bugs" encontrados ser melhor, o valor de referência da métrica não deve ser menor, o que descarta esse elemento como fator de contexto.

Por outro lado, a "Criticidade do projeto" é um dos fatores de contexto para a métrica "Número de Bugs", uma vez que quanto mais crítico for o projeto, menor será o "Número de Bugs" aceitos antes da entrega do produto e, consequentemente, menor será o valor de referência.

Uma vez definidos os fatores de contexto para a métrica m, suas escalas deverão ser

ajustadas considerando a escala intervalar definida na **Etapa 2**, e os valores de cada fator devem ser discretizados. Por exemplo, para a métrica "Número de Bugs", considerando a escala [Bom, Médio, Ruim], os valores podem ser discretizados da seguinte forma: Bom (≤ 5), Médio ($6 - 10$) e Ruim (> 10).

3.1.2 Construção do Grafo Acíclico Dirigido (DAG)

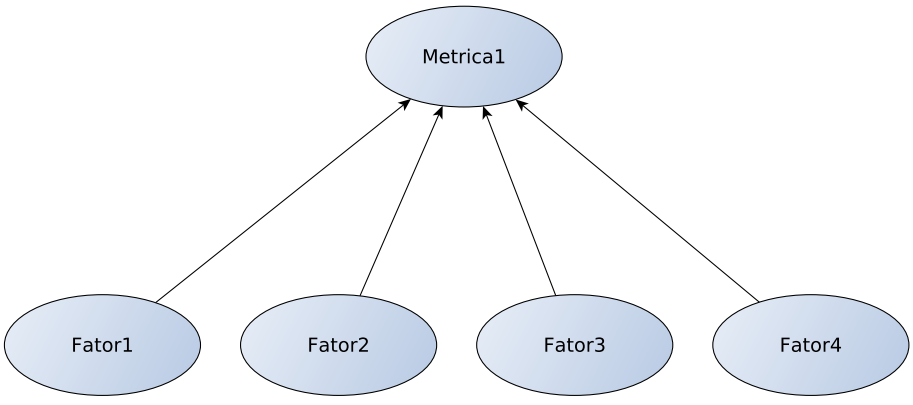
Após a identificação do conjunto de fatores F da métrica m, o ajuste de escalas e a discretização de valores, inicia-se a **Subetapa 4.2**, que trata da construção do Grafo Acíclico Dirigido (DAG).

Inicialmente, um nó Nn, correspondente à métrica m, é adicionado ao grafo. Os estados desse nó devem corresponder à escala intervalar discretizada definida nas **Etapas 2 e 3**. Esse nó representa a saída da rede, ou seja, o valor de referência da métrica m.

Para cada fator $f \in F$, adiciona-se um nó à rede. Cada nó adicionado terá uma relação de paternidade com o nó Nn. Os estados de cada nó também devem corresponder à escala intervalar discretizada definida nas **Etapas 2 e 3**.

Para exemplificar, suponha-se que para a métrica $Metrica1$ foram encontrados o conjunto de fatores definido por $F = \{Fator1, Fator2, Fator3, Fator4\}$. Ao adicionar o nó correspondente à métrica ao grafo, assim como um nó para cada fator, tem-se o grafo representado na Figura 3.2.

Figura 3.2: Exemplo da Construção do DAG (a)



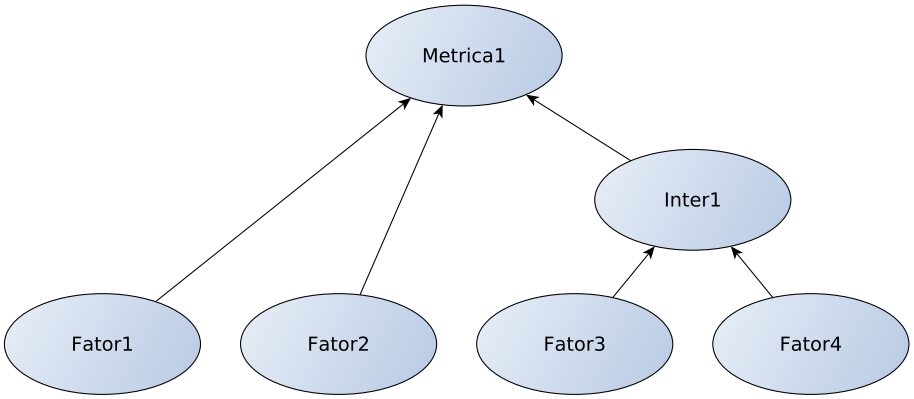
Fonte: Do autor.

Por fim, dado que o tamanho de uma Tabela de Probabilidade de Nós (TPN) cresce

exponencialmente de acordo com o número de nós-pai [25], para diminuir a complexidade da rede e facilitar a sua construção, se necessário, deve-se criar nós intermediários utilizando a técnica de Fenton *et al.* [13], de forma que nenhum nó fique com mais de três pais.

No exemplo citado acima, seria necessário adicionar um nó intermediário, de forma que o nó $Metrica1$ não tenha mais que três pais. O grafo resultante pode ser visto na Figura 3.3. Nela, o nó $Inter1$ foi adicionado como filho dos nós $Fator3$ e $Fator4$ e pai do nó $Metrica1$, ficando esse com apenas três nós-pais.

Figura 3.3: Exemplo da Construção do DAG (b)



Fonte: Do autor.

3.1.3 Refatoração da Discretização das Escalas

Na **Subetapa 4.3**, todas as discretizações das escalas feitas na **Subetapa 4.1** devem ser refatoradas. Ao final dessa etapa, o especialista do projeto deve estar de acordo com todos os estados e valores definidos. Assim, é verificado se o especialista concorda com todos os pontos de fronteira. Por exemplo, se todos os pais são baixos o filho vai ser baixo, etc. Caso o especialista discorde de algum ponto, altera-se apenas a discretização do filho, ou seja, o que aquele nó representa [25].

3.1.4 Definição das Funções de Probabilidade

Uma vez que o DAG está construído e refatorado, é necessário definir as TPN dos nós-folhas do grafo (**Subetapa 4.4**). De acordo com Zhou *et al.* [54], uma TPN pode ser representada

pela Equação 3.1, em que X_i representa um nó e $pa(X_i)$ representa o conjunto de pais desse nó.

$$P(X_i|pa(X_i)) \tag{3.1}$$

Nesse caso, a TPN contém todos os valores da variável X_i dada cada combinação de valores diferentes dos seus pais $pa(X_i)$.

Outra forma de representar uma TPN é definir manualmente uma matriz nxm de probabilidades, onde n é o número de estados possíveis e m é o produto de estados dos seus pais. Conforme o tamanho da rede aumenta, o tamanho da matriz cresce exponencialmente, tornando-se muito trabalhoso definir as TPNs manualmente. Por esse motivo, utilizou-se a técnica da de nós ranqueados proposta por Fenton *et al.* [14], com o auxílio da ferramenta *AgenaRisk* [3].

De forma geral, o *AgenaRisk* permite criar TPN a partir de dados qualitativos. Para definir as funções de probabilidade em nós ranqueado, ela utiliza uma distribuição normal duplamente truncada (TNormal [14]). A distribuição TNormal é caracterizada por dois parâmetros: **média** (μ) e **variância** (σ^2).

A média μ da distribuição TNormal, para a maioria dos nós, é calculada por meio de uma expressão ponderada que reflete a influência que os pais de determinado nó exercem sobre ele. A variância σ^2 indica o grau de confiança no resultado dos cálculos das probabilidades. Nos projetos estudados no decorrer da pesquisa, a variância utilizada foi de $0,0005$, a menor possível no *AgenaRisk*, indicando um alto grau de confiança.

A média μ da distribuição TNormal, na maioria dos nós, é calculada por meio de uma expressão ponderada que reflete a influência que os pais de determinado nó exercem sobre ele. No contexto deste trabalho, foram utilizados três tipos de funções ponderadas suportadas pelo *AgenaRisk*:

- **Ponderada mínima (wmin):** utilizada quando o valor calculado para o nó tende para Baixo se algum de seus pais for classificado como Baixo;
- **Ponderada máxima (wmax):** utilizada quando o valor calculado para o nó tende para Alto se algum de seus pais for classificado como Alto;

- **Ponderada média (wmean):** utilizada quando o valor do fator filho deve ser calculado a partir do cálculo da probabilidade ponderada dos pais.

Dessa forma, para criar as funções de probabilidade para cada nó-filho f da rede, deve-se capturar dos especialistas o resultado esperado para $3k$ combinações de estados possíveis de nós-pai (e.g., qual o resultado esperado para a combinação Bom, Ruim?), em que k é o número de nós-pais de f. A definição dos tipos das funções ponderadas (média, máxima ou mínima) deve ser inferido dos cenários capturados.

Da amostra capturada, $2k$ deve ser usado para calibrar a rede, definindo as funções de probabilidade utilizando a abordagem proposta em Fenton *et al.* [14]. O $1k$ restante deve ser usado para validar a rede.

Suponha que para o exemplo da Figura 3.3 foi definido uma escala de três fatores (Bom, Médio, Ruim), neste exemplo seria necessário coletar do especialista 6 cenários para calibrar o nó $Metrica1$ e mais quatro cenários para calibrar o nó intermediário $Inter1$. Se fosse preciso definir a TPN completa, só para o nó $Metrica1$ seria necessário definir 3^3 cenários, um número até pequeno, no entanto se a escala escolhida fosse de cinco fatores, o número de cenários aumentaria para 3^5, o que seria totalmente inviável.

Ainda no exemplo da Figura 3.3, suponha que foi coletado do especialista os seis cenários para o nó $Metrica1$ definido na Tabela 3.1.

Tabela 3.1: Seis cenários do nó $Metrica1$

$Fator1$	$Fator2$	$Inter1$	**Saída**
Bom	Ruim	Ruim	Bom
Ruim	Bom	Ruim	Bom
Ruim	Ruim	Bom	Ruim
Ruim	Bom	Bom	Médio
Bom	Ruim	Bom	Médio
Bom	Bom	Ruim	Bom

Fonte: Do autor.

Analisando os cenários da Tabela 3.1, pode-se inferir que a função de ponderada a ser utilizada seria a **máxima**, uma vez que na maioria dos cenários, o valor calculado para o nó

$Metrica1$ tende para Bom quando algum de seus pais é classificado como Bom.

A definição dos valores das funções toma como base o procedimento publicado em [38] por Perkusich *et al.*. Ao analisar os cenários é observado que o nó $Fator2$ (B) tem maior influência sobre os demais, e que que o nó $Iter1$ (C) tem maior influência sobre o nó $Fator1$ (A). Neste caso, a expressão ponderada do nó $Metria1$ (X) seria representada por $X = A + 3 \times B + 2 \times C$.

A seguinte função ponderada $wmax(1, A, 2, B, 3, C)$ é inserida no *AgenaRisk* gerando a TPN apresentada na Tabela 3.2.

Tabela 3.2: TPN gerada pelo *AgenaRisk* usando a função ponderada $wmax(1, A, 2, B, 3, C)$

$Fator1$	Bom								
$Fato2$	Ruim			Médio			Bom		
$Inter1$	Bom	Médio	Rum	Bom	Médio	Rum	Bom	Médio	Rum
Ruim	0,99	0,37	0,00	0,50	0,08	0,00	0,01	0,00	0,00
Bom	0,01	0,63	0,88	0,50	0,92	0,63	0,98	0,85	0,30
Médio	0,00	0,00	0,12	0,00	0,00	0,37	0,01	0,15	0,70
$Fator1$	Médio								
$Fato2$	Ruim			Médio			Bom		
$Inter1$	Bom	Médio	Rum	Bom	Médio	Rum	Bom	Médio	Rum
Ruim	0,76	0,12	0,00	0,15	0,01	0,00	0,00	0,00	0,00
Bom	0,24	0,88	0,63	0,85	0,98	0,37	0,85	0,50	0,08
Médio	0,00	0,00	0,37	0,00	0,01	0,63	0,15	0,50	0,92
$Fator1$	Ruim								
$Fato2$	Ruim			Médio			Bom		
$Inter1$	Bom	Médio	Rum	Bom	Médio	Rum	Bom	Médio	Rum
Ruim	0,23	0,01	0,00	0,01	0,00	0,00	0,00	0,00	0,00
Bom	0,77	0,98	0,37	0,99	0,76	0,12	0,50	0,15	0,01
Médio	0,00	0,01	0,63	0,01	0,24	0,88	0,50	0,85	0,99

Fonte: Do autor.

Os valores escolhidos foram testados em todos os modelos produzidos no decorrer da

pesquisa e calibrados até que se achasse um valor satisfatório. É possível que haja a necessidade de ajustar esses valores no decorrer do projeto de acordo com as necessidades ou características deste.

A validação da abordagem é feita no Capítulo 4 por meio de um estudo piloto com três especialistas em um projeto de desenvolvimento de software.

Capítulo 4

Estudo Piloto

A partir do estudo piloto realizado, buscou-se responder a seguinte questão de pesquisa: **Q:** Qual a precisão da abordagem para definir valores de referência representativos no contexto do projeto em questão?

O estudo piloto foi realizado com três especialistas envolvidos diretamente em um projeto de desenvolvimento executado no VIRTUS - Núcleo de Pesquisa, Desenvolvimento e Inovação em Tecnologia da Informação, Comunicação e Automação - da Universidade Federal de Campina Grande. Todos os especialistas têm mais de cinco anos de experiência em gestão de projetos. Um deles tem certificação profissional em gerenciamento de projetos ágeis e especialização em gerenciamento de projetos baseados no PMBoK [20]; outro tem a certificação Project Management Professional (PMP)[1].

Em uma reunião com os três especialistas, algumas características do projeto que estava sendo desenvolvido foram identificadas por meio do questionário apresentado no Apêndice A.1. Todos os especialistas estraram em consenso quanto as suas respostas.

O software que estava sendo desenvolvido era um aplicativo para o sistema operacional *Android*. A linguagem de programação utilizada era *JavaScript* e o processo de desenvolvimento tinha como base o framework Scrum [43]. A equipe que trabalhava no projeto era formada por quatro desenvolvedores e um testador. O projeto teve duração de um ano.

Alguns aspectos do uso de métricas de software utilizadas no projeto foram identificados em um segundo conjunto de perguntas. Os especialistas relataram a utilização das seguintes métricas: “Análise Estática”, mais precisamente, alertas de faltas não justificadas; “Número

[1] https://www.pmi.org/certifications/types/project-management-pmp

de Bugs Minor" (i.e., pequenos ajustes que impactam poucos usuários); e "Cobertura de Código". Todas as três métricas tinham os seguintes valores de referência: 5 para "Análise Estática", 5 para "Número de Bugs Minor" e 80% para "Cobertura de Código".

De acordo com os especialistas, os valores de referência adotados para as métricas supracitadas foram reutilizados de outros projetos. No entanto, em alguns casos, eles não eram adequados para o projeto em questão. Esse fato foi percebido após algumas tomadas de decisão consideradas errôneas em virtude de valores de referência não representativos.

Uma vez que foram identificadas todas as métricas utilizadas no projeto em questão, iniciou-se o processo de execução da abordagem. Nas Seções 4.1, 4.2 e 4.3 são descritos os detalhes desse processo para cada métrica usada. Além disso, na Seção 4.4 é apresentada uma breve análise dos resultados, finalizando o Capítulo com a Seção 4.5, na qual são discutidas as ameaças à validade no que diz respeito à abordagem apresentada.

4.1 Análise Estática (Alertas de Faltas Não Justificadas)

A métrica **Análise Estática** corresponde ao número de alertas de faltas em aberto, produzidos pelas ferramentas *CheckStyle*[2], *FindBugs*[3] e *PMD*[4], que não tinham sido justificadas pela equipe de desenvolvimento. O valor de referência usado para essa métrica era de, no máximo, cinco alertas.

As **Etapas 2 e 3** da abordagem, que correspondem a definição e discretização da escala, foram executadas em paralelo em uma reunião com os três especialistas. Em consenso, foi decidido usar uma escala *Likert* (i.e., ordinal) de três pontos (Bom, Médio e Ruim) para essa métrica. Quanto à discretização da escala (**Etapa 3**), foi classificada em: **Bom** (≤ 5), **Médio** ($]5, 10]$) e **Ruim** (> 10).

A construção da rede (**Etapa 4**) foi iniciada na mesma reunião. Nesse momento, os fatores que influenciam o valor de referência da métrica **Análise Estática** (**Subetapa 4.1**) foram coletados com os especialistas, que entraram em consenso quanto aos fatores e suas justificativas.

O fator **(i) Complexidade do Produto** foi sugerido com a justificativa de que, quanto

[2]http://checkstyle.sourceforge.net/

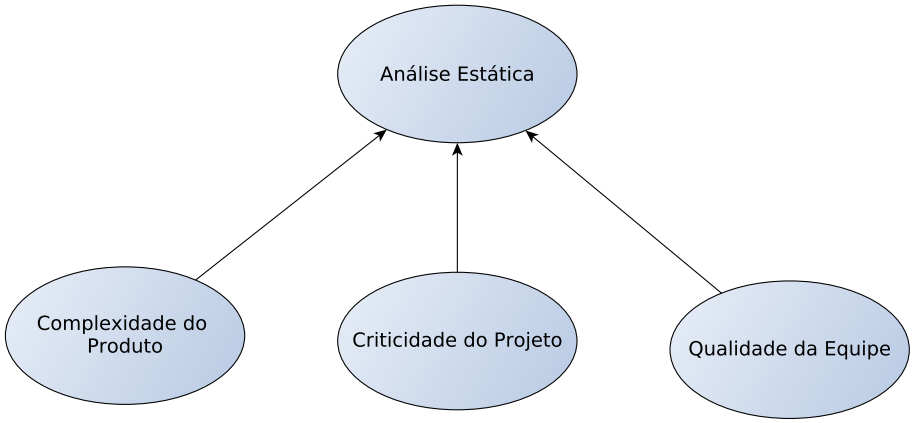
[3]http://findbugs.sourceforge.net/

[4]http://pmd.sourceforge.net/

mais complexo for o produto que está sendo desenvolvido, maior será o controle exercido sobre ele. Logo, menor será o número de alertas de faltas não justificadas aceitos. A mesma justificativa foi dada ao fator **(ii) Criticidade do Projeto**. O último fator identificado foi a **(iii) Qualidade da Equipe**. Uma equipe de qualidade tende a passar maior confiança no que é produzido, mesmo em cenários nos quais o número de alertas de faltas não justificadas é grande.

Como todos os fatores encontrados eram subjetivos, não foi necessário discretizar os seus valores, sendo finalizada assim a **Subetapa 4.1** e dando sequência à construção do DAG (**Subetapa 4.2**). A ferramenta *AgenaRisk* [3] foi utilizada durante todo o estudo piloto e o DAG criado para a métrica **Análise Estática** é apresentado na Figura 4.1.

Figura 4.1: Rede Bayesiana Referente à Métrica Análise Estática (Alertas de Faltas Não Justificadas)



Fonte: Do autor.

Assim como a **Subetapa 4.1**, a **Subetapa 4.3**, refatoração da discretização das escalas, não foi executada, uma vez que todos os fatores encontrados eram subjetivos.

A fim de coletar dados para compor alguns cenários e definir as funções de probabilidade (**Subetapa 4.4**) da rede Bayesiana, um questionário (Apêndice A.2.1) foi aplicado em conjunto aos especialistas do projeto, que discutiram entre eles a melhor resposta para cada cenário.

Para o nó "Análise Estática", foram capturados do usuário $3k$ cenários, em que k é o número de nós-pais definidos para o nó em questão, ou seja, três, totalizando nove cenários. Dos nove cenários, $2k$ foram usados para calibrar a rede, definindo as funções de probabili-

dade utilizando o algoritmo proposto por Fenton *et al.* [14].

Após a construção e calibração da rede Bayesiana, foi feita uma avaliação que consistiu da comparação entre o valor de referência resultante da execução da rede e a opinião dos especialistas para os três cenários restantes.

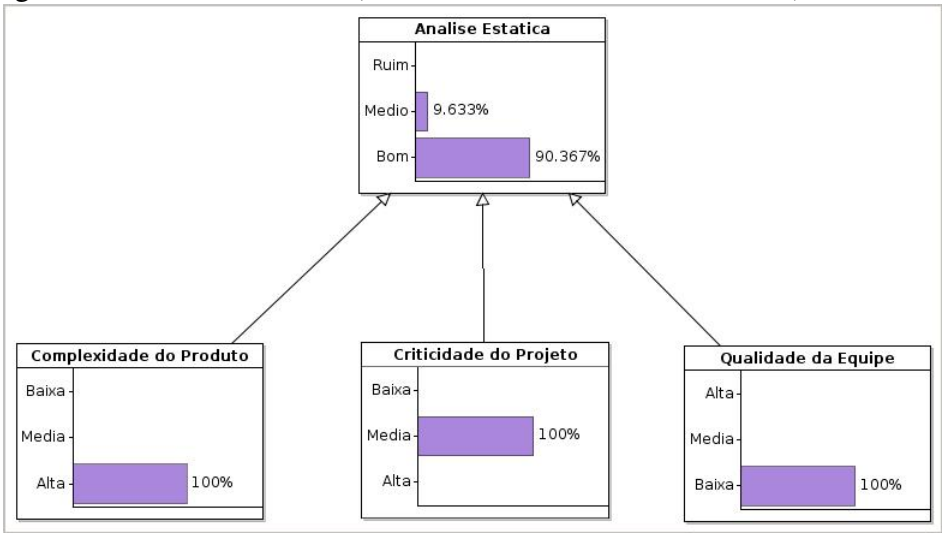
Nas subseções seguintes serão detalhados os resultados obtidos em cada cenário de validação.

4.1.1 Cenário 1

O primeiro cenário trata-se de um projeto em que a **Complexidade do Produto** é **Alta**, a **Criticidade do Projeto** foi definida como **Média** e a **Qualidade da Equipe** é **Baixa**. Nesse cenário, o resultado esperado segundo a opinião dos especialistas é que o valor de referência seja **Bom**, ou seja, para esse cenário, um valor de referência representativo seria de, no máximo, **cinco** alertas de faltas não justificadas.

Ao executar a rede criada com as entradas do Cenário 1, foi obtida a probabilidade do resultado ser **Bom** (≤ 5) de **90,367%** e de **9,633%** do resultado ser **Médio** ($]5, 10]$), como mostra a Figura 4.2, corroborando com o resultado esperado pelos especialistas.

Figura 4.2: Análise Estática (Alertas de Faltas Não Justificadas) - Cenário 1



Fonte: Do autor.

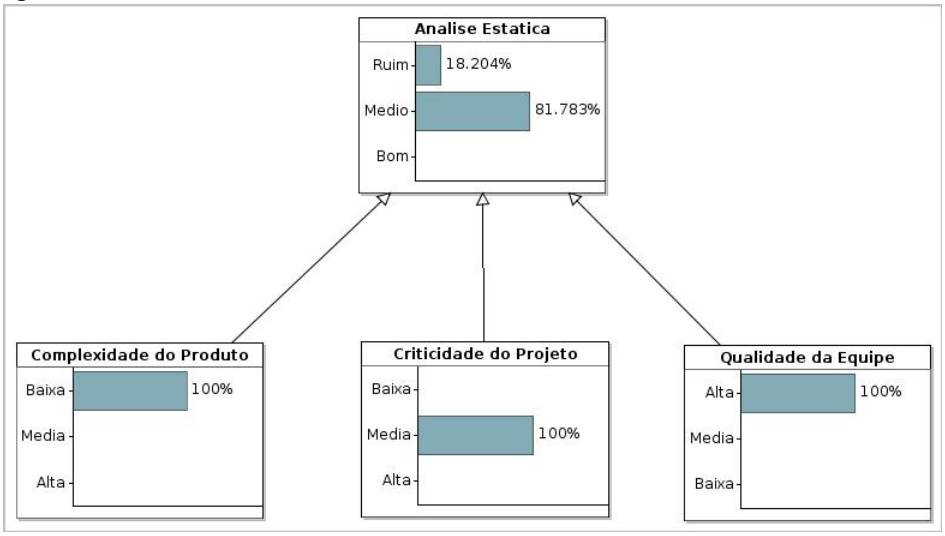
4.1.2 Cenário 2

O segundo cenário trata-se de um projeto em que a **Complexidade do Produto** é **Baixa**, a **Criticidade do Projeto** foi definida como **Média** e a **Qualidade da Equipe** é **Alta**. Neste cenário, o resultado esperado segundo a opinião dos especialistas é que o valor de referência seja **Médio**, ou seja, para esse cenário, um valor de referência representativo estaria entre **cinco** e **dez** alertas de faltas não justificadas.

Ao executar a rede criada com as entradas do Cenário 2, foi obtida a probabilidade do resultado ser **Médio (**$]5, 10]$**)** de **81,783%** e de **18,204%** do resultado ser **Ruim (**> 10**)**, como mostra a Figura 4.3.

Como a maior probabilidade é do resultado ser **Médio**, pode-se dizer que a rede obteve o resultado esperado pelos especialistas.

Figura 4.3: Análise Estática (Alertas de Faltas Não Justificadas) - Cenário 2



Fonte: Do autor.

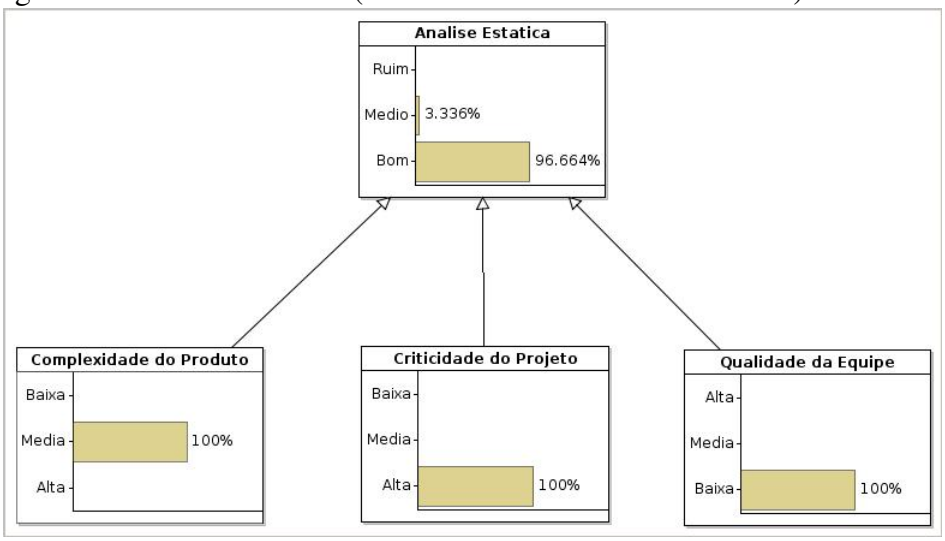
4.1.3 Cenário 3

O terceiro e último cenário trata-se de um projeto em que a **Complexidade do Produto** é **Média**, a **Criticidade do Projeto** foi definida como **Alta** e a **Qualidade da Equipe** é **Baixa**. Neste cenário, o resultado esperado segundo a opinião dos especialistas é que o valor de referência seja **Bom**, ou seja, para esse cenário, um valor de referência representativo seria de no máximo **cinco** alertas de faltas não justificadas.

Ao executar a rede criada com as entradas do Cenário 3, foi obtida a probabilidade do resultado ser **Bom** (≤ 5) de **96,664%** e de **3,336%** do resultado ser **Médio** ($]5, 10]$), como mostra a Figura 4.4.

Como a maior probabilidade é do resultado ser **Bom**, pode-se dizer que a rede também obteve o resultado esperado pelos especialistas.

Figura 4.4: Análise Estática (Alertas de Faltas Não Justificadas) - Cenário 3



Fonte: Do autor.

4.2 Número de Bugs Minor

A métrica **Número de Bugs Minor** usada no projeto diz respeito ao número de *bugs* de categoria *minor* encontrados pela equipe de testes e cadastrados na ferramenta *BugZilla*[5]. O valor de referência usado para essa métrica era de, no máximo, cinco *bugs minor*.

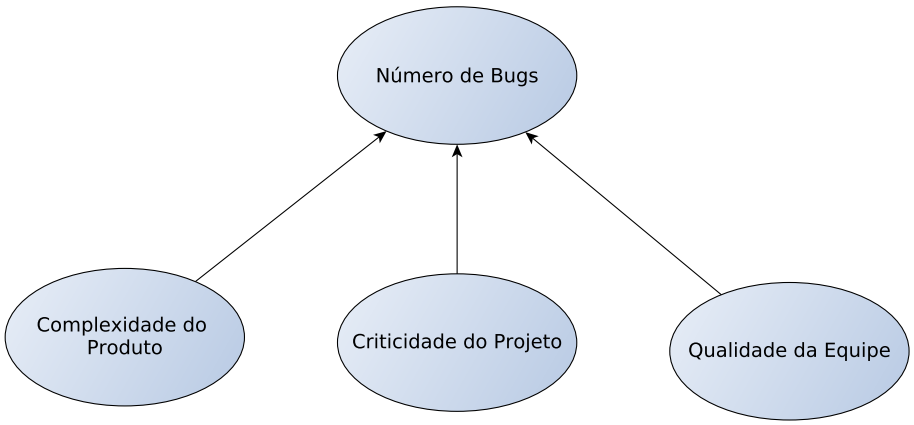
Uma vez identificada a métrica, conclui-se a **Etapa 1** da abordagem. As **Etapas 2 e 3** foram executadas em paralelo em uma reunião com os três especialistas que, em consenso, também decidiram usar uma escala **likert** de três pontos (bom, médio e ruim) para essa métrica. Quanto à discretização da escala (**Etapa 3**), eles usaram a mesma classificação adotada para a métrica **Análise Estática**, classificando-os em: **bom** (≤ 5), **médio** ($]5, 10]$) e **ruim** (> 10).

[5]https://www.bugzilla.org/

Ainda na reunião, iniciou-se a construção da rede (**Etapa 4**), momento no qual foram coletados com os especialistas os fatores que influenciam o valor de referência da métrica **Número de Bugs Minor** (**Subetapa 4.1**). Esses, juntamente com suas justificativas, foram os mesmo definidos para a métrica **Análise Estática**.

Como todos os fatores encontrado eram subjetivos, não foi necessário discretizar seus valores, finalizando, assim, a **Subetapa 4.1** e dando sequência a construção do DAG (**Subetapa 4.2**), cujo resultado é apresentado na Figura 4.5.

Figura 4.5: Rede Bayesiana Referente à Métrica Número de Bugs Minor



Fonte: Do autor.

Assim como a **Subetapa 4.1**, a **Subetapa 4.3** não foi executada, uma vez que todos os fatores encontrado eram subjetivos, não havendo a necessidade de discretizar seus valores.

Dando continuidade à execução da abordagem, um questionário (Apêndice A.2.2) foi aplicado aos especialistas a fim de coletar os dados de alguns cenários para a definição das funções de probabilidade (**Subetapa 4.4**), assim como para validar a rede criada. Houve uma discussão entre eles a fim de definir a melhor resposta para cada cenário.

Para o nó "Número de Bugs", foram capturados do usuário $3k$ cenários, em que k é o número de nós-pais definidos para o nó em questão. Dos nove cenários, $2k$ foram usados para calibrar a rede, definindo as funções de probabilidade.

Um vez que a rede foi construída e calibrada, foi utilizado como método de validação uma comparação entre o valor de referência resultante da sua execução e a opinião dos especialistas para os três cenários restantes.

Nas subseções seguintes serão detalhados os resultados obtidos em cada cenário de vali-

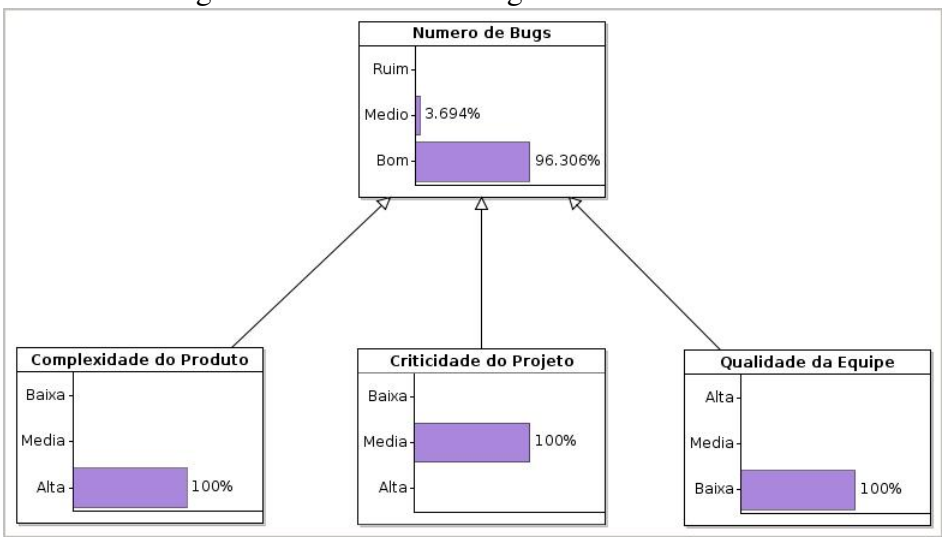
dação.

4.2.1 Cenário 1

O primeiro cenário trata-se de um projeto em que a **Complexidade do Produto** é **Alta**, a **Criticidade do Projeto** foi definida como **Média**, e a **Qualidade da Equipe** é **Baixa**. Nesse cenário, o resultado esperado segundo a opinião dos especialistas é que o valor de referência seja **Bom**, ou seja, para esse cenário, um valor de referência representativo seria de, no máximo, **cinco** *bugs minor*.

Ao executar a rede criada com as entradas do Cenário 1, foi obtida a probabilidade do resultado ser **Bom** (≤ 5) de **96,306%** e de **3,694%** do resultado ser **Médio** ($]5, 10]$), como mostra a Figura 4.6, corroborando com o resultado esperado pelos especialistas.

Figura 4.6: Número de Bugs Minor - Cenário 1



Fonte: Do autor.

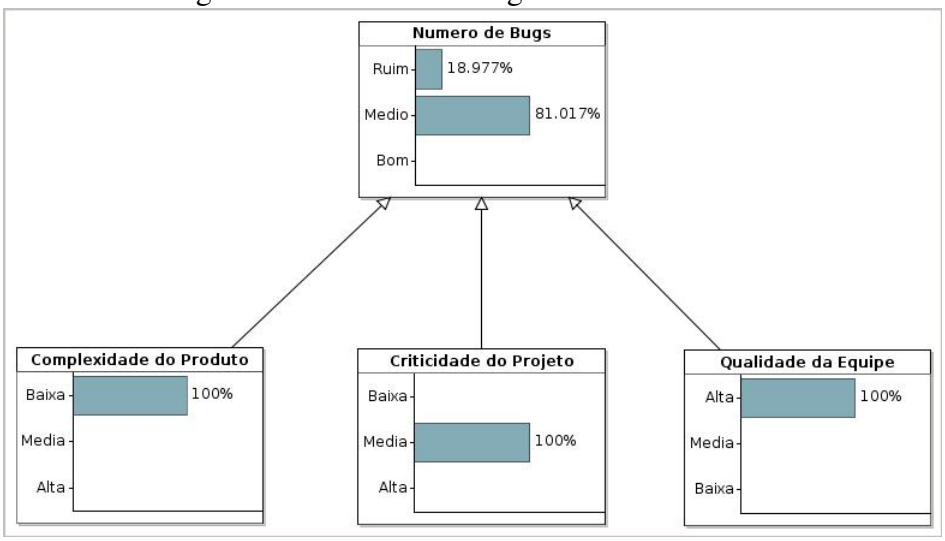
4.2.2 Cenário 2

O segundo cenário trata-se de um projeto onde a **Complexidade do Produto** é **Baixa**, a **Criticidade do Projeto** foi definida como **Média**, e a **Qualidade da Equipe** é **Alta**. Nesse cenário, o resultado esperado segundo a opinião dos especialistas é que o valor de referência seja **Ruim**, ou seja, para esse cenário, um valor de referência representativo seria maior que **dez** *bugs minor*.

Ao executar a rede criada com as entradas do Cenário 2, foi obtida a probabilidade do resultado ser **Médio** ($]5, 10]$) de **81,017%** e de **18,977%** do resultado ser **Ruim** (> 10), como mostra a Figura 4.7.

Nesse cenário, o resultado esperado difere do resultado encontrado pela rede. É possível elencar três possíveis motivos para isso acontecer: (i) a rede não foi calibrada corretamente; (ii) os especialistas não informaram corretamente o resultado esperado para esse cenário; (iii) os fatores encontrados não foram suficientes para predizer a medida do valor de referência.

Figura 4.7: Número de Bugs Minor - Cenário 2



Fonte: Do autor.

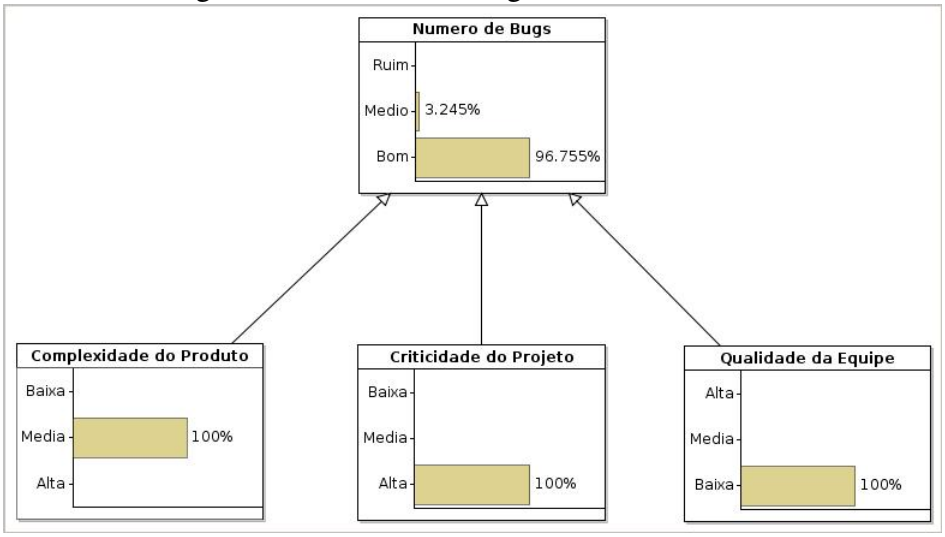
4.2.3 Cenário 3

O terceiro e último cenário trata-se de um projeto onde a **Complexidade do Produto** é **Média**, a **Criticidade do Projeto** foi definida como **Alta**, e a **Qualidade da Equipe** é **Baixa**. Nesse cenário, o resultado esperado segundo a opinião dos especialistas é que o valor de referência seja **Bom**, ou seja, para esse cenário, um valor de referência representativo seria de, no máximo, **cinco** *bugs minor*.

Ao executar a rede criada com as entradas do Cenário 3, foi obtida a probabilidade do resultado ser **Bom** (≤ 5) de **96,755%** e de **3,245%** do resultado ser **Médio** ($]5, 10]$), como mostra a Figura 4.8.

Como a maior probabilidade é do resultado ser **Bom**, pode-se dizer que a rede obteve o resultado esperado pelos especialistas.

Figura 4.8: Número de Bugs Minor - Cenário 3



Fonte: Do autor.

4.3 Cobertura de Código

De modo geral, a métrica **Cobertura de Código** usada no projeto se refere à porcentagem de código coberto por testes sem a utilização de filtros. O valor de referência usado para essa métrica era de 80%.

Uma vez identificada a métrica, conclui-se a **Etapa 1** da abordagem. As **Etapas 2 e 3** foram executadas em paralelo em uma reunião com os três especialistas que, em consenso, decidiram usar uma escala **likert** de cinco pontos (muito baixo, baixo, médio, alto e muito alto) para essa métrica. Quanto à discretização da escala (**Etapa 3**), eles classificaram em: **muito baixo (**$0 - 30\%$**)**, **baixo (**$31 - 50\%$**)**, **médio (**$51 - 70\%$**)**, **alto (**$71 - 80\%$**)** e **muito alto (**$81 - 100\%$**)**.

Ainda na reunião, iniciou-se a construção da rede (**Etapa 4**), ocasião na qual foram coletados com os especialistas os fatores que influenciam a medida do valor de referência da métrica **Cobertura de Código** (**Subetapa 4.1**).

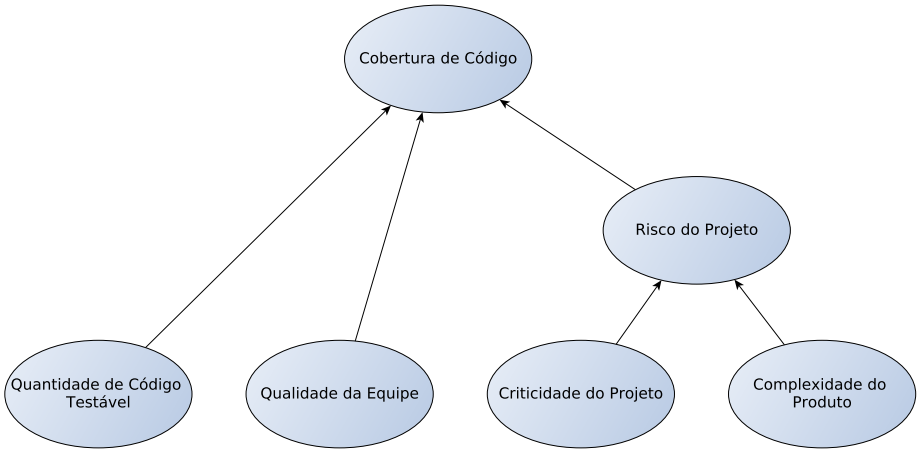
O fator **(i) Complexidade do Produto** foi sugerido com a justificativa de que, quanto mais complexo for o produto que está sendo desenvolvido, maior será o controle exercido sobre ele e, por fim, maior será a cobertura de código necessária para aprová-lo. A mesma justificativa foi dada ao fator **(ii) Criticidade do Projeto**.

O uso de *frameworks*, APIs, códigos de terceiros ou até mesmo páginas HTML tende a

diminuir o valor de referência da **Cobertura de Código**, uma vez que não é possível testar esse tipo de código. Dessa forma, adicionou-se também o fator **(iii) Quantidade de Código Testável**. O último fator identificado foi a **(iv) Qualidade da Equipe**, com a justificativa de que uma equipe de qualidade tende a passar maior confiança no que é produzido, mesmo em cenários nos quais a cobertura de código é baixa.

Como todos os fatores encontrado eram subjetivos, não foi necessário discretizar seus valores, finalizando, assim, a **Subetapa 4.1** e dando sequência à construção do DAG (**Subetapa 4.2**), cujo resultado pode ser visto na Figura 4.9.

Figura 4.9: Rede Bayesiana Referente à Métrica Cobertura de Código



Fonte: Do autor.

Ainda na **Subetapa 4.2**, o nó "Risco do Projeto" foi acrescentado ao DAG como nó intermediário entre o filho "Cobertura de Código" e os nós "Criticidade do Projeto" e "Complexidade do Projeto" (Figura 4.9). O nó foi adicionado com o intuito de reduzir a complexidade da rede e facilitar a coleta de alguns cenários para a definição das funções de probabilidade.

Assim como a **Subetapa 4.1**, a **Subetapa 4.3** não foi executada, uma vez que todos os fatores encontrado eram subjetivos, não havendo a necessidade de discretizar seus valores.

Dando continuidade à execução da abordagem, um questionário (Apêndice A.2.3) foi aplicado aos especialistas com o intuito de coletar os dados de alguns cenários para a definição das funções de probabilidade (**Subetapa 4.4**), assim como para validar a rede criada. Este foi respondido em conjunto, definindo apenas uma resposta para cada cenário.

Para o nó "Risco do Projeto", foram capturados do usuário $3k$ cenários, onde k é o número de nós-pais definidos para o nó em questão, ou seja, dois, totalizando seis cenários.

Dos seis cenários, $2k$ foram usados para calibrar a sub-rede do nó "Risco do Projeto", e dois cenários foram usados para validá-la.

Ainda no Questionário A.2.3, coletou-se dos especialistas os dados de alguns cenários para definir a função de probabilidade do nó "Cobertura de Código". Foram capturados nove cenários, tendo em vista que o nó possui três pais.

Um vez que a rede foi construída e calibrada, foi utilizado como método de validação uma comparação entre o valor de referência resultante da sua execução e a opinião dos especialistas para os três cenários restantes do nó "Cobertura de Código".

Nas subseções seguintes serão detalhados os resultados obtidos em cada cenário de validação.

4.3.1 Cenário 1

O primeiro cenário trata-se de um projeto onde a **Quantidade de Código Testável** é **Muito Alta**, a **Qualidade da Equipe** foi definida como **Média**, e o **Risco do Projeto** é **Muito Baixo**. Nesse cenário, o resultado esperado segundo a opinião dos especialistas é que o valor de referência seja **Alto**, ou seja, para esse cenário, um valor de referência representativo estaria entre $71 - 80\%$.

Ao executar a rede criada com as entradas do Cenário 1, foi obtida a probabilidade do resultado ser **Alto (**$71 - 80\%$**)** de **52,11%** e de **47,89%** do resultado ser **Muito Alto (**$81 - 100\%$**)**, como mostra a Figura 4.10.

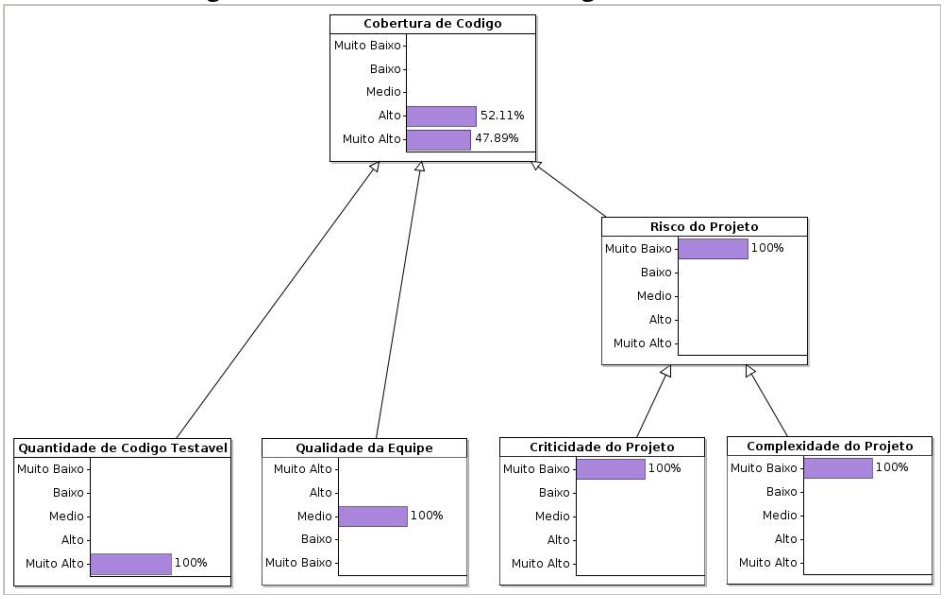
Como a maior probabilidade é do resultado ser **Alto**, pode-se afirmar que a rede obteve o resultado esperado pelos especialistas.

4.3.2 Cenário 2

O segundo cenário trata-se de um projeto onde a **Quantidade de Código Testável** é **Muito Baixa**, a **Qualidade da Equipe** foi definida como **Média**, e o **Risco do Projeto** é **Muito Alto**. Nesse cenário, o resultado esperado segundo a opinião dos especialistas é que o valor de referência seja **Muito Alto**, em outras palavras, para esse cenário, um valor de referência representativo estaria entre $81 - 100\%$.

Ao executar a rede criada com as entradas do Cenário 2, foi obtida a probabilidade do

Figura 4.10: Cobertura de Código - Cenário 1



Fonte: Do autor.

resultado ser **Muito Alto** (81 − 100%) de **53,648%** e de **46,352%** do resultado ser **Alto** (71 − 80%), como mostra a Figura 4.11.

Como a maior probabilidade é do resultado ser **Muito Alto**, pode-se dizer que a rede obteve, de fato, o resultado esperado pelos especialistas.

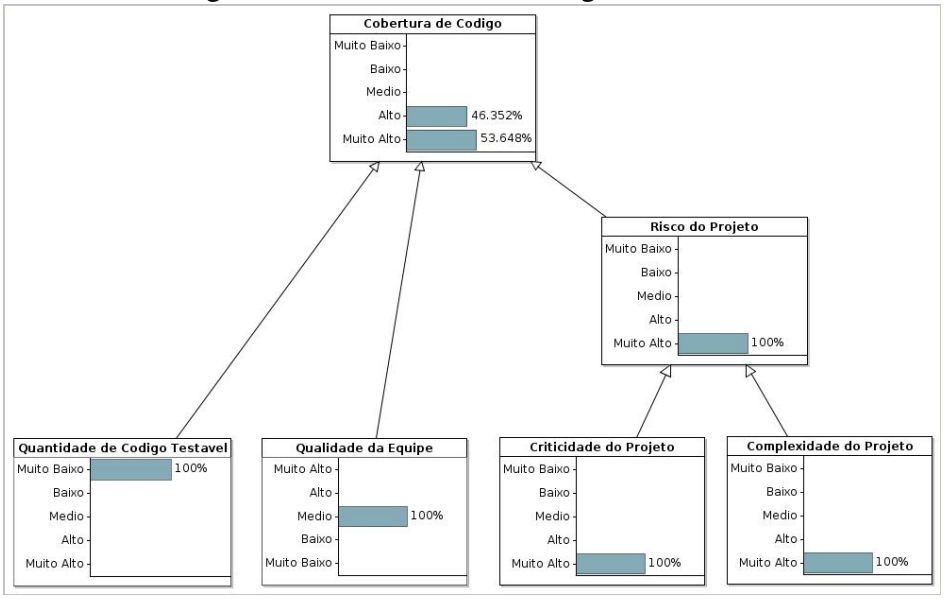
4.3.3 Cenário 3

O terceiro e último cenário trata-se de um projeto onde a **Quantidade de Código Testável** é **Média**, a **Qualidade da Equipe** foi definida como **Muito Alta**, e o **Risco do Projeto** é **Muito Baixo**. Nesse cenário, o resultado esperado segundo a opinião dos especialistas é que o valor de referência seja **Médio**, ou seja, para esse cenário, um valor de referência representativo estaria entre 51 − 70%.

Ao executar a rede criada com as entradas do Cenário 3, foi obtida a probabilidade do resultado ser **Médio** (51 − 70%) de **64,726%** e de **35,271%** do resultado ser **Baixo** (31 − 50%), como mostra a Figura 4.12.

Como a maior probabilidade é do resultado ser **Médio**, pode-se dizer que a rede obteve o resultado esperado pelos especialistas.

Figura 4.11: Cobertura de Código - Cenário 2



Fonte: Do autor.

4.4 Análise dos Resultados

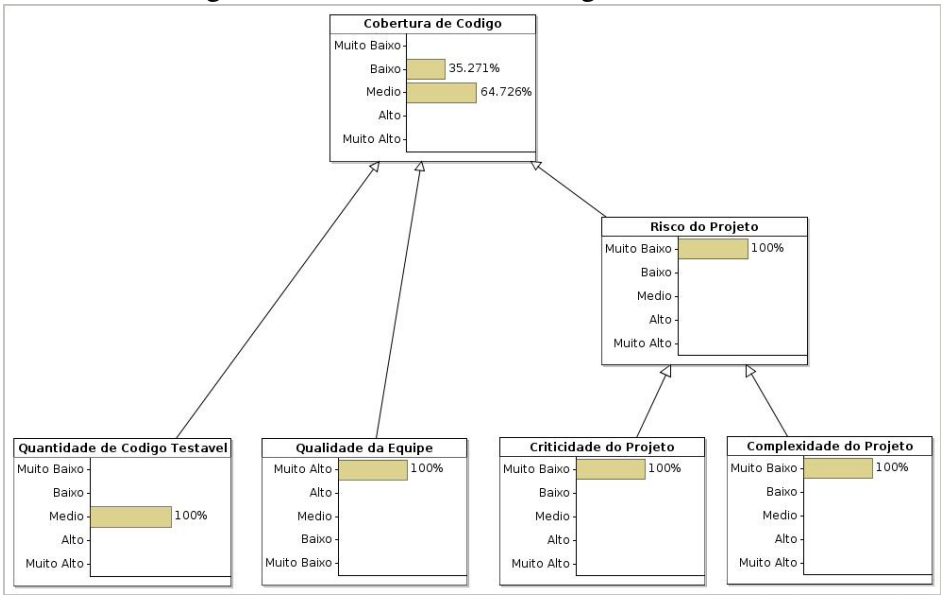
É possível concluir que, para os nove cenários verificados, em oito o modelo calculou o resultado esperado. Como discutido na Seção 4.2.2, os motivos para essa divergência podem ser problemas na construção da rede Bayesiana ou elicitação do conhecimento. Sendo assim, a abordagem obteve bons resultados para o cenário em que ela foi validada. Por outro lado, destaca-se a necessidade de ter um processo maduro para construir e validar a rede Bayesiana antes de utilizá-la para auxiliar a tomada de decisão. Além disso, destaca-se a necessidade de ter especialistas que possuam conhecimento dos fatores de contexto para cada métrica.

4.5 Ameaças à Validade

O estudo apresenta ameaças à validade externa por ter sido realizado com especialistas de apenas uma empresa. Por outro lado, os especialistas têm experiência e conhecimento teórico em gestão de projetos de software e a abordagem foi aplicada para diferentes métricas.

Além disso, foram identificadas ameaças à validade de conclusão devido ao seu baixo poder estatístico, uma vez que a abordagem foi executada por apenas três especialistas em um projeto e validada em nove cenários. Por outro lado, a abordagem foi aplicada para

Figura 4.12: Cobertura de Código - Cenário 3



Fonte: Do autor.

métricas diferentes e em um contexto maduro de desenvolvimento de software.

Com relação às ameaças internas, a experiência dos gerentes com a tomada de decisão pode ter influenciado os resultados. Essa ameaça foi minimizada selecionando especialistas com experiência em gerenciamento de projetos de software e aplicando a abordagem para métricas que eles usualmente utilizavam para auxiliar na tomada de decisões. Por outro lado, a não aleatoriedade na seleção dos especialistas pode ter enviesado os resultados.

A ameaça à validade de construção foi minimizada, uma vez que o método utilizado para inferir a precisão foi apenas comparar o resultado esperado (i.e., um estado) com o resultado calculado (i.e., probabilidade da variável estar naquele estado).

Capítulo 5

Conclusão

Neste trabalho foi apresentada uma abordagem para a definição de valores de referência de métricas de software na ausência de dados passados e de acordo com o contexto do projeto. Além de considerar fatores de contexto, a presente abordagem se destaca também pela não necessidade de dados históricos e por viabilizar um processo sistemático de definição de valores de referência de métricas de software. Com isso, é possível definir tais valores durante a fase inicial dos projetos ou quando seus dados não são facilmente coletados.

A abordagem foi avaliada a partir de um estudo piloto realizado com três especialistas que atuavam em um projeto real de desenvolvimento de software. A questão de pesquisa que buscou-se responder foi: **Q:** Qual a precisão da abordagem para definir valores de referência representativos no contexto do projeto em questão?

No estudo piloto foram construídas redes Bayesianas para três métricas: **Análise Estática**, **Número de Bugs Minor** e **Cobertura de Código**. Para responder a questão de pesquisa supracitada, foram elaborados alguns cenários a fim de comparar os resultados esperados definidos pelos especialistas com os resultados gerados a partir da execução das redes.

Em todos os cenários, exceto o Cenário 2 da métrica **Número de Bugs Minor**, o resultado gerado foi igual ao esperado. Dessa forma, é possível concluir que a abordagem proposta mostrou-se promissora para auxiliar os profissionais a identificar valores de referência representativos para seus projetos.

A principal implicação para o estado da arte é complementá-lo com uma abordagem inovadora para a definição de valores de referência para métricas de software de acordo

com o contexto e sem ser limitada a métricas de código-fonte. Para o estado da prática, foi apresentado um processo sistemático para auxiliar profissionais na tomada de decisão baseadas em métricas.

Para trabalhos futuros, pretende-se desenvolver uma ferramenta para que o uso da solução proposta possa ser inserida no dia-a-dia dos profissionais e, dentro do possível, para que seja possível abstrair a complexidade da rede Bayesiana. Atualmente, o processo de construção das redes Bayesianas acontece por meio do uso da ferramenta *AgenaRisk*, que exige o pagamento de uma licença anual. Além disso, sua utilização pode ser muito complexa para um usuário comum. Dessa forma, a construção de uma ferramenta intuitiva, por meio da qual os especialistas possam facilmente identificar os fatores de contexto, construir o DAG e responder os questionários referentes aos cenários usados para calcular as funções de probabilidade, apresenta-se como uma ideia extremamente válida.

Uma vez criada a ferramenta, pretende-se tornar a calibração das funções de probabilidade mais transparente e otimizada usando, por exemplo, algoritmos genéticos, que já se mostraram bastante efetivos para resolver diversas tarefas de otimização, além de já serem usados para modelar a estrutura de redes Bayesianas [26].

Por fim, pretende-se realizar um estudo de caso com mais projetos e em diferentes contextos para avaliar a utilidade da abordagem, gerando valores de referência e verificando se as decisões tomadas ao longo do projeto, utilizando esses valores, são assertivas.

Bibliografia

[1] T. L. Alves, C. Ypma, and J. Visser. Deriving metric thresholds from benchmark data. In *2010 IEEE International Conference on Software Maintenance*, pages 1–10. IEEE, sep 2010.

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

[3] N. Birtles, N. Fenton, M. Neil, and E. Tranham. Agenarisk manual (version 6.1) computer software, 2014.

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

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

[6] C. Calero, J. Ruiz, and M. Piattini. Classifying web metrics using the web quality model. *Online Information Review*, 29(3):227–248, 2005.

[7] S. R. Chidamber and C. F. Kemerer. A metrics suite for object oriented design. *Software Engineering, IEEE Transactions on*, 20(6):476–493, 1994.

[8] M. Convertino, K. Baker, C. Lu, J. T. Vogel, K. McKay, and I. Linkov. Metric selection for ecosystem restoration. Technical report, DTIC Document, 2013.

[9] B. Das. Generating Conditional Probabilities for Bayesian Networks: Easing the Knowledge Acquisition Problem. *Computing Research Repository*, cs.AI/0411034, 2004.

[10] S. Dragicevic, S. Celar, and M. Turic. Bayesian network model for task effort estimation in agile software development. *Journal of Systems and Software*, 127:109–119, 2017.

[11] B. Efron. Bootstrap methods: another look at the jackknife. *The annals of Statistics*, pages 1–26, 1979.

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

[13] N. Fenton and M. Neil. *Risk assessment and decision analysis with Bayesian networks*. Crc Press, 2012.

[14] N. E. Fenton, M. Neil, and J. G. Caballero. Using ranked nodes to model qualitative judgments in bayesian networks. *IEEE Transactions on Knowledge and Data Engineering*, 19(10):1420–1432, 2007.

[15] K. A. Ferreira, M. A. Bigonha, R. S. Bigonha, L. F. Mendes, and H. C. Almeida. Identifying thresholds for object-oriented software metrics. *Journal of Systems and Software*, 85(2):244–257, feb 2012.

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

[17] M. Foucault, M. Palyart, J.-R. Falleri, and X. Blanc. Computing contextual metric thresholds. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing - SAC '14*, pages 1120–1125, New York, New York, USA, mar 2014. ACM Press.

[18] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine learning*, 29(2-3):131–163, 1997.

[19] C. Gencel, K. Petersen, A. A. Mughal, and M. I. 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.

[20] P. Guide. A guide to the project management body of knowledge. In *Project Management Institute*, volume 3, 2004.

[21] D. Heckerman et al. A tutorial on learning with bayesian networks. *Nato Asi Series D Behavioural And Social Sciences*, 89:301–354, 1998.

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

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

[24] B. Kitchenham, S. L. Pfleeger, and N. Fenton. Towards a framework for software measurement validation. *IEEE Transactions on software Engineering*, 21(12):929–944, 1995.

[25] P. Laitila and K. Virtanen. Improving construction of conditional probability tables for ranked nodes in bayesian networks. *IEEE Transactions on Knowledge and Data Engineering*, 28(7):1691–1705, 2016.

[26] P. Larranaga, C. M. Kuijpers, R. H. Murga, and Y. Yurramendi. Learning bayesian network structures by searching for the best ordering with genetic algorithms. *IEEE transactions on systems, man, and cybernetics-part A: systems and humans*, 26(4):487–493, 1996.

[27] I. Linkov, D. Loney, S. Cormier, F. K. Satterstrom, and T. Bridges. Weight-of-evidence evaluation in environmental assessment: review of qualitative and quantitative approaches. *Science of the Total Environment*, 407(19):5199–5205, 2009.

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

[29] R. Martin. Oo design quality metrics. *An analysis of dependencies*, 12:151–170, 1994.

[30] K. S. Mathias, J. H. Cross II, T. D. Hendrix, and L. 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.

[31] A. Medeiros. Uma abordagem baseada em redes bayesianas para auxiliar a interpretação de métricas de software, 2015.

[32] E. Mendes, P. Rodriguez, V. Freitas, S. Baker, and M. A. Atoui. Towards improving decision making and estimating the value of decisions in value-based software engineering: the value framework. *Software Quality Journal*, pages 1–50, 2017.

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

[34] P. Oliveira, F. P. Lima, M. T. Valente, and A. 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.

[35] P. Oliveira, M. T. Valente, A. Bergel, and A. Serebrenik. Validating metric thresholds with developers: An early result. In *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*, pages 546–550. IEEE, 2015.

[36] P. Oliveira, M. T. Valente, and F. P. Lima. Extracting relative thresholds for source code metrics. In *2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*, pages 254–263. IEEE, feb 2014.

[37] J. Pearl. Bayesian networks. *Department of Statistics, UCLA*, 2011.

[38] M. Perkusich, A. Perkusich, and H. O. 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.

[39] R. Pressman and B. Maxim. *Engenharia de Software-8ª Edição*. McGraw Hill Brasil, 2016.

[40] F. Ruggeri, R. S. Kenett, and F. W. Faltin. Encyclopedia of statistics in quality and reliability, 2007.

[41] L. C. Santos, R. Saraiva, M. Perkusich, H. O. Almeida, and A. Perkusich. An empirical study on the influence of context in computing thresholds for chidamber and kemerer

metrics. In *Proceedings of the 29th International Conference on Software Engineering and Knowledge Engineering*. KSI Research Inc. and Knowledge Systems Institute Graduate School, jul 2017.

[42] N. F. Schneidewind. Methodology for validating software metrics. *IEEE Transactions on software engineering*, 18(5):410–422, 1992.

[43] K. Schwaber and M. Beedle. *Agile software development with Scrum*, volume 1. Prentice Hall Upper Saddle River, 2002.

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

[45] R. Shatnawi, W. Li, J. Swain, and T. Newman. Finding software metrics threshold values using ROC curves. *Journal of Software Maintenance and Evolution: Research and Practice*, 22(1):1–16, jan 2010.

[46] I. Sommerville, S. S. S. Melnikoff, R. Arakaki, and E. de Andrade Barbosa. *Engenharia de software*, volume 9. Addison Wesley São Paulo, 2011.

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

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

[49] T. Tahir, G. Rasool, and C. Gencel. A systematic literature review on software measurement programs. *Information and Software Technology*, 73:101–121, 2016.

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

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

[52] K. Verbert, R. Babuška, and B. De Schutter. Bayesian and dempster–shafer reasoning for knowledge-based fault diagnosis–a comparative study. *Engineering Applications of Artificial Intelligence*, 60:136–150, 2017.

[53] F. Zhang, A. Mockus, Y. Zou, F. Khomh, and A. E. Hassan. How Does Context Affect the Distribution of Software Maintainability Metrics? In *2013 IEEE International Conference on Software Maintenance*, pages 350–359. IEEE, sep 2013.

[54] Y. Zhou, N. Fenton, and M. Neil. Bayesian network approach to multinomial parameter learning using data and expert judgments. *International Journal of Approximate Reasoning*, 55(5):1252–1268, 2014.

[55] H. Ziv and D. 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

Questionários Aplicados

Neste Apêndice, encontram-se todos os questionários respondidos pelos especialistas participantes desta pesquisa. Estes, entraram em consenso quanto as respostas apresentadas. O primeiro questionário (Apêndice A.1) trata dos dados de contextualização do projeto. O segundo questionário (Apêndice A.2) se refere aos cenários para a definição das TPN e validação das redes Bayesianas.

A.1 Contexto do Projeto

Sobre o projeto
Qual é o domínio da aplicação que está sendo desenvolvida? • Resposta: *Aplicação Android.*
Qual é a linguagem de programação que está sendo usada? • Resposta: *JavaScript.*
Qual é o processo de desenvolvimento usado? • Resposta: *Scrum.*

Continua na próxima página

Quantos desenvolvedores formam a equipe? • Resposta: *4 desenvolvedores.*
Quantos testadores formam a equipe? • Resposta: *1 testador.*
Qual é a duração prevista para este projeto? • Resposta: *1 ano.*

Sobre as métricas/valores de referência usados no projeto
Quais métricas são usadas? Quais valores de referência são usados para essas métricas? • Respostas: – *Cobertura de Código: 80%;* – *Análise Estática: 5;* – *Número de Bugs Minor: 5.*
Os valores de referência foram reutilizados de outros projetos? • Resposta: *Todos.*
Vocês consideram que os valores de referência são adequados para este projeto? • Resposta: *Em alguns casos, não.*

Continua na próxima página

Você já obteve resultados ruins provenientes das decisões tomadas usando esses valores de referência?

- Resposta: *Sim, em cerca de 20% dos casos.*

A.2 Cenários

A.2.1 Análise Estática (Alertas de Faltas Não Justificados)

Juntamente com os questionários, os especialistas receberam o seguinte texto, com o intuito de esclarecer eventuais dúvidas a respeito do cenário:

A escala definida foi:

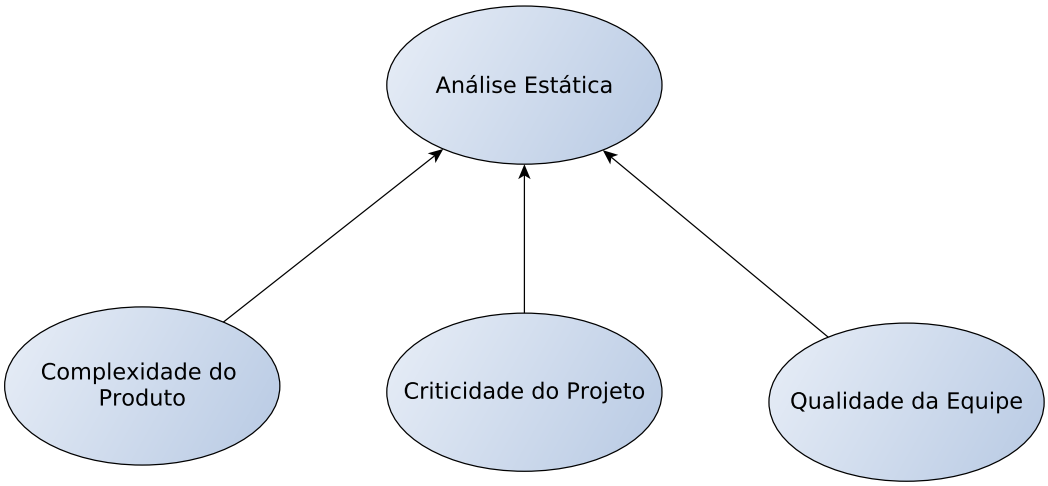
- Bom: (≤ 5)
- Médio: $(]5, 10])$
- Ruim: (> 10)

Legenda:

- **Complexidade do Produto:** Trata-se de quão complexo o *software* produzido é. Por exemplo, a complexidade "Baixa"poderia ser usada para uma agenda telefônica que conta apenas com cadastros;
- **Criticidade do Projeto:** Se for alta, trata-se de um projeto crítico, por exemplo, sistema de controle aéreo, bomba de insulina, etc; se for baixa, não é um projeto crítico, por exemplo, aplicativo de notícias;
- **Qualidade da Equipe:** Define o nível da equipe em todos os quesitos. Se for alta, trata-se de uma equipe integrada, motivada, que conhece as ferramentas e domina bem as tecnologias usadas.

Continua na próxima página

Segue, abaixo, a Rede Bayesiana criada.



Para cada cenário, selecione a saída esperada

Para cada cenário, informe a saída esperada para o valor de referência da métrica Análise Estática. Por exemplo, se: Complexidade do Produto: **Alta** | Criticidade do Projeto: **Alta** | Qualidade da Equipe: **Alta**, o valor de referência deve ser ≤ 5.

Complexidade do Produto: **Alta** | Criticidade do Projeto: **Baixa** | Qualidade da Equipe: **Baixa**:

- Resposta: *Bom (≤ 5).*

Complexidade do Produto: **Baixa** | Criticidade do Projeto: **Alta** | Qualidade da Equipe: **Baixa**:

- Resposta: *Bom (≤ 5).*

Complexidade do Produto: **Baixa** | Criticidade do Projeto: **Baixa** | Qualidade da Equipe: **Alta**:

- Resposta: *Ruim (> 10).*

Continua na próxima página

Complexidade do Produto: **Baixa** | Criticidade do Projeto: **Alta** | Qualidade da Equipe: **Alta**: • Resposta: *Bom (*≤ 5*).*
Complexidade do Produto: **Alta** | Criticidade do Projeto: **Baixa** | Qualidade da Equipe: **Alta**: • Resposta: *Médio (*$]5, 10]$*).*
Complexidade do Produto: **Alta** | Criticidade do Projeto: **Alta** | Qualidade da Equipe: **Baixa**: • Resposta: *Bom (*≤ 5*).*
Complexidade do Produto: **Alta** | Criticidade do Projeto: **Média** | Qualidade da Equipe: **Baixa**: • Resposta: *Bom (*≤ 5*).*
Complexidade do Produto: **Baixa** | Criticidade do Projeto: **Média** | Qualidade da Equipe: **Alta**: • Resposta: *Médio (*$]5, 10]$*).*
Complexidade do Produto: **Média** | Criticidade do Projeto: **Alta** | Qualidade da Equipe: **Baixa**: • Resposta: *Bom (*≤ 5*).*

A.2.2 Número de Bugs Minor

Juntamente com os questionários, os especialistas receberam o seguinte texto, com o intuito de esclarecer eventuais dúvidas a respeito do cenário:

A escala definida foi:

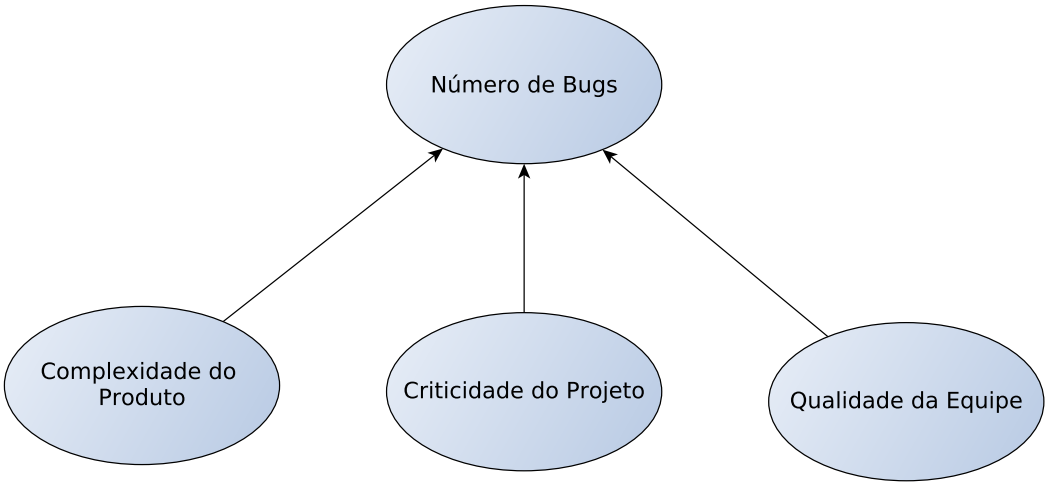
- Bom: (≤ 5)
- Médio: ($]5, 10]$)
- Ruim: (> 10)

Legenda:

- **Complexidade do Produto:** Trata-se de quão complexo o *software* produzido é. Por exemplo, a complexidade "Baixa"poderia ser usada para uma agenda telefônica que conta apenas com cadastros;
- **Criticidade do Projeto:** Se for alta, trata-se de um projeto crítico, por exemplo, sistema de controle aéreo, bomba de insulina, etc; se for baixa, não é um projeto crítico, por exemplo, aplicativo de notícias;
- **Qualidade da Equipe:** Define o nível da equipe em todos os quesitos. Se for alta, trata-se de uma equipe integrada, motivada, que conhece as ferramentas e domina bem as tecnologias usadas.

Continua na próxima página

Segue, abaixo, a Rede Bayesiana criada.



Para cada cenário, selecione a saída esperada

Para cada cenário, informe a saída esperada para o valor de referência da métrica Número de Bugs Minor. Por exemplo, se: Complexidade do Produto: **Alta** | Criticidade do Projeto: **Alta** | Qualidade da Equipe: **Alta**, o valor de referência deve ser ≤ 5.

Complexidade do Produto: **Alta** | Criticidade do Projeto: **Baixa** | Qualidade da Equipe: **Baixa**:

- Resposta: *Bom (≤ 5).*

Complexidade do Produto: **Baixa** | Criticidade do Projeto: **Alta** | Qualidade da Equipe: **Baixa**:

- Resposta: *Bom (≤ 5).*

Complexidade do Produto: **Baixa** | Criticidade do Projeto: **Baixa** | Qualidade da Equipe: **Alta**:

- Resposta: *Bom (≤ 5).*

Continua na próxima página

Complexidade do Produto: **Baixa** | Criticidade do Projeto: **Alta** | Qualidade da Equipe: **Alta**: • Resposta: *Bom (*≤ 5*).*
Complexidade do Produto: **Alta** | Criticidade do Projeto: **Baixa** | Qualidade da Equipe: **Alta**: • Resposta: *Médio (*$]5, 10]$*).*
Complexidade do Produto: **Alta** | Criticidade do Projeto: **Alta** | Qualidade da Equipe: **Baixa**: • Resposta: *Bom (*≤ 5*).*
Complexidade do Produto: **Alta** | Criticidade do Projeto: **Média** | Qualidade da Equipe: **Baixa**: • Resposta: *Bom (*≤ 5*).*
Complexidade do Produto: **Baixa** | Criticidade do Projeto: **Média** | Qualidade da Equipe: **Alta**: • Resposta: *Ruim (*> 10*).*
Complexidade do Produto: **Média** | Criticidade do Projeto: **Alta** | Qualidade da Equipe: **Baixa**: • Resposta: *Bom (*≤ 5*).*

A.2.3 Cobertura de Código

Juntamente com os questionários, os especialistas receberam o seguinte texto, com o intuito de esclarecer eventuais dúvidas a respeito do cenário:

A escala definida foi:

- Muito Baixo: $(0 - 30\%)$
- Baixo: $(31 - 50\%)$
- Médio: $(51 - 70\%)$
- Alto: $(71 - 80\%)$
- Muito Alto: $(81 - 100\%)$

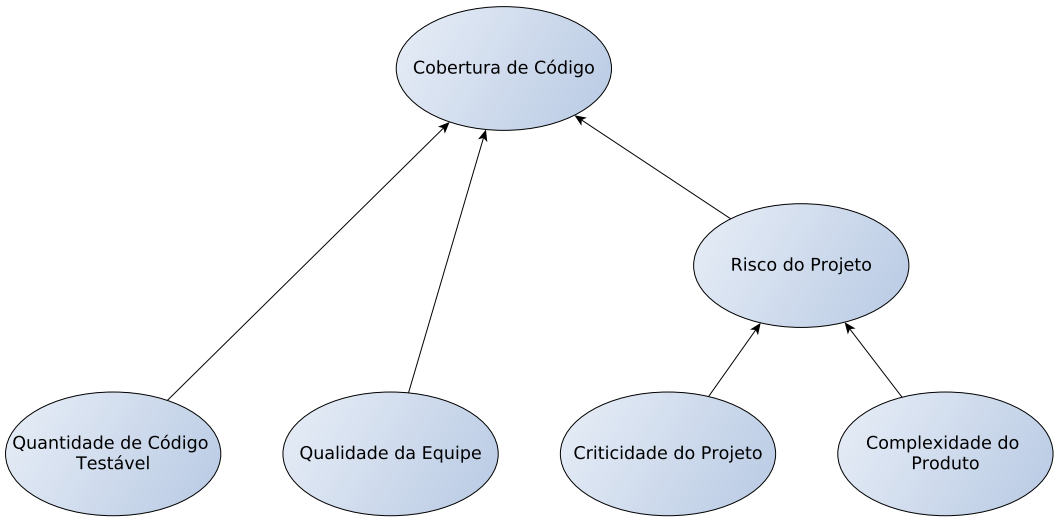
Continua na próxima página

Legenda:

- **Criticidade do Projeto:** Se for alta, trata-se de um projeto crítico, por exemplo, sistema de controle aéreo, bomba de insulina, etc; se for baixa, não é um projeto crítico, por exemplo, aplicativo de notícias;
- **Complexidade do produto:** Trata-se de quão complexo o *software* produzido é. Por exemplo, A complexidade "Muito Baixa" poderia ser usada para uma agenda telefônica que conta apenas com cadastros;
- **Quantidade de Código Testável:** Define a quantidade de código do projeto que deve ser testado. É importante destacar que na Cobertura de Código não deve-se usar filtros. Por exemplo, se for muito alto, trata-se de um *software desktop* ou *mobile* que possui quase todo o seu código é inscrito em Java, havendo, assim, muito código a ser testado;
- **Qualidade da Equipe:** Define o nível da equipe em todos os quesitos. Se for alta, trata-se de uma equipe integrada, motivada, que conhece as ferramentas e domina bem as tecnologias usadas;
- **Risco do Projeto:** É a combinação de alta criticidade e alta complexidade do projeto.

Continua na próxima página

Segue, abaixo, a Rede Bayesiana criada.



Para cada cenário, selecione é a saída esperada

Para cada cenário, selecione a saída esperada para o valor de referência da métrica Cobertura de Código. Por exemplo, se: Criticidade do Projeto: **Muito Alta** | Complexidade do Produto: **Muito Alta**, o valor de referência deve ser Muito Alto ($81 - 100\%$). É importante ressaltar que nessa etapa só são levados em consideração os fatores Criticidade do Projeto e Complexidade do Projeto.

Criticidade do Projeto: **Muito Alta** | Complexidade do Produto: **Muito Baixa**:

- Resposta: *Alto (*$71 - 80\%$*).*

Criticidade do Projeto: **Muito Baixa** | Complexidade do Produto: **Muito Alta**:

- Resposta: *Alto (*$71 - 80\%$*).*

Criticidade do Projeto: **Muito Baixa** | Complexidade do Produto: **Média**:

- Resposta: *Baixo (*$31 - 50\%$*).*

Continua na próxima página

Criticidade do Projeto: **Média** | Complexidade do Produto: **Muito Baixa**: • Resposta: *Baixo (*$31 - 50\%$*).*
Criticidade do Projeto: **Média** | Complexidade do Produto: **Alta**: • Resposta: *Alto (*$71 - 80\%$*).*
Criticidade do Projeto: **Baixa** | Complexidade do Produto: **Alta**: • Resposta: *Médio (*$51 - 70\%$*).*

Para cada cenário, selecione a saída esperada
Para cada cenário, selecione a saída esperada para o valor de referência da métrica Cobertura de Código. Por exemplo, se: Quantidade de Código Testável: **Muito Alta** | Qualidade da Equipe: **Muito Baixa** | Risco do Projeto: **Muito Alto**, o valor de referência deve ser Muito Alto ($81 - 100\%$). Destaque-se que a Qualidade da Equipe é inversamente proporcional à Cobertura de Código. É importante ressaltar que nessa etapa só são levados em consideração os fatores Quantidade de Código Testável, Qualidade da Equipe e o fator intermediário Risco do Projeto.
Quant. de Código Testável: **Muito Alta** | Qualidade da Equipe: **Muito Baixa** | Risco do Projeto: **Muito Baixo**: • Resposta: *Alto (*$71 - 80\%$*).*
Quant. de Código Testável: **Muito Baixa** | Qualidade da Equipe: **Muito Alta** | Risco do Projeto: **Muito Baixo**: • Resposta: *Muito Baixo (*$0 - 30\%$*).*

Continua na próxima página

Quant. de Código Testável: **Muito Baixa** | Qualidade da Equipe: **Muito Baixa** | Risco do Projeto: **Muito Alto**: • Resposta: *Muito Alto (*$81 - 100\%$*).*
Quant. de Código Testável: **Muito Baixa** | Qualidade da Equipe: **Muito Alta** | Risco do Projeto: **Muito Alto**: • Resposta: *Muito Alto (*$81 - 100\%$*).*
Quant. de Código Testável: **Muito Alta** | Qualidade da Equipe: **Muito Baixa** | Risco do Projeto: **Muito Alto**: • Resposta: *Muito Alto (*$81 - 100\%$*).*
Quant. de Código Testável: **Muito Alta** | Qualidade da Equipe: **Muito Alta** | Risco do Projeto: **Muito Baixo**: • Resposta: *Muito Alto (*$81 - 100\%$*).*
Quant. de Código Testável: **Muito Alta** | Qualidade da Equipe: **Média** | Risco do Projeto: **Muito Baixo**: • Resposta: *Alto (*$71 - 80\%$*).*
Quant. de Código Testável: **Muito Baixa** | Qualidade da Equipe: **Média** | Risco do Projeto: **Muito Alto**: • Resposta: *Muito Alto (*$81 - 100\%$*).*

Continua na próxima página

Quant. de Código Testável: **Média** | Qualidade da Equipe: **Muito Alta** | Risco do Projeto: **Muito Baixo**:

- Resposta: *Médio (*$51 - 70\%$*).*