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

Uma Abordagem Baseada em Redes Bayesianas para Auxiliar a Interpretação de Métricas de Software

Amaury Bartolomeu Carneiro de Medeiros

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

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Metodologia e Técnicas da Computação

Kyller Costa Gorgônio
Hyggo Oliveira de Almeida
(Orientadores)

Campina Grande, Paraíba, Brasil

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

M488a Medeiros, Amaury Bartolomeu Carneiro de.

Uma abordagem baseada em redes bayesianas para auxiliar a interpretação de métricas de software / Amaury Bartolomeu Carneiro de Medeiros. – Campina Grande, 2015.

128 f. : il. color.

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

"Orientação: Prof. Kyller Costa Gorgônio, Prof. Hyggo Oliveira de Almeida".

Referências.

1. Métrica de Programas. 2. Software - Métricas. 3. Software - Desenvolvimento. 4. Redes Bayesianas. I. Gorgônio, Kyller Costa. II. Almeida, Hyggo Oliveira de. III. Título.

CDU 004.412(043)

"UMA ABORDAGEM BASEADA EM REDES BAYESIANAS PARA AUXILIAR A INTERPRETAÇÃO DE MÉTRICAS DE SOFTWARE"

AMAURY BARTOLOMEU CARNEIRO DE MEDEIROS

DISSERTAÇÃO APROVADA EM 02/12/2015

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

HYGGO OLIVEIRA DE ALMEIDA, D.Sc, UFCG
Orientador(a)

ANGELO PERKUSICH, D.Sc, UFCG
Examinador(a)

AYLA DÉBORA DANTAS DE SOUZA REBOUÇAS, D.Sc, UFPB
Examinador(a)

CAMPINA GRANDE - PB

Resumo

Apesar do alto número de métricas de software que vêm sendo apresentadas desde a década de 1960, sua adoção e implantação ainda é limitada em diversas situações. Um desafio encontrado ao se usar métricas é interpretá-las para se fazer análises e predições em projetos de desenvolvimento de software. Alguns pesquisadores propuseram abordagens para definir limiares que determinam se um valor medido para uma métrica é aceitável ou não, com o intuito de auxiliar desenvolvedores e gerentes a interpretá-la. Essas abordagens, no entanto, não consideram riscos e outros fatores subjetivos que têm impacto no processo de medição e que podem influenciar a interpretação das métricas e, consequentemente, nas decisões do gerente. Outros pesquisadores propuseram modelos que combinam métricas de software e fatores subjetivos para auxiliar o processo de tomada de decisões, mas eles não consideraram riscos na interpretação, como problemas nos processos de coleta e relatório de métricas ou o mau uso destas. Nesta pesquisa, é proposta uma abordagem para se construir redes Bayesianas para auxiliar a interpretação de métricas considerando esses riscos. As redes Bayesianas construídas auxiliam os gerentes a identificar riscos relacionados a métricas e fatores controladores para mitigá-los. O objetivo é maximizar a acurácia das métricas e minimizar o número de decisões erradas tomadas com base em métricas de software. A abordagem foi validada com sucesso em um estudo de caso aplicado em quatro projetos e foi concluído que se trata de uma abordagem promissora para auxiliar gerentes e desenvolvedores a interpretar métricas e dar suporte ao processo de tomada de decisão em projetos de software.

Abstract

Despite the large amount of software metrics that has been proposed since the 1960s, their adoption and application is still limited in many situations. A challenge in using metrics is to interpret them to make assessments and predictions regarding software development projects. Several researchers proposed approaches to define thresholds to determine whether the value of a metric is acceptable, in order to help the developers and managers to interpret it. These approaches, however, do not consider risks and other subjective factors that have impact in the measurement process and might influence the metrics' interpretation and consequently the manager's decision. Other researchers proposed models combining software metrics and subjective factors to assist on decision-making, but they did not consider interpretation risks such as problems in metrics' collection and reporting process and metrics misuse. In this research, we propose an approach to construct Bayesian networks to assist on metrics interpretation considering these risks. The Bayesian networks constructed help the managers to identify risks related to the metrics and controller factors to mitigate them. The goal is to maximize the metrics' accuracy and minimize wrong decisions based on software metrics. The approach was successfully validated with a case study performed with four projects and we concluded that it's a promising approach to assist practitioners to interpret metrics and support software projects managerial decision-making.

Agradecimentos

A realização deste trabalho não seria possível sem o apoio da minha família. Agradeço-os do fundo do coração por terem me dado educação e me criado de forma que eu me tornasse um homem de bem que está sempre em busca de expandir o seu conhecimento.

Agradeço aos meus amigos (em especial a OFF Thread) e namorada por todo amor, carinho e companheirismo fornecidos. Vocês me ajudaram a manter o foco e me deram forças para que eu continuasse me esforçando ao máximo para concluir este trabalho.

Grande parte desta pesquisa se deve à orientação que obtive durante a sua realização. Agradeço ao professor Kyller Gorgônio pela orientação, eventuais puxões de orelha e suas colaborações com o trabalho. Agradeço também ao professor Hyggo Almeida pela co-orientação e por me ajudar a tornar-me um melhor pesquisador. Além deles, sou muito grato a Mirko Perkusich, que, especialmente no momento da gênese deste trabalho, me ajudou bastante a conceber a ideia central da pesquisa.

Aos amigos do CGHackspace, meu muito obrigado. Tanto pelos conselhos profissionais e pessoais fornecidos no decorrer do mestrado, quanto pela companhia e discussões durante a escrita desta dissertação.

Registro também o meu profundo agradecimento aos gerentes de projeto do Laboratório de Sistemas Embarcados e Computaçãoo Pervasiva (*Embedded*) pelo tempo dedicado à participação na pesquisa.

Finalmente, agradeço ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro parcial para o desenvolvimento desta pesquisa.

Conteúdo

Lista de Símbolos

GAD - *Grafo Acíclico Dirigido*

GQM - *Goal-Question-Metric*

LOC - *Linhas de Código*

MAT - *Modelo de Aceitação de Tecnologia*

MUSE - *Máxima da Incerteza em Engenharia de Software*

P&D - *Pesquisa e Desenvolvimento*

SMILE - *Structural Modeling, Inference, and Learning Engine*

TPN - *Tabela de Probabilidade dos Nós*

UFCG - *Universidade Federal de Campina Grande*

UPG - *Unidade de Processamento Gráfico*

XP - *Extreme Programming*

Lista de Figuras

Lista de Tabelas

Lista de Códigos Fonte

Capítulo 1

Introdução

Métricas de software possibilitam a medição, avaliação, controle e melhoria de produtos e processos de software [14]. Essas métricas podem ser utilizadas para se obter uma vasta variedade de informação sobre a qualidade do produto entregue ao cliente, o progresso de um projeto de software, a complexidade (ou tamanho) do sistema e estimativa de custos. Portanto, é muito importante que suas medições sejam monitoradas com frequência, principalmente considerando que os requisitos de um sistema podem mudar ao longo de seu desenvolvimento [16].

De acordo com Fenton e Neil [34], o conceito de métricas de software pode ser dividido em dois componentes distintos: o primeiro é a definição da métrica, em si; o segundo componente, o qual é abordado mais amplamente neste trabalho, foca na coleta das métricas, no seu gerenciamento e em como essas métricas devem ser utilizadas no contexto de projetos de desenvolvimento de software.

Apesar de toda a sua importância, de acordo com Martin Fowler[1], métricas são simplificações de atributos muito mais complexos. Comportamentos, ou características, difíceis de se mensurar são representados de forma simples por números e/ou gráficos. O ato de reduzir essa complexidade implica no custo de perder de vista os reais objetivos do projeto, o que pode fazer com que este atinja resultados subótimos.

O uso de métricas para se avaliar projetos de desenvolvimento de software não é algo recente. Dezenas de métricas foram propostas desde a década de 1960 [34] [4]. Elas são consideradas recursos essenciais para melhorar a qualidade e controlar os custos durante

[1] `http://martinfowler.com/articles/useOfMetrics.html`

o andamento de projetos. No entanto, apesar da alta quantidade de métricas existentes, sua adoção e aplicação em projetos ainda é limitada [18]. Um dos desafios encontrados ao utilizar métricas de software é interpretá-las de forma a dar suporte no processo de tomada de decisões acerca dos projetos de desenvolvimento. Métricas são muito importantes, mas só são úteis se forem entendidas pelos envolvidos na sua coleta, análise e relatórios.

Nesse contexto, alguns pesquisadores produziram trabalhos com o objetivo de definir limiares para determinar se os valores observados para uma métrica são aceitáveis, ou seja, se a métrica é confiável o suficiente para ser utilizada no processo de tomada de decisões acerca do projeto. Por exemplo, Foucault et al. [26] apresentaram uma abordagem estatística para calcular limiares para métricas de software em qualquer contexto. No entanto, os autores não consideram riscos e outros fatores subjetivos que também podem impactar no processo de medição e, consequentemente, influenciar a interpretação da métrica.

Neste trabalho, três principais fatores são considerados como riscos no processo de medição de uma métrica: (i) processo de coleta; (ii) processo de relatório da métrica, como o processo de reportar *bugs*; e (iii) mau uso da métrica, como por exemplo a escrita de métodos falsos com o intuito de aumentar a cobertura dos testes. Esses riscos impactam na confiança de usar a métrica no processo de tomada de decisão. Por exemplo, uma baixa qualidade no processo de testes e no processo de relatar *bugs* poderia reduzir significativamente a confiança da métrica *Número de bugs*.

1.1 Problemática

De acordo com Wallace e Sheetz [18], mais de 80% das iniciativas de se usar métricas de software falham nos primeiros 18 meses. A explicação para esse fenômeno, em geral, baseia-se no argumento de que as medições são complicadas de se entender e que a implantação das métricas nos projetos é um processo difícil. Além disso, algumas métricas possuem aplicabilidade limitada, não são validadas com muito rigor, ou sua utilidade não é muito aparente. Idealmente, a confiança no valor observado de uma métrica deve ser a maior possível para que esta seja utilizada com sucesso no contexto de projetos de desenvolvimento de software.

A confiança, ou acurácia, de uma métrica pode sofrer influência de problemas com ferra-

mentas ou comportamentos humanos indesejados. Por exemplo, uma ferramenta de análise estática pode apresentar falsos positivos. Além disso, os desenvolvedores podem produzir erros na medição por falta de capacidade técnica e/ou experiência, ou ainda para mascarar os dados e apresentar melhores números ao gerente. Esse último caso geralmente acontece se o desenvolvedor não concorda com as metas estabelecidas pelo gerente ou pelo cliente. De todo modo, os valores de métricas observados podem fazer com que profissionais tomem decisões erradas.

Alguns estudos já apontaram que alguns riscos podem impactar na confiança de métricas. Por exemplo, Çalıklı e Benner [10] relataram que desenvolvedores tendem a escrever e executar testes de unidade para que eles passem, ao invés de apontarem erros indicando que o software não está de acordo com as especificações. Nesse caso, mesmo que as métricas *Número de testes* ou *Cobertura de código* estejam dentro da meta estabelecida e, consequentemente, indicando uma boa situação no que diz respeito a testes, estes podem não estar refletindo a realidade, pois a cobertura dos testes medida foi deturpada pelos desenvolvedores.

Em uma outra pesquisa, Cavalcanti et al. [42] observaram que fatores subjetivos, como o perfil do relator de *bugs*, podem ocasionar em um número incorreto de *bugs* relatados, por causa da presença de duplicatas. Em um estudo empírico realizado pelos autores, identificou-se que, em um dos projetos observados, 49% dos *bugs* eram duplicados. Nesse caso, portanto, a confiança da métrica *Número de bugs* é reduzida.

Erros e riscos associados com as métricas podem causar uma má interpretação destas e, consequentemente, um processo de tomada de decisões longe do ideal. Além disso, métricas com baixa confiança possuem utilidade e aplicabilidade limitadas, portanto há uma perda de tempo e de recursos para implantá-las e utilizá-las. É preciso, então, considerar os fatores subjetivos e fontes de erro envolvidos com as métricas para auxiliar a sua interpretação.

Conforme já mencionado, métricas têm sua utilidade limitada se suas medições não forem bem interpretadas e utilizadas no processo de tomada de decisão. Nesse contexto, alguns pesquisadores, como Ahmed e Muzaffar [22], Spasic e Onggo [5] e Wagner [41] propuseram modelos que aliam métricas de software e fatores subjetivos para auxiliar a tomada de decisão em projetos.

Em seus trabalhos, esses autores utilizaram diferentes técnicas, como redes Bayesianas

e simulação baseada em agentes, para tentar predizer problemas e fazer recomendações com relação a produtos, processos e projetos de software em geral. Por outro lado, nenhum deles considerou erros associados com a medição, que podem impactar na acurácia das métricas.

Portanto, há espaço para a investigação do uso de fatores subjetivos relacionados a métricas, que podem inserir erro no processo de medição em projetos de software. Essas fontes de erro podem estar relacionadas com o processo de coleta da métrica, com o processo de relatá-la, ou com o mau uso da métrica por parte dos envolvidos no projeto. A identificação das fontes de erro é o primeiro passo para contorná-las, a fim de se aumentar a confiança da métricas sobre as quais elas têm impacto.

1.2 Objetivos

Considerando o que foi abordado nas seções anteriores, o objetivo principal deste trabalho é mitigar os problemas levantados, principalmente na Seção 1.1, e aumentar a acurácia e confiança das métricas coletadas em projetos de desenvolvimento, propondo uma abordagem para auxiliar os gerentes e times de desenvolvimento a interpretar métricas de software em seus projetos, considerando fatores subjetivos e fontes de erro envolvidos no processo de medição dessas métricas.

De forma a modelar as métricas dos projetos e os riscos a estas associados, optou-se pelo uso de redes Bayesianas (estruturas representadas por grafos acíclicos dirigidos, nas quais cada nó representa uma variável aleatória, que pode assumir determinados valores calculados probabilisticamente de acordo com os valores dos seus pais) no desenvolvimento da abordagem, visto que outras pesquisas já mostraram que estas são adequadas para se modelar incerteza em um domínio [13]. Como fatores subjetivos inserem incerteza com relação à confiança da métrica, o uso dessas estruturas se mostrou adequado na pesquisa.

Esse objetivo geral da pesquisa pode ser dividido nos seguintes objetivos específicos:

1. propor uma abordagem baseada em redes Bayesianas para auxiliar a interpretação de métricas de software;

2. proporcionar aos gerentes de projetos de desenvolvimento de software uma forma mais objetiva de lidar com os fatores subjetivos relacionadas com as métricas utilizadas no

projeto, por meio da modelagem de tais fatores juntamente com as métricas;

3. aplicar a abordagem em projetos de desenvolvimento de software para observar sua utilidade e seu custo-benefício em projetos reais;

4. modelando as métricas e os fatores subjetivos envolvidos nos projetos, permitir que os gerentes possam identificar e corrigir fontes de problema nos seus projetos com o objetivo de se aumentar a confiança nas métricas coletadas.

1.3 Contribuições e Resultados

A abordagem proposta neste trabalho é dividida em etapas que englobam desde a identificação de métricas coletadas no projeto até a construção das redes Bayesianas que modelam tais métricas. Com a construção desses modelos, é possível calcular, de forma probabilística, o impacto de fatores subjetivos envolvidos na medição, como riscos na coleta e ao relatar as métricas. Aplicando a abordagem no seu projeto, o gerente pode observar as saídas dos modelos, que irão indicar se as métricas correspondentes são confiáveis para serem utilizadas no processo de tomada de decisão ou se suas medições devem ser desconsideradas, e assim tomar decisões mais sólidas.

Um efeito colateral positivo observado na pesquisa foi a identificação de possíveis fontes de problemas no projeto como um todo. Como os modelos construídos com a aplicação da abordagem proposta contêm riscos envolvidos com as métricas coletadas no projeto e fatores que possam controlar e/ou mitigar tais riscos, é possível que um gerente visualize potenciais causas de erro, como uma ferramenta mal configurada ou um time com baixa experiência, ao analisar um modelo construído para um determinado grupo de métricas. Isso pode fazer com que os problemas sejam corrigidos e a confiança na métrica envolvida com estes riscos aumente.

A abordagem proposta foi validada com sucesso por meio de um estudo de caso que utilizou quatro projetos de desenvolvimento de software como unidades de análise. De acordo com os gerentes dos projetos (ou sujeitos, no contexto do estudo de caso), a abordagem se mostrou útil em auxiliar o time na interpretação das métricas e o custo-benefício, considerando o esforço em aplicá-la, foi considerado satisfatório.

1.4 Relevância

A abordagem proposta se apresenta como uma alternativa promissora para auxiliar o processo de tomada de decisão em projetos de desenvolvimento de software. Com as recomendações obtidas por meio da observação das saídas dos modelos gerados para os projetos, os gerentes podem interpretar as métricas coletadas de uma forma mais objetiva e, assim, diagnosticar problemas e propor soluções para contorná-los.

Esta pesquisa também pode originar importantes benefícios no sentido de diminuir, de forma indireta, os custos de um projeto. Como as redes Bayesianas construídas por meio da aplicação da abordagem indicam se as métricas coletadas possuem confiança suficiente para serem consideradas no processo de tomada de decisão, os gerentes podem optar por não realizar a coleta de métricas consideradas irrelevantes, dessa forma economizando recursos. Por exemplo, a métrica *Burndown* da *Sprint* exige um esforço considerável do time, que deve elencar bem as tarefas, dedicando um tempo para fazer a estimativa de esforço de cada uma delas, atualizar com frequência o estado de cada uma das tarefas e tentar não iniciar tarefas que não estejam no escopo da *Sprint* (esforço adicionado). Em alguns casos, portanto, o uso dessa métrica pode consumir mais tempo e recursos do que trazer benefícios. Além disso, os gerentes podem tomar certas iniciativas com o intuito de mitigar os riscos envolvidos com as métricas, fazendo com que o investimento necessário para as suas medições não seja desperdiçado.

Por fim, este trabalho pode ainda ser expandido com o objetivo de simplificar a abordagem proposta de modo que o envolvimento direto do pesquisador se torne desnecessário. Por exemplo, automatizando o processo de construção de redes Bayesianas, é possível fazer com que a construção dos modelos seja feita de forma transparente para os gerentes de projeto, que iriam apenas alimentar os modelos e observar a saída. Uma evolução do trabalho que simplificaria ainda mais o esforço dos gerentes seria a obtenção dos valores das métricas de forma automática, automatizando assim a alimentação dos modelos.

1.5 Estrutura da Dissertação

Os demais conteúdos da dissertação são divididos em outros cinco capítulos:

- no **Capítulo 2**, são apresentados os conceitos relacionados com o presente trabalho e utilizados na pesquisa;
- no **Capítulo 3**, a abordagem de auxílio à interpretação de métricas é apresentada;
- a validação da abordagem é feita no **Capítulo 4**, no qual se descreve o estudo de caso realizado, utilizando quatro projetos de desenvolvimento de software, assim como uma discussão dos resultados obtidos;
- no **Capítulo 5**, são discutidas algumas pesquisas relacionadas com este trabalho, suas limitações e as contribuições desta pesquisa no intuito de melhorá-las;
- por fim, as considerações finais, limitações do trabalho e propostas de trabalhos futuros são apresentadas no **Capítulo 6**.

Capítulo 2

Fundamentação Teórica

Neste capítulo são discutidos os principais conceitos e técnicas que foram utilizados no decorrer da pesquisa. Na Seção 2.1, é feita uma introdução a métricas de software e discute-se o seu uso no contexto de projetos de desenvolvimento, enquanto na Seção 2.2 aborda-se a temática de redes Bayesianas.

2.1 Métricas de Software

Métricas são utilizadas com frequência no cotidiano de quase todos os indivíduos. O preço de produtos, por exemplo, é uma métrica comum e geralmente define qual item deve ser comprado. Outras métricas, como altura e largura, podem ser utilizadas para encontrar roupas de tamanho adequado, por exemplo. O processo de coleta de métricas é, portanto, o ato de associar números e símbolos a entidades do mundo real, de forma a descrevê-las utilizando regras bem definidas [38].

Existem métricas, geralmente um pouco mais complexas, que são aplicadas em áreas específicas. Por exemplo, investidores de ações podem utilizar métricas para os auxiliar a montar sua carteira. Métricas em sistemas de radar, por sua vez, podem auxiliar a detecção de outras aeronaves quando o tempo estiver fechado.

No contexto da engenharia de software, métricas são essenciais. Elas podem tentar mensurar diversos fatores de projetos de desenvolvimento, como: estimativa de custo e esforço; qualidade, complexidade e segurança do sistema; maturidade; e qualidade de métodos e ferramentas. Portanto, muitos dos melhores desenvolvedores utilizam métricas para ter noção

da qualidade do sistema, se o software está pronto para ser implantado, ou até mesmo se o *design* do sistema foi feito de forma satisfatória. Gerentes, por sua vez, podem analisar essas métricas para verificar o que pode ser melhorado ou ajustado em um projeto de desenvolvimento de software.

Uma das principais vantagens de se utilizar métricas de software em um projeto de desenvolvimento é o aumento da mantenabilidade do sistema [32], isto é, a facilidade com que o software pode ser modificado para corrigir falhas, melhorar o desempenho ou se adaptar a um novo ambiente. O bom uso de métricas coletadas auxilia no controle do projeto e proporciona uma boa mantenabilidade ao sistema, o que, consequentemente, implica em menos custos e esforços durante a evolução do software.

Segundo Fenton e Bieman [38], a coleta de métricas não é só útil, mas necessária. Afinal, não há como afirmar que tudo está bem com um projeto se não existem métricas que apontem isso. Portanto, fazer medições é uma prática importante para, no mínimo, conhecer o estado de projetos, processos e produtos.

Com relação ao gerenciamento de projetos de desenvolvimento de software, clientes e desenvolvedores tendem a observar gráficos de métricas para verificar o andamento do projeto e para tomar decisões acerca deste. Segundo Fenton e Neil [34], métricas devem prover informações para dar suporte quantitativo à tomada de decisões gerenciais, o que também implica em análise e redução de riscos.

Muitas empresas e organizações definem um conjunto padrão de métricas e métodos de relatório com o objetivo de auxiliar no planejamento e na qualidade do software produzido. Adotar as mesmas métricas em diferentes projetos também facilita a comparação entre eles. Nesse contexto, métricas podem ser apresentadas de uma forma que desenvolvedores e clientes possam observar o estado dos projetos, mesmo que não tenham um conhecimento profundo das tecnologias utilizadas, como linguagens de programação ou *hardware*.

De acordo com Fenton e Bieman [38], métricas, no seu cerne, consistem em dados que são coletados de alguma forma seguindo determinada regra. Por exemplo, a métrica *linhas de código* pode ser medida pela soma do número de linhas do código fonte com exceção de linhas em branco ou comentários. Para se determinar se os dados coletados são uma boa fonte de medição, estes devem seguir quatro características importantes:

- **corretude:** os dados devem ser coletados de acordo com as regras de definição da

métrica;

- **acurácia:** os dados coletados não devem divergir muito do valor real;
- **precisão:** o número de casas decimais dos dados coletados deve ser apropriado para a representação da métrica;
- **consistência:** os dados não devem apresentar diferenças alarmantes caso sejam medidos por outro sujeito ou ferramenta.

Existem diversos riscos associados com a acurácia da métrica. Umarji e Seaman [33], por exemplo, identificaram que desenvolvedores tendem a deturpar valores de métricas que medem a sua própria eficácia, por medo de reações adversas que podem ocorrer, como uma eventual demissão. Os autores também identificaram que diferentes grupos possuem diferentes interpretações sobre determinada métrica, o que pode fazer com que os valores reportados não condigam com os valores reais.

Nesta pesquisa, focou-se justamente em melhorar a **acurácia** das métricas. Foi proposta uma abordagem para auxiliar os gerentes de projeto a identificar riscos envolvidos com a medição e mitigadores/controladores (práticas, métodos ou ferramentas) para contorná-los, com o objetivo de maximizar a acurácia das métricas coletadas nos seus projetos. Dessa forma, com uma maior confiança nos valores medidos, os gerentes podem tomar decisões coerentes acerca do projeto.

2.2 Redes Bayesianas

Redes Bayesianas são um tipo de modelos gráficos probabilísticos e são utilizadas para representar conhecimento acerca de um domínio incerto [13]. Este tipo de modelo foi aplicado nesta pesquisa justamente para se trabalhar com as incertezas associadas com métricas de software coletadas em projetos de desenvolvimento.

Formalmente, uma rede Bayesiana B pode ser representada pela tupla $\langle G, \Theta \rangle$ [40]. Dado um conjunto finito de variáveis aleatórias $U = \{X_1, ..., X_{|U|}\}$, o primeiro componente da rede Bayesiana, G, é um Grafo Acíclico Dirigido (GAD) cujos vértices correspondem às variáveis aleatórias $X_1, ..., X_{|U|}$ e cujas arestas representam os relacionamentos diretos entre

essas variáveis. O componente Θ representa o conjunto de parâmetros que quantificam a rede. Para cada possível valor x_i de cada variável aleatória $X_i \in U$, Θ contém 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. Uma rede Bayesiana B define ainda uma distribuição de probabilidade conjunta sobre o conjunto U, que pode ser visualizada na equação 2.1.

$$P_B(X_1, ..., X_{|U|}) = \prod_{i=1}^{|U|} P_B(X_i|\Pi_{X_i}) = \prod_{i=1}^{|U|} \theta_{X_i|\Pi_{X_i}} \tag{2.1}$$

Graficamente, redes Bayesianas são geralmente representadas por estruturas similares a uma árvore. As funções de probabilidade podem ser representadas por tabelas-verdade. Na Figura 2.1, ilustra-se um exemplo fictício trivial de uma rede Bayesiana. Nota-se que o valor do nó *Sucesso do Piloto* depende do valor dos seus pais. Apesar de as arestas das redes Bayesianas representarem o relacionamento causal entre as variáveis, a informação pode propagar em qualquer direção no grafo [23]. Ou seja, é possível calcular as probabilidades de um nó pai se apenas o valor do seu filho for conhecido. Por exemplo, na Figura 2.1, caso o piloto tenha obtido sucesso e a qualidade do carro fosse ruim, é provável que o piloto fosse bom.

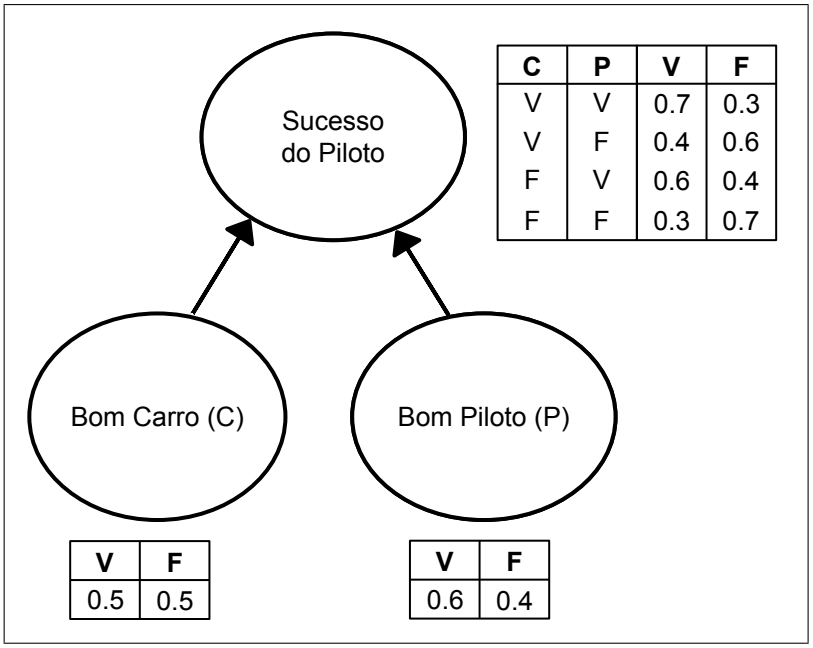

Figura 2.1: Exemplo de rede Bayesiana

Redes Bayesianas possuem diversas características positivas, como facilidade de lidar

com bases de dados pequenas e/ou incompletas, possibilidade de aprendizado, combinação de diferentes fontes de conhecimento, tratamento explícito de incertezas e rápido tempo de respostas [19]. Por esse motivo, elas são utilizadas em sistemas que possuem algum tipo de incerteza envolvida [8]. O uso de redes Bayesianas pode ser observado em sistemas especialistas, por exemplo para auxiliar a tomada de decisões seguras em ambientes de projetos complexos [20] ou ainda para predizer o desempenho em projetos de inovação que adotam liderança transformacional [24].

De acordo com a Máxima da Incerteza em Engenharia de Software (MUSE) [12], incerteza é algo inerente e inevitável em produtos e processos de desenvolvimento de software. Ziv e Richardson [12] propuseram um corolário que diz que incertezas de software devem ser modeladas e gerenciadas explicitamente por meio de técnicas de modelagem de incerteza, como redes Bayesianas. Os autores ainda mencionaram que redes Bayesianas provêem uma maneira matemática e computacional para modelar incertezas e que sua estrutura gráfica é adequada para representar sistemas de software. Além disso, engenheiros de software e cientistas da computação possuem conhecimento acerca de grafos e árvores, o que facilita a visualização de redes Bayesianas e permite uma melhor análise dos modelos, bem como a execução de modificações nas redes, caso necessário.

Por outro lado, o fato de incluir funções de probabilidade pode fazer com que o uso de redes Bayesianas tenha uma complexidade maior principalmente para engenheiros de software sem um conhecimento profundo e sólido em matemática. Felizmente, existem algumas abordagens para simplificar a definição das funções de probabilidade, como a utilização de nós ranqueados [36], assim como se faz neste trabalho.

Existem dois principais desafios ao se construir redes Bayesianas: a construção do GAD e a definição das funções de probabilidade [34]. Fenton e Neil [35] propuseram o que eles chamaram de conjunto de idiomas para auxiliar na construção do GAD e, portanto, tentar amenizar o primeiro problema. Esse idiomas são fragmentos de redes Bayesianas que representam graficamente tipos genéricos de modelagem de incerteza. Neste trabalho, foram utilizados os idiomas de causa-consequência, medição e síntese.

O idioma causa-consequência possibilita a construção de uma relação visual entre determinadas consequências e suas respectivas causas. Esse relacionamento causal pode partir das causas para se obter a consequência, ou partir de uma consequência observável para que

as probabilidades das causas sejam calculadas. O idioma causa-consequência é ilustrado em um exemplo na Figura 2.2. É possível que uma previsão acerca da frequência de sucesso de um piloto de corrida (consequência) seja feita tomando como base o conhecimento da qualidade do piloto e da qualidade do carro. Na abordagem proposta nesta pesquisa, o idioma causa-consequência foi utilizado principalmente para modelar o relacionamento entre riscos e fatores controladores.

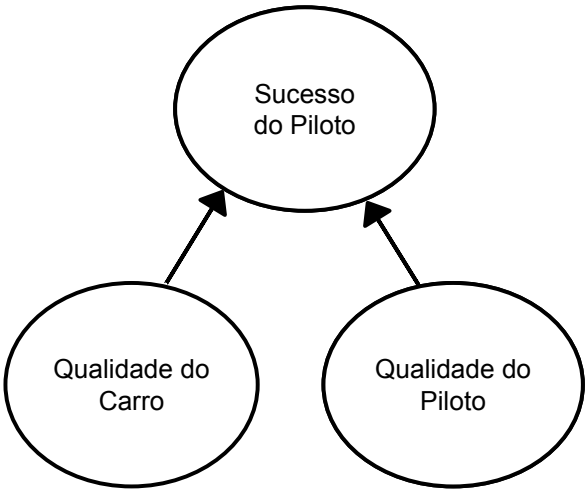

Figura 2.2: Idioma causa-consequência

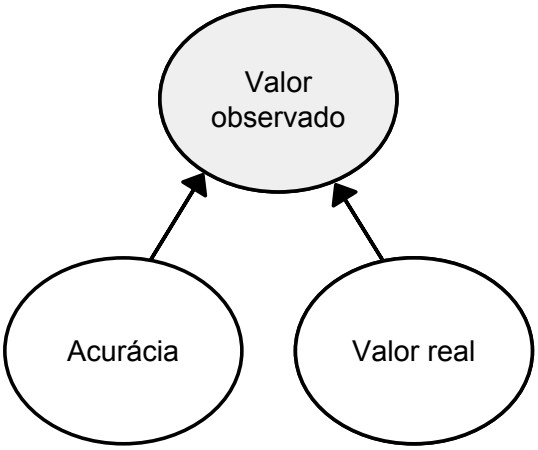

Figura 2.3: Idioma de medição

O idioma de medição modela a incerteza envolvida com a acurácia de qualquer tipo de

medição. Nesse idioma, como ilustrado na Figura 2.3, diz-se que o valor observado na medição possui influência da acurácia da medição e do valor real, geralmente desconhecido. O idioma de medição foi utilizado nesta pesquisa para modelar o erro das métricas coletadas nos projetos de desenvolvimento de software e o seu impacto na interpretação dessas métricas.

O idioma de síntese modela a combinação (ou síntese) de múltiplos nós em um só, com o propósito de organização da rede Bayesiana. O idioma de síntese, ilustrado na Figura 2.4, pode também ser utilizado para aumentar a eficiência da rede Bayesiana, evitando que alguns nós possuam uma quantidade grande de pais, o que adicionaria complexidade aos cálculos. Esse idioma foi utilizado na abordagem proposta neste trabalho pelos dois motivos mencionados.

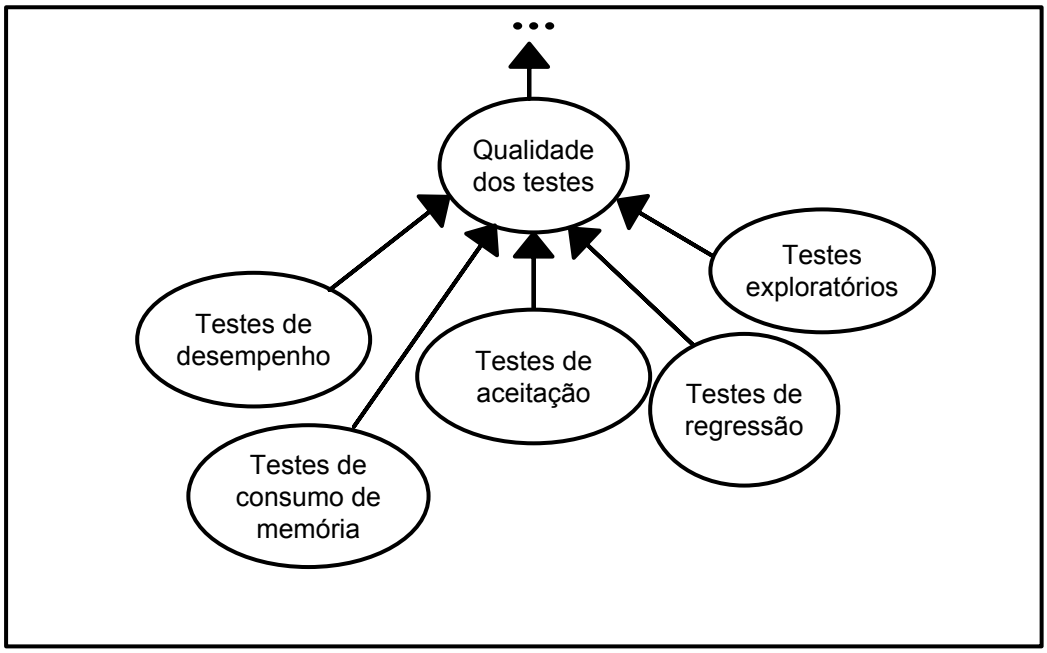

Figura 2.4: Idioma de síntese

Já com relação ao desafio da definição das funções de probabilidade, estas são geralmente representadas como Tabelas de Probabilidade dos Nós (TPN). Existem duas formas de se coletar dados e definir as TPN de uma rede Bayesiana: base de dados ou opinião de especialistas [29]. A definição das TPN de uma rede utilizando base de dados pode ser automatizada por meio de um processo chamado aprendizado em lotes [7]. No entanto, é

muito raro encontrar uma base de dados adequada para problemas práticos. Com relação a definir as TPN de uma rede manualmente por meio de opinião de especialistas, Fenton et al. [36] mostraram que vários tipos de inconsistências podem ocorrer se especialistas tentarem, exaustivamente, definir TPN de nós com um número muito alto (por exemplo, 125) de estados. Nesse caso, esse método de definição de TPN pode ser inviável.

Alguns pesquisadores conduziram trabalhos com o objetivo de diminuir a complexidade de utilizar a opinião de especialistas para definir TPN de redes Bayesianas. Das [3] propôs um algoritmo para popular TPN que utiliza heurísticas e estimativas para facilitar a aquisição de conhecimento do especialista. Fenton et al. [36] propuseram uma abordagem para utilizar nós ranqueados em redes Bayesianas. Essa abordagem é baseada numa distribuição normal duplamente truncada que usa como média, invariavelmente, um tipo de função ponderada que utiliza os nós pai. Perkusich et al. [29] apresentou, ainda, uma extensão de [36] na qual os autores propõem um método para coletar dados por meio de questionários e analisá-los estatisticamente para construir as TPN de uma rede Bayesiana.

Capítulo 3

Descrição da Abordagem Proposta

O objetivo principal da pesquisa realizada durante a concepção deste trabalho é propor uma abordagem de auxílio à interpretação de métricas coletadas no decorrer da execução de projetos de desenvolvimento de software. No contexto desta pesquisa, entende-se como interpretação de uma métrica o aumento da acurácia da sua medição e, consequentemente, da confiança do valor observado, ou seja, a proximidade desse valor com o valor real da métrica.

Por meio da abordagem proposta, gerentes e desenvolvedores podem identificar fatores subjetivos, como a qualidade da coleta da métrica ou do time responsável pela manutenção da métrica, que exercem influência sobre o valor observado da métrica. Esses fatores podem ser entendidos como fontes de erro envolvidas na medição. Ao identificá-las, os profissionais podem também adotar medidas para contorná-las. Dessa forma, a confiança das métricas coletadas nos projetos aumenta.

Concluiu-se que o idioma causa-consequência, descrito na Seção 2.2, se enquadra bem com a abordagem proposta. Então, optou-se pela utilização de redes Bayesianas para a construção dos modelos.

A abordagem proposta foi introduzida em Perkusich et al [28], podendo ser dividida em quatro macro etapas principais:

i – identificação das métricas do projeto;

ii – agrupamento de métricas do mesmo escopo;

iii – construção do Grafo Acíclico Dirigido (GAD);

iv – definição das funções de probabilidade.

Na Seção 3.1, discute-se uma visão geral da abordagem proposta, enquanto as seções seguintes se aprofundam um pouco mais em detalhes de cada uma das etapas.

3.1 Visão Geral

Como já mencionado, a abordagem proposta nesta pesquisa pode ser dividida em quatro etapas claramente distintas. A Figura 3.1 ilustra o fluxo da abordagem como um todo e as interações entre as etapas.

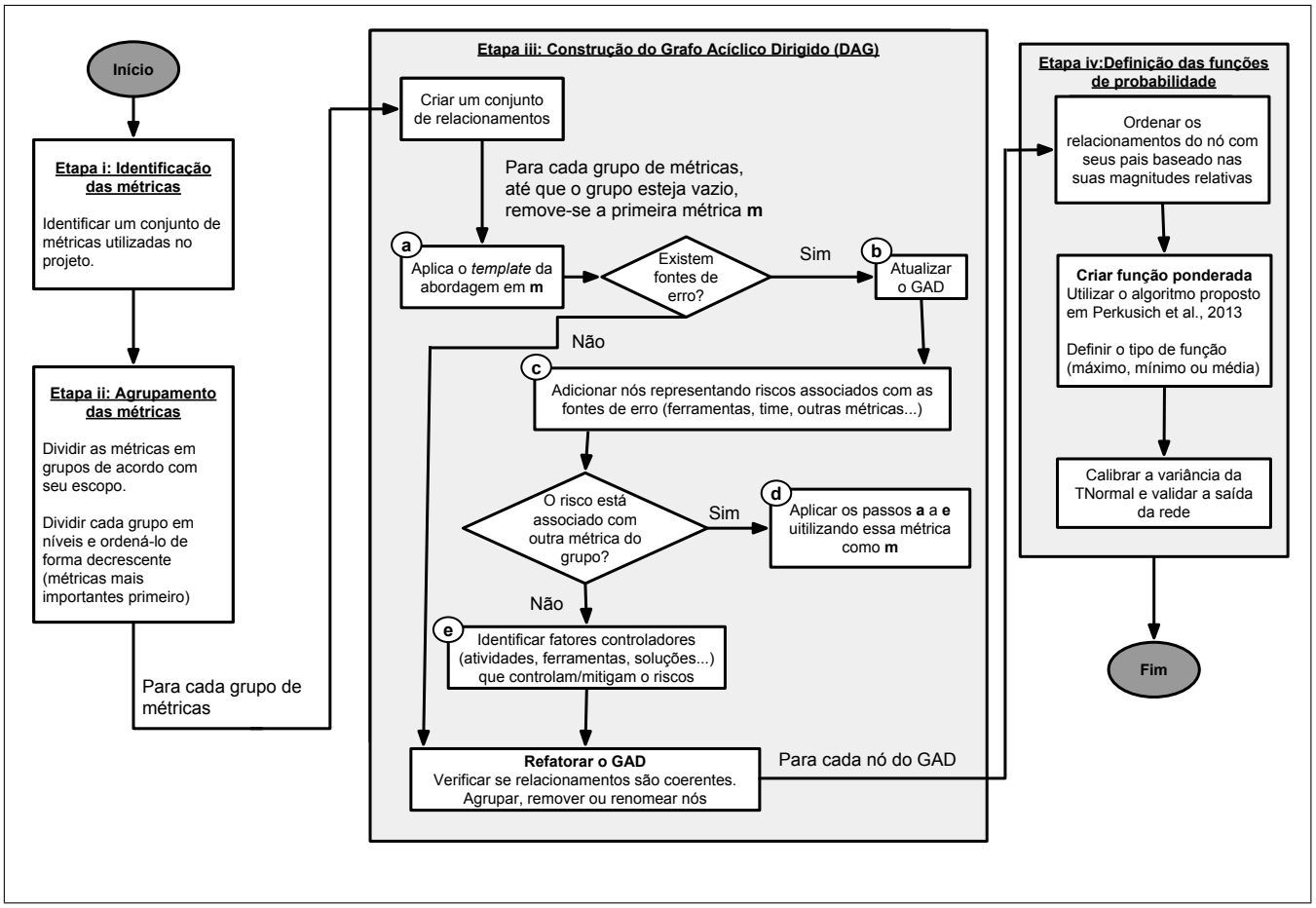

Figura 3.1: Método completo

Em um primeiro momento, faz-se uma reunião com o gerente do projeto que está sendo estudado. Este deve identificar as métricas utilizadas no seu projeto. Em seguida, as métricas

devem ser agrupadas em escopos. No contexto desta pesquisa, métricas possuem o mesmo escopo se elas se relacionam, de alguma maneira, uma com a outra. Por exemplo, em alguns projetos utilizados na pesquisa, os gerentes agruparam as métricas **Número de *bugs***, **Número de testes** e **Cobertura de código** no mesmo escopo, pois, de acordo com eles, as duas últimas métricas têm influência (direta ou indireta) sobre a primeira.

Em seguida, ainda na segunda etapa, o gerente deve ordenar os grupos por ordem de importância das métricas dentro do seu escopo de forma que as métricas mais importantes de um escopo apareçam primeiro no grupo correspondente.

Depois disso, tem início a etapa iii, na qual constrói-se um GAD para cada conjunto de métricas do projeto. O primeiro passo da construção do GAD é a criação de um conjunto de relacionamentos para representar o relacionamento entre os nós do grafo. Em seguida, até que o grupo de métricas atual esteja vazio, deve-se remover a primeira métrica deste e aplicar o que se chama de *template* da abordagem proposta sobre ela. O *template* da abordagem, em linhas gerais, representa a métrica, a interpretação do seu valor e as fontes de erro associados à medição. A construção do *template* é explicada de forma mais detalhada na Seção 3.4.1.

Ainda na terceira etapa da abordagem, caso existam fontes de erro associadas à métrica, o GAD deve ser atualizado de forma a conter os nós que representam tais fontes. Em seguida, devem ser adicionados nós representando riscos associados com cada fonte de erro. Tais riscos podem ser relacionados com ferramentas, com o time e até com outras métricas. Caso o risco esteja associado com outras métricas do grupo, o *template* da abordagem deve também ser aplicado nessas métricas e elas devem ser removidas do grupo. Se o risco não for associado com outras métricas, devem ser adicionados nós representando fatores controladores, que são atividades, ferramentas, práticas ou soluções com a função de controlar ou mitigar os riscos.

Adicionalmente, existe um modificador que pode, ou não, ser aplicado no cálculo da influência sobre cada nó. Esse modificador pode assumir o valor de *máximo* ou *mínimo*. O modificador *máximo* deve ser aplicado se o gerente considerar que o nó deve apresentar uma tendência para resultados mais positivos se pelo menos um dos seus pais apresentar resultado positivo. Por exemplo, considerando que três fatores (revisão de código, coleta de alertas análise estática e arquitetura bem definida) tenham sido elencados como mitigadores da fonte de erro código duplicado, aplica-se o modificador *máximo* se o gerente considerar

que se pelo menos um dos mitigadores for aplicado de forma satisfatória, a quantidade código duplicado já deve ser considerada satisfatória.

De maneira análoga, o modificador *mínimo* deve ser aplicado se o gerente achar que o valor do nó deve tender para resultados mais negativos se pelo menos um fator pai apresentar resultado negativo. Para exemplificar, presume-se que existam quatro fatores que influenciam a qualidade dos testes (qualidade dos testes de regressão, de usabilidade, funcionais e de unidade). Se a qualidade de um desses tipos de teste estiver ruim e o gerente considerar que isso deve implicar em uma qualidade geral dos testes igualmente ruim, o modificador *mínimo* deve ser utilizado.

O último passo da etapa de construção do GAD é o refatoramento do grafo. Deve-se revisar os relacionamentos para assegurar que estes são coerentes, além de agrupar, renomear ou remover nós. Os tipos de refatoramento do GAD são discutidos em detalhe na Seção 3.4.2.

A última etapa da abordagem consiste na definição das funções de probabilidade dos nós do GAD. Esta etapa é discutida mais amplamente na Seção 3.4.3. Para cada nó do GAD, deve-se primeiramente ordenar os relacionamentos do nó com seus pais baseado nas suas magnitudes relativas, de forma que os pais que exerçam mais influência sobre o nó sejam listados primeiro. Em seguida, baseado no algoritmo proposto por Perkusich et al. em [29], deve ser criada uma função ponderada a ser usada como média de uma distribuição duplamente truncada [36]. Em seguida, deve-se definir a variância da distribuição e esta pode ser calibrada pelo gerente posteriormente. A própria distribuição, no intervalo $[0, 1]$, representa a função de probabilidade do nó.

Finalmente, ao fim da última etapa, a saída da rede deve ser observada a fim de se identificar inconsistências evidentes e de corrigir eventuais falhas com certa antecedência, efetuando-se eventuais calibrações necessárias quando cabível.

No que diz respeito às próximas seções, na **Seção 3.2**, formaliza-se a etapa de identificação das métricas, na **Seção 3.3** é explicado como estas métricas devem ser agrupadas e na **Seção 3.4**, a mais extensa, o processo de construção das redes Bayesianas é detalhado.

3.2 Identificação das Métricas

O primeiro passo da abordagem proposta neste trabalho é a identificação das métricas utilizadas no projeto ao qual a abordagem é aplicada. As métricas devem ser apontadas pelo gerente do projeto ou por outro responsável cabível. O conjunto de métricas é definido pela equação 3.1, onde m_i representa uma métrica e $|M|$ representa o número de métricas coletadas no projeto.

$$M = \{m_1, \ldots, m_{|M|}\} \tag{3.1}$$

3.3 Agrupamento das Métricas

Depois que o conjunto de métricas do projeto é definido, inicia-se a segunda etapa, na qual as métricas são agrupadas de acordo com o seu escopo.

Diante disto, cria-se um conjunto de grupos de métricas G, representado pela equação 3.2, contendo todos os grupos de métrica X pertencentes ao projeto. Cada grupo X, por sua vez, é representado por um conjunto de métricas pertencentes a um mesmo escopo e obedece às restrições representadas pelas equações 3.3 e 3.4 que, respectivamente, asseguram que todas as métricas pertencentes a um grupo também pertencem ao conjunto de métricas M e todos os grupos de métricas são representados por conjuntos disjuntos.

Com relação à restrição representada pela equação 3.4, decidiu-se que nenhuma métrica pode pertencer a mais de um grupo com o intuito de se evitar, principalmente, duas coisas: a criação de grupos muito gerais e pouco representativos, o que pode causar uma baixa qualidade na execução da abordagem proposta, pois o gerente pode acabar escolhendo os grupos sem exercer o esforço adequado para produzir grupos de qualidade; e o fato de uma mesma métrica ser considerada múltiplas vezes e, assim, inserir uma complexidade maior na construção da rede Bayesiana, além de serem criados múltiplos nós de saída para uma mesma métrica, dificultando (e não auxiliando) a interpretação destas.

$$G = \{X_1, \ldots, X_{|G|}\} \tag{3.2}$$

$$\forall X \in G,\ X \subseteq M \tag{3.3}$$

$$\forall X_i, X_j \in G,\ i \neq j \Rightarrow X_i \cap X_j = \varnothing \tag{3.4}$$

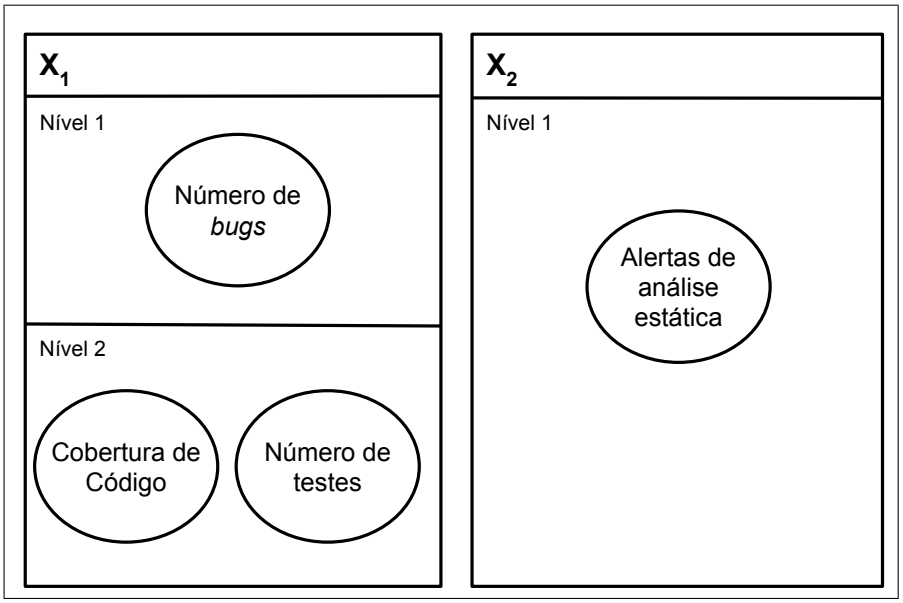

Figura 3.2: Exemplo de grupos de métricas de um projeto

É preciso, ainda, dividir as métricas de cada grupo em níveis. A distribuição de níveis é feita de forma hierárquica. As métricas de determinado nível dependem (isto é, sofrem influência) das métricas dos próximos níveis. As métricas do último nível não dependem de métrica alguma, ou seja, apenas possuem influência sobre as métricas dos níveis anteriores. No exemplo mencionado anteriormente, a métrica **Número de *bugs*** pertenceria ao nível 1 do seu respectivo grupo, enquanto as métricas **Número de testes** e **Cobertura de código** pertenceriam ao nível 2. Na Figura 3.2, expõe-se um possível cenário de grupos de métricas extraídas de um projeto. Cada grupo é dividido em níveis, mesmo que só exista um.

3.4 Construção da Rede Bayesiana

3.4.1 Construção do GAD

O terceiro passo da abordagem proposta consome um pouco mais de tempo que os demais. Nesta etapa são construídos os GAD associados a cada grupo de métricas do projeto. O pseudocódigo exposto no Código Fonte 3.1 descreve o passo a passo dessa fase da abordagem proposta em linhas gerais.

Código Fonte 3.1: Pseudocódigo para a construção do GAD

```
1  ConstroiGAD():
2      Q ← ∅
3      Para cada grupo de métricas X ∈ G:
4          Até que X esteja vazio, faça:
5              F ← primeiro elemento de X
6              ConstroiSubGADDaMetrica(F, X, Q)
```

Como pode ser observado no pseudocódigo, a primeira ação a ser realizada é a criação de um conjunto de relacionamentos (Q) da forma $p(x, y)$, que, semanticamente, significa que o grafo possui um relacionamento no qual o nó x é pai do nó y.

Em seguida, para cada grupo de métricas (X) utilizadas no projeto, algumas ações são realizadas. Até que o grupo X esteja vazio, suas métricas são passadas como parâmetro para a função **ConstroiSubGADDaMetrica**, declarada no Código Fonte 3.2. Como os grupos de métricas foram construídos de forma que estas sejam ordenadas de forma decrescente em termos de importância, **ConstroiSubGADDaMetrica** receberá as métricas mais importantes primeiro.

Código Fonte 3.2: Pseudocódigo para a construção do sub GAD de uma métrica

```
1 ConstroiSubGADDaMetrica(F, G, Q):
2     Remove a métrica F do grupo G
3     I_F ← interpretação de F
4     I_{F-M} ← medição-limiar de F
5     I_{F-E} ← erro associado à medição de F
6     AdicionaFontesPrimariasDeErro(F, I_F, I_{F-M}, I_{F-E}, Q)
7     AdicionaSubmetricasERiscos(F, I_{F-E}, G, Q)
8     AdicionaControladores(I_{F-E}, Q)
```

A função **ConstroiSubGADDaMetrica** inicia com a criação de três nós referentes à métrica F recebida como parâmetro:

- I_F, que representa a interpretação da métrica, a ser calculada pela rede Bayesiana posteriormente.
- I_{F-E}, que representa o fator de erro associado à métrica, ou seja, a incerteza inserida no valor coletado.
- I_{F-M}, que representa a relação métrica-limiar, ou seja, a opinião do gerente (ou outro responsável) sobre o valor observado quando a métrica é coletada. Os nós utilizados nesta pesquisa são do tipo ranqueado, como apresentado por Fenton et al. em [36]. Como as métricas, em sua maioria, são coletadas de forma quantitativa, faz-se necessária uma forma de converter o valor coletado em um valor qualitativo, que alimentará os nós ranqueados. Essa conversão é feita tomando como base um limiar, que auxilia o gerente a atribuir ao nó um dos três possíveis valores, ou estados, considerados na abordagem proposta: **Alto**, **Moderado** ou **Baixo**. Por exemplo, um gerente pode decidir que a métrica **Cobertura de código** possui o valor **Baixo** se for menor que 50%, **Moderado** se entre 50% e 80% (inclusive) e **Alto** se for maior que 80%.

Após criar esses três nós, **ConstroiSubGADDaMetrica** ainda chama três outras funções: **AdicionaFontesPrimariasDeErro** (Código Fonte 3.3), **AdicionaSubmetricasERiscos** (Código Fonte 3.4) e **AdicionaControladores** (Código Fonte 3.6).

A função **AdicionaFontesPrimariasDeErro** recebe como parâmetros a métrica F, a sua interpretação I_F, a medição-limiar I_{F_M}, o erro associado com a medição I_{F_E} e o conjunto de relacionamentos Q. Na linha 2 do Código Fonte 3.3, criam-se os relacionamentos $p(I_F, I_{F_M})$ e $p(I_F, I_{F_E})$.

A esta altura o grafo toma uma forma semelhante ao idioma de medição, descrito na Seção 2.2. No caso desta pesquisa, o valor observado é representado por I_{F-M}, a acurácia por I_{F-E} e o valor real pela interpretação da métrica (I_F).

Código Fonte 3.3: Pseudocódigo para a adição de fontes primárias de erro ao GAD

```
1  AdicionaFontesPrimariasDeErro(F, I_F, I_{F-M}, I_{F-E}, Q):
2      Q ← Q ∪ {p(I_F, I_{F-M}), p(I_{F-E}, I_{F-M})}
3      Se a qualidade da coleta de F pode causar erro:
4          QPCM ← qualidade do procedimento da coleta de F
5          Q ← Q ∪ {p(QPCM, I_{F-E})}
6      Se a qualidade do relatório de F pode causar erro:
7          QRM ← qualidade do procedimento de relatar F
8          Q ← Q ∪ {p(QRM, I_{F-E})}
9      Se o mau uso de F pode causar erro:
10         MUM ← qualidade do procedimento da coleta de F
11         Q ← Q ∪ {p(MUM, I_{F-E})}
```

As principais fontes de erro identificados na pesquisa foram: *Qualidade do procedimento da coleta da métrica*, *Qualidade do procedimento de relatar a métrica* e *Mau uso da métrica*. Por exemplo, no contexto de **Número de *bugs***, a *Qualidade do procedimento da coleta da métrica* representa a qualidade do processo de testes, a *Qualidade do procedimento de relatar a métrica* pode ser representada pela qualidade do processo da geração de relatórios de *bugs* e um exemplo de *Mau uso da métrica* é a escrita de métodos falsos (métodos com um grande número linhas que não são realmente utilizados na lógica do sistema) pelos desenvolvedores, que são utilizados para mascarar o verdadeiro valor da cobertura de código.

No Código Fonte 3.3, observa-se a função **AdicionaFontesPrimariasDeErro**, que representa a adição das fontes de erro (processo de coleta, processo de relatório e mau uso da métrica) como nós-pai do nó que representa o erro associado à medição. Naturalmente, a fonte de erro só deve ser adicionada como pai do erro se inserir incerteza na medição.

Após a inserção das fontes de erro, o modelo construído passa a ter a forma do *template* da abordagem, já mencionado na Seção 3.1, ilustrado na Figura 3.3.

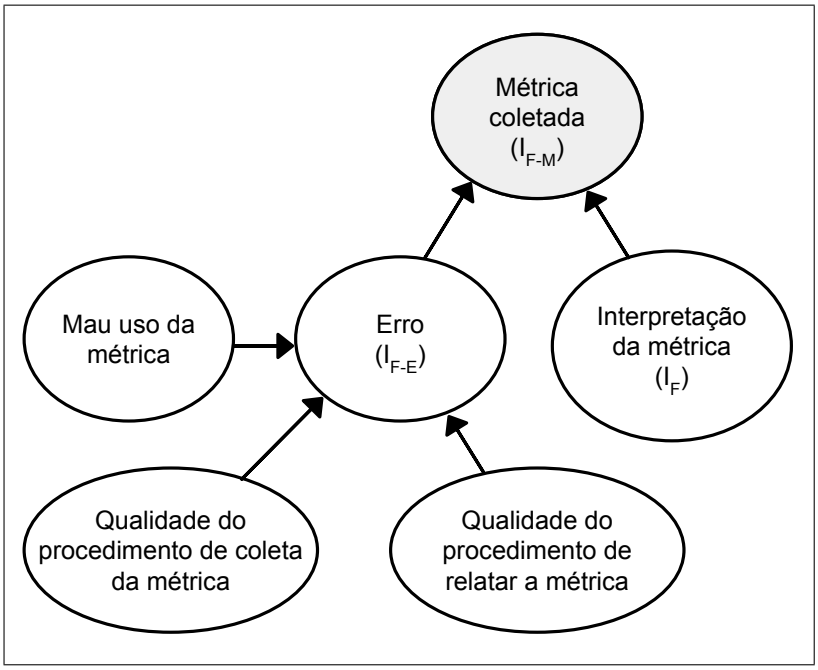

Figura 3.3: *Template* da abordagem

A função **AdicionaSubmetricasERiscos**, exibida no Código Fonte 3.4, recebe como parâmetro uma métrica (F), o nó que representa o erro associado à medição dessa métrica (I_{F-E}), o grupo que contém a métrica (G) e o conjunto de relacionamentos do grafo (Q). Cada nó P_E representando uma fonte de risco é visitado e, se houver uma ou mais métricas que exercem influência em F e condizem com a fonte de erro P_E, estas devem ser removidas de G e a função **ConstroiSubGADDaMetrica** deve ser chamada recebendo esta(s) métrica(s) como parâmetro. Finalmente, a função **AdicionaRiscos** é chamada, recebendo P_E como parâmetro.

Código Fonte 3.4: Pseudocódigo para a adição de submétricas e riscos ao GAD

```
1 AdicionaSubmetricasERiscos(F, I_{F-E}, G, Q):
2     Para cada pai (P_E) de I_{F-E}, tal que P_E ∈ {x|p(x, I_{F-E}) ∈ Q}:
3         Para cada métrica m_j ∈ G que influencia F e se relaciona com P_E:
4             ConstroiSubGADDaMetrica(m_j, G, Q)
5         AdicionaRiscos(Q, P_E)
```

A função **AdicionaRiscos** (Código Fonte 3.5) recebe uma fonte de erro P e o conjunto de relacionamentos Q como parâmetros e cria relacionamentos entre a fonte de erro P e riscos identificados no projeto que são associados a P. Riscos são mais específicos que fontes de erro e podem existir devido a ferramentas, times, atividades, ou até mesmo outra métrica. Cada fonte de erro, ou até mesmo macro riscos, devem ser decompostos em riscos que podem ser identificados pelo gerente. Portanto, a função **AdicionaRiscos** é chamada recursivamente recebendo cada um dos riscos como parâmetro (linha 4). Por exemplo, supondo que R_1 representa um macro risco *Qualidade do time*, este pode ser decomposto em riscos como *Comunicação entre os membros da equipe*, *Experiência dos membros da equipe* e *Perícia dos membros da equipe*.

Código Fonte 3.5: Pseudocódigo para a adição riscos ao GAD

```
1  AdicionaRiscos(Q, P):
2      Para todo risco R_i identificado e associado com P:
3          Q ← Q ∪ {p(R_i, P)}
4          AdicionaRiscos(Q, R_i)
```

Finalmente, o Código Fonte 3.6 contém a função **AdicionaControladores**, que recebe um nó representando o erro na acurácia de uma métrica I_{F-E} e o conjunto de relacionamentos Q. Para cada fator de risco R_i, ancestral do nó I_{F-E} e que não contém um nó pai, deve-se adicionar um ou mais fatores controladores que controlam, amenizam ou mitigam o risco. Fatores controladores, em geral, são atividades, ferramentas ou soluções adotadas com o intuito de diminuir o impacto do risco sobre a medição. Por exemplo, *Treinamento do time* e *Qualidade do processo de seleção dos membros do time* podem ser considerados fatores controladores do risco *Perícia do time*. Como mostrado nas linhas 3 e 4 do Código Fonte 3.6, deve ser adicionado em Q um relacionamento de parentesco entre cada fator controlador e o risco R_i.

Código Fonte 3.6: Pseudocódigo para a adição de controladores de riscos ao GAD

```
1 AdicionaControladores(I_{F-E}, Q):
2     Para cada ancestral órfão (R_k) de I_{F-E}:
3         Para cada controlador C_j que pode controlar ou mitigar R_k:
4             Q ← Q ∪ {p(C_j, R_k)}
```

3.4.2 Refatoramento do GAD

O GAD resultante do passo iii pode apresentar algumas inconsistências. Por exemplo, alguns nós podem precisar ser substituídos, outros nós relacionados podem ser combinados em um filho comum e alguns nós podem ser nomeados de uma melhor forma. Portanto, uma revisão deve ser feita e, se cabível, um refatoramento deve ser aplicado para que o grafo seja otimizado.

Se um nó possui apenas um pai e este não é um nó controlador, o nó filho deve ser descartado e o nó pai deve tomar seu lugar. Este tipo de refatoramento pode ser visualizado na Figura 3.4.

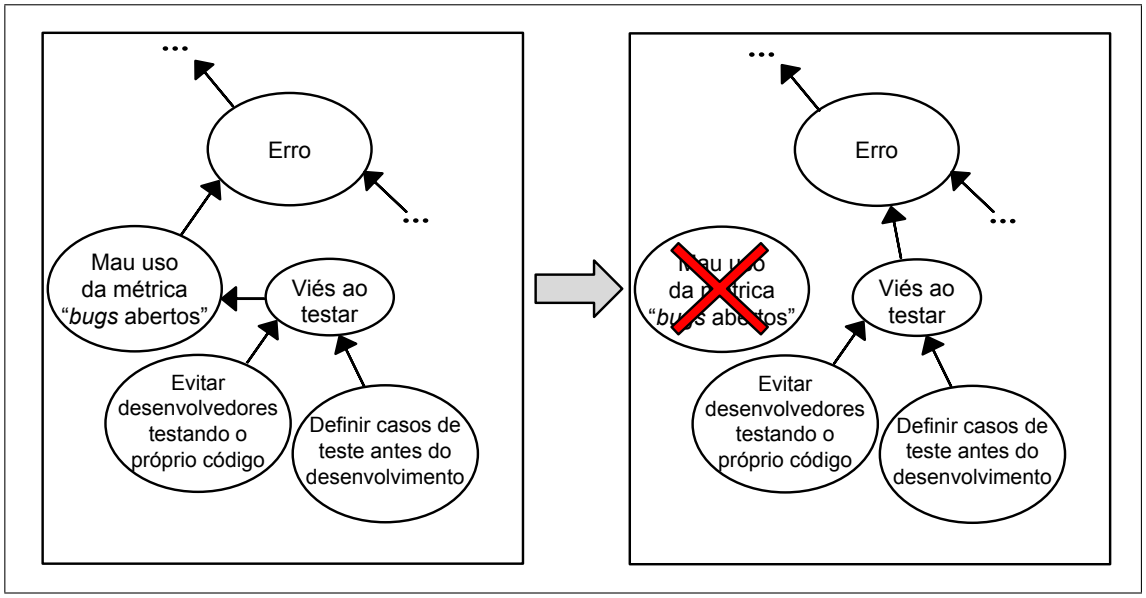

Figura 3.4: Exemplo de refatoramento: substituição de filho único

Além disso, o nome que representa cada nó pode ser modificado de modo a auxiliar a visualização e interpretação do grafo. Um exemplo desse tipo de refatoramento pode ser

observado no nó que representa a fonte de erro *Qualidade do procedimento da coleta da métrica* da métrica **Número de *bugs***. Os *bugs* são coletados, ou identificados, por meio da criação e execução de testes no projeto.

Por fim, por motivos de desempenho dos cálculos da rede Bayesiana e para tornar o GAD mais organizado e legível, deve-se tentar aplicar o idioma de síntese (mencionado na Seção 2.2) para agrupar alguns nós a fim de evitar nós com um grande número de pais (no máximo quatro). Essa sintetização de múltiplos nós em um só já foi apresentada por Fenton e Neil em [39] e é ilustrada na Figura 3.5.

Após o refatoramento do GAD, a terceira etapa é concluída. Um exemplo de GAD completo, gerado por meio da aplicação da abordagem proposta na pesquisa em um projeto de desenvolvimento de software, pode ser visto na Figura 3.6.

3.4.3 Definição das Funções de Probabilidade

A última etapa da abordagem proposta é a definição das funções de probabilidade para cada um dos nós dos grafos gerados por meio da execução das etapas anteriores. As funções de probabilidade, no contexto da pesquisa, servem para definir a influência que os nós do grafo exercem sobre outros nós. Essas funções de probabilidade são representadas por Tabelas de Probabilidade dos Nós (TPN). De acordo com Zhou et al. [43], uma TPN pode ser representada pela equação 3.5, onde X_i representa uma variável (isto é, um nó) e $pa(X_i)$ representa o conjunto de pais desse nó. A TPN, nesse caso, contém todos os valores da variável X_i dada cada combinação de valores diferentes dos seus pais $pa(X_i)$.

$$P(X_i|pa(X_i)) \tag{3.5}$$

Uma outra representação da TPN de um nó x é feita por uma matrix $n \times m$, onde n é o número de estados possíveis de x e m é o produto de estados dos seus pais. Nota-se que, conforme o tamanho da rede aumenta, o tamanho de m cresce exponencialmente. Portanto, torna-se muito trabalhoso definir as TPN manualmente e, por esse motivo, a ferramenta *AgenaRisk* [21] foi utilizada para auxiliar a construção das tabelas.

No *AgenaRisk*, é possível criar TPN a partir de dados qualitativos de maneira simples. Como os nós utilizados na pesquisa são ranqueados, essa funcionalidade se torna muito útil,

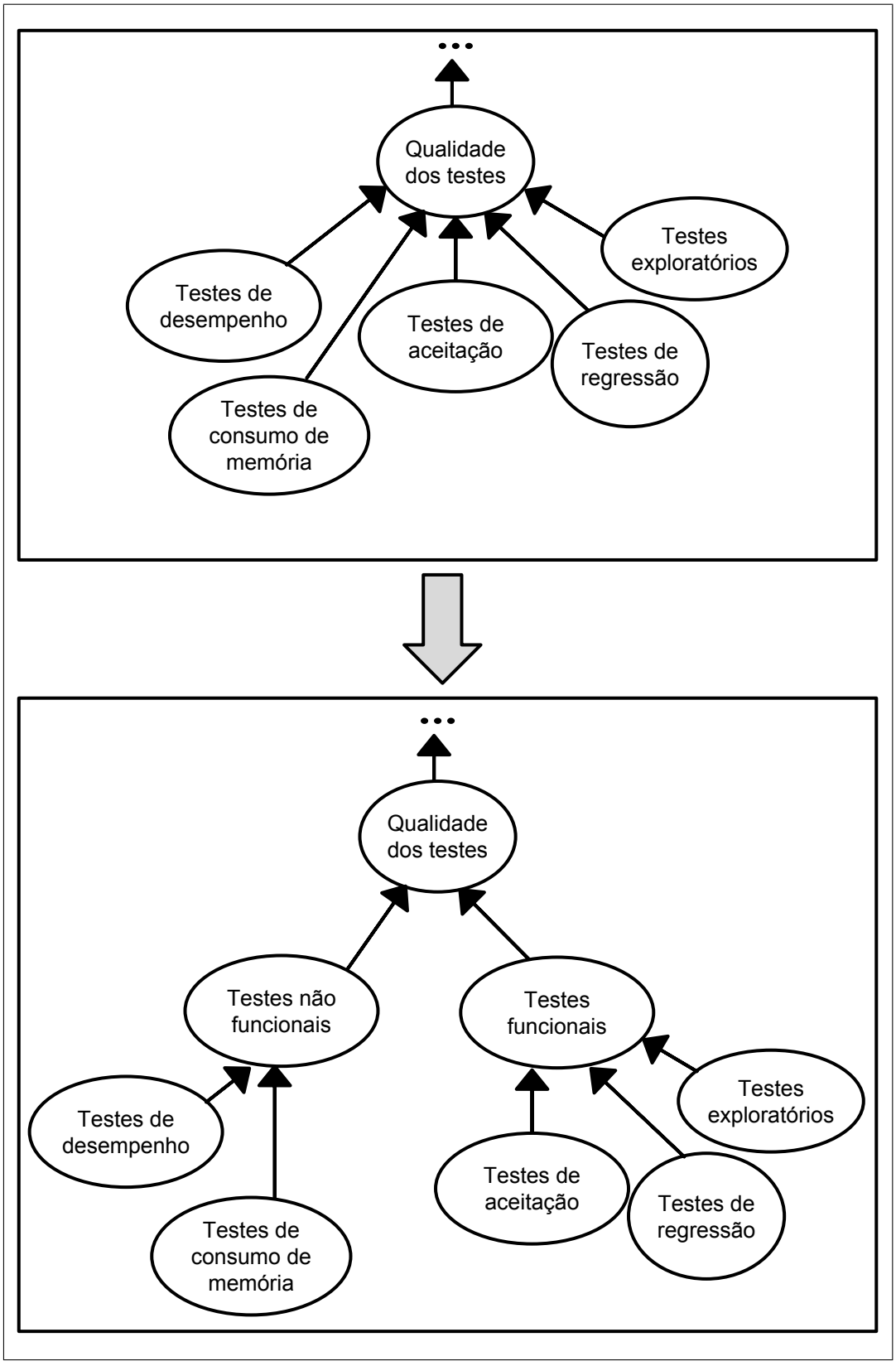

Figura 3.5: Exemplo de refatoramento: agrupamento de nós

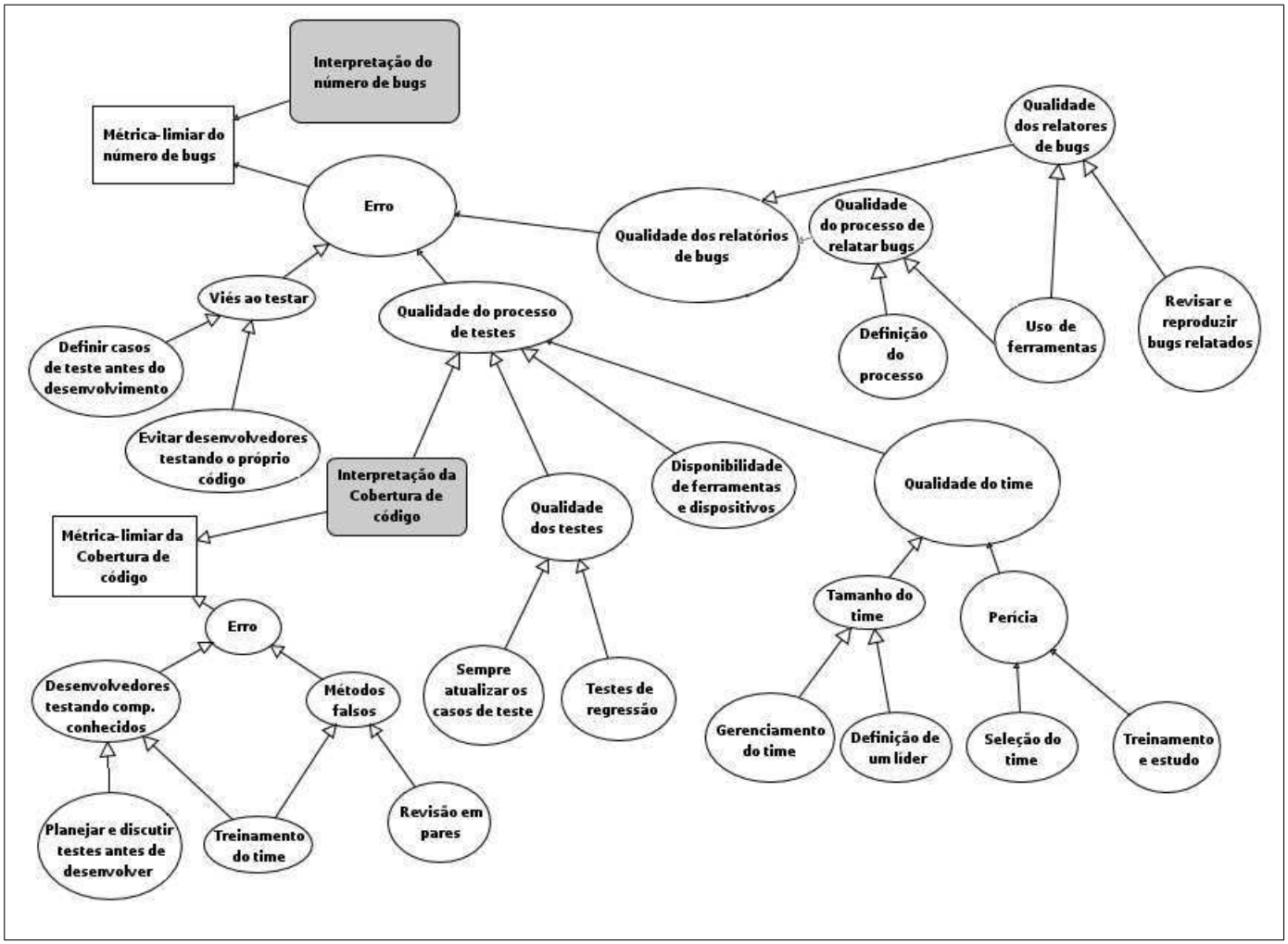

Figura 3.6: Exemplo de GAD criado a partir da aplicação da abordagem proposta

já que os estados dos nós seguem uma escala ordinal.

Para definir as funções de probabilidade em nós ordinais, o *AgenaRisk* utiliza uma distribuição normal duplamente truncada (neste caso, no intervalo $[0, 1]$), também conhecida como TNormal [36]. A distribuição TNormal é caracterizada por dois parâmetros: **média** (μ) e **variância** (σ^2).

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**: utilizada quando o gerente quer que o valor calculado para o nó tenda para **Baixo** se algum de seus pais for classificado como **Baixo**. Por exemplo, um gerente pode interpretar a qualidade dos testes como ruim se a qualidade de qualquer tipo de testes, como testes de unidade ou integração, for considerada baixa, mesmo que a qualidade de todos os outros testes seja classificada como boa.

- **Ponderada máxima**: utilizada quando o gerente quer que o valor calculado para o nó tenda para **Alto** se algum de seus pais for classificado como **Alto**. Por exemplo, um gerente pode interpretar a qualidade dos testadores como boa se houver uma boa seleção da equipe, mesmo que os membros da equipe não recebam um treinamento considerado satisfatório.

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

Para criar as funções ponderadas, utilizou-se uma modificação do algoritmo apresentado por Perkusich et al. em [29]. Primeiramente, os pais de um nó são ordenados de acordo com a sua magnitude relativa (influência exercida sobre o nó filho). Por exemplo, se um nó B exerce maior influência do que um nó A sobre um nó C, diz-se que o relacionamento $B - C$ tem uma magnitude relativa maior do que o relacionamento $A - C$. Para realizar a ordenação das magnitudes relativas de cada nó, foram utilizados questionários, nos quais gerentes dos projetos ordenaram os pais de cada nó de acordo com a sua influência no nó-filho. Todos os questionários formulados, assim como as suas respostas, foram organizados no Apêndice A. A definição dos tipos das funções ponderadas (média, máxima ou mínima) também é capturada por meio de respostas aos questionários.

Posteriormente, as funções ponderadas são, de fato, definidas. A definição dessas funções toma como base o procedimento publicado em [29] por Perkusich et al. Um gerente pode julgar que a influência do **Mau uso da métrica** (B) sobre o erro (A) é maior que a influência exercida pela **Qualidade do procedimento da coleta da métrica** (C). Neste caso, a expressão ponderada do erro seria representada por $A = 2 \times B + C$. Nota-se que, como são duas métricas (B e C), o peso da métrica com maior influência é 2. Também é possível que alguns fatores exerçam a mesma influência que outros. No exemplo anterior, se a **Qualidade do procedimento de relatar a métrica** (D) exercer a mesma influência que B sobre o A, a

expressão ponderada de A seria $A = 3 \times B + C + 3 \times D$. Nesse caso, se as influências de B, C e D sobre A fossem todas diferentes, os pesos seriam 3, 2 e 1, ao invés de 3, 3 e 1.

Além da média, a variância σ^2 da distribuição TNormal associada a cada nó também deve ser configurada. Nos projetos estudados no decorrer desta pesquisa, a variância utilizada em todos os nós foi $0,0005$, a menor possível no *AgenaRisk*. Uma baixa variância indica uma alta confiança no resultado dos cálculos das probabilidades. Sabe-se que, muitas vezes, a confiança nos dados não é tão alta. Portanto, a fim de assegurar que os resultados apresentados reflitam a realidade, a variância pode ser calibrada pelo gerente. Ao calibrar a variância, o modelo deve ser cuidadosamente testado, para evitar que sejam escolhidos valores indesejados.

Os nós que representam relações métrica-limiar merecem uma atenção especial. As tabelas de probabilidade de cada um desses nós (I_{F-M}), ao contrário dos demais, são definidas por uma expressão particionada. A expressão é particionada em três partes e cada uma delas é definida por uma distribuição TNormal cujo valor de μ é o valor da interpretação da métrica associada com o nó I_{F-M} (I_F) e σ^2 depende do valor do erro associado à medição (I_{F-E}).

- $1,0$, se o erro for **Alto**
- $0,3$, se o erro for **Moderado**
- $0,0005$, se o erro for **Baixo**

Nota-se que, conforme o erro aumenta, a variância escolhida também aumenta. Esse efeito é natural, pois conforme o erro associado à métrica aumenta, a confiança na medição diminui. 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 um gerente queira ajustar esses valores de acordo com as necessidades ou características do seu projeto.

A validação do método é feita no Capítulo 4 por meio de um estudo de caso com quatro projetos de desenvolvimento de software.

Capítulo 4

Estudo de Caso

A validação da abordagem proposta foi realizada por meio da execução de um estudo de caso realizado em quatro projetos de desenvolvimento de software. Todos os projetos foram executados no Laboratório de Sistemas Embarcados e Computação Pervasiva (*Embedded*) na Universidade Federal de Campina Grande (UFCG). Cada um dos projetos foi considerado uma unidade experimental.

Todos os projetos considerados no estudo de caso adotam *Scrum* [15] como processo de desenvolvimento de software, mas cada um apresenta suas próprias características e peculiaridades. Para cada projeto, foram coletados dados referentes a três *sprints* não necessariamente consecutivas. *Sprints* são iterações que, quando chegam ao fim, um incremento (novas funcionalidades, correção de *Bugs*, etc.) deve ser apresentado. A duração das *sprints* nos projetos estudados é de quinze dias, portanto a duração total do estudo de caso para cada um deles foi de quarenta e cinco dias. Essa duração foi escolhida pela disponibilidade de tempo e conveniência.

Com a realização do estudo de caso, objetivou-se responder as questões de pesquisa a seguir.

1. A abordagem apresentada é útil no sentido de guiar, ou auxiliar, o time na interpretação de métricas?

2. O custo-benefício da utilização da abordagem é satisfatório, considerando o esforço para aplicá-la?

4.1 Visão Geral do Estudo de Caso

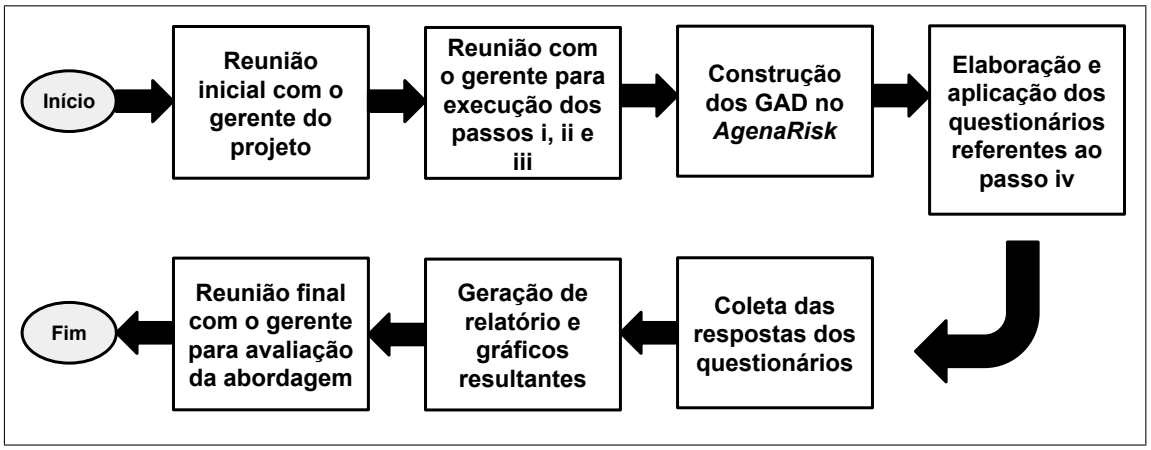

Figura 4.1: Atividades executadas em todos os projetos

Em cada um dos projetos acompanhados no decorrer da pesquisa, uma sequência de atividades, ilustradas na Figura 4.1, foi realizada. Nas Seções 4.2 a 4.5, detalham-se as especificidades de cada projeto, bem como a aplicação da abordagem proposta nestes. Além disso, na Seção 4.7, discutem-se as ameaças à validade no que diz respeito à abordagem apresentada. As atividades comuns a todos os projetos estudados na pesquisa foram as seguintes:

- reunião com o gerente do projeto com o objetivo de explicar a abordagem como um todo e esclarecer eventuais dúvidas;
- reunião com o gerente para a identificação e agrupamento de métricas do projeto, bem como a construção dos GAD correspondentes a cada grupo de métricas (etapas i, ii e iii da abordagem);
- construção dos GAD das redes Bayesianas utilizando o *AgenaRisk*;
- elaboração dos questionários, a serem respondidos pelos gerentes, utilizados para a definição das funções de probabilidade de cada nó (etapa iv da abordagem);
- coleta das respostas dos questionários e definição das funções de probabilidade de cada nó no *AgenaRisk*;

- geração do relatório e gráficos resultantes dos cálculos realizados pelas redes Bayesianas;

- reunião com o gerente para apresentar os relatórios e gráficos produzidos e para coletar as respostas das questões de pesquisa.

Como já mencionado, nas seções a seguir se relata a aplicação da abordagem proposta em quatro projetos de desenvolvimento de software distintos. Todos eles são projetos de Pesquisa e Desenvolvimento (P&D) financiados por empresas privadas. Sendo assim, os projetos podem estar sujeitos a um acordo de confidencialidade e, portanto, seus nomes foram omitidos e eles são referenciados pelas alcunhas *Projeto A*, *Projeto B*, *Projeto C* e *Projeto D*.

4.2 Projeto A

O foco do Projeto A era o desenvolvimento de um protótipo com a finalidade de demonstrar ao cliente que uma determinada técnica poderia ser agregada à reprodução de filmes para melhorar a experiência do espectador. No total, a equipe contou com 7 desenvolvedores, 2 testadores e 1 gerente de projeto. Todos formados e trabalhando em regime integral.

A reunião com o gerente do Projeto A, na qual foram construídos os GAD relativos ao projeto, durou aproximadamente 1 hora e 20 minutos. Nessa reunião, o gerente identificou quatro distintos grupos de métricas. Os questionários elaborados para determinar os pesos dos nós das redes Bayesianas do Projeto A são apresentados no Apêndice A.1.

O primeiro grupo de métricas do Projeto A é composto por métricas relacionadas a testes, anomalias e/ou *Bugs*. As métricas coletadas no projeto que pertencem a esse grupo são: **Número de *Bugs***, **Cobertura de Código** e **Número de Testes**. A primeira, segundo o gerente, possui importância maior que as demais, sendo classificada em um nível hierárquico acima das outras. A justificativa do gerente é que um bom número de testes e uma alta cobertura de código podem contribuir diretamente para o número de defeitos identificados. O GAD desse grupo de métricas é ilustrado na Figura 4.2.

Observando o GAD do Grupo 1, nota-se que as fontes de erro do **Número de *bugs*** são a qualidade do processo de testes e a qualidade dos relatórios de *Bugs*. A **Cobertura de**

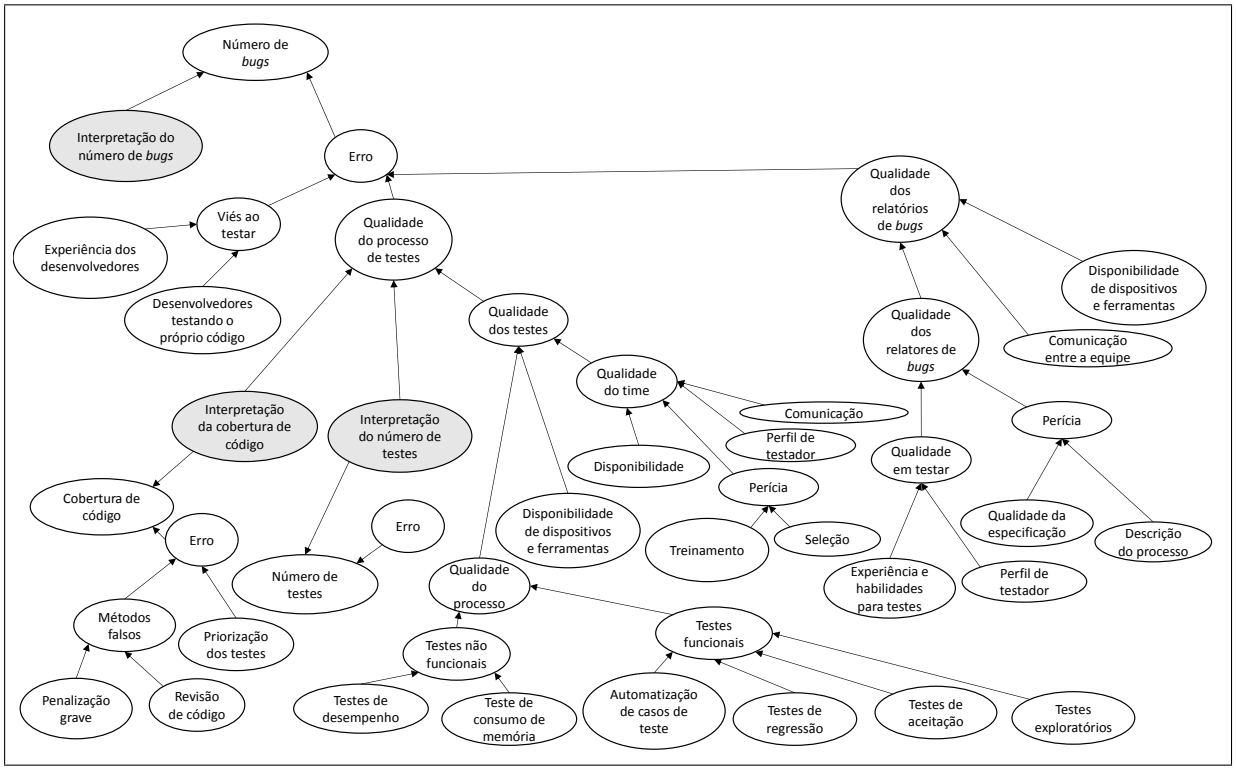

Figura 4.2: GAD construído para o primeiro grupo de métricas do Projeto A

Código e o **Número de Testes** se encaixam na primeira categoria, portanto os nós correspondentes a essas métricas foram inseridos como pais do nó **Qualidade do Processo de Testes**. Ainda no grafo, é possível notar os fatores de risco e os seus respectivos controladores, ou mitigadores. Por exemplo, uma das fontes de erro da métrica **Cobertura de Código** é o fato de os desenvolvedores implementarem métodos falsos com o intuito de mascarar o real valor da métrica. Para reduzir esse risco, todo o código submetido no projeto deve passar pela revisão de um desenvolvedor que não participou do desenvolvimento do código. Além disso, o gerente aplicaria uma penalização bastante grave no desenvolvedor que eventualmente desenvolvesse um método falso.

A Figura 4.3 exibe um gráfico com a interpretação da métrica **Cobertura de Código** na primeira *Sprint* considerada no Projeto A. Gráficos desse tipo foram gerados para cada uma das métricas coletadas (em cada *Sprint*) nos projetos estudados e todos eles podem ser visualizados no Apêndice B.

Os gráficos gerados possuem uma barra azul, que sempre terá o valor de 100% e representa o valor da métrica (*Baixo*, *Médio* ou *Alto*) observado pelo gerente para determinada

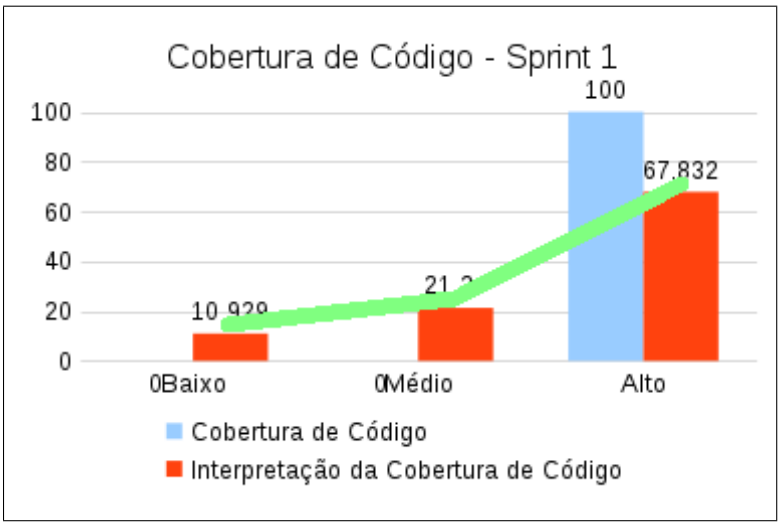

Figura 4.3: Gráfico correspondente à interpretação da métrica Cobertura de Código na primeira *Sprint* do Projeto A

Sprint. A interpretação do valor real da métrica, calculada pela rede Bayesiana construída por meio da aplicação da abordagem proposta, é representada pelas barras laranjas do gráfico. A confiança da métrica, de acordo com a abordagem, é medida por meio da observação do valor da barra laranja correspondente à resposta dada pelo gerente. Por exemplo, caso o gerente tenha observado o valor *Alto* para uma métrica, o valor da barra laranja para o valor *Alto* deve ser observado. Nesta pesquisa, definiu-se que, caso esse valor seja maior que 65% a confiança da métrica é alta; se o valor observado na barra laranja estiver entre 40% e 65% (inclusive) a confiança da métrica é média; e se esse valor for menor que 40% a confiança é considerada baixa.

Esses limiares foram determinados de forma subjetiva, num processo dividido em duas partes. A princípio, foram criadas algumas redes Bayesianas simulando projetos fictícios, sempre seguindo a abordagem proposta. Em seguida, foram elaborados alguns cenários representativos (caótico, perfeito, balanceado, etc.) para se alimentar as redes construídas. Após a execução dos cenários nos modelos construídos, os valores medidos para os nós que representam métricas foram analisados e já se pôde ter uma aproximação dos valores que seriam utilizados. A segunda parte da definição dos limiares foi a aplicação dos mesmos cenários utilizados na etapa anterior em algumas redes construídas para os projetos acompanhados na pesquisa. Os nós representando métricas foram novamente analisados e, por fim,

foram selecionados os valores de 40% e 65% para diferenciar uma confiança ruim de uma confiança média e uma confiança média de uma confiança boa, respectivamente.

Nos gráficos apresentados nesta seção, traçou-se ainda uma linha verde tangenciando as barras laranjas para auxiliar a visualização da confiança da métrica. Quanto maior a confiança, observa-se que essa linha converge mais acentuadamente para o valor observado pelo gerente.

Isso posto, pelo que pode ser observado na Figura 4.3, o gerente considerou o valor observado da métrica **Cobertura de Código** na primeira *Sprint* considerada no Projeto A como *Alto*. A linha verde auxilia na observação de que a interpretação da métrica visivelmente tende para o valor *Alto*, fazendo com que a confiança da métrica **Cobertura de Código** seja alta. Portanto, de acordo com a abordagem, o gerente pode considerar o valor medido para essa métrica para tomar decisões relacionadas ao projeto, como aumentar a cobertura ou manter (ou modificar) o tamanho do time de testes.

Um exemplo oposto ao anterior é ilustrado na Figura 4.4. Na primeira *Sprint*, o gerente considerou a métrica **Número de *Bugs*** como *Médio*. A linha verde, que acompanha a interpretação calculada pela rede Bayesiana construída, está, aproximadamente, perpendicular às barras do gráfico. Como a linha não converge visivelmente para *Médio*, isto quer dizer que a confiança da medição não é alta. Portanto, não é ideal que o gerente considere essa métrica para tomar decisões no projeto. Na discussão com o gerente, este mencionou que já era sabido que essa métrica não refletia bem na realidade, principalmente pela baixa qualidade dos relatórios de *bugs* produzidos.

O segundo grupo de métricas do Projeto A tem foco na documentação do projeto. O GAD gerado para esse grupo é ilustrado na Figura 4.5. Segundo o gerente, apenas uma métrica se enquadra no grupo 2: **Número de Alertas de *Checkstyle***. Os alertas considerados nessa métrica são coletados pela ferramenta de análise estática *Checkstyle* [1] referentes à documentação de código. A ferramenta não analisa a semântica da documentação. Portanto, a documentação de alguns trechos de código pode estar presente, porém com uma semântica errada. No entanto, o gerente identificou a Revisão de Código como um fator que pode mitigar esse risco.

A Figura 4.6 exibe um gráfico com a interpretação da métrica **Número de Alertas de**

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

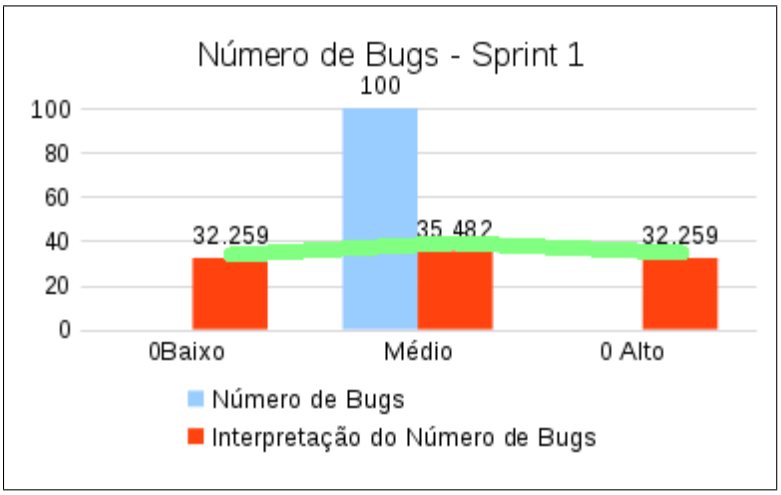

Figura 4.4: Gráfico correspondente à interpretação da métrica Número de *Bugs* na primeira *Sprint* do Projeto A

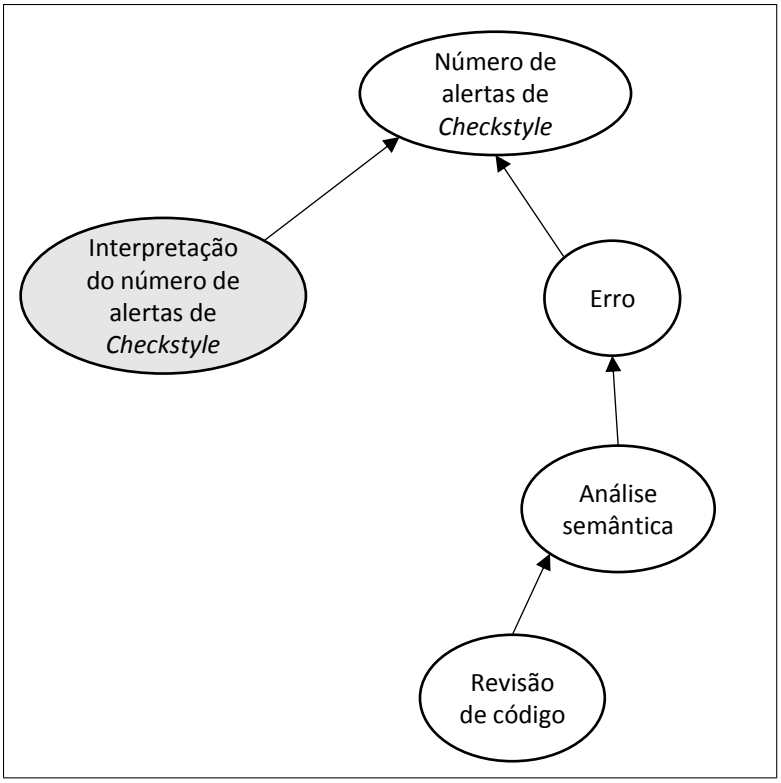

Figura 4.5: GAD construído para o segundo grupo de métricas do Projeto A

Checkstyle na primeira *Sprint* considerada no Projeto A. Como pode ser visto na figura, a interpretação calculada pela aplicação da abordagem converge para o valor observado pelo gerente. Conclui-se, então, que o valor medido para essa métrica pode ser considerado pelo gerente para que este tome decisões relacionadas ao projeto, no que diz repeito à documentação.

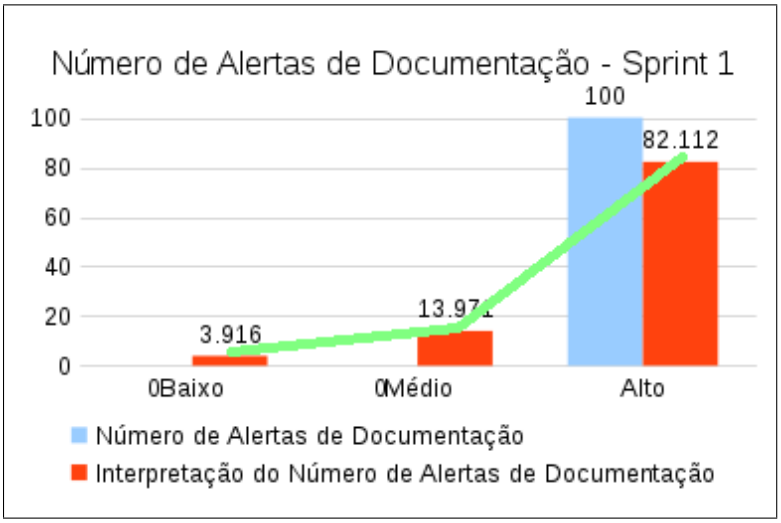

Figura 4.6: Gráfico correspondente à interpretação da métrica Número de Alertas de *Checkstyle* na primeira *Sprint* do Projeto A

O terceiro grupo de métricas do Projeto A diz respeito à análise estática. O GAD correspondente a esse grupo de métricas pode ser visto na Figura 4.7. O gerente do projeto classificou apenas uma métrica nesse grupo: **Número de Alertas de Análise Estática**. Essa métrica leva em consideração alertas coletados pela ferramenta de análise estática *Checkstyle* (não referentes à documentação), *Findbugs* [2], *Lint* [3] e *PMD* [4] (apenas alertas de código duplicado). As ferramentas podem apontar falsos positivos, ou seja, alertas que, na verdade não refletem em má qualidade do produto desenvolvido. Para tentar reduzir esse risco, o gerente identificou que a ferramenta adotada pode contribuir significativamente para a redução. Além disso, uma boa configuração da ferramenta pode diminuir a ocorrência de falsos positivos.

A Figura 4.8 exibe um gráfico com a interpretação da métrica **Número de Alertas de**

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

[3]http://developer.android.com/tools/help/lint.html

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

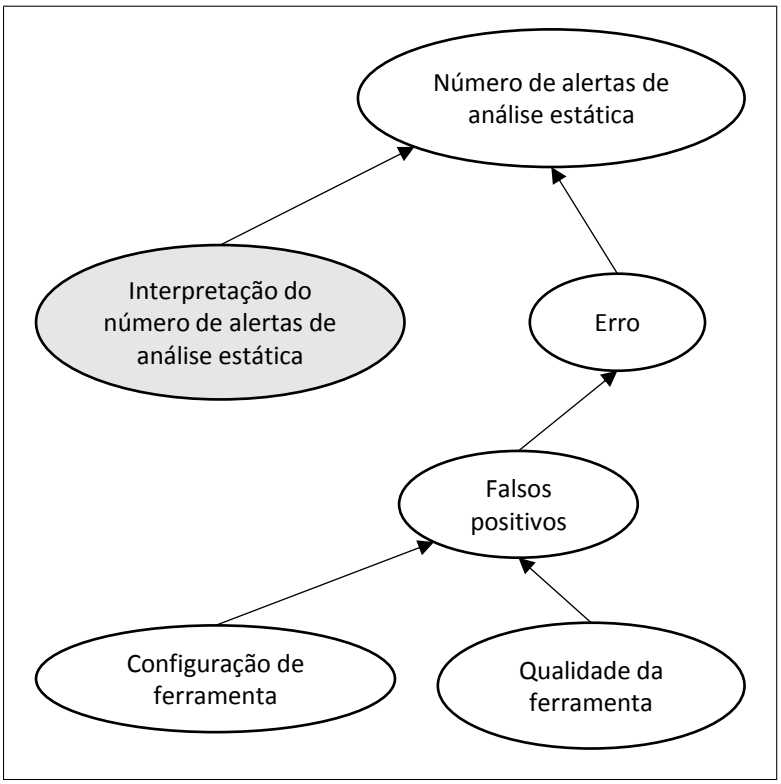

Figura 4.7: GAD construído para o terceiro grupo de métricas do Projeto A

Análise Estática na segunda *Sprint* considerada no Projeto A. Como observado na figura, a interpretação calculada utilizando a abordagem proposta na pesquisa converge muito fracamente para o valor observado pelo gerente. Isso ocorre principalmente pela baixa qualidade da ferramenta adotada no projeto. O gerente informou que outras opções de ferramenta já estavam sendo estudadas para substituir a ferramenta utilizada em projetos futuros. A boa configuração da ferramenta, incluindo o silenciamento de alguns tipos de alertas, auxilia em fazer com que a interpretação da métrica tenha uma leve inclinação na direção do valor observado pelo gerente.

O quarto grupo de métricas do Projeto A envolve o gerenciamento do projeto. O GAD correspondente a esse grupo de métricas é ilustrado na Figura 4.9. Novamente, apenas uma métrica foi classificada nesse grupo pelo gerente: ***Burndown* da *Sprint*** [1]. Essa métrica exibe o número de tarefas completadas e o número de tarefas planejadas para a *Sprint*, então o gerente pode ter uma noção do estado de completude da *Sprint* e estimar se esta vai ser concluída no tempo previsto. Dependendo da observação do gráfico, o gerente pode tomar decisões relacionadas com o processo de desenvolvimento do projeto, como alocar novos

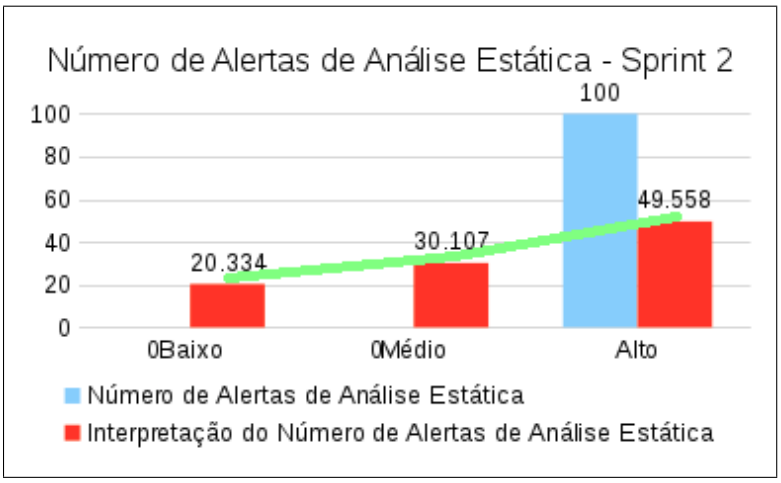

Figura 4.8: Gráfico correspondente à interpretação da métrica Número de Alertas de Análise Estática na segunda *Sprint* do Projeto A

desenvolvedores para determinadas tarefas, priorizar tarefas críticas ou bloqueadas, dentre outros.

A Figura 4.10 exibe um gráfico com a interpretação da métrica ***Burndown* da *Sprint*** na primeira *Sprint* considerada no Projeto A. É possível notar que a interpretação calculada converge para o valor observado pelo gerente. Isso ocorre principalmente pelo bom gerenciamento de tarefas por parte do gerente, perícia do time, descrição do processo de desenvolvimento e configuração da ferramenta utilizada para controlar o processo de desenvolvimento. Esses fatores auxiliam na minimização de alguns riscos, como a baixa qualidade do processo e da ferramenta utilizada. A Figura 4.11 ilustra a interpretação da métrica com uma observação diferente do gerente (*Alta*, ao invés de *Média*). A interpretação da métrica ainda assim converge para o valor observado. Portanto, recomenda-se considerar a métrica ***Burndown* da *Sprint*** no processo de tomada de decisões acerca do projeto.

Ao fim da aplicação da abordagem em cada um dos projetos, os gerentes receberam um pequeno questionário contendo as duas questões de pesquisa do estudo de caso. As respostas de todos os gerentes foram utilizadas na análise dos resultados. Relembrando o que foi dito no começo do capítulo, as questões de pesquisa são as seguintes:

1. A abordagem apresentada é útil no sentido de guiar, ou auxiliar, o time na interpretação de métricas?

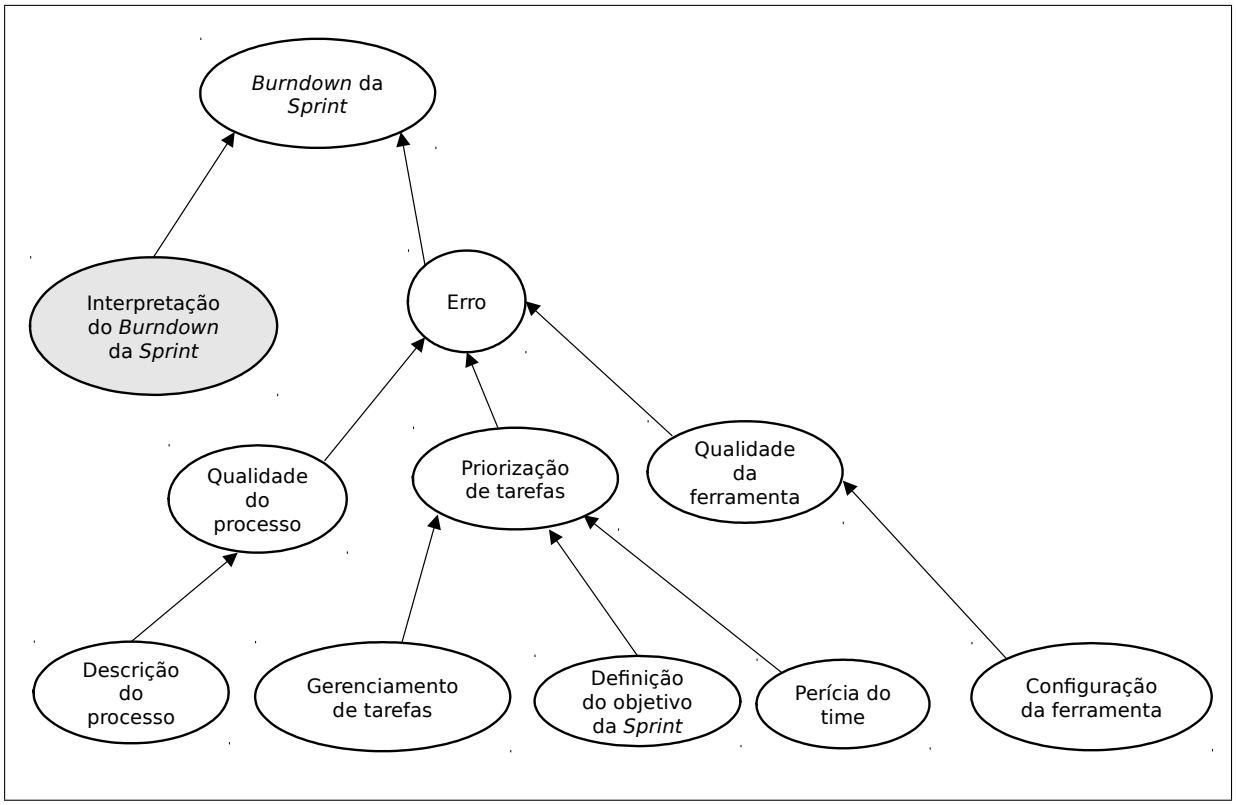

Figura 4.9: GAD construído para o quarto grupo de métricas do Projeto A

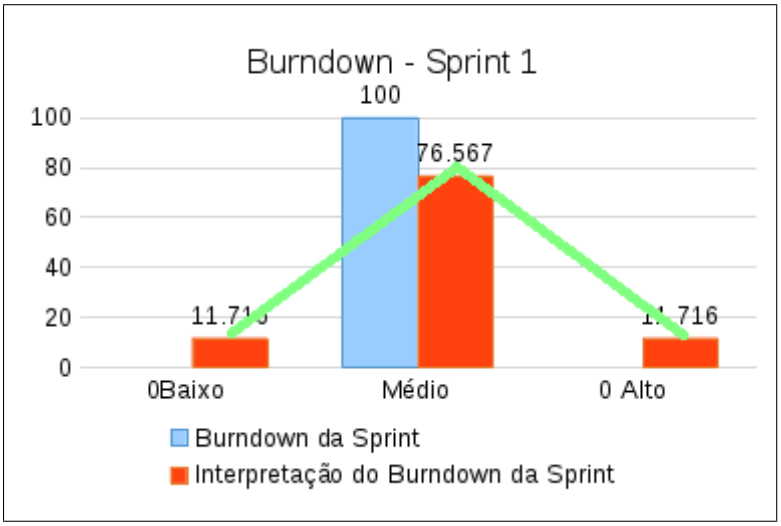

Figura 4.10: Gráfico correspondente à interpretação da métrica *Burndown* da *Sprint* na primeira *Sprint* do Projeto A

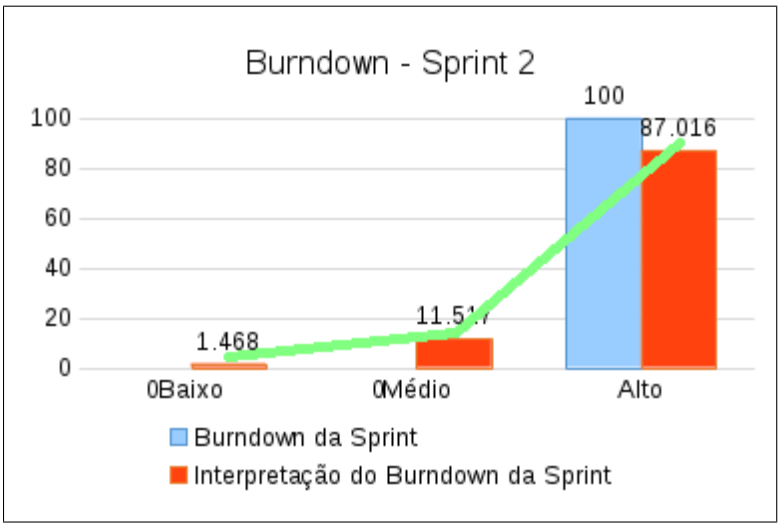

Figura 4.11: Gráfico correspondente à interpretação da métrica *Burndown* da *Sprint* na segunda *Sprint* do Projeto A

2. O custo-benefício da utilização da abordagem é satisfatório, considerando o esforço para aplicá-la?

Para a primeira questão de pesquisa, a resposta do gerente do Projeto A foi: *"Sim, pois o método proposto explicitou algo que já desconfiávamos. Mais especificamente, por algum motivo nós sabíamos que não podíamos confiar na métrica* ***Número de Bugs****. Portanto, nós sempre fomos muito cautelosos e alertávamos ao cliente que um baixo número de defeitos relatados fosse baixo [sic] não necessariamente indicaria que o produto não poderia apresentar comportamentos indesejados"*.

O gerente respondeu a segunda questão de pesquisa da seguinte forma: *"Sim, pois agora nós compreendemos de forma mais profunda os problemas em nosso processo. A informação que os modelos forneceram serão utilizadas para definir um plano de ações corretivas"*.

Os gráficos correspondentes às probabilidades calculadas em todos os grupos de métricas do Projeto A, em todas as *Sprints*, encontram-se no Apêndice B.1.

4.3 Projeto B

O Projeto B tinha como foco o desenvolvimento de uma aplicação para dispositivos vestíveis, cada vez mais populares nos mercados comerciais[17], como relógios e pulseiras inte-

ligentes. A aplicação desenvolvida foi publicada na Loja *online Google Play*[5]. Esse projeto contou com 3 desenvolvedores, 2 testadores e 1 gerente. Todos formados e trabalhando em regime integral. Uma característica bastante peculiar do Projeto B é o fato de o gerente estar fisicamente distante dos demais membros do projeto, comunicando-se com eles por meio de videoconferências.

A reunião com o foco de construir os GAD do Projeto B teve uma duração aproximada de 1 hora. O gerente identificou três grupos de métricas no escopo do projeto. Os questionários elaborados para o projeto, referentes à última etapa da abordagem apresentada na pesquisa, encontram-se no Apêndice A.2.

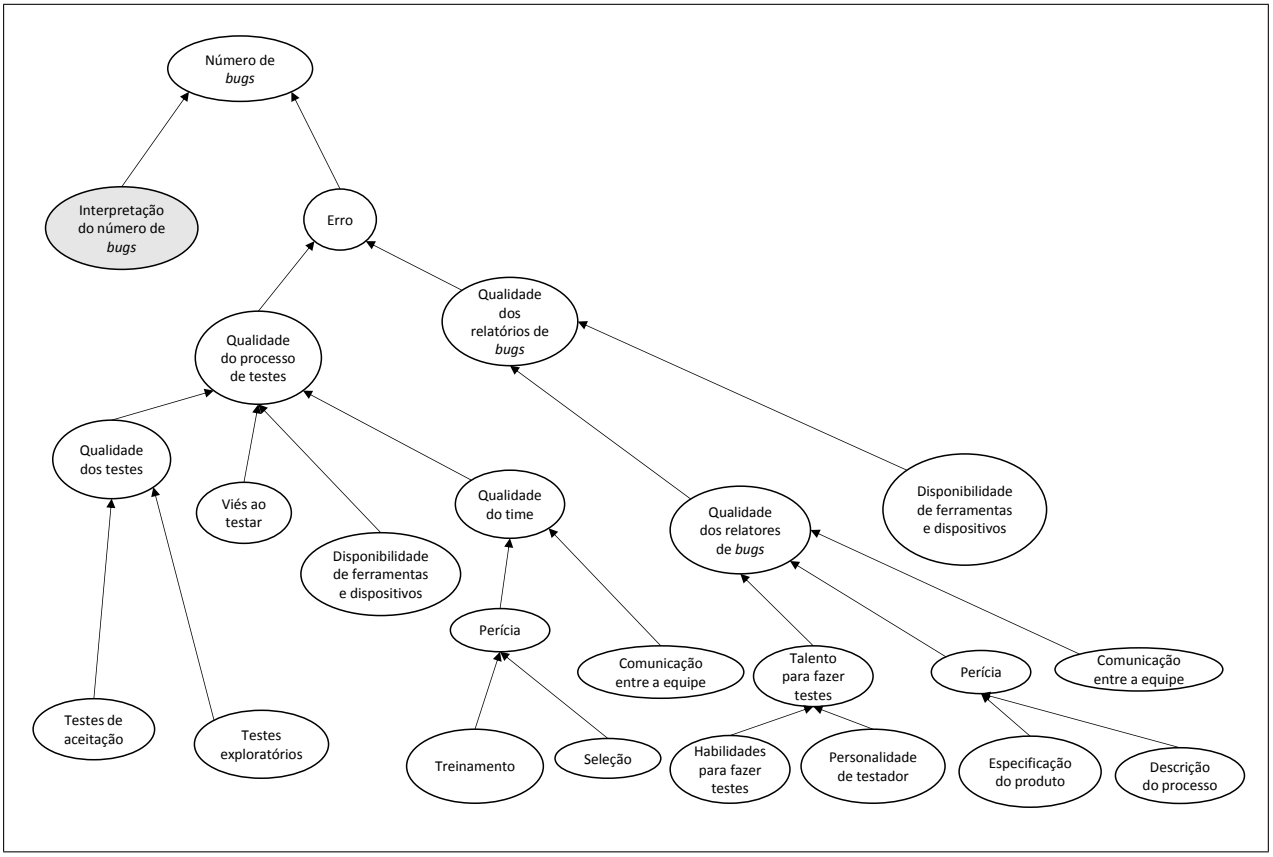

Figura 4.12: GAD construído para o primeiro grupo de métricas do Projeto B

O primeiro grupo de métricas do Projeto B engloba métricas coletadas no projeto no escopo de testes, anomalias e/ou *Bugs*. A única métrica pertencente a esse grupo é **Número de *Bugs***. O GAD desse grupo de métricas é ilustrado na Figura 4.12. Observando o GAD

[5]http://play.google.com

construído, pode-se notar que as principais fontes de erro considerados pelo gerente que podem prejudicar a acurácia da medição do **Número de *Bugs*** são a qualidade do processo de testes e a qualidade dos relatórios de *bugs*. Algumas abordagens adotadas pelo gerente para mitigar esses riscos são a seleção e treinamento da equipe de testes, se certificar que o produto esteja bem especificado e garantir uma boa descrição do processo de relatar *bugs*.

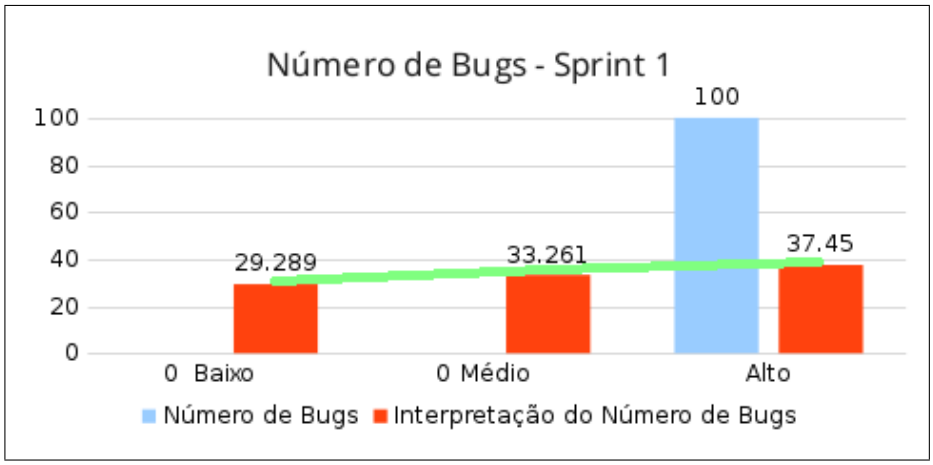

Figura 4.13: Gráfico correspondente à interpretação da métrica Número de *Bugs* na primeira *Sprint* do Projeto B

A Figura 4.13 exibe um gráfico com a interpretação da métrica **Número de *Bugs*** na primeira *Sprint* do Projeto B. É possível perceber que a interpretação da métrica não possui uma tendência a convergir para o valor observado pelo gerente. Essa baixa acurácia pode ser justificada por inúmeros fatores preocupantes observados no projeto. Por exemplo, não há uma descrição de um processo que padronize a criação de relatórios de *bugs*, o que diminui a qualidade dos relatórios produzidos. Além disso, os desenvolvedores escrevem testes de integração para o seu próprio código, o que pode causar um viés no desenvolvimento dos testes, visto que eles podem testar apenas fluxos conhecidos. A falta de uma ferramenta de relatório de *bugs* também prejudica a qualidade dos relatórios, pois a organização destes decai e o número de *bugs* relatados pode não condizer com a realidade. Finalmente, a falta de dispositivos pode prejudicar o time, pois *bugs* que ocorrem especificamente em determinados dispositivos não são reproduzidos.

O segundo grupo de métricas do Projeto B, referente à análise estática do código produzido, possui apenas uma métrica: Número de Alertas de Análise Estática. O GAD referente

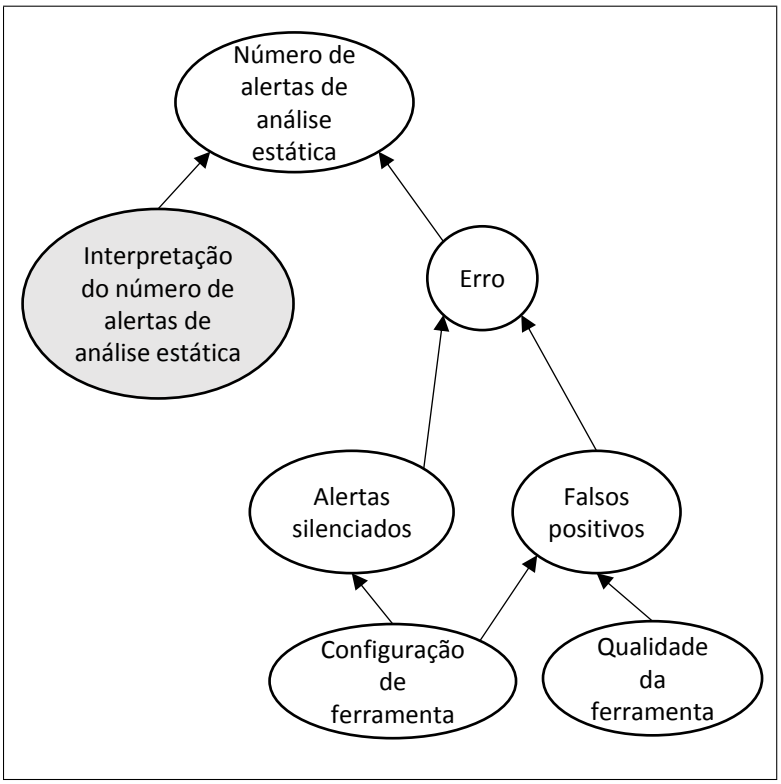

Figura 4.14: GAD construído para o segundo grupo de métricas do Projeto B

a esse grupo de métricas pode ser visualizado na Figura 4.14. Assim como o Projeto A, os alertas em questão são aqueles coletados pelas ferramenta de análise estática *Checkstyle* (não referentes à documentação), *Findbugs*, *Lint* e *PMD* (dessa vez, todos os alertas e não só os de código duplicado). As principais fontes de erro identificadas pelo gerente são os falsos positivos acusados pela ferramenta e os alertas erroneamente silenciados. Para mitigar esse risco, a ferramenta deve ser configurada da melhor forma possível, para evitar a ocorrência de erros. A escolha de uma ferramenta de boa qualidade também pode ajudar a mitigar a ocorrência de falsos positivos.

A Figura 4.15 exibe um gráfico com a interpretação da métrica **Número de Alertas de Análise Estática** na terceira *Sprint* do Projeto B. Observando o gráfico, nota-se que a interpretação da métrica converge moderadamente para o valor observado pelo gerente. Isso se justifica pelo fato de que, embora a qualidade da ferramenta adotada não seja tão boa, os desenvolvedores desempenham bem a função de configurá-la da melhor forma possível, de diminuir os riscos associados à medição. É indicado que o gerente utilize essa métrica para analisar a qualidade do código produzido, porém sempre atentando para os riscos que podem

introduzir erro na medição, visto que a sua acurácia não é tão alta.

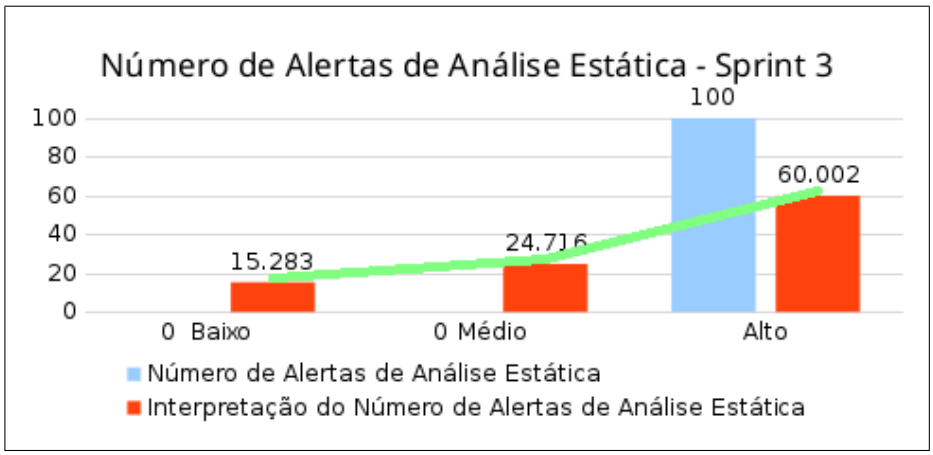

Figura 4.15: Gráfico correspondente à interpretação da métrica Número de Alertas de Análise Estática na terceira *Sprint* do Projeto B

O último grupo de métricas do Projeto B tem o escopo de gerenciamento do processo de desenvolvimento do produto. O grafo relativo a esse grupo pode ser visto na Figura 4.16. A métrica ***Burndown* da *Sprint***, já explicada no Projeto A, é a única métrica pertencente a esse grupo. As fontes de erro da medição identificadas pelo gerente do Projeto B foram uma eventual baixa qualidade no processo de desenvolvimento e a má priorização de tarefas. Para controlar esses riscos, o gerente listou alguns fatores, como uma boa descrição do processo de desenvolvimento, definição do objetivo da *Sprint* e treinamento específico em *Scrum* (processo de desenvolvimento de software utilizado no projeto) para time de desenvolvimento.

A Figura 4.17 exibe um gráfico com a interpretação da métrica ***Burndown* da *Sprint*** na terceira *Sprint* do Projeto B. É possível perceber, pela observação da Figura, que a interpretação da métrica converge moderadamente para o valor observado pelo gerente. A convergência não é tão acentuada porque, apesar da boa experiência do time e da boa definição do objetivo da *Sprint*, não há um treinamento em *Scrum* muito bom para a equipe e a descrição do processo de desenvolvimento não é tão satisfatório. Assim, o gerente pode utilizar essa métrica para analisar o andamento da *Sprint*, mas é fortemente recomendado que os fatores que podem controlar os riscos sejam revistos de forma a aumentar a acurácia da medição.

O gerente do Projeto B foi o mais breve ao responder as questões de pesquisa. A primeira

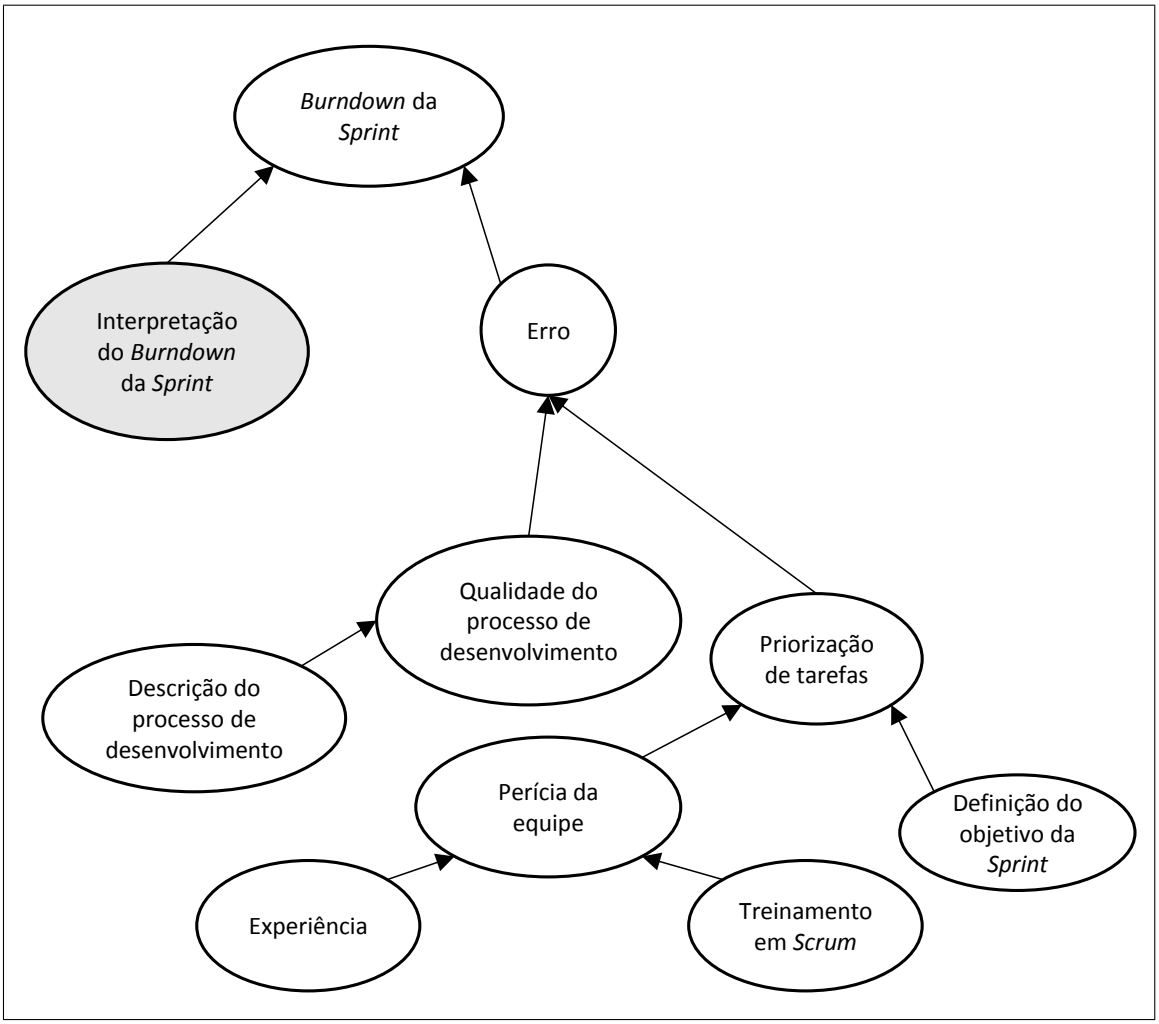

Figura 4.16: GAD construído para o terceiro grupo de métricas do Projeto B

delas foi respondida da seguinte maneira: *"Sim. Existem vários métodos de coleta de métricas, mas eles não ajudam na sua interpretação, como faz o método proposto. Por outro lado, eu acho que a etapa de definir as funções de probabilidade foi abstrata, o que pode levar a erros"*. A segunda resposta foi ainda mais curta: *"Sim, pois eu acho que os benefícios das informações providas pelas redes Bayesianas pesam mais que o tempo investido"*.

Os gráficos correspondentes às probabilidades calculadas em todos os grupos de métricas do Projeto B, em todas as *Sprints*, encontram-se no Apêndice B.2.

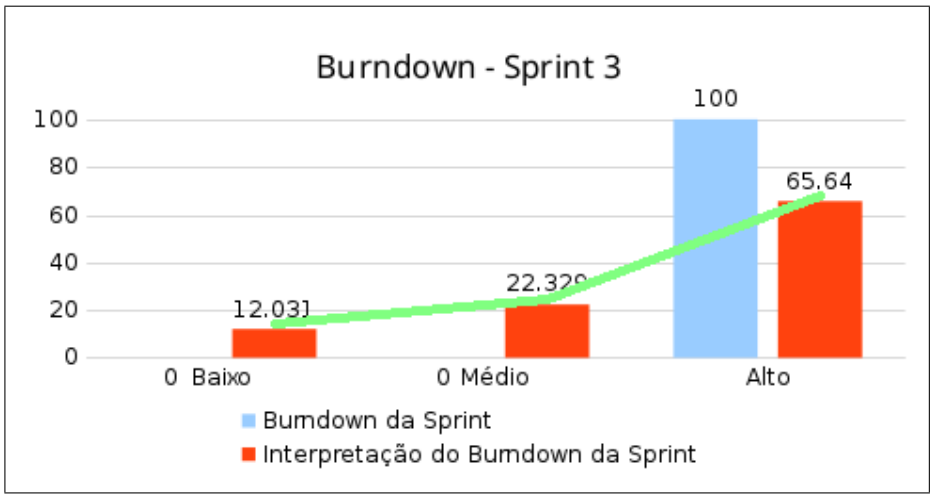

Figura 4.17: Gráfico correspondente à interpretação da métrica *Burndown* da *Sprint* na terceira *Sprint* do Projeto B

4.4 Projeto C

Provavelmente o mais complexo dos projetos considerados na pesquisa, o Projeto C tinha o objetivo de desenvolver um sistema de computação voluntária distribuída para que usuários fornecessem poder de processamento dos seus celulares, durante as horas que estes estivessem ociosos, para auxiliar na simulação de fenômenos moleculares [2]. A aplicação desenvolvida também foi disponibilizada na *Google Play*. A equipe desse projeto era composta por 6 desenvolvedores, 3 testadores e 2 gerentes. Todos formados e trabalhando em regime integral.

O gerente de projeto que participou das reuniões e respondeu aos questionários referentes à pesquisa foi aquele que ficou à frente do Projeto C por um maior período de tempo, caracterizando-o como o gerente de maior domínio e conhecimento acerca do projeto.

A reunião com o gerente do Projeto C com foco em construir os GAD do projeto foi mais longa que as demais, durando por cerca de 1 hora e 50 minutos. Durante esse tempo, foram identificados 3 grupos de métricas referentes ao projeto. Os questionários correspondentes ao Projeto C são apresentados no Apêndice A.3.

O primeiro grupo de métricas do Projeto C contém métricas relacionadas a testes, anomalias e/ou *Bugs*. As métricas coletadas no projeto que pertencem a esse grupo são: **Número de *Bugs***, **Cobertura de Código**, ***Status* dos Testes** e **Número de Testes**. O gerente classi-

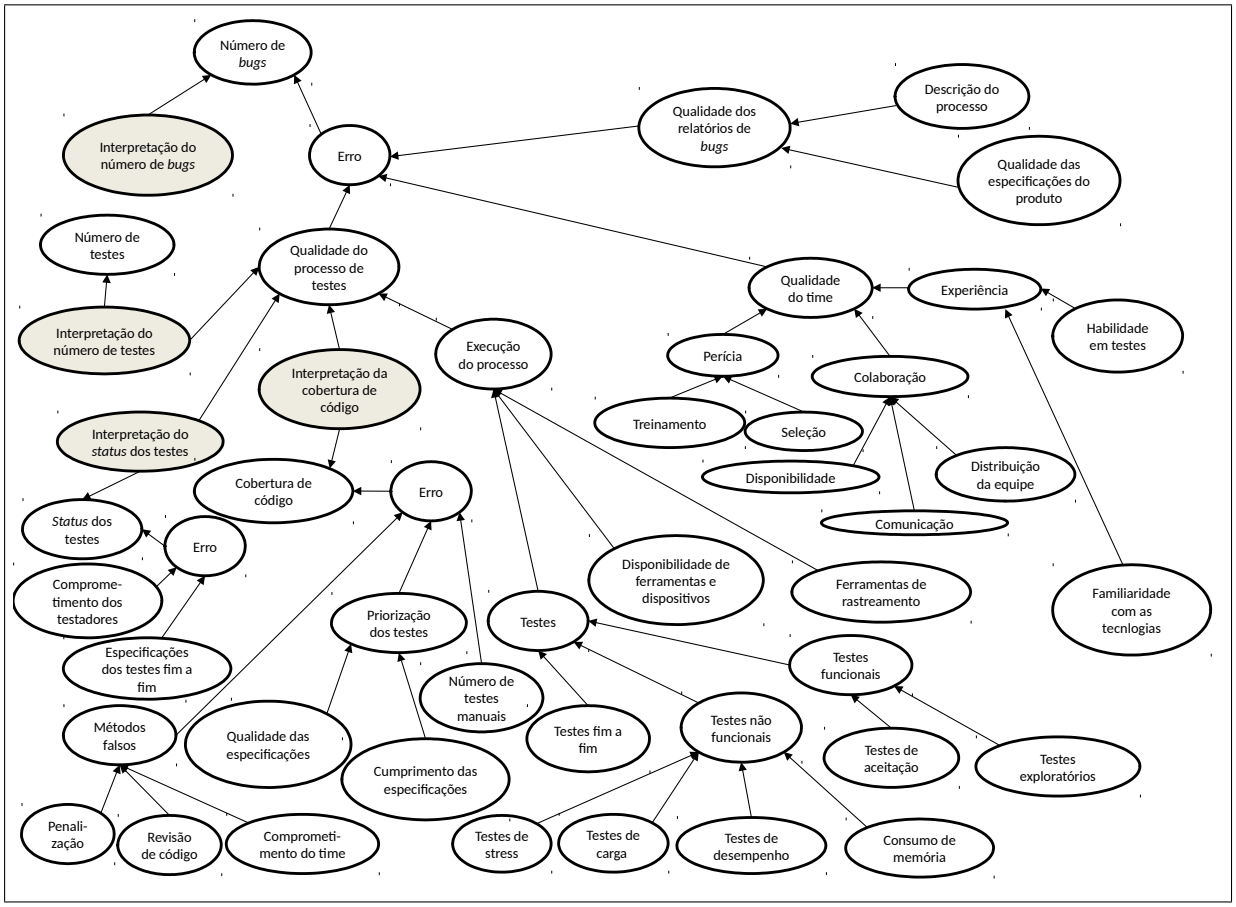

Figura 4.18: GAD construído para o primeiro grupo de métricas do Projeto C

ficou a primeira como um nível acima na hierarquia quando comparada com as três demais. Sua justificativa é que um bom número de testes, com *status* satisfatório e uma alta cobertura de código influenciam diretamente no número de defeitos identificados. O GAD desse grupo de métricas é ilustrado na Figura 4.18.

O GAD gerado para este grupo, dentre os outros grafos gerados nesta pequisa, é o que apresenta o maior número de nós. Observando-o, é possível perceber que a interpretação das métricas **Cobertura de Código**, ***Status* dos Testes** e **Número de Testes** se apresentam como fonte de erro da métrica **Número de *Bugs***, caracterizando-se como qualidade do processo de testes.

No Projeto C, pela complexidade deste e por exigência do cliente, diversos tipos de teste foram desenvolvidos: Testes fim a fim, testes exploratórios, testes de aceitação, testes de consumo de memória, testes de desempenho, testes de carga e testes de *stress*. Para cada

um desses testes, um nó correspondente à sua qualidade foi criado no grafo. Se todos esses nós fossem pais de um nó representando a qualidade geral dos testes, os cálculos da rede Bayesiana seriam muito complexos. Portanto, esses testes foram divididos em três grupos: testes funcionais, testes não funcionais e testes fim a fim. Dessa forma, o nó representando a qualidade geral dos testes só possui três pais e a rede Bayesiana não é prejudicada.

Outro ponto relevante nesse grupo de métricas é a confiança do gerente no seus times de desenvolvimento e de testes. O comprometimento do time foi citado mais de uma vez como um fator controlador de riscos, por exemplo na criação de métodos falsos para aumentar a cobertura de código e o risco de mascarar o verdadeiro *status* dos testes.

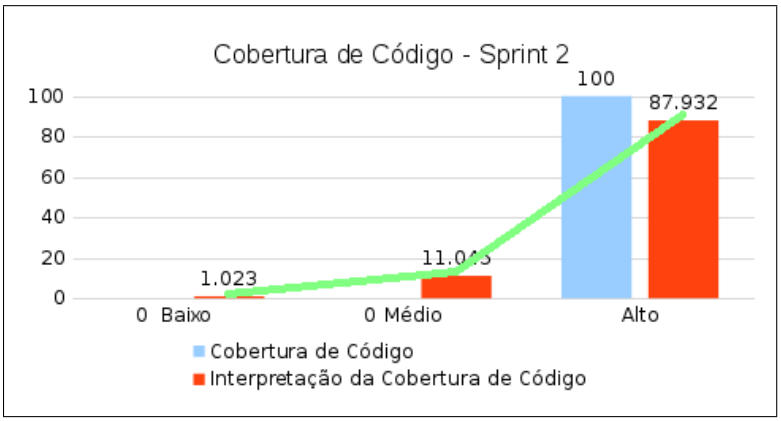

Figura 4.19: Gráfico correspondente à interpretação da métrica Cobertura de Código na segunda *Sprint* do Projeto C

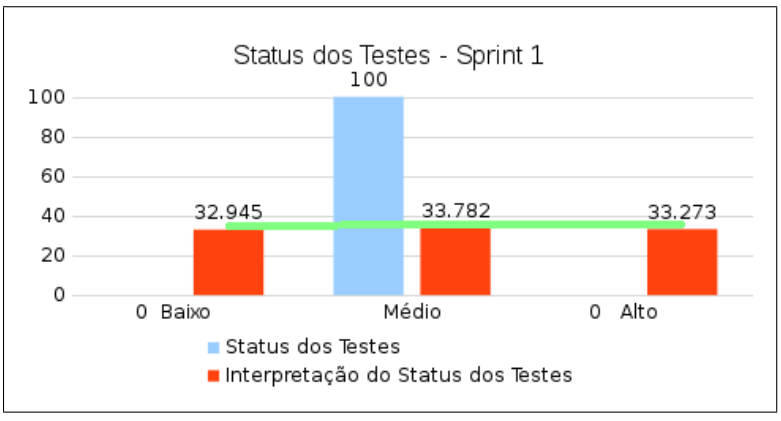

Figura 4.20: Gráfico correspondente à interpretação da métrica *Status* dos Testes na primeira *Sprint* do Projeto C

A Figura 4.19 exibe um gráfico com a interpretação da métrica **Cobertura de Código** na segunda *Sprint* considerada no Projeto C. É possível perceber que a interpretação da métrica converge para o valor observado pelo gerente. Dessa forma, recomenda-se que essa métrica seja considerada pelo gerente ao se tomar decisões no âmbito de testes e anomalias do projeto. A qualidade das especificações e o fato de os desenvolvedores a respeitarem durante o desenvolvimento fazem com que os testes sejam bem priorizados, reduzindo o risco que poderia ser inserido com uma priorização precária. Além disso, um baixo número de testes manuais e o comprometimento do time para que não se implementem métodos falsos colaboram positivamente para a acurácia da cobertura de código.

Em contrapartida, a Figura 4.20 ilustra uma perspectiva menos animadora. A curva que auxilia a visualização da tendência da interpretação da métrica *Status* dos testes está quase totalmente perpendicular às barras do gráfico. Isso implica em um alto erro na interpretação do valor medido. Como vê-se na Figura 4.18, essa métrica possui apenas duas fontes de risco. As especificações dos testes fim a fim não foram muito bem feitas, o que acabou diminuindo o comprometimento do time no que diz respeito a se importar com a medição do *status* dos testes. Portanto, como já vinha sendo feito no projeto, não é recomendado que, na primeira *Sprint*, essa métrica seja tomada como base para a tomada de decisões por parte do gerente. O gerente decidiu atacar as fontes de erro e a especificação dos testes fim a fim melhoraram gradualmente. Assim, em *Sprints* posteriores, a confiança nessa métrica aumentou.

O segundo grupo de métricas do Projeto A tem foco na documentação do projeto. O GAD gerado para esse grupo é ilustrado na Figura 4.21. A medição da documentação do projeto, de acordo com o gerente, acontece por meio da observação de apenas uma métrica: **Número de Alertas de *Javadoc***. A ferramenta que coleta os alertas de *Javadoc* pode apresentar falsos positivos, por se tratar de um mecanismo automatizado. Além disso, ela não analisa a semântica da documentação. Essa fonte de risco pode fazer com que a ferramenta não alerte a documentação errada de determinados trechos de código, mas o gerente identificou a revisão de código como um fator que pode mitigar o impacto dessa fonte de erro.

A Figura 4.22 exibe um gráfico com a interpretação da métrica **Número de Alertas de *Javadoc*** na terceira *Sprint* considerada no Projeto C. A figura deixa claro que a interpretação calculada converge para o valor observado pelo gerente. Dessa maneira, a sugestão feita para

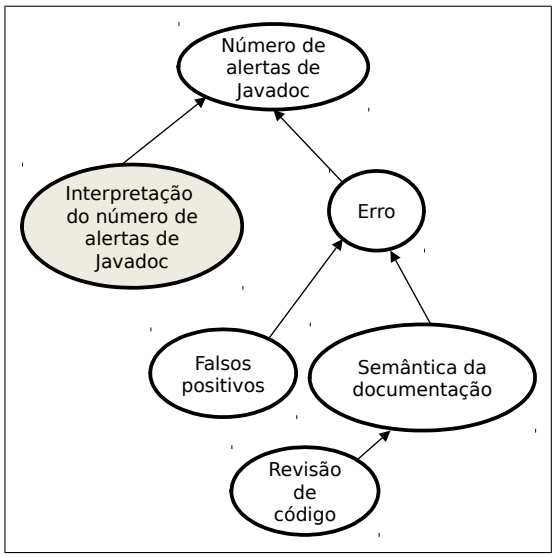

Figura 4.21: GAD construído para o segundo grupo de métricas do Projeto C

o gerente, de acordo com a abordagem proposta, é de considerar essa métrica ao se tomar decisões referentes à documentação do projeto.

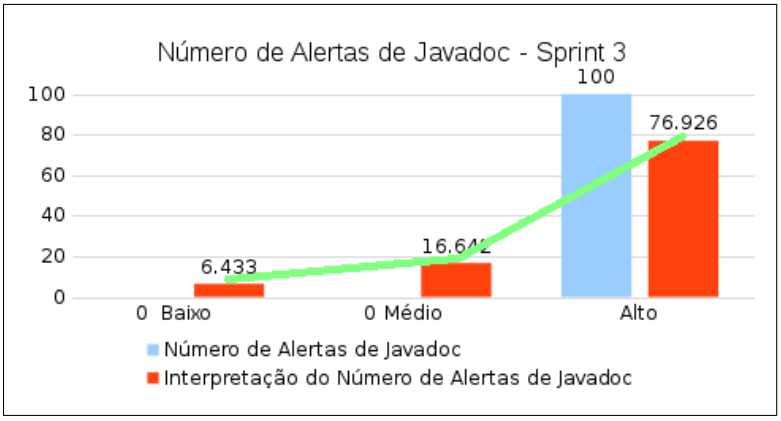

Figura 4.22: Gráfico correspondente à interpretação da métrica Número de Alertas de *Javadoc* na terceira *Sprint* do Projeto C

O terceiro grupo de métricas do Projeto C diz respeito à análise estática. O GAD que representa esse grupo de métricas é ilustrado na Figura 4.23. Apenas uma métrica foi classificada nesse grupo pelo gerente: **Número de Alertas de Análise Estática**. Essa métrica leva em consideração alertas coletados pelas ferramenta de análise estática *Checkstyle*, *Findbugs* e *Lint*. A única fonte de erro identificada pelo gerente foi a produção de falsos positivos pela

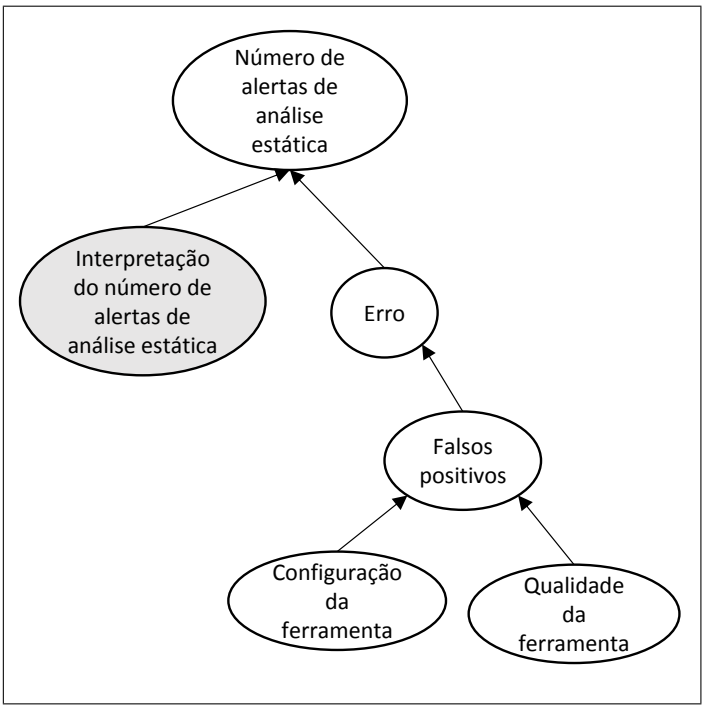

Figura 4.23: GAD construído para o terceiro grupo de métricas do Projeto C

ferramenta. Os fatores identificados pelo gerente com o objetivo de mitigar esse risco seriam a escolha de uma ferramenta de boa qualidade e a configuração apropriada da mesma.

A Figura 4.24 exibe um gráfico com a interpretação da métrica **Número de Alertas de Análise Estática** na terceira *Sprint* considerada no Projeto C. Nota-se, pela observação da figura, que a interpretação calculada pela abordagem não converge para o valor observado pelo gerente. A baixa confiança nessa métrica pode ser justificada pela baixa qualidade da ferramenta adotada no projeto, o que aumenta as chances da presença de falsos positivos. Ainda segundo o gerente, a configuração da ferramenta não recebe esforços significativos, pois o foco dos desenvolvedores são a realização de testes e a produção de código. Então, considerando as características do projeto, não se recomenda o uso dessa métrica para se tomar decisões no contexto da qualidade do código fonte.

A resposta dada pelo gerente entrevistado no Projeto C para a primeira questão de pesquisa foi: *"Sim. A aplicação da abordagem proporcionou uma compreensão muito útil acerca do que estava acontecendo no projeto. Baseado no parecer fornecido, nós passamos a dar atenção em algumas métricas em detrimento de outras. Para as métricas que demonstraram uma baixa confiança, nós tivemos discussões internas para identificar quais razões*

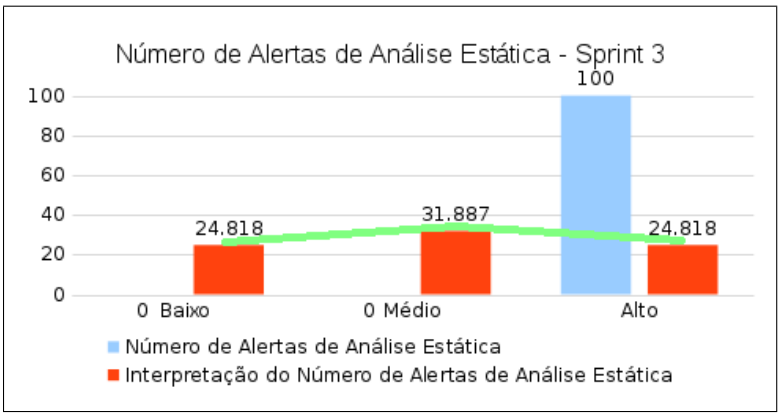

Figura 4.24: Gráfico correspondente à interpretação da métrica Número de Alertas de *Checkstyle* na segunda *Sprint* do Projeto A

poderiam ser causas desses resultados. Em seguida, trabalhamos para tentar consertar alguns aspectos que contribuíam para diminuir o nível de confiança dessas métricas".

A resposta da segunda questão de pesquisa foi: *"Com certeza. O esforço para criar o modelo tomou algum tempo no início. Depois disso, o uso do modelo foi realmente bem simples e objetivo. Os resultados obtidos aumentaram o nosso entendimento acerca do nosso processo de desenvolvimento de software. Realmente vale a pena"*.

Os gráficos correspondentes às probabilidades calculadas em todos os grupos de métricas do Projeto C, em todas as *Sprints*, encontram-se no Apêndice B.3.

4.5 Projeto D

Assim como o Projeto B, o Projeto D também focou no desenvolvimento de uma aplicação para dispositivos vestíveis. A aplicação desenvolvida, assim como a do projeto citado anteriormente, foi publicada na *Google Play*. A aplicação desenvolvida foi um pouco mais complexa que a do Projeto B e a equipe também foi maior: 8 desenvolvedores, 2 testadores e 2 gerentes. Todos formados e trabalhando em regime integral.

O gerente de projeto que participou das atividades desta pesquisa foi aquele responsável pela maior quantidade de módulos e componentes desenvolvidos no Projeto D, além de ter gerenciado o projeto sozinho por um certo período de tempo. Portanto, entende-se que esse gerente possui um entendimento maior com relação ao projeto em questão.

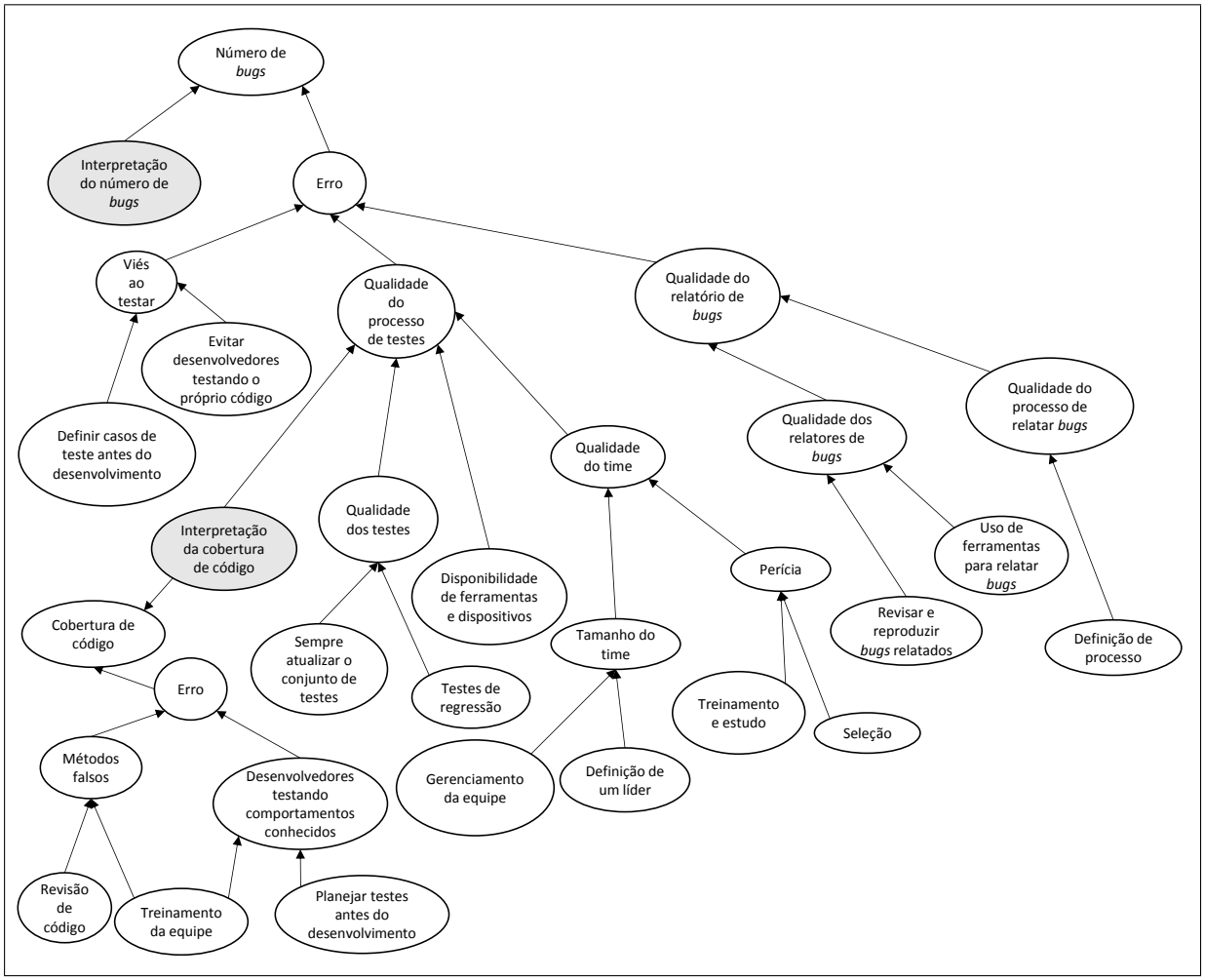

Figura 4.25: GAD construído para o primeiro grupo de métricas do Projeto D

A reunião com o gerente do Projeto D que objetiva construir os GAD correspondentes ao projeto teve duração aproximada de 50 minutos. Nessa reunião, o gerente identificou quatro distintos grupos de métricas. Os questionários respondidos pelo gerente do Projeto D, referentes à distribuição do peso de cada um dos nós dos GAD construídos, encontram-se no Apêndice A.4.

O primeiro grupo de métricas do Projeto D, de forma similar aos demais, compõe-se de métricas relacionadas a testes, anomalias e/ou *Bugs*. As métricas identificadas pelo gerente que pertencem a esse grupo são: **Número de *Bugs*** e **Cobertura de Código**. A segunda, de acordo com o gerente, possui influência direta no valor da primeira. Portanto, **Número de *Bugs*** está um nível hierárquico acima de **Cobertura de Código**. O grafo gerado para esse grupo de métricas pode ser visualizado na Figura 4.25.

Observando o GAD do Grupo 1, percebe-se que as fontes de erro do **Número de *bugs*** são a qualidade do processo de testes, a qualidade dos relatórios de *Bugs* e o viés ao realizar os testes. Como exemplo de práticas ou atitudes que o gerente citou para controlar os riscos que podem afetar a acurácia da medição, podem ser citados: planejamento de testes antes do desenvolvimento, bom gerenciamento da equipe, treinamento/estudo, revisão e reprodução de todos os *bugs* relatados para assegurar a existência do defeito, disponibilidade de ferramentas e dispositivos e definição do processo de relatar *bugs*.

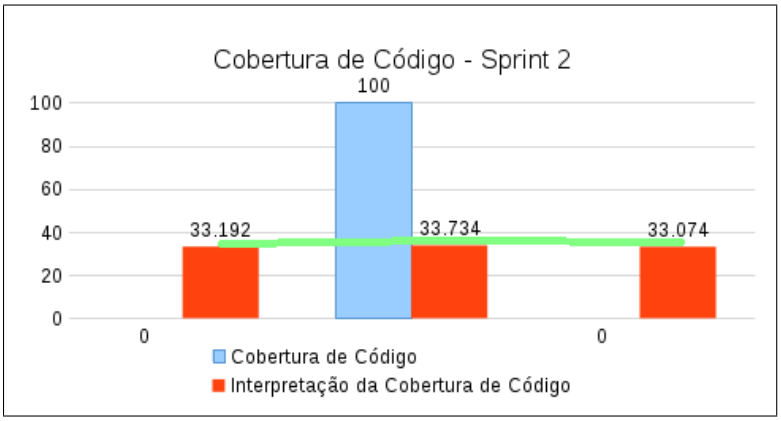

Figura 4.26: Gráfico correspondente à interpretação da métrica Cobertura de Código na segunda *Sprint* do Projeto D

A Figura 4.26 ilustra um gráfico com a interpretação da métrica **Cobertura de Código** na segunda *Sprint* levada em consideração no Projeto D. A observação do gerente foi que o valor observado caracterizava-se como *Médio*, porém a tendência da interpretação da métrica, representada pela linha verde no gráfico, apresenta-se quase inteiramente perpendicular às barras do gráfico. Isso demonstra uma alta taxa de erro na interpretação da métrica e, consequentemente, uma baixa confiança na medição. Isso pode ser justificado pelo fraco, ou mesmo inexistente, planejamento dos testes antes da etapa de desenvolvimento e da ausência de treinamento da equipe com relação ao desenvolvimento de testes para maximizar a cobertura de código. Esses dois fatores controladores poderiam controlar as duas fontes de erro identificadas para a métrica: os desenvolvedores testando apenas comportamentos conhecidos e a implementação de métodos falsos para tentar mascarar o valor medido. Portanto, o uso da métrica **Cobertura de Código** para a tomada de decisões relativas a testes e afins.

O segundo grupo de métricas do Projeto D possui o escopo de documentação do projeto.

O grafo correspondente a esse grupo produzido pela aplicação da abordagem é ilustrado na Figura 4.27. Só uma métrica foi classificada no grupo 2: **Número de Alertas de *Checkstyle***. Assim como no Projeto A, os alertas considerados nessa métrica são aqueles coletados pela ferramenta *Checkstyle* que são referentes à documentação de código. A fonte de erro é a mesma do Projeto A, ou seja, a falta de uma análise semântica da documentação. A forma de mitigar esse risco identificada pelo gerente é a revisão de código em pares, na qual um código produzido é revisado por um desenvolvedor que não participou do desenvolvimento do código.

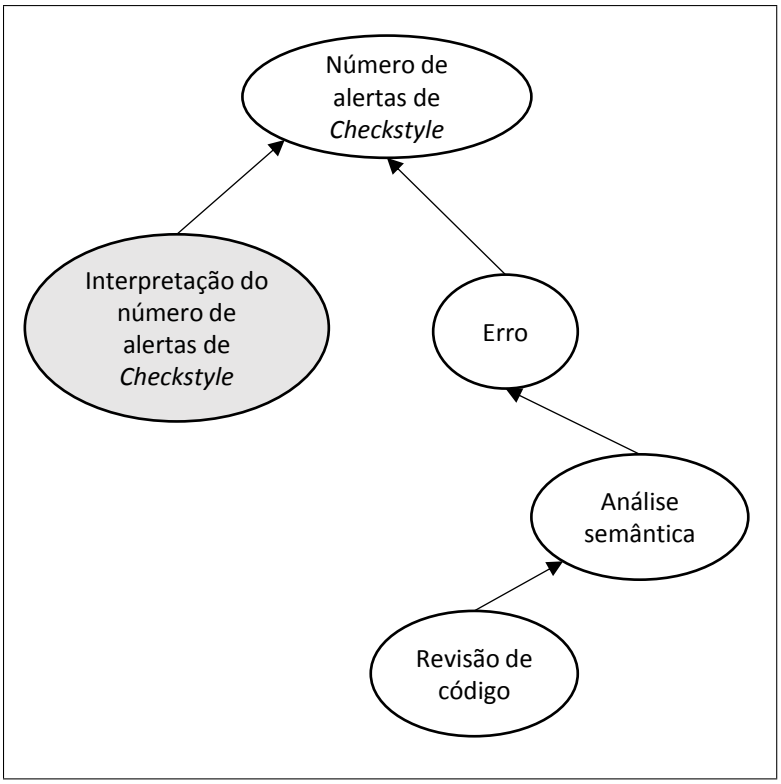

Figura 4.27: GAD construído para o segundo grupo de métricas do Projeto D

A Figura 4.28 exibe um gráfico com a interpretação da métrica **Número de Alertas de *Checkstyle*** na terceira *Sprint* estudada no Projeto D. Conforme a visualização da figura permite concluir, a interpretação calculada converge para o valor observado pelo gerente, mesmo que não seja de forma muito acentuada. Portanto, o valor medido para essa métrica pode ser considerado pelo gerente para que este tome decisões relacionadas no que diz repeito à documentação, mas sempre tentando manter o código bem revisado.

O terceiro grupo de métricas do Projeto D corresponde à análise estática. O GAD gerado

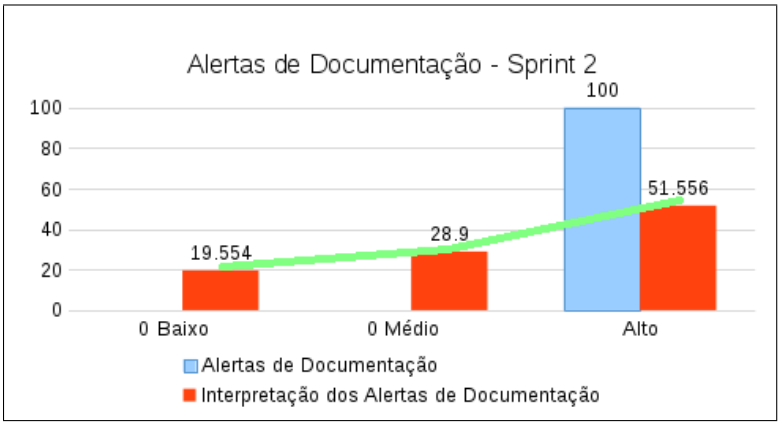

Figura 4.28: Gráfico correspondente à interpretação da métrica Número de Alertas de *Checkstyle* na terceira *Sprint* do Projeto D

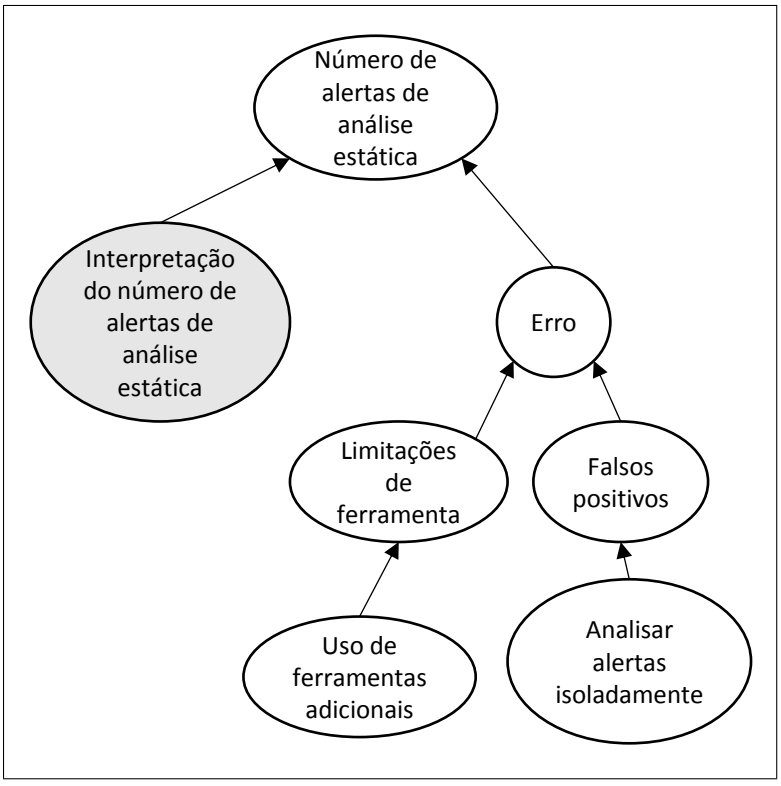

Figura 4.29: GAD construído para o terceiro grupo de métricas do Projeto D

para esse grupo de métricas pode ser visualizado na Figura 4.29. Apenas uma métrica se enquadra nesse grupo, segundo o gerente: **Número de Alertas de Análise Estática**. Os alertas levados em consideração nessa métrica são aqueles coletados pelas ferramentas *Checkstyle* (apenas os que não têm relação com documentação de código), *Findbugs*, *Lint* e *PMD*. As ferramentas podem apontar falsos positivos e os números observados não refletirem no número real de alertas. Para contornar esse risco, o gerente sugere a análise isolada de cada alerta para verificar se este não se trata de um falso positivo. Esta abordagem pode parecer surreal à primeira vista. No entanto, como o número de alertas, idealmente, deve ser mantido perto de zero, um baixo número de alertas é analisado isoladamente. Além disso, a ferramenta utilizada possui certas limitações que, segundo o gerente, devem tentar ser contornadas por meio do uso de ferramentas adicionais.

A Figura 4.30 exibe um gráfico com a interpretação da métrica **Número de Alertas de Análise Estática** na primeira *Sprint* estudada no Projeto D. Como pode ser observado na figura, a interpretação calculada com a abordagem converge moderadamente para o valor observado pelo gerente. Segundo o gerente, a prática de analisar os alertas isoladamente é bem empregada no projeto. No entanto, o uso de ferramentas adicionais para contornar as limitações das ferramentas que coletam os alertas ainda é moderado. Portanto, recomenda-se o uso da métrica para se tomar decisões em relação à qualidade do código fonte do projeto e também é encorajado o uso de mais ferramentas adicionais para contornar as limitações das ferramentas já utilizadas, com o intuito de aumentar a acurácia da medição.

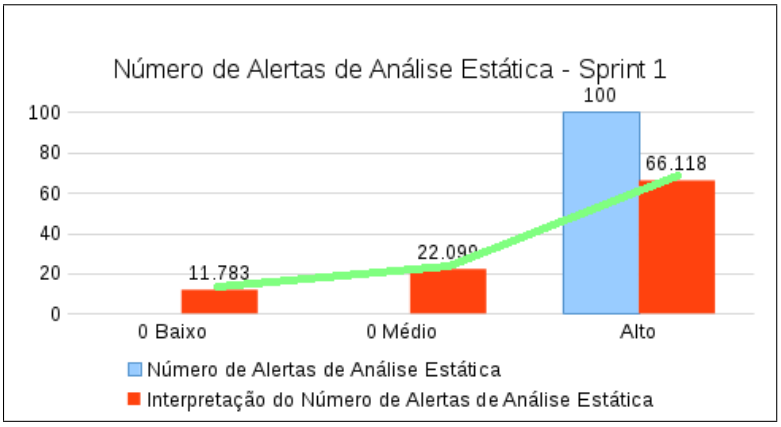

Figura 4.30: Gráfico correspondente à interpretação da métrica Número de Alertas de Análise Estática na primeira *Sprint* do Projeto D

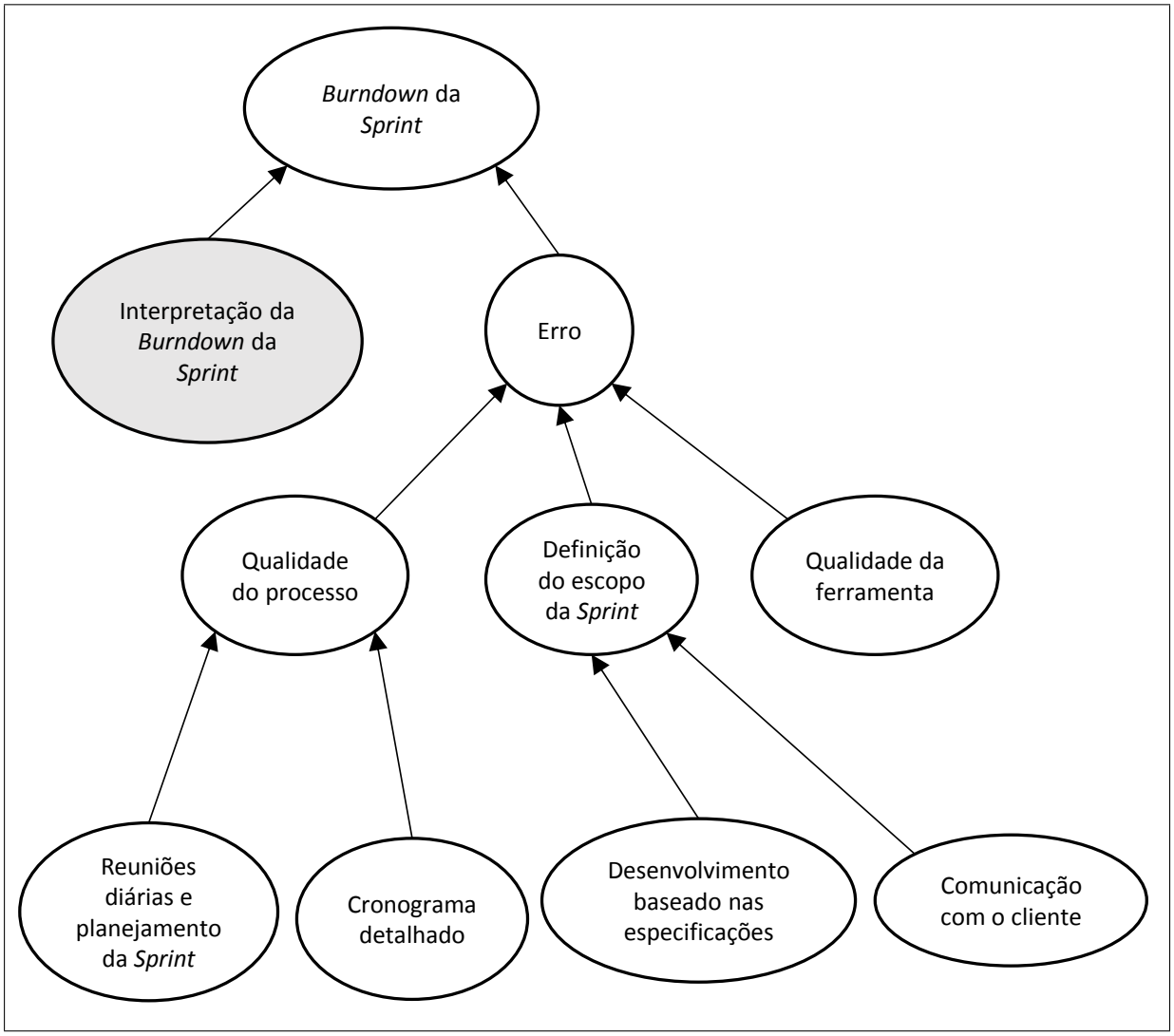

Figura 4.31: GAD construído para o quarto grupo de métricas do Projeto D

O quarto grupo de métricas do Projeto D corresponde ao gerenciamento do projeto, mais especificamente à gerência de atividades realizadas na *Sprint*. O grafo relativo a esse grupo de métricas pode ser visualizado na Figura 4.31. Só uma métrica pertence a esse grupo segundo o gerente: ***Burndown* da *Sprint***. As fontes de erro da medição dessa métrica, identificadas pelo gerente, são uma precária qualidade do processo de desenvolvimento, uma imprópria definição do escopo da *Sprint* e uma ferramenta de gerenciamento de baixa qualidade. Com o intuito de diminuir o impacto desses riscos sobre a medição, algumas técnicas foram identificadas pelo gerente: reuniões diárias com a equipe de desenvolvimento, um bom planejamento da *Sprint*, definição de um cronograma bem detalhado, desenvolvimento

baseado nas especificações do projeto e comunicação constante com o cliente, com o intuito de sanar eventuais dúvidas.

A Figura 4.32 exibe um gráfico com a interpretação da métrica ***Burndown* da *Sprint*** na primeira *Sprint* considerada no Projeto D. É possível notar que a interpretação calculada converge claramente para o valor observado pelo gerente. A justificativa para uma convergência tão evidente é a boa aplicação dos fatores mitigadores dos riscos associados com a coleta da métrica. No caso dessa *Sprint*, o gerente considerou que todas as medidas de prevenção de riscos foram bem aplicadas. No caso da Figura 4.33, ainda se observa uma clara convergência para o valor observado pelo gerente, porém, um pouco menos acentuada que o exemplo anterior. A justificativa para isso é que o gerente considerou que a definição de um cronograma bem detalhado foi feita de maneira moderada e não tão boa quanto na *Sprint* anterior. De toda forma, a recomendação é que a métrica ***Burndown* da *Sprint*** seja levada em consideração no processo de tomada de decisões acerca do projeto, no que diz respeito ao processo de desenvolvimento e gerenciamento de tarefas.

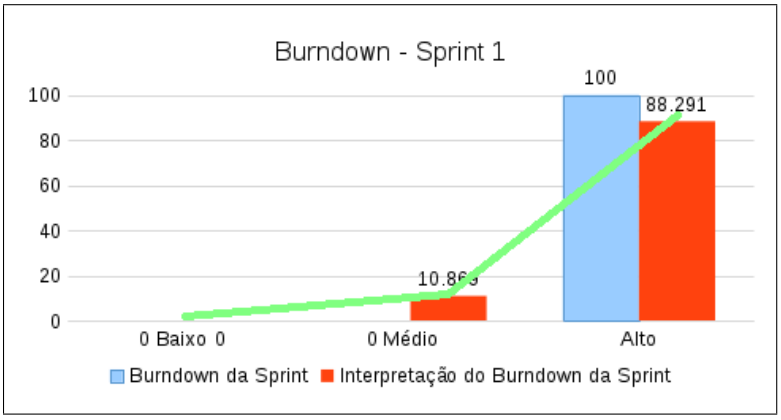

Figura 4.32: Gráfico correspondente à interpretação da métrica *Burndown* da *Sprint* na primeira *Sprint* do Projeto D

O gerente entrevistado no projeto D respondeu a primeira questão de pesquisa dessa maneira: *"Sim, pois ele usa como base para a interpretação dados concretos, diferentemente da avaliação subjetiva que é realizada atualmente. Com isso é possível demonstrar com maior clareza e precisão o que necessita mellhoramento no processo utilizado. Por exemplo, sabemos que temos problemas relacionados a testes, porém não conseguimos dizer com objetividade onde esse problema está de fato. Pela análise ficou mais claro que o problema*

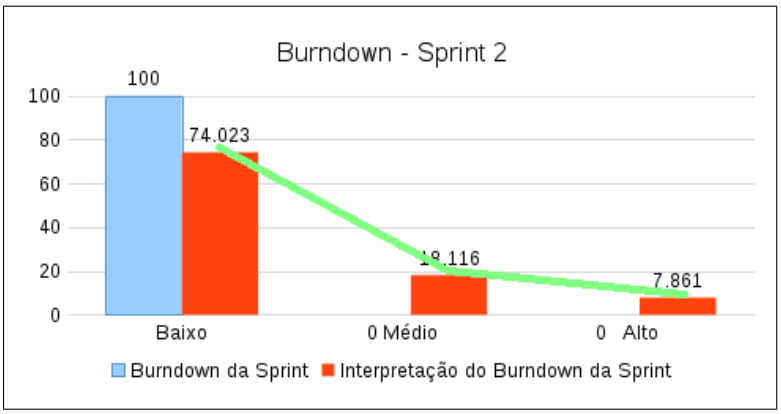

Figura 4.33: Gráfico correspondente à interpretação da métrica *Burndown* da *Sprint* na segunda *Sprint* do Projeto D

é geral (para toda a equipe), possivelmente por não termos grande familiaridade com elaboração de testes, ou seja, não é algo que esteja localizado apenas no time de testes".

A seguinte resposta foi dada à segunda questão de pesquisa: *"Sim, pois o tempo despendido para popular a base compensa as ánalises feitas, as quais acabam elencando os pontos mais falhos do processo. Com esses pontos bem definidos é possível tomar medidas pontuais para melhorar como todo (a equipe se aprimora, o processo melhora, aumenta o controle dos resultados obtidos, melhora a qualidade do produto entregue, melhora a satisfação do cliente, etc)"*.

Os gráficos correspondentes às probabilidades calculadas em todos os grupos de métricas do Projeto D, em todas as *Sprints*, encontram-se no Apêndice B.4.

4.6 Discussão Geral dos Resultados

Levando em consideração as respostas de todos os gerentes de projeto que participaram da pesquisa, pode-se considerar que ambas as questões de pesquisa foram respondidas positivamente. Em todos os projetos estudados, a abordagem proposta se mostrou útil para auxiliar os gerentes a interpretar as métricas coletadas no projeto e a tomar decisões referentes ao projeto. Segundo os gerentes, o custo-benefício da aplicação da abordagem nos seus projetos também é satisfatório.

A fim de fortalecer as respostas dadas pelos gerentes para as questões de pesquisa, foi

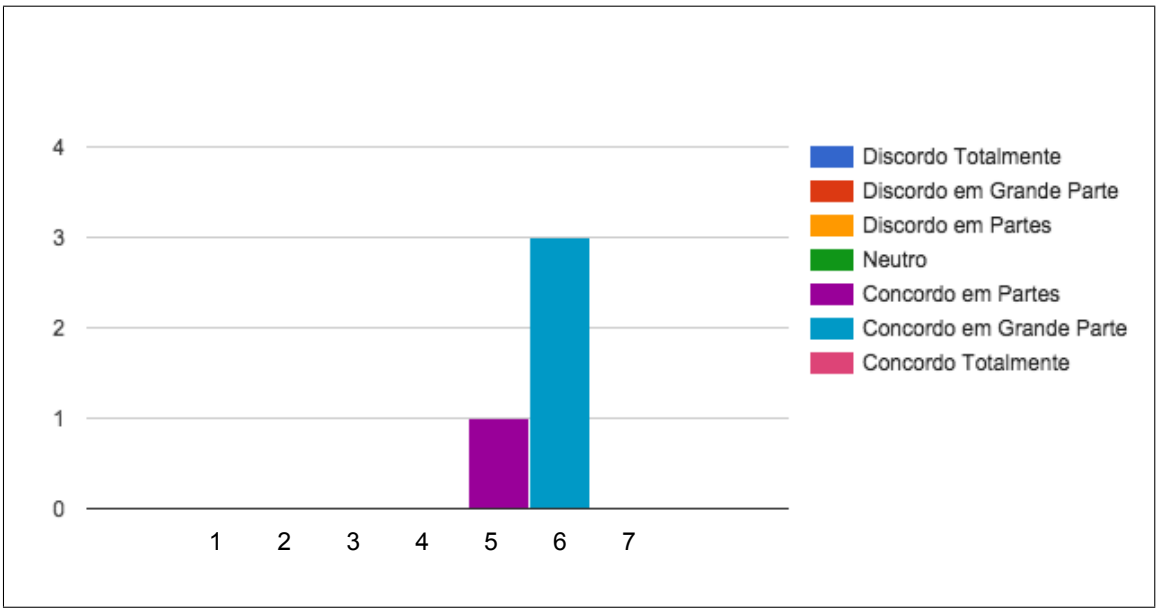

Figura 4.34: Escala de Likert de 7 pontos referente à questão de pesquisa *"A abordagem apresentada é útil no sentido de guiar, ou auxiliar, o time na interpretação de métricas?"*

feito ainda um questionário utilizando escala de Likert de 7 pontos a fim de agregar uma medição mais quantitativa aos resultados. Na Figura 4.34, nota-se que todos os gerentes que responderam ao questionário concordam (em partes ou em grande parte) que a abordagem proposta é útil no sentido de guiar o time na interpretação das métricas.

Já com relação ao custo-benefício da abordagem considerando o esforço em aplicá-la ao projeto, pode ser visto na Figura 4.35 que a maioria dos gerentes concorda em grande parte que este custo benefício é satisfatório, um dos gerentes concorda em partes e outro gerente manteve uma opinião neutra. Usando algumas medidas de tendência central, percebe-se que de uma maneira geral o custo-benefício da aplicação do método é considerado satisfatório pelos gerentes:

- A **moda** das respostas é 6, equivalente a *"Concordo em grande parte"*;
- A **média** das respostas é $\left(\frac{4+5+6+6}{4}\right) = 5{,}25$, o que seria equivalente a algo entre *"Concordo em partes"* e *"Concordo em grande parte"*;
- A **mediana** das respostas é $\left(\frac{5+6}{2}\right) = 5{,}5$, o que também seria equivalente a algo entre *"Concordo em partes"* e *"Concordo em grande parte"*.

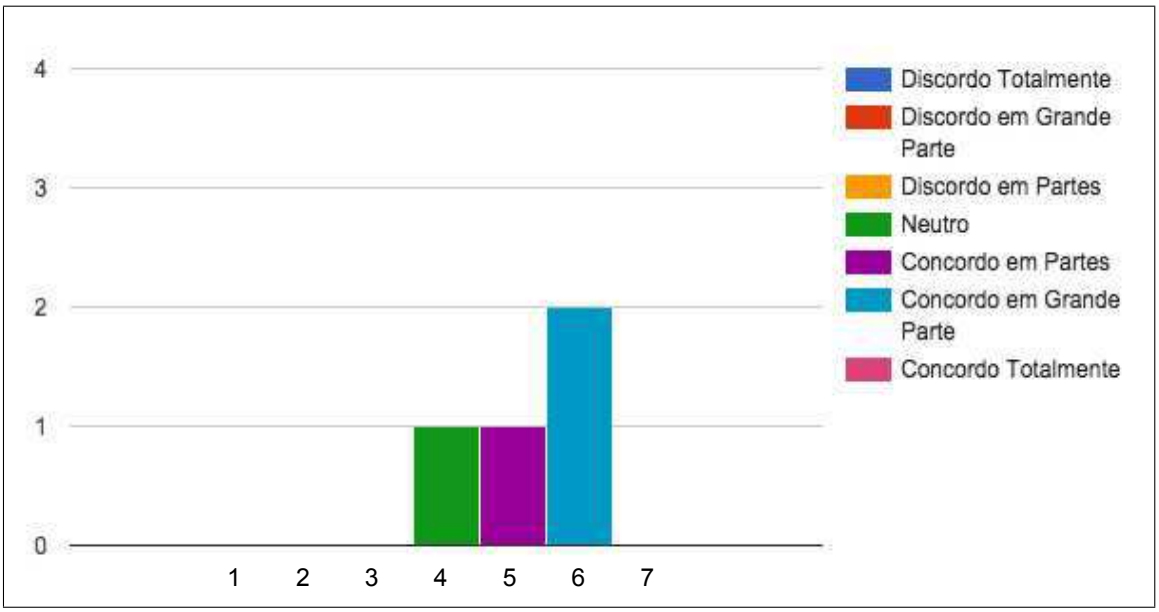

Figura 4.35: Escala de Likert de 7 pontos referente à questão de pesquisa *"O custo-benefício da utilização da abordagem é satisfatório, considerando o esforço para aplicá-la?"*

Portanto, verificou-se que o uso da abordagem proposta, além de ter um custo-benefício satisfatório, auxilia a interpretação de métricas (ou seja, indica a confiança das medições e aponta meios de contornar as fontes de erro que afetam a acurácia das métricas). Por exemplo, uma rede Bayesiana produzida com a aplicação da abordagem em um determinado projeto pode identificar uma baixa qualidade do processo de desenvolvimento. O gerente, por sua vez, pode identificar que a causa desse problema é uma baixa qualidade na seleção, ou treinamento, da equipe e focar esforços para melhorar ambos os fatores.

4.7 Ameaças à validade

A abordagem proposta nesta pesquisa possui ameaças à validade interna e externa. No que diz respeito à validade interna, algumas das reuniões realizadas com os gerentes (principalmente durante a execução das etapas i, ii e iii) se estenderam por um período relativamente longo, chegando a durar pouco mais de uma hora. Isso pode ter causado uma certa fadiga nos gerentes, fazendo com que estes não apresentassem a dedicação adequada no processo de construção do modelo que representa o seu projeto. Acredita-se que, pelo fato do sucesso da abordagem implicar em uma melhora significativa no andamento do projeto como um

todo, o gerente possui uma motivação extra para contribuir da melhor forma possível. Além disso, procurou-se dar suporte aos gerentes durante toda a reunião, esclarecendo eventuais dúvidas e iniciando discussões quando não se notava avanço na conversa.

Ainda com relação à ameaça à validade interna, pode-se citar a qualidade das respostas dos questionários referentes ao passo iv da abordagem proposta. Como esta etapa não ocorreu de forma presencial, não foi possível realizar um acompanhamento direto. Assim, alguns gerentes podem ter respondido algumas perguntas sem tanta confiança, pelo simples motivo de não achar que sua dúvida era digna de uma troca de *emails* ou de realizar uma nova reunião. Entretanto, buscou-se elaborar os questionários evitando ambiguidades, explicando detalhadamente no que consistia essa etapa. Os modificadores **mínimo** e **máximo** também foram explicados em detalhes, citando exemplos bastante compreensíveis do seu uso. Para auxiliar ainda mais os gerentes, a representação gráfica dos GAD referentes a cada grupo de métricas foi anexada ao questionário correspondente.

Outro tipo de ameaça à validade interna ocorre na definição dos parâmetros **média** (μ) e **variância** (σ^2) na distribuição duplamente truncada. A suposição que o valor de σ^2 deve ser o mínimo possível pode estar errada, assim como determinar que o valor de μ é calculado por uma expressão ponderada. Essas suposições, no entanto, foram baseadas no trabalho de Perkusich et al. [31] e foram validadas anteriormente. Outra suposição feita foi a dos valores de σ^2 nas funções de probabilidade dos nós que representam o erro na medição. Para se calibrar esses valores e chegar nos números determinados, alguns cenários hipotéticos foram construídos e executados no *AgenaRisk* até que as probabilidades calculadas apresentassem resultados satisfatórios.

Finalmente, no caso de ameaças à validade externa, é válido lembrar que a abordagem proposta foi validada em apenas quatro projetos, de um mesmo laboratório, em um período de quarenta e cinco dias cada. Considerando o universo de todos os projetos de desenvolvimento de software espalhados pelo mundo, esse número é ínfimo, o que pode significar que o resultado positivo obtido na pesquisa não reflita na realidade de grande parte dos projetos existentes. Tentar abranger todos, ou a maioria dos projetos de desenvolvimento de software é algo fora de cogitação, pois foge bastante à realidade de uma pesquisa. Entretanto, os projetos escolhidos, apesar de serem da mesma empresa, apresentam características bem distintas e cada um deles possui suas particularidades. Isso pode ser notado ao comparar os GAD

resultantes em cada um deles. Espera-se que, escolhendo projetos com diferentes perfis, essa ameaça possa ser amenizada e que a abordagem possa ser aplicada de forma satisfatória em projetos similares aos utilizados nesta pesquisa, com um time de 5 a 12 integrantes, processo de desenvolvimento ágil e focados no desenvolvimento de aplicações para dispositivos móveis (plataforma *Android*). Outro caso de ameaça à validade externa é o fato de a execução da abordagem ter sido conduzida por um especialista, então a sua aplicação futura, se feita por profissionais não tão experientes, pode tomar mais tempo do que foi gasto na presente pesquisa.

Capítulo 5

Trabalhos Relacionados

Outros trabalhos na literatura foram investigados durante o desenvolvimento desta pesquisa. Essas fontes abordam a importância da adoção de métricas no contexto de projetos de desenvolvimento de software e propõem modelos e/ou abordagens para que estas métricas sejam utilizadas no intuito de auxiliar os projetos nos quais elas são coletadas e não causem apenas perda de tempo e recursos. Afinal, de acordo com Fenton e Neil [34], o requisito mais importante para uma métrica de software é que esta proveja informação quantitativa para dar suporte ao processo de tomada de decisão gerencial durante o ciclo de vida de um software.

Wallace e Sheetz [18] propuseram um Modelo de Aceitação de Tecnologia (MAT) para tentar identificar e explicar problemas ao se adotar métricas em projetos de software. Um MAT tem como objetivo desvendar as razões de indivíduos adotarem, ou não, determinada tecnologia ao executar uma tarefa. Esse tipo de modelo considera que duas variáveis possuem impacto na adoção de uma tecnologia: a utilidade da tecnologia e a facilidade do seu uso.

De acordo com os autores de [18], muitas métricas são difíceis de se entender e sua implantação em projetos de desenvolvimento de software não é uma tarefa simples. O modelo proposto no trabalho teve como objetivo entender a razão da adoção de métricas de software em projetos ser tão difícil. A conclusão dos autores é que os gerentes devem deixar claro para os desenvolvedores a importância e a utilidade das métricas no escopo do projeto e que eles tentem facilitar ao máximo o seu uso (provendo treinamentos, por exemplo).

Apesar de destacar a importância do uso de métricas em seu trabalho e propor maneiras da inserção destas em projetos de desenvolvimento, Wallace e Sheetz não consideraram pro-

blemas que podem ocorrer após a implantação da métrica, como a baixa acurácia devido a fatores subjetivos.

Outros tipos de modelo também podem ser usados na representação de métricas. Por exemplo, modelos causais, como redes Bayesianas, podem ser utilizados para representar métricas sem introduzir ou exigir custos adicionais em termos da quantidade de dados coletados ou da complexidade das métricas [37]. Além disso, esses modelos dão suporte a:

- diversas variáveis representando produtos e processos;
- evidência empírica e julgamento de especialistas;
- relacionamentos entre causa e efeito;
- incerteza;
- informação incompleta.

Alguns pesquisadores realizaram trabalhos nos q uais combinaram métricas de software com redes Bayesianas para construir modelos causais com o objetivo de dar suporte à tomada de decisões em projetos. Fenton e Neil [37] propuseram uma abordagem para modelar e predizer defeitos e recursos necessários em projetos de software considerando fatores subjetivos, como a qualidade da equipe e o esforço necessário para se testar o sistema. Os autores relataram que os modelos construídos podem ser vistos como ferramentas de apoio a decisões e gerenciamento de riscos baseadas em métricas que fazem uso das métricas que já são coletadas nos projetos.

Seguindo o mesmo escopo, Wagner [41] apresentou uma abordagem para construir modelos de qualidade baseados em atividade derivados de redes Bayesianas. Os modelos construídos por meio da abordagem proposta por Wagner têm o objetivo de predizer e tentar aumentar a qualidade de *softwares*, utilizando métricas como indicatores. Abouelela e Benedicenti [25], por sua vez, utilizaram métricas (como número de defeitos e velocidade do time) para modelar o famoso processo de desenvolvimento de software ágil *Extreme Programming (XP)* [1]. O objetivo do trabalho foi utilizar o modelo para tentar predizer a qualidade e o tempo gasto por projetos que adotam *XP*.

[1] `http://www.extremeprogramming.org/`

Em outro trabalho, Perskusich et al. [31] propuseram um modelo baseado em redes Bayesianas para detectar problemas em projetos de desenvolvimento de software. O principal objetivo do modelo é deixar o gerente a par dos problemas dos projetos para que este possa guiar o time e aumentar as chances de sucesso, assim como se faz na presente pesquisa. A diferença desse trabalho para o trabalho proposto nesta dissertação é que este foca na coleta e uso das métricas coletadas no projeto.

Uma das limitações de [31] é o fato de o modelo conter apenas valores qualitativos para as métricas que usa. Por exemplo, a qualidade dos testes automatizados e da análise estática do código são nós de entrada da rede Bayesiana apresentada no trabalho. A abordagem proposta nesta dissertação pode ser aliada ao modelo proposto por Perkusich et al. justamente para expandir nós de entrada que representam métricas, fazendo uma decomposição desses nós em um submodelo que irá conter os riscos associados com as métrica e os fatores controladores que têm a função de mitigar tais riscos.

Aproximando-se ainda mais com o trabalho proposto nesta dissertação, Montini et al. [6] propuseram uma extensão do GQM, aplicando redes Bayesianas para modelar o relacionamento entre objetivos (*Goals*), perguntas (*Questions*) e métricas (*Metrics*). O grande diferencial de [6] é a inserção de um nó para representar fatores subjetivos que afetam a medição das métricas.

O trabalho de Montini et al., no entanto, possui um grau de abstração muito elevado e não se aprofunda muito nos conceitos apresentados. Os nós que representam o valor qualitativo da métrica, que englobam os fatores subjetivos que afetam a acurácia da medição, não são bem explicados e tampouco se menciona como devem ser feitas a sua decomposição ou até mesmo, alternativamente, como estes nós devem ser alimentados. O capítulo 3 desta dissertação, em contrapartida, explica com detalhes como os fatores subjetivos que exercem influência sobre as métricas devem ser decompostos e como são os seus relacionamentos com as métricas em si.

Algumas outras técnicas também foram utilizadas com o intuito de se apresentar abordagens para criar modelos baseados em métricas de software e, assim, tentar fazer predições e/ou análises no âmbito de projetos de desenvolvimento. Spasic e Onggo [5], por exemplo, utilizaram simulação baseada em agentes para modelar as fases de *design* e desenvolvimento de um projeto de desenvolvimento de software. O modelo foi construído com o intuito de

estimar a duração dos projetos. Para isso, os autores utilizaram uma base de dados contendo informação de 300 projetos, incluindo a sua duração, o esforço estimado e o esforço real de cada um deles.

Mesmo sendo considerado relativamente simples, o modelo utilizado em [5] foi calibrado e em seguida validado, mostrando-se útil para estimar a duração de projetos de desenvolvimento. Spasic e Onggo concluíram, então, que simulação baseada em agentes se apresenta como uma técnica simples e útil para se modelar projetos de desenvolvimento de software utilizando métricas (nesse caso, esforço medido em pessoas-hora). No entanto, fontes de erro associadas com a medição não foram sequer mencionados no trabalho. Outro diferencial do trabalho proposto é que este foca principalmente em métricas de qualidade de software e não de estimativa de tempo de desenvolvimento, como faz o trabalho de Spasic e Onggo.

Ahmed e Muzaffar [22] apresentaram um arcabouço baseado em lógica nebulosa do tipo 2 com o objetivo de construir modelos capazes de predizer o esforço necessário (medido em pessoas-mês) em projetos de desenvolvimento de software. Métricas e opiniões de especialistas foram utilizadas na alimentação dos modelos construídos por meio do uso do arcabouço. Os autores concluíram que o uso de lógica nebulosa do tipo 2 como base do arcabouço foi satisfatório, pois essa técnica é capaz de lidar com três tipos de imprecisão e incerteza associados com a coleta de métricas de software:

- a imprecisão resultante do uso de dados qualitativos;
- a incerteza decorrente da existência de métricas que são computadas por meio de opiniões de especialistas (que podem ser conflitantes);
- e a incerteza inserida na medição pelo que os autores chamam de preguiça/ignorância do *designer* da métrica.

De acordo com o que é relatado em [22], o fator **preguiça** ocorre quando o *designer*, ou seja, o indivíduo que propõe uma nova métrica, não considera todos os fatores envolvidos com a característica do projeto que está sendo mensurada, por considerar que não seria prático ou demandaria muito trabalho. Entre esses fatores estão inclusos riscos que podem afetar na acurácia da métrica. Já a ignorância significa uma falta de conhecimento teórico acerca dos fatores associados à característica mensurada.

Os trabalhos relacionados identificados auxiliaram bastante na condução desta pesquisa, apresentando ideias reutilizadas e algumas limitações que puderam ser exploradas no presente trabalho. No entanto, em nenhum deles são observadas formas de se identificar e contornar riscos que podem influenciar na acurácia da métrica, como o processo de coleta, processo em relatar a métrica e o mau uso, ou anti-padrões de projeto, que podem ocorrer por comportamentos humanos indesejados.

Ignorar riscos relacionados com métricas pode afetar o gerenciamento do projeto, fazendo com que o gerente tome decisões erradas. Por exemplo, Fenton e Neil [34] ignoram os riscos associados com o número de defeitos detectados para o sistema. Por exemplo, um relator de *bugs* pode não ter um grande conhecimento acerca das especificações do sistema (e especificidades de todas as plataformas nas quais ele deve funcionar) e classificar um comportamento esperado do sistema como indesejado por engano. Nesse caso, o número de defeitos medido não irá representar a realidade.

Outro problema relacionado a riscos pode ser encontrado no trabalho de Spasic e Onggo [5]. Os autores utilizaram a mudança no tamanho dos arquivos do código fonte como indicativo de esforço de desenvolvimento. Existem riscos claros associados com essa decisão. Um desenvolvedor pode escrever um método com 100 linhas de código (*LOC*), enquanto um outro desenvolvedor pode escrever um método com o mesmo comportamento em apenas 10 *LOC*. Na prática, o esforço de desenvolvimento é o mesmo, pois os métodos são similares e o desenvolvedor responsável pelo método com 10 *LOC* provavelmente focaria em buscar a solução mais eficiente ao passo em que o outro iria escrever a solução trivial. Entretanto, as mudanças no tamanho do arquivo teriam uma diferença considerável nesse cenário, o que indicaria que o esforço do método com 100 *LOC* foi maior.

Inúmeros problemas podem ainda ocorrer devido ao não entendimento do significado de uma métrica ou da motivação para usá-la. Dois exemplos não observados nos trabalhos relacionados, mas que são bem conhecidos como anti-padrões da métrica *Cobertura de código* são a escrita de grandes métodos falsos (sem uma lógica útil para o sistema) que não causam problemas nos testes ou ainda a escrita de métodos de testes sem *asserts*, que "cobrem"várias partes do código fonte, porém sem realmente testá-las.

Em todos esses casos, a má interpretação das métricas pode causar uma falsa noção da realidade, o que pode conduzir o gerente à tomada de decisões erradas. Nesta dissertação,

os riscos envolvidos com a medição em projetos de desenvolvimento de software foram considerados, em virtude dos problemas que podem ser encontrados caso contrário. Além disso, a abordagem proposta pode guiar o gerente na identificação tanto desses riscos como de fatores que podem amenizá-los.

Capítulo 6

Conclusão

Nesta dissertação de mestrado, abordou-se a problemática relacionada à confiabilidade de métricas coletadas em projetos de desenvolvimento de software. As métricas coletadas nos projetos são afetadas por fontes de erro associadas com a medição, o que pode diminuir sua acurácia e confiança, causando problemas na interpretação do valor medido. Métricas, se interpretadas de maneira indevida, podem indicar que o projeto esteja em uma situação diferente da qual realmente está, o que impactaria na tomada de decisões incorretas por parte dos gerentes. Outras métricas, ainda, são coletadas apenas a pedido dos clientes e sequer são utilizadas no processo de tomada de decisão por parte dos gerentes.

A fim de amenizar esses problemas, foi apresentada uma abordagem para auxiliar gerentes (bem como times de desenvolvimento de software, como um todo) na interpretação de métricas coletadas em projetos, fornecendo uma análise mais objetiva, modelando com redes Bayesianas as métricas e os fatores subjetivos que exercem influência sobre estas.

A abordagem proposta na pesquisa pode ser dividida em quatro macro etapas: (i) identificação das métricas do projeto, na qual os gerentes devem identificar quais as métricas coletadas no seu projeto; (ii) agrupamento de métricas, na qual as métricas devem ser agrupadas de acordo com o seu escopo; (iii) construção de um GAD para cada grupo grupo de métricas, de forma a modelar os relacionamentos entre as métricas, os riscos associados com estas e os mitigadores de tais riscos; e (iv) definição das funções de probabilidades para cada um dos nós definidos na etapa anterior, concluindo a construção das redes Bayesianas. Ao fim dessas etapas, o gerente precisa apenas alimentar a rede com os valores coletados da métrica para obter uma recomendação do uso ou da desconsideração da métrica no processo

de tomada de decisões acerca do projeto.

A validação da abordagem foi feita por meio de um estudo de caso utilizando quatro projetos do laboratório *Embedded* como unidades de análise. Como discutido na Seção 4, a abordagem proposta se mostrou útil no auxílio à interpretação de métricas e o custo-benefício para aplicá-la em projetos de desenvolvimento de software foi considerado satisfatório pelos sujeitos utilizados na pesquisa (os gerentes dos projetos acompanhados), considerando o esforço exigido.

Como a validação da abordagem proposta nesta pesquisa foi realizada em quatro projetos com diferentes características, conclui-se que a sua aplicação é útil para projetos com atributos comuns aos acompanhados no decorrer do trabalho: utilização do processo de desenvolvimento ágil, com times de 5 a 12 integrantes e focados no desenvolvimento de aplicações para dispositivos móveis. No entanto, como a quantidade de projetos estudados foi relativamente baixa, não há como concluir que a abordagem proposta nesta pesquisa é útil para todos os projetos de desenvolvimento de software existentes, pois há uma incontável quantidade de diferentes perfis de projeto. Ainda assim, acredita-se que o objetivo geral, bem como os quatro objetivos específicos da pesquisa, foram atingidos.

6.1 Limitações

Apesar de se concluir que a abordagem apresentada nesta pesquisa é promissora no sentido de auxiliar a interpretação de métricas em projetos de desenvolvimento de software, existem alguns fatores limitantes envolvidos no trabalho. Como já mencionado, o número de projetos acompanhados não foi ideal e o tempo de coleta dos dados em cada um deles foi de quarenta e cinco dias, o que também pode não refletir os resultados para a duração total dos projetos. Entretanto, buscou-se coletar dados em três intervalos, não necessariamente consecutivos, de quinze dias para minimizar o risco de selecionar um período anormal do projeto.

Uma outra limitação da pesquisa é o fato da necessidade do pesquisador estar envolvido em algumas partes do processo, para explicar a abordagem aos gerentes, reunir-se com estes durante o processo de identificação e agrupamento das métricas, construir manualmente as redes Bayesianas e auxiliar os gerentes a alimentar a rede e interpretar as saídas. Como discutido na Seção 6.2, existe a ideia da criar ferramentas para auxiliar os gerentes em cada

uma dessas etapas, assim como automatizar o processo da criação das redes Bayesianas, preenchimento das suas entradas e visualização da recomendação obtida como saídas das redes.

6.2 Trabalhos Futuros

Como trabalhos futuros, pode-se citar o desenvolvimento de uma ferramenta *Open Source* para facilitar o uso da abordagem proposta nesta pesquisa. Atualmente, o processo de construção das Redes *Bayesianas*, que modelam os projetos de desenvolvimento de software que adotam a abordagem proposta, acontece por meio do uso da ferramenta *AgenaRisk*, que exige o pagamento de uma licença anual. Além disso, a utilização do *AgenaRisk* pode ser muito complicada para um usuário comum. Portanto, a construção de uma ferramenta intuitiva, na qual os gerentes possam facilmente identificar as métricas, riscos e fatores controladores dos seus projetos e responder os questionários referentes à distribuição de peso da funções de probabilidade, se apresenta como uma ideia bastante válida.

Idealmente, a ferramenta supracitada poderia receber os dados do gerente e, de forma transparente para o mesmo, construir as Redes *Bayesianas* correspondentes aos projetos. Isso pode ser feito com o desenvolvimento de uma extensão para o *SMILE* (*Structural Modeling, Inference, and Learning Engine*) [27] que dê suporte a nós ranqueados. *SMILE* é uma biblioteca de classes em C++ que implementam métodos gráficos de apoio à decisão, como diagramas de influência e Redes *Bayesianas*. Além disso, a biblioteca *SMILE* permite que os dados sejam exportados de maneira simples para o ambiente gráfico *GeNIe*. A ferramenta construída poderia ser integrada a diversas soluções *Open Source* para o gerenciamento do processo e de artefatos produzidos em projetos de desenvolvimento de software, tais como *Taiga* [1], *Jenkins* [2], *TRAC* [3] e *GitLab* [4] (este último utilizado em mais de 100.000 organizações, incluindo grandes empresas como NASA e IBM). O caráter *Open Source* permitiria que diversos desenvolvedores espalhados pelo mundo pudessem contribuir de forma a deixar a ferramenta mais genérica, dando suporte a uma maior diversidade de projetos.

[1] http://taiga.io
[2] http://jenkins-ci.org
[3] http://trac.edgewall.org
[4] http://about.gitlab.com

Além da possibilidade do desenvolvimento da ferramenta sugerida anteriormente, espera-se que, com a publicação deste estudo, outros pesquisadores sintam-se motivados a aplicar a abordagem proposta em diversos outros projetos, possivelmente por uma duração maior de tempo, permitindo que seja feita uma avaliação da aplicação da abordagem a longo prazo. Isso aprimoraria a confiança na pesquisa, visto que o número de projetos acompanhados no decorrer desta e o tempo total de acompanhamento, conforme discutido na Seção 4.7, foram relativamente curtos.

Os modelos construídos por meio da utilização da abordagem proposta podem também ser integrados como componentes ao modelo construído para detectar problemas no processo de projetos de desenvolvimento de software que utilizam *Scrum*, apresentado por Perkusich et al. em [31] e [30]. Também é possível a utilização da abordagem para expandir o *Goal-Question-Metric* (GQM) [11], um método muito popular para usar as métricas de uma maneira orientada a objetivos. Montini et al. [6] propuseram uma extensão do GQM, aplicando Redes *Bayesianas* para a inserção de subjetividade nas métricas, demonstrando que pesquisas nesse âmbito podem se mostrar promissoras.

Finalmente, a calibração das funções de probabilidade pode ser otimizada. Algoritmos Genéticos, por exemplo, já se mostraram bastante efetivos em resolver diversas tarefas de otimização, podendo, inclusive, ser executados paralelamente na Unidade de Processamento Gráfico (UPG) do computador, apresentando um ganho de velocidade [9]. Também é válido analisar se o uso de dados históricos pode apresentar algum tipo de ganho no escopo da definição de funções de probabilidade.

Bibliografia

[1] Systems and software engineering – developing user documentation in an agile environment. *ISO/IEC/IEEE 26515 First edition 2011-12-01; Corrected version 2012-03-15*, pages 1–36, Março 2012.

[2] A. L. Beberg, D. L. Ensign, G. Jayachandran, S. Khaliq e V. S. Pande. Folding@home: Lessons from eight years of volunteer distributed computing. In *IPDPS*, pages 1–8. IEEE, 2009.

[3] B. Das. Generating conditional probabilities for bayesian networks: Easing the knowledge acquisition problem. *CoRR*, cs.AI/0411034, 2004.

[4] B. Kitchenham. What's up with software metrics? - a preliminary mapping study. *J. Syst. Softw.*, 83(1):37–51, January 2010.

[5] B. Spasic e B. S. S. Onggo. Agent-based simulation of the software development process: A case study at AVL. In *Proceedings of the Winter Simulation Conference*, WSC '12, pages 400:1–400:11. Winter Simulation Conference, 2012.

[6] D. A. Montini, F. R. M. Cardoso, F. S. Marcondes, P. M. Tasinaffo, L. A. V. Dias e A. M. da Cunha. Using GQM hypothesis restriction to infer bayesian network testing. In *Information Technology: New Generations, 2009. ITNG '09. Sixth International Conference on*, pages 1436–1441, Abril 2009.

[7] D. Heckerman. A tutorial on learning with bayesian networks. Technical report, Learning in Graphical Models, 1996.

[8] F. Lee, Y. Park e J. G. Shin. Large engineering project risk management using a bayesian belief network. *Expert Syst. Appl.*, 36(3):5880–5887, April 2009.

[9] F. M. Johar, F. A. Azmin, M. K. Suaidi, A. S. Shibghatullah, B. H. Ahmad, S. N. Salleh, M. Z. A. A. Aziz e M. M. Shukor. A review of genetic algorithms and parallel genetic algorithms on graphics processing unit (GPU). In *ICCSCE*, pages 264–269. IEEE, 2013.

[10] G. Çalıklı e A. B. Bener. Influence of confirmation biases of developers on software quality: An empirical study. *Software Quality Control*, 21(2):377–416, June 2013.

[11] H. Koziolek. Goal, question, metric. In *Dependability metrics*, pages 39–42. Springer, 2008.

[12] H. Ziv, D. J. Richardson. Constructing bayesian-network models of software testing and maintenance uncertainties. In *Proceedings of the International Conference on Software Maintenance*, ICSM '97, pages 100–, Washington, DC, USA, 1997. IEEE Computer Society.

[13] I. Ben-Gal. *Bayesian Networks*. John Wiley & Sons, Ltd, 2008.

[14] K. A. M. Ferreira, M. A. S. Bigonha, R. S. Bigonha, L. F. O. Mendes e H. C. Almeida. Identifying thresholds for object-oriented software metrics. *J. Syst. Softw.*, 85(2):244–257, February 2012.

[15] K. Schwaber e M. Beedle. *Agile Software Development with Scrum*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2001.

[16] K. V. J. Padmini, H. M. N. D. Bandara e I. Perera. Use of software metrics in agile software development process. In *Moratuwa Engineering Research Conference (MERCon), 2015*, pages 312–317, April 2015.

[17] L. E. Dunne, H. Profita, C. Zeagler, J. Clawson, S. Gilliland, E. Y.-L. Do e J. Budd. The social comfort of wearable technology and gestural interaction. In *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*, pages 4159–4162, Agosto 2014.

[18] L. G. Wallace e S. D. Sheetz. The adoption of software measures: A technology acceptance model (tam) perspective. *Inf. Manage.*, 51(2):249–259, March 2014.

[19] L. Uusitalo. Advantages and challenges of Bayesian networks in environmental modelling. *Ecological Modelling*, 203(3-4):312–318, May 2007.

[20] L. Zhang, X. Wu, L. Ding, M.J. Skibniewski e Y. Yan. Decision support analysis for safety control in complex project environments based on bayesian networks. *Expert Syst. Appl.*, 40(11):4273–4282, September 2013.

[21] Agena Ltd. Agenarisk 6.1. `http://www.agena.co.uk`.

[22] M. A. Ahmed e Z. Muzaffar. Handling imprecision and uncertainty in software development effort prediction: A type-2 fuzzy logic based framework. *Inf. Softw. Technol.*, 51(3):640–654, March 2009.

[23] M. A. Arbib. *The Handbook of Brain Theory and Neural Networks*. MIT Press, Cambridge, MA, USA, 1st edition, 1995.

[24] M. A. de Oliveira, O. Possamai, L. V. O. D. Valentina e C. A. Flesch. Applying bayesian networks to performance forecast of innovation projects: A case study of transformational leadership influence in organizations oriented by projects. *Expert Syst. Appl.*, 39(5):5061–5070, April 2012.

[25] M. Abouelela e L. Benedicenti. Bayesian network based XP process modelling. *CoRR*, abs/1007.5115, 2010.

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

[27] M. J. Druzdzel. SMILE: Structural modeling, inference, and learning engine and GeNIe: A development environment for graphical decision-theoretic models. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*, AAAI '99/IAAI '99, pages 902–903, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.

[28] M. Perkusich, A. Medeiros, L. Chaves, K. Gorgonio, H. O. de Almeida e A. Perkusich. A bayesian network approach to assist on the interpretation of software metrics. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. ACM, 2015.

[29] M. Perkusich, A. Perkusich e H. O. de Almeida. Using survey and weighted functions to generate node probability tables for bayesian networks. In *Proceedings of the 2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence*, BRICS-CCI-CBIC '13, pages 183–188, Ipojuca, Brazil, 2013. IEEE Computer Society.

[30] M. Perkusich, G. Soares, H. O. de Almeida e A. Perkusich. A procedure to detect problems of processes in software development projects using bayesian networks. *Expert Systems with Applications*, 42(1):437 – 450, 2015.

[31] M. Perkusich, H. O. de Almeida e A. Perkusich. A model to detect problems on scrum-based software development projects. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pages 1037–1042. ACM, 2013.

[32] M. Riaz, E. Mendes e E. Tempero. A systematic review of software maintainability prediction and metrics. In *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, ESEM '09, pages 367–377, Washington, DC, USA, 2009. IEEE Computer Society.

[33] M. Umarji e C. Seaman. Gauging acceptance of software metrics: Comparing perspectives of managers and developers. In *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, ESEM '09, pages 236–247, Washington, DC, USA, 2009. IEEE Computer Society.

[34] N. E. Fenton e M. Neil. Software metrics: Roadmap. In *Proceedings of the Conference on The Future of Software Engineering*, ICSE '00, pages 357–370, New York, NY, USA, 2000. ACM.

[35] N. E. Fenton, M. Neil e D. A. Lagnado. A general structure for legal arguments about evidence using bayesian networks. *Cognitive Science*, 37(1):61–102, 2013.

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

[37] N. Fenton and M. Neil. Software metrics and risk. In *2nd European Software Measurement Conference (FESMA 99)*, 1999.

[38] N. Fenton e J. Bieman. *Software Metrics: A Rigorous and Practical Approach, Third Edition*. CRC Press, Inc., Boca Raton, FL, USA, 3rd edition, 2014.

[39] N. Fenton e M. Neil. *Risk Assessment and Decision Analysis with Bayesian Networks*. CRC Press, 1 edition, 11 2012.

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

[41] S. Wagner. A bayesian network approach to assess and predict software quality using activity-based quality models. *Inf. Softw. Technol.*, 52(11):1230–1241, November 2010.

[42] Y. C. Cavalcanti, P. A. M. Silveira Neto, D. Lucrédio, T. Vale, E. S. Almeida e S. R. L. Meira. The bug report duplication problem: An exploratory study. *Software Quality Control*, 21(1):39–66, March 2013.

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

Apêndice A

Questionários Aplicados

Neste Apêndice, encontram-se todos os questionários respondidos pelos gerentes de projeto participantes desta pesquisa. Os questionários estão divididos primeiramente por projeto e em seguida pelos grupos de métricas correspondentes a cada projeto. Juntamente com os questionários, os gerentes receberam o seguinte texto, com o intuito de esclarecer eventuais dúvidas:

"Neste questionário, foram elencados alguns riscos ou fatores controladores/mitigadores presentes no seu projeto. Esses fatores, ou riscos, possuem alguns nós-pai, que podem influenciar o nó filho de alguma forma. Para cada um dos pais de um nó, você deve ordená-los, linha-a-linha, por ordem decrescente de importância, ou seja, os fatores que possuem mais influência no nó pai devem aparecer nas linhas superiores. Caso existam dois ou mais fatores com a mesma importância, estes devem ser colocados na mesma linha, separados por vírgula (,).

Adicionalmente, existe um modificador que pode, ou não, ser aplicado no cálculo da inflluência sobre cada nó. Esse modificador pode ser ***MAX*** *ou* ***MIN****. O modificador* ***MAX*** *deve ser aplicado se você acha que o nó deve apresentar uma tendência para resultados mais positivos se pelo menos um nó-pai apresentar resultado positivo. Por exemplo, suponha-se que três fatores para evitar o código duplicado (revisão de código, coleta de alertas análise estática e arquitetura bem definida) tenham sido elencados e você acha que se pelo menos um deles está sendo aplicado de forma satisfatória, o risco de ter código duplicado já deve ser baixo. Nesse caso, o modificador* ***MAX*** *deve ser utilizado.*

O modificador ***MIN*** *deve ser aplicado se você acredita que o nó tenha uma tendência*

para resultados mais negativos se pelo menos um fator-pai apresentar resultado negativo. Para exemplificar, presume-se que existam quatro fatores que influenciam a qualidade dos testes (qualidade dos testes de regressão, de usabilidade, funcionais e de unidade). Se a qualidade de um desses tipos de teste estiver ruim e você achar que isso deve implicar em uma qualidade geral dos testes igualmente ruim, o modificador ***MIN*** *deve ser utilizado".*

Nota-se que nem todos os grupos de métricas possuem um questionário associado. Isso ocorre porque determinados grupos não contêm nós com mais de um pai, fazendo com que a aplicação de um questionário não seja necessária.

A.1 Projeto A

A.1.1 Grupo de Métricas 1

PERGUNTAS
1) Quais os fatores que mais podem controlar e/ou mitigar a perícia dos relatores de *bugs*? • Descrição do processo de relatar *bugs* • Definição das especificações do produto
2) Quais os fatores que mais podem controlar e/ou mitigar a (má) adequação dos testadores? • Habilidade e experiência em testes • Perfil de testador

3) Quais os fatores de risco que mais podem contribuir para melhorar/piorar a qualidade dos relatores de *bugs*?

- Perícia dos relatores de *bugs*
- Adequação dos testadores
- Comunicação

4) Quais os fatores que mais podem contribuir para melhorar/piorar a qualidade dos relatórios de *bugs*?

- Qualidade dos relatores de *bugs*
- Disponibilidade de ferramentas e dispositivos

5) Quais os fatores que mais podem controlar e/ou mitigar a perícia do time de testes?

- Seleção do time
- Treinamento do time

6) Quais os fatores que mais podem controlar e/ou mitigar a qualidade do time de testes?

- Perícia do time de testes
- Comunicação
- Perfil de testador
- Disponibilidade

7) Quais tipos de teste mais colaboram para a (má) qualidade dos testes não funcionais?

- Testes de desempenho
- Testes de consumo de memória

8) Quais tipos de teste mais colaboram para a (má) qualidade dos testes funcionais?

- Testes de regressão
- Testes exploratórios
- Testes de aceitação
- Automatização dos casos de teste

9) Quais os fatores que mais podem controlar e/ou mitigar a qualidade da execução do processo de testes?

- Disponibilidade de ferramentas e dispositivos
- Qualidade dos testes funcionais
- Qualidade dos testes não funcionais

10) Quais os fatores que mais podem controlar e/ou mitigar o fato de os desenvolvedores escreverem métodos falsos para aumentar a cobertura de código?

- Penalidade bastante severa
- Revisão de código

11) Quais os fatores de risco que podem inserir erro na medição da cobertura de código?

- Priorização dos testes
- Implementação de métodos falsos

12) Quais os fatores que mais podem contribuir para melhorar/piorar o viés ao desenvolver testes?

- Experiência dos desenvolvedores
- Evitar que desenvolvedores testem o próprio código

13) Quais os fatores de risco que podem inserir erro na medição do número de *bugs*?

- Qualidade do processo de testes
- Qualidade dos relatórios de *bugs*
- Viés no desenvolvimento de testes

RESPOSTAS

Ordenação referente à pergunta 1:

1. Definição das especificações do produto
2. Descrição do processo de relatar *bugs*

Modificador utilizado: MIN

Ordenação referente à pergunta 2:

1. Habilidade e experiência em testes
2. Perfil de testador

Modificador utilizado: Nenhum

Ordenação referente à pergunta 3:

1. Perícia dos relatores de *bugs*
2. Comunicação
3. Adequação dos testadores

Modificador utilizado: Nenhum

Ordenação referente à pergunta 4:

1. Qualidade dos relatores de *bugs*
2. Disponibilidade de ferramentas e dispositivos

Modificador utilizado: MIN

Ordenação referente à pergunta 5:

1. Seleção do time
2. Treinamento do time

Modificador utilizado: Nenhum

Ordenação referente à pergunta 6:

1. Perícia do time de testes
2. Comunicação
3. Disponibilidade
4. Perfil de testador

Modificador utilizado: Nenhum

Ordenação referente à pergunta 7:

1. Testes de desempenho
2. Testes de consumo de memória

Modificador utilizado: MIN

Ordenação referente à pergunta 8:

1. Testes de aceitação
2. Automatização dos casos de teste
3. Testes de regressão
4. Testes exploratórios

Modificador utilizado: Nenhum

Ordenação referente à pergunta 9:

1. Disponibilidade de ferramentas e dispositivos
2. Qualidade dos testes funcionais
3. Qualidade dos testes não funcionais

Modificador utilizado: Nenhum

Ordenação referente à pergunta 10:

1. Revisão de código
2. Penalidade bastante severa

Modificador utilizado: Nenhum

Ordenação referente à pergunta 11: 1. Implementação de métodos falsos 2. Priorização dos testes Modificador utilizado: Nenhum
Ordenação referente à pergunta 12: 1. Experiência dos desenvolvedores 2. Evitar que desenvolvedores testem o próprio código Modificador utilizado: Nenhum
Ordenação referente à pergunta 13: 1. Qualidade do processo de testes 2. Qualidade dos relatórios de *bugs* 3. Viés no desenvolvimento de testes Modificador utilizado: Nenhum

Tabela A.1: Questionário referente ao primeiro grupo de métricas do projeto A.

A.1.2 Grupo de Métricas 3

PERGUNTAS
1) Quais os fatores que mais podem inserir erros na medição de alertas de análise estática? • Limitações de ferramenta • Falsos positivos

RESPOSTAS
Ordenação referente à pergunta 1: 1. Limitações de ferramenta 2. Falsos positivos Modificador utilizado: MAX

Tabela A.2: Questionário referente ao terceiro grupo de métricas do projeto A.

A.1.3 Grupo de Métricas 4

PERGUNTAS
1) Quais os fatores que mais podem melhorar/piorar a priorização das tarefas por parte dos desenvolvedores? • Definição do objetivo da *Sprint* • Perícia do time de desenvolvimento • Gerenciamento de tarefas
2) Quais os fatores de risco que mais podem inserir erro na medição do *Burndown* da *Sprint*? • Priorização de tarefas • Qualidade do processo • Qualidade da ferramenta

RESPOSTAS
Ordenação referente à pergunta 1: 1. Definição do objetivo da *Sprint* 2. Gerenciamento de tarefas 3. Perícia do time de desenvolvimento Modificador utilizado: Nenhum
Ordenação referente à pergunta 2: 1. Qualidade do processo 2. Priorização de tarefas 3. Qualidade da ferramenta Modificador utilizado: Nenhum

Tabela A.3: Questionário referente ao quarto grupo de métricas do projeto A.

A.2 Projeto B

A.2.1 Grupo de Métricas 1

PERGUNTAS
1) Quais os fatores que mais podem controlar e/ou mitigar a perícia dos relatores de *bugs*? • Descrição do processo de relatar *bugs* • Definição das especificações do produto

2) Quais os fatores que mais podem controlar e/ou mitigar a (má) adequação dos testadores?

- Habilidade em desenvolver testes
- Perfil de testador

3) Quais os fatores de risco que mais podem contribuir para melhorar/piorar a qualidade dos relatores de *bugs*?

- Perícia dos relatores de *bugs*
- Adequação dos testadores
- Comunicação

4) Quais os fatores que mais podem contribuir para melhorar/piorar a qualidade dos relatórios de *bugs*?

- Qualidade dos relatores de *bugs*
- Disponibilidade de ferramentas e dispositivos

5) Quais os fatores que mais podem controlar e/ou mitigar a perícia do time de testes?

- Seleção do time
- Treinamento do time

6) Quais os fatores que mais podem controlar e/ou mitigar a qualidade do time de testes?

- Perícia do time de testes
- Comunicação

7) Quais os fatores que mais podem melhorar/piorar a qualidade da execução do processo de testes?

- Viés ao desenvolver o teste
- Testes exploratórios
- Testes de aceitação

8) Quais tipos de teste mais colaboram para a (má) qualidade do processo de testes?

- Qualidade da execução do processo de testes
- Disponibilidade de ferramentas e/ou dispositivos
- Qualidade do time de testes

9) Quais os fatores de risco que podem inserir erro na medição do número de *bugs*?

- Qualidade do processo de testes
- Qualidade dos relatórios de *bugs*

RESPOSTAS
Ordenação referente à pergunta 1: 1. Definição das especificações do produto 2. Descrição do processo de relatar *bugs* Modificador utilizado: MIN
Ordenação referente à pergunta 2: 1. Habilidade em desenvolver testes 2. Perfil de testador Modificador utilizado: MIN
Ordenação referente à pergunta 3: 1. Perícia dos relatores de *bugs* 2. Comunicação 3. Adequação dos testadores Modificador utilizado: MIN
Ordenação referente à pergunta 4: 1. Qualidade dos relatores de *bugs* 2. Disponibilidade de ferramentas e dispositivos Modificador utilizado: MIN
Ordenação referente à pergunta 5: 1. Seleção do time 2. Treinamento do time Modificador utilizado: MIN

Ordenação referente à pergunta 6: 1. Perícia do time de testes 2. Comunicação Modificador utilizado: MIN
Ordenação referente à pergunta 7: 1. Viés ao desenvolver o teste, Testes exploratórios 2. Testes de aceitação Modificador utilizado: MIN
Ordenação referente à pergunta 8: 1. Qualidade do time de testes 2. Qualidade da execução do processo de testes 3. Disponibilidade de ferramentas e/ou dispositivos Modificador utilizado: MIN
Ordenação referente à pergunta 9: 1. Qualidade dos relatórios de *bugs* 2. Qualidade do processo de testes Modificador utilizado: MIN

Tabela A.4: Questionário referente ao primeiro grupo de métricas do projeto B.

A.2.2 Grupo de Métricas 2

PERGUNTAS
1) Quais os fatores que mais podem controlar e/ou mitigar o risco inserido por falsos positivos na análise estática? • Qualidade da ferramenta utilizada • Configuração da ferramenta
2) Quais os fatores de risco que mais podem inserir erro na medição do número de alertas de análise estática? • Alertas silenciados • Falsos positivos
RESPOSTAS
Ordenação referente à pergunta 1: 1. Configuração da ferramenta 2. Qualidade da ferramenta utilizada Modificador utilizado: Nenhum
Ordenação referente à pergunta 2: 1. Alertas silenciados 2. Falsos positivos Modificador utilizado: MIN

Tabela A.5: Questionário referente ao segundo grupo de métricas do projeto B.

A.2.3 Grupo de Métricas 3

PERGUNTAS
1) Quais os fatores que mais podem melhorar/piorar a perícia da equipe de desenvolvimento? • Experiência • Treinamento em *Scrum*
2) Quais os fatores que mais podem controlar e/ou mitigar a (má) priorização de tarefas? • Perícia da equipe • Definição do objetivo da *Sprint*
3) Quais os fatores de risco que mais podem inserir erro na medição do *Burndown* da *Sprint*? • (Má) Qualidade do processo de desenvolvimento • (Inadequada) Priorização de tarefas
RESPOSTAS
Ordenação referente à pergunta 1: 1. Experiência 2. Treinamento em *Scrum* Modificador utilizado: MIN

Ordenação referente à pergunta 2: 1. Perícia da equipe 2. Definição do objetivo da *Sprint* Modificador utilizado: MIN
Ordenação referente à pergunta 3: 1. (Má) Qualidade do processo de desenvolvimento, (Inadequada) Priorização de tarefas Modificador utilizado: MIN

Tabela A.6: Questionário referente ao terceiro grupo de métricas do projeto B.

A.3 Projeto C

A.3.1 Grupo de Métricas 1

PERGUNTAS
1) Quais os fatores que mais podem controlar e/ou mitigar a perícia dos relatores de *bugs*? • Descrição do processo de relatar *bugs* • Definição das especificações do produto

2) Quais os fatores que mais podem controlar e/ou mitigar a colaboração dos relatores de *bugs*?

- Comunicação
- Distribuição do time
- Disponibilidade

3) Quais os fatores que mais podem controlar e/ou mitigar a experiência dos relatores de *bugs*?

- Habilidade em testes
- Familiaridade com as tecnologias de relatório de *bugs*

4) Quais os fatores que mais podem controlar e/ou mitigar a qualidade dos relatores de *bugs*?

- Perícia dos relatores de *bugs*
- Colaboração dos relatores de *bugs*
- Experiência dos relatores de *bugs*

5) Quais os fatores que mais podem controlar e/ou mitigar a qualidade dos relatórios de *bugs*?

- Qualidade dos relatores de *bugs*
- Disponibilidade de ferramentas e dispositivos

6) Quais os fatores que mais podem controlar e/ou mitigar a perícia do time de testes?

- Seleção do time
- Treinamento

7) Quais os fatores que mais podem controlar e/ou mitigar a experiência do time de testes?

- Habilidade em testes
- Familiaridade com as tecnologia de testes

8) Quais os fatores que mais podem contribuir para melhorar/piorar a colaboração do time de testes?

- Comunicação
- Distribuição do time
- Disponibilidade

9) Quais os fatores que mais podem controlar e/ou mitigar a qualidade do time de testes?

- Perícia do time de testes
- Experiência do time de testes
- Colaboração do time de testes

10) Quais tipos de teste mais colaboram para a (má) qualidade dos testes não funcionais?

- Testes de desempenho
- Testes de consumo de memória
- Testes de estresse
- Testes de carga

11) Quais tipos de teste mais colaboram para a (má) qualidade dos testes funcionais?

- Testes de aceitação
- Testes exploratórios

12) Quais os fatores que mais podem contribuir para melhorar/piorar a (má) qualidade geral dos testes?

- Qualidade dos testes fim a fim
- Qualidade da execução dos testes funcionais
- Qualidade da execução dos testes não funcionais

13) Quais os fatores que mais podem contribuir para melhorar/piorar a (má) qualidade da execução do processo de testes?

- Qualidade dos testes
- Disponibilidade de dispositivos e ferramentas
- Ferramentas de rastreabilidade

14) Quais os fatores que mais podem controlar e/ou mitigar o fato de os desenvolvedores escreverem métodos falsos para aumentar a cobertura de código?

- Penalidade bastante severa
- Revisão de código
- Comprometimento do time

15) Quais os fatores que mais podem contribuir para melhorar/piorar a (inadequada) priorização dos testes?

- Qualidade dos requisitos
- Cumprimento dos requisitos

16) Quais os fatores de risco que podem inserir erro na medição da cobertura de código?

- Priorização dos testes
- Implementação de métodos falsos
- Número de testes manuais

17) Quais os fatores de risco que podem inserir erro na medição do *status* dos testes?

- Comprometimento do time de testes
- Especificação dos testes fim a fim

18) Quais os fatores que mais podem contribuir para melhorar/piorar a qualidade do processo de testes?

- Interpretação do número de testes
- Interpretação do *status* dos testes
- Interpretação da cobertura de código
- Qualidade da execução do processo de testes

19) Quais os fatores de risco que podem inserir erro na medição do número de *bugs*?

- Qualidade do processo de testes
- Qualidade dos relatórios de *bugs*
- Qualidade do time de testes

RESPOSTAS

Ordenação referente à pergunta 1:

1. Definição das especificações do produto
2. Descrição do processo de relatar *bugs*

Modificador utilizado: Nenhum

Ordenação referente à pergunta 2:

1. Comunicação
2. Disponibilidade
3. Distribuição do time

Modificador utilizado: Nenhum

Ordenação referente à pergunta 3:

1. Habilidade em testes, Familiaridade com as tecnologias de relatório de *bugs*

Modificador utilizado: Nenhum

Ordenação referente à pergunta 4:

1. Perícia dos relatores de *bugs*
2. Experiência dos relatores de *bugs*
3. Colaboração dos relatores de *bugs*

Modificador utilizado: Nenhum

Ordenação referente à pergunta 5:

1. Qualidade dos relatores de *bugs*
2. Disponibilidade de ferramentas e dispositivos

Modificador utilizado: Nenhum

Ordenação referente à pergunta 6:

1. Seleção do time
2. Treinamento

Modificador utilizado: Nenhum

Ordenação referente à pergunta 7:

1. Habilidade em testes, Familiaridade com as tecnologia de testes

Modificador utilizado: Nenhum

Ordenação referente à pergunta 8:

1. Comunicação
2. Disponibilidade
3. Distribuição do time

Modificador utilizado: Nenhum

Ordenação referente à pergunta 9:

1. Perícia do time de testes
2. Experiência do time de testes
3. Colaboração do time de testes

Modificador utilizado: Nenhum

Ordenação referente à pergunta 10:

1. Testes de desempenho
2. Testes de consumo de memória, Testes de carga
3. Testes de estresse

Modificador utilizado: Nenhum

Ordenação referente à pergunta 11:

1. Testes exploratórios
2. Testes de aceitação

Modificador utilizado: Nenhum

Ordenação referente à pergunta 12:

1. Qualidade dos testes fim a fim
2. Qualidade da execução dos testes funcionais
3. Qualidade da execução dos testes não funcionais

Modificador utilizado: Nenhum

Ordenação referente à pergunta 13:

1. Qualidade dos testes
2. Disponibilidade de dispositivos e ferramentas, Ferramentas de rastreabilidade

Modificador utilizado: Nenhum

Ordenação referente à pergunta 14:

1. Comprometimento do time
2. Revisão de código
3. Penalidade bastante severa

Modificador utilizado: Nenhum

Ordenação referente à pergunta 15:

1. Qualidade dos requisitos
2. Cumprimento dos requisitos

Modificador utilizado: Nenhum

Ordenação referente à pergunta 16:

1. Priorização dos testes, Número de testes manuais
2. Implementação de métodos falsos

Modificador utilizado: Nenhum

Ordenação referente à pergunta 17: 1. Comprometimento do time de testes, Especificação dos testes fim a fim Modificador utilizado: Nenhum
Ordenação referente à pergunta 18: 1. Qualidade da execução do processo de testes 2. Interpretação do *status* dos testes 3. Interpretação do número de testes 4. Interpretação da cobertura de código Modificador utilizado: Nenhum
Ordenação referente à pergunta 19: 1. Qualidade do processo de testes 2. Qualidade dos relatórios de *bugs*, Qualidade do time de testes Modificador utilizado: Nenhum

Tabela A.7: Questionário referente ao primeiro grupo de métricas do projeto C.

A.3.2 Grupo de Métricas 2

PERGUNTAS
1) Quais os fatores de risco que podem inserir erro na medição do número de alertas de Javadoc? • Análise de falsos positivos • Análise da semântica da documentação

RESPOSTAS
Ordenação referente à pergunta 1: 1. Análise da semântica da documentação 2. Análise de falsos positivos Modificador utilizado: Nenhum

Tabela A.8: Questionário referente ao segundo grupo de métricas do projeto C.

A.3.3 Grupo de Métricas 3

PERGUNTAS
1) Quais os fatores que mais podem controlar e/ou mitigar o risco inserido por falsos positivos na análise estática? • Qualidade da ferramenta utilizada • Configuração da ferramenta
RESPOSTAS
Ordenação referente à pergunta 1: 1. Qualidade da ferramenta utilizada, Configuração da ferramenta Modificador utilizado: Nenhum

Tabela A.9: Questionário referente ao terceiro grupo de métricas do projeto C.

A.4 Projeto D

A.4.1 Grupo de Métricas 1

PERGUNTAS
1) Quais os fatores que mais podem controlar e/ou mitigar a (má) qualidade dos relatores de *bugs*? • Revisar e reproduzir *bugs* reportados • Uso de ferramentas de relatório de *bugs*
2) Quais os fatores que mais podem controlar e/ou mitigar a (má) qualidade do processo de relatar *bugs*? • Uso de ferramentas de relatório de *bugs* • Definição do processo de relatar *bugs*
3) Quais os fatores de risco que mais podem contribuir para melhorar/piorar a qualidade dos relatórios de *bugs*? • (Má) Qualidade do processo de relatar *bugs* • (Má) Qualidade dos relatores de *bugs*
4) Quais os fatores que mais podem controlar e/ou mitigar o viés no desenvolvimento dos testes? • Definir casos de teste antes do desenvolvimento • Evitar que desenvolvedores testem seu próprio código

5) Quais os fatores que mais podem controlar e/ou mitigar a qualidade dos testes?

- Sempre atualizar o conjunto de testes (criar novo teste ao consertar um *bug* interessante, remover testes obsoletos, etc.)
- Realizar testes de regressão

6) Quais os fatores que mais podem controlar e/ou mitigar a perícia do time de testes?

- Seleção do time
- Treinamento e estudo

7) Quais os fatores que mais podem controlar e/ou mitigar o tamanho (inapropriado) do time de testes?

- Definição de um líder
- Gerenciamento do time

8) Quais os fatores que mais podem contribuir para melhorar/piorar a qualidade do time de testes?

- Perícia do time
- Tamanho do time

9) Quais os fatores que mais podem controlar e/ou mitigar a implementação de métodos falsos para o aumento da cobertura de código?

- Treinamento do time
- Revisão de código em pares

10) Quais os fatores que mais podem controlar e/ou mitigar o fato de os desenvolvedores testarem apenas comportamentos já conhecidos?

- Planejar e discutir os testes antes do desenvolvimento
- Treinamento da equipe

11) Quais os fatores de risco que podem inserir erro na medição da cobertura de código?

- Implementação de métodos falsos
- Desenvolvedores testarem apenas comportamentos já conhecidos

12) Quais os fatores que mais podem contribuir para melhorar/piorar a qualidade do processo de testes?

- Interpretação da cobertura de código
- Qualidade dos testes
- Disponibilidade de dispositivos e/ou ferramentas
- Qualidade do time de testes

13) Quais os fatores de risco que podem inserir erro na medição do número de *bugs*?

- Viés no desenvolvimento de testes
- Qualidade do processo de testes
- Qualidade dos relatórios de *bugs*

RESPOSTAS
Ordenação referente à pergunta 1: 1. Revisar e reproduzir *bugs* reportados 2. Uso de ferramentas de relatório de *bugs* Modificador utilizado: MIN
Ordenação referente à pergunta 2: 1. Definição do processo de relatar *bugs* 2. Uso de ferramentas de relatório de *bugs* Modificador utilizado: MIN
Ordenação referente à pergunta 3: 1. (Má) Qualidade do processo de relatar *bugs* 2. (Má) Qualidade dos relatores de *bugs* Modificador utilizado: Nenhum
Ordenação referente à pergunta 4: 1. Definir casos de teste antes do desenvolvimento 2. Evitar que desenvolvedores testem seu próprio código Modificador utilizado: MAX
Ordenação referente à pergunta 5: 1. Sempre atualizar o conjunto de testes (criar novo teste ao consertar um *bug* interessante, remover testes obsoletos, etc.) 2. Realizar testes de regressão Modificador utilizado: MIN

Ordenação referente à pergunta 6:

1. Treinamento e estudo
2. Seleção do time

Modificador utilizado: MIN

Ordenação referente à pergunta 7:

1. Definição de um líder
2. Gerenciamento do time

Modificador utilizado: Nenhum

Ordenação referente à pergunta 8:

1. Perícia do time
2. Tamanho do time

Modificador utilizado: Nenhum

Ordenação referente à pergunta 9:

1. Treinamento do time
2. Revisão de código em pares

Modificador utilizado: Nenhum

Ordenação referente à pergunta 10:

1. Planejar e discutir os testes antes do desenvolvimento
2. Treinamento da equipe

Modificador utilizado: MAX

Ordenação referente à pergunta 11: 1. Implementação de métodos falsos 2. Desenvolvedores testarem apenas comportamentos já conhecidos Modificador utilizado: MIN
Ordenação referente à pergunta 12: 1. Qualidade dos testes 2. Disponibilidade de dispositivos e/ou ferramentas 3. Qualidade do time de testes 4. Interpretação da cobertura de código Modificador utilizado: MIN
Ordenação referente à pergunta 13: 1. Qualidade do processo de testes 2. Qualidade dos relatórios de *bugs* 3. Viés no desenvolvimento de testes Modificador utilizado: MIN

Tabela A.10: Questionário referente ao primeiro grupo de métricas do projeto D.

A.4.2 Grupo de Métricas 3

PERGUNTAS
1) Quais os fatores que mais podem inserir erros na medição de alertas de análise estática? • Limitações de ferramenta • Falsos positivos
RESPOSTAS
Ordenação referente à pergunta 1: 1. Limitações de ferramenta 2. Falsos positivos Modificador utilizado: MAX

Tabela A.11: Questionário referente ao terceiro grupo de métricas do projeto D.

A.4.3 Grupo de Métricas 4

PERGUNTAS
1) Quais os fatores que mais podem controlar e/ou mitigar a (má) qualidade do processo de desenvolvimento? • Reuniões diárias e planejamento da *Sprint* • Adoção de um cronograma bem detalhado

2) Quais os fatores que mais podem controlar e/ou mitigar a (má) definição do escopo da *Sprint*?

- Guiar o desenvolvimento pelo *blueprint* do produto
- Comunicação com o cliente

3) Quais os fatores de risco que mais podem inserir erro na medição do *Burndown* da *Sprint*?

- (Má) Definição do escopo da *Sprint*
- (Má) Qualidade do processo de desenvolvimento
- (Má) qualidade da ferramenta de medição

RESPOSTAS

Ordenação referente à pergunta 1:

1. Adoção de um cronograma bem detalhado
2. Reuniões diárias e planejamento da *Sprint*

Modificador utilizado: MIN

Ordenação referente à pergunta 2:

1. Guiar o desenvolvimento pelo *blueprint* do produto
2. Comunicação com o cliente

Modificador utilizado: MIN

Ordenação referente à pergunta 3:

1. (Má) Definição do escopo da *Sprint*
2. (Má) Qualidade do processo de desenvolvimento
3. (Má) qualidade da ferramenta de medição

Modificador utilizado: MIN

Tabela A.12: Questionário referente ao quarto grupo de métricas do projeto D.

Apêndice B

Probabilidades Calculadas

B.1 Projeto A

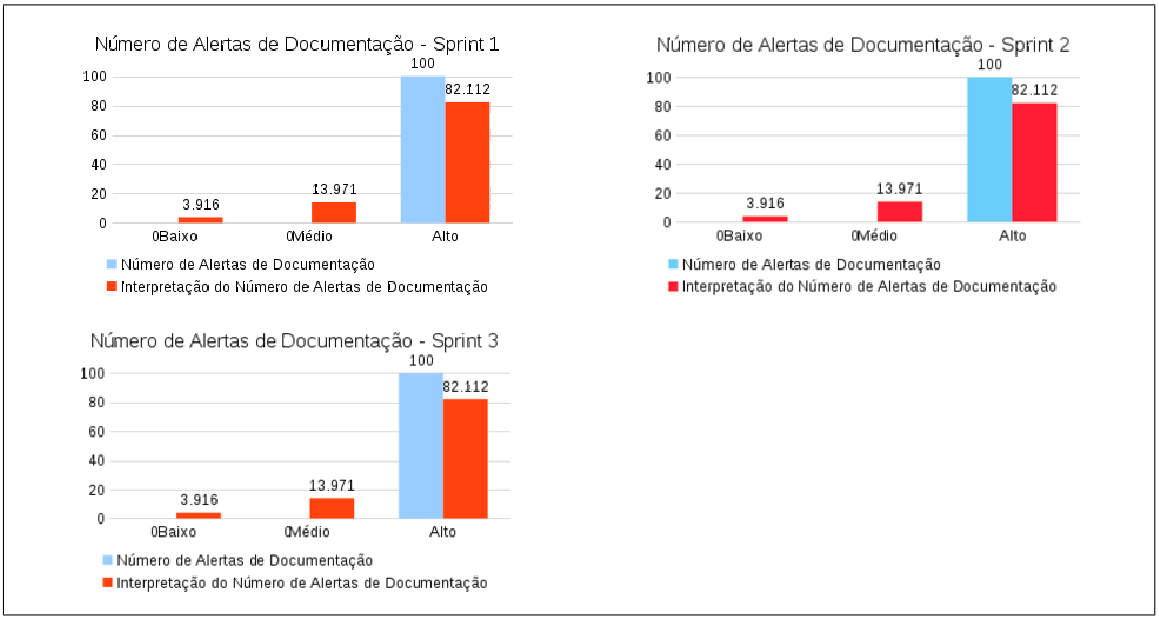

Figura B.1: Probabilidades calculadas para o grupo de métricas 2 do Projeto A

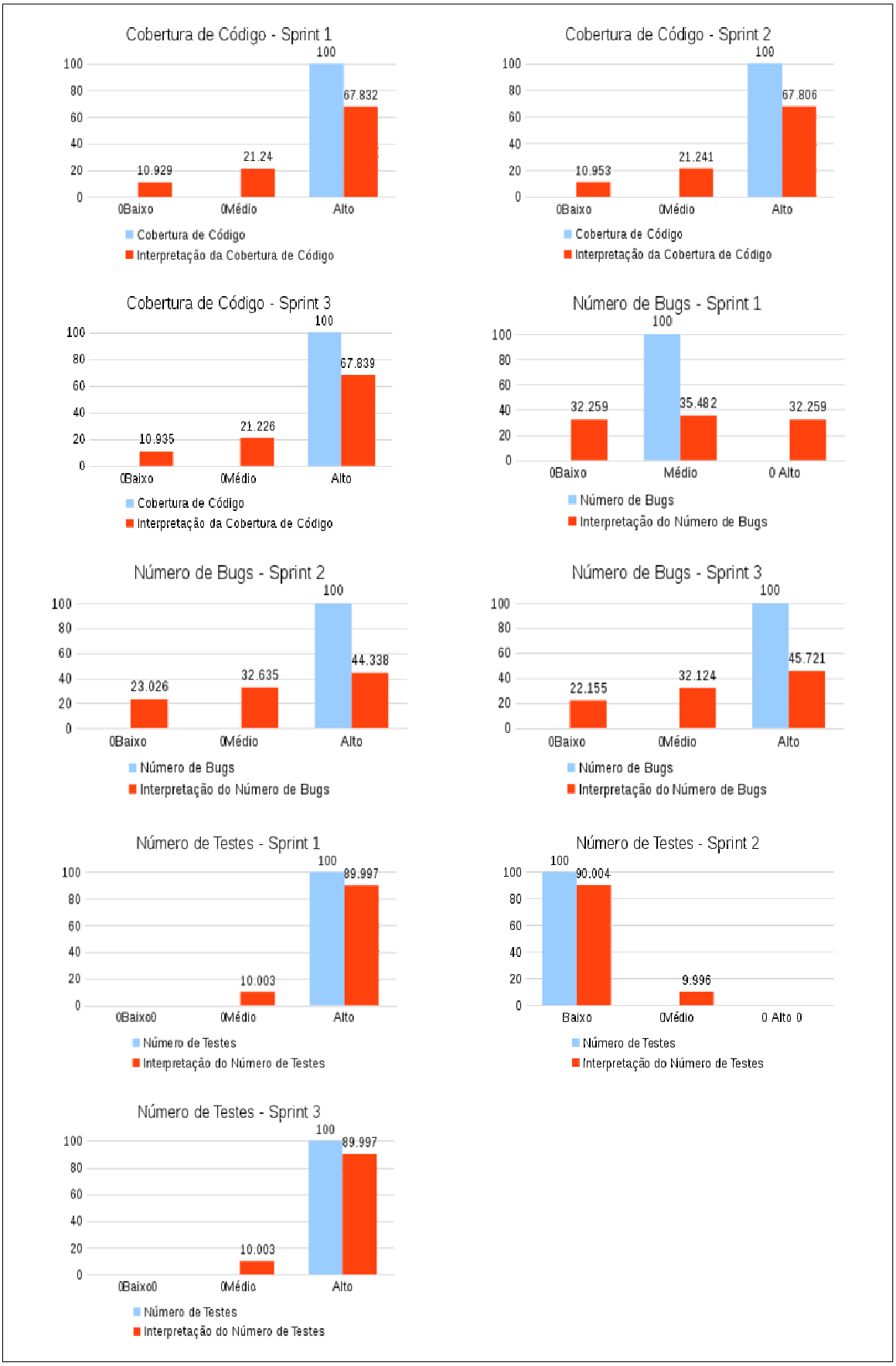

Figura B.2: Probabilidades calculadas para o grupo de métricas 1 do Projeto A

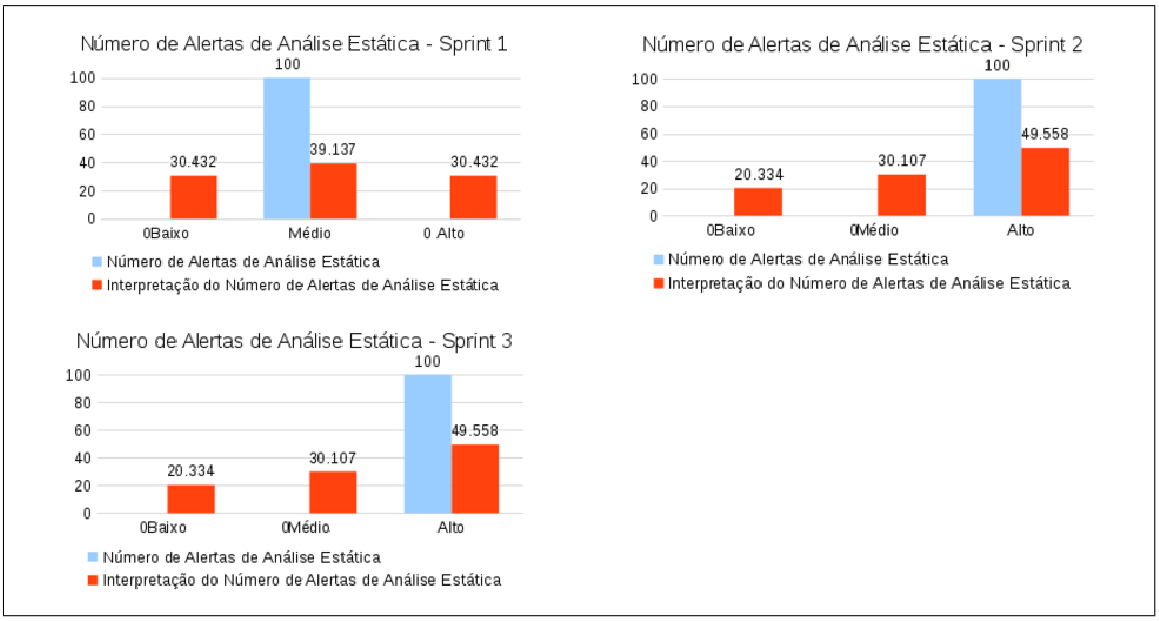

Figura B.3: Probabilidades calculadas para o grupo de métricas 3 do Projeto A

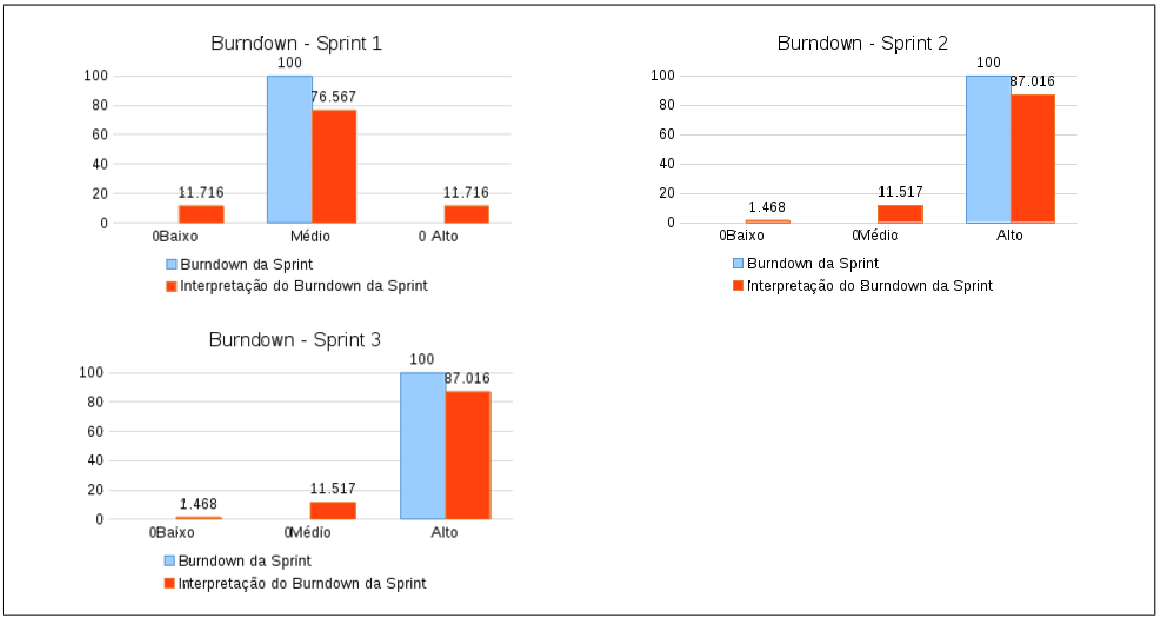

Figura B.4: Probabilidades calculadas para o grupo de métricas 4 do Projeto A

B.2 Projeto B

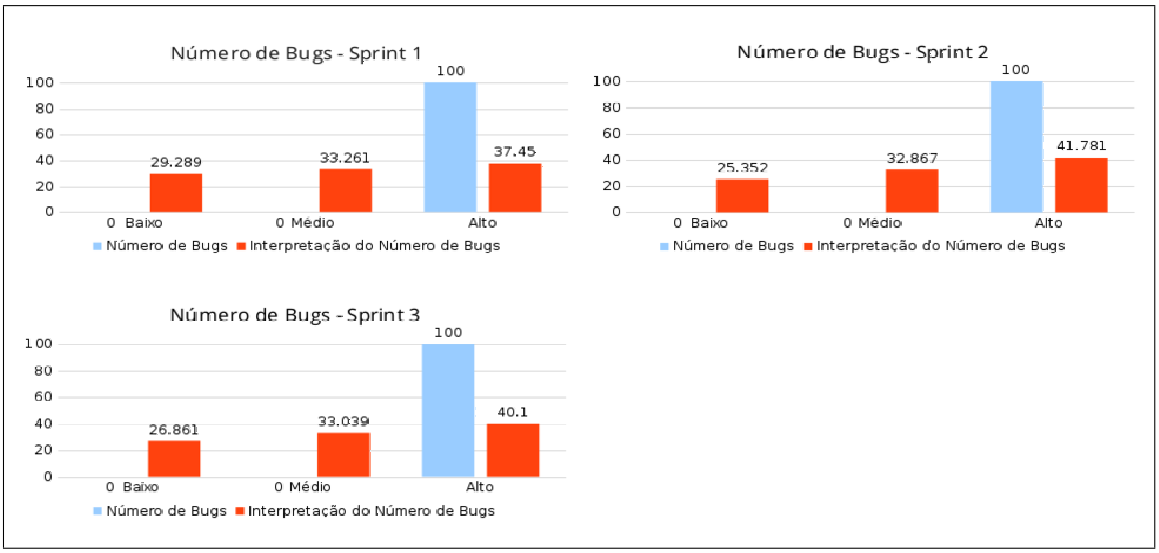

Figura B.5: Probabilidades calculadas para o grupo de métricas 1 do Projeto B

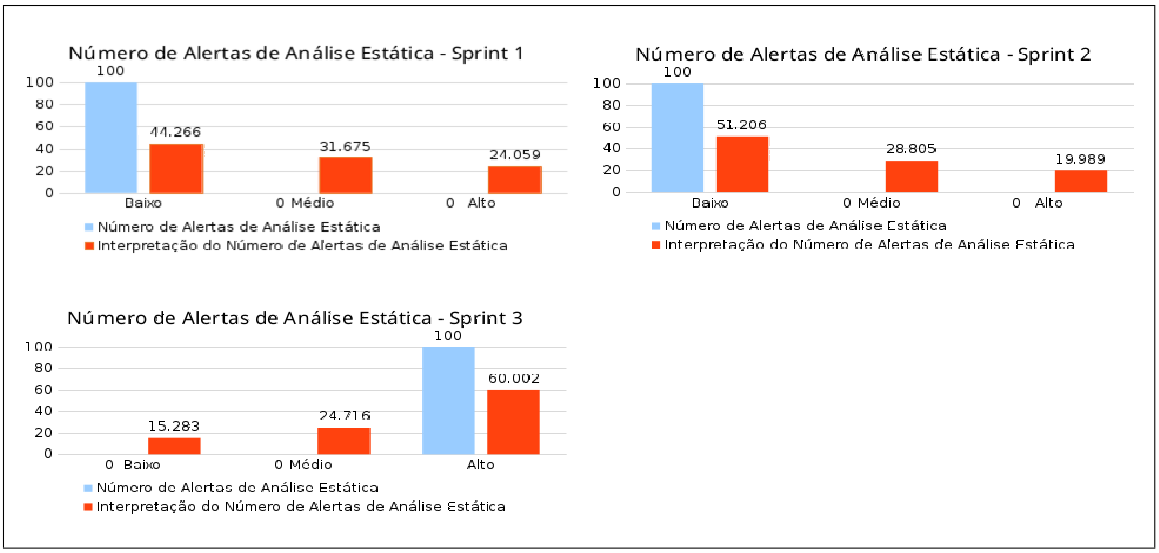

Figura B.6: Probabilidades calculadas para o grupo de métricas 2 do Projeto B

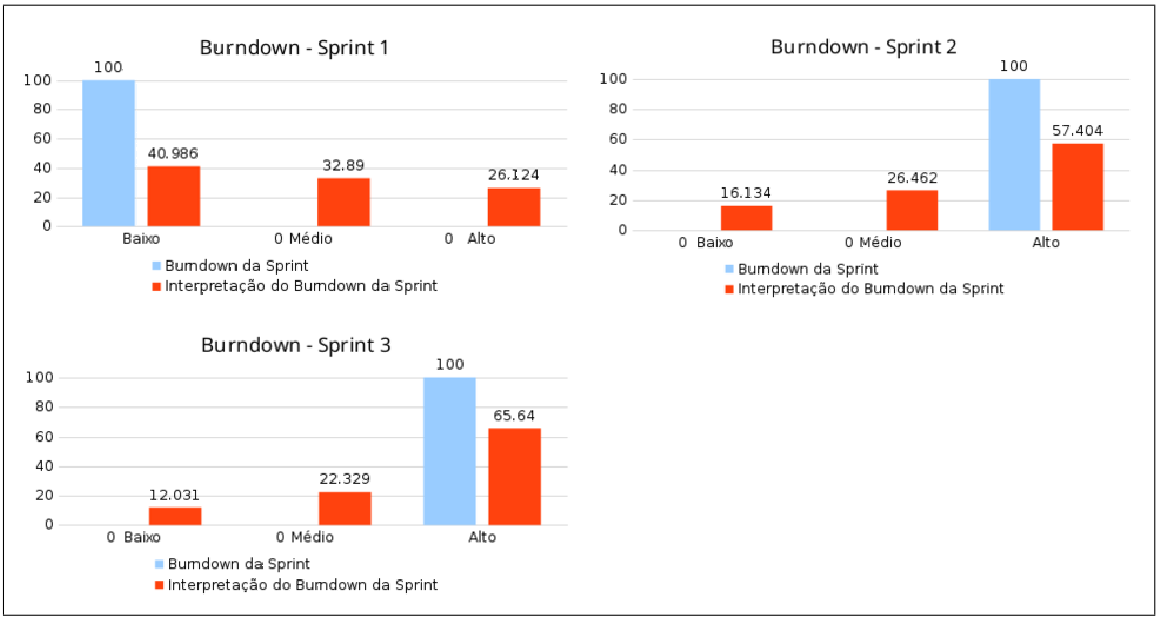

Figura B.7: Probabilidades calculadas para o grupo de métricas 3 do Projeto B

B.3 Projeto C

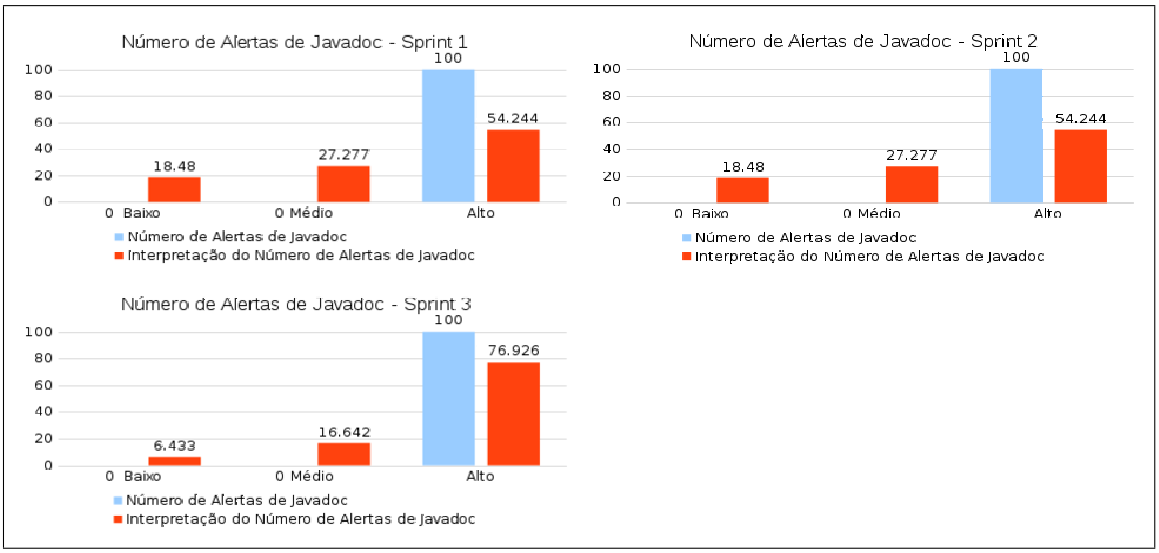

Figura B.8: Probabilidades calculadas para o grupo de métricas 2 do Projeto C

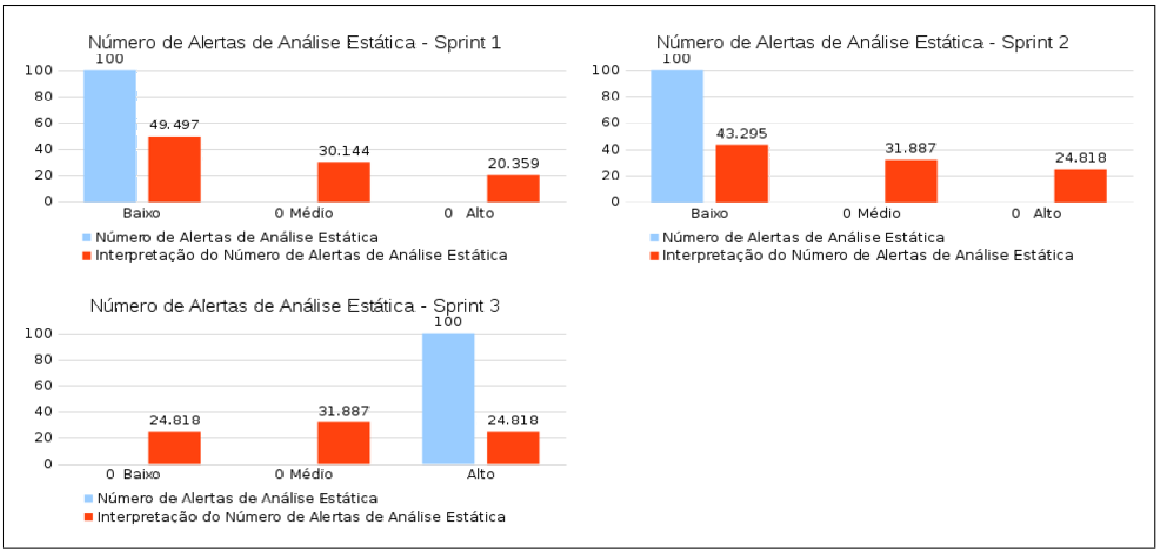

Figura B.9: Probabilidades calculadas para o grupo de métricas 3 do Projeto C

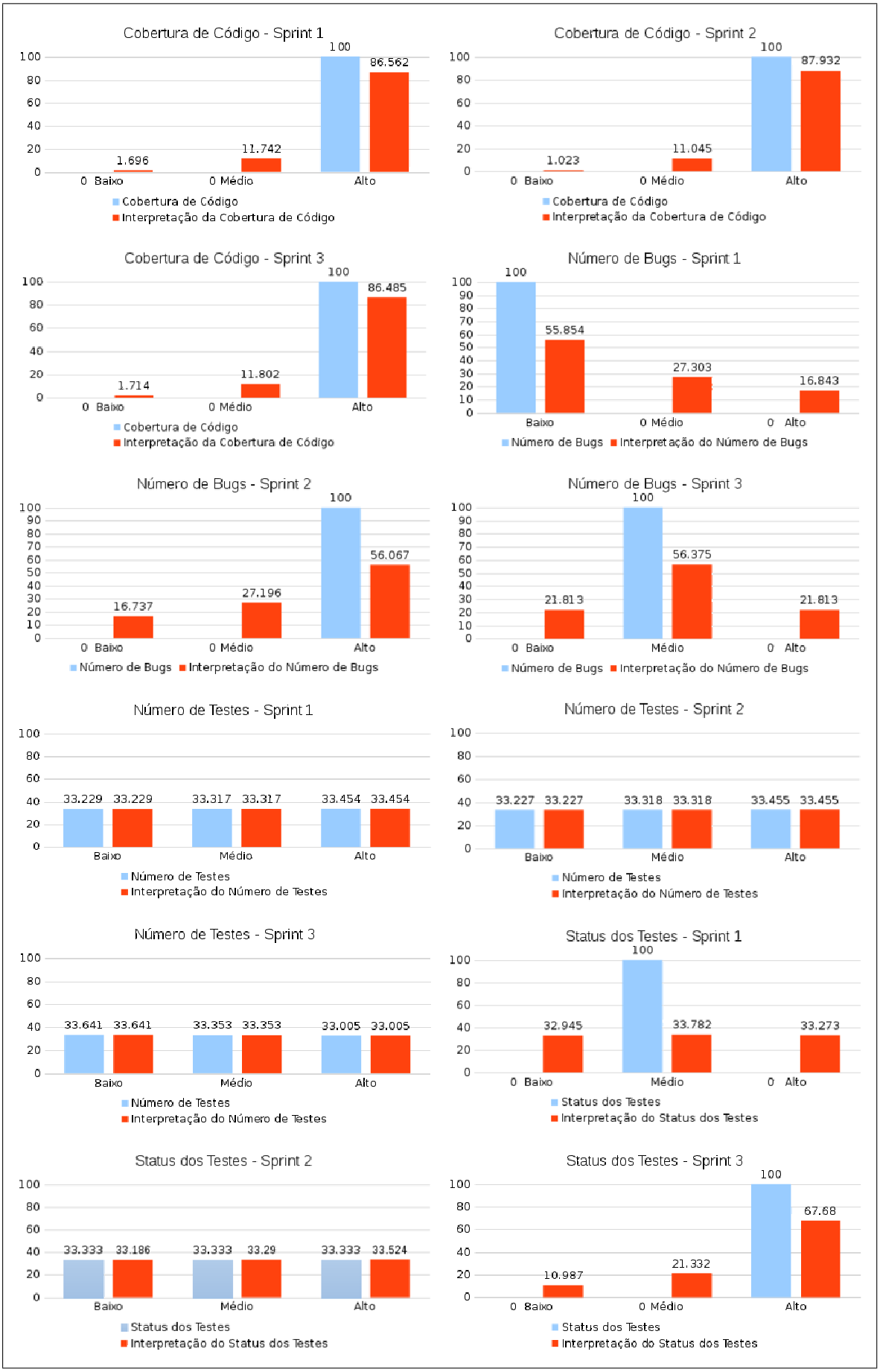

Figura B.10: Probabilidades calculadas para o grupo de métricas 1 do Projeto C

B.4 Projeto D

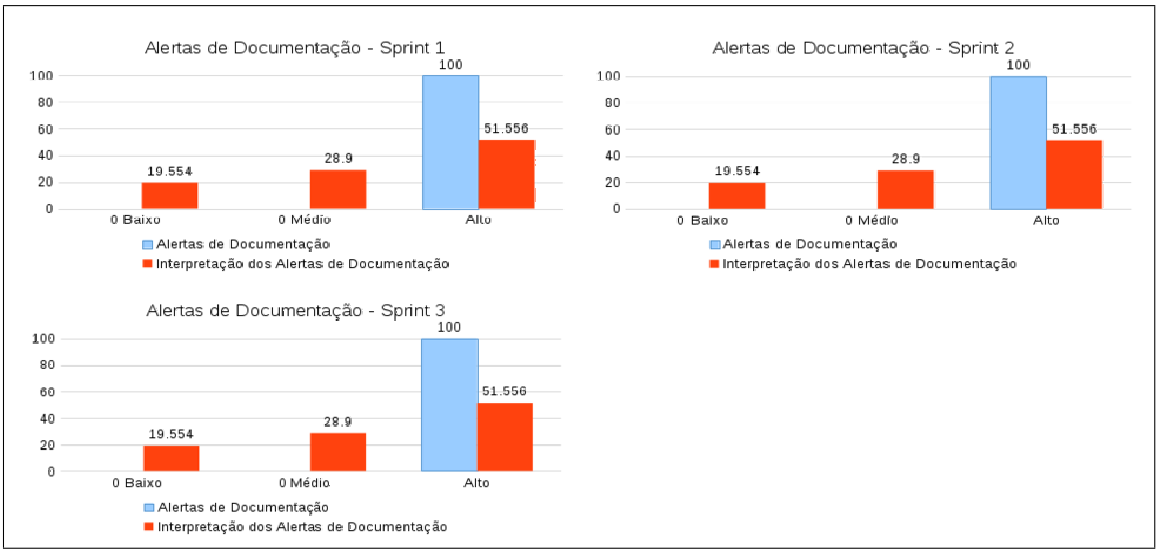

Figura B.11: Probabilidades calculadas para o grupo de métricas 2 do Projeto D

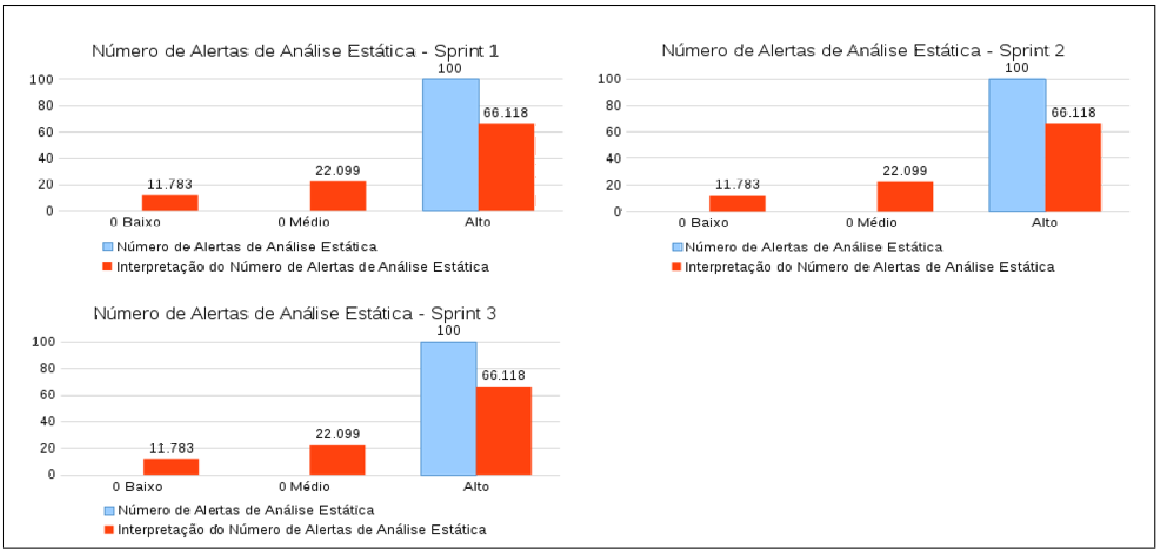

Figura B.12: Probabilidades calculadas para o grupo de métricas 3 do Projeto D

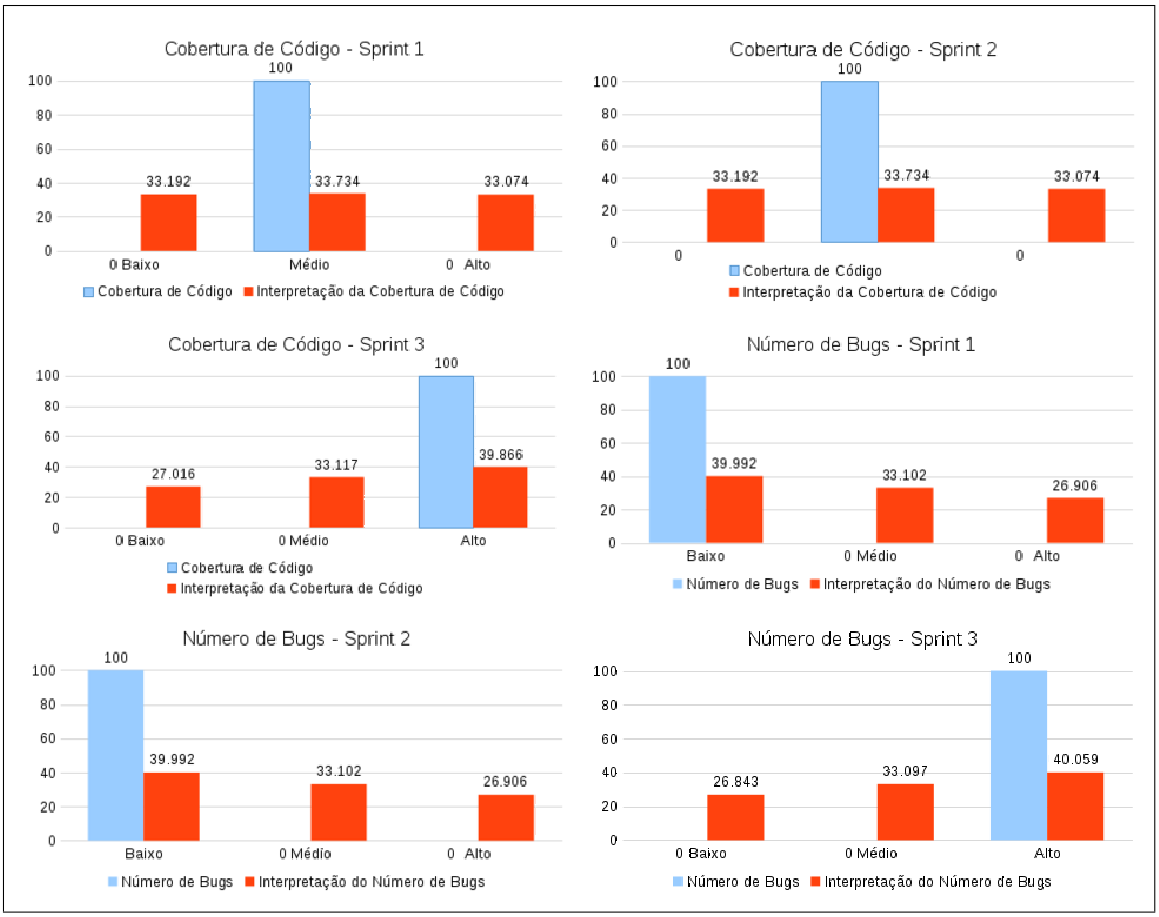

Figura B.13: Probabilidades calculadas para o grupo de métricas 1 do Projeto D

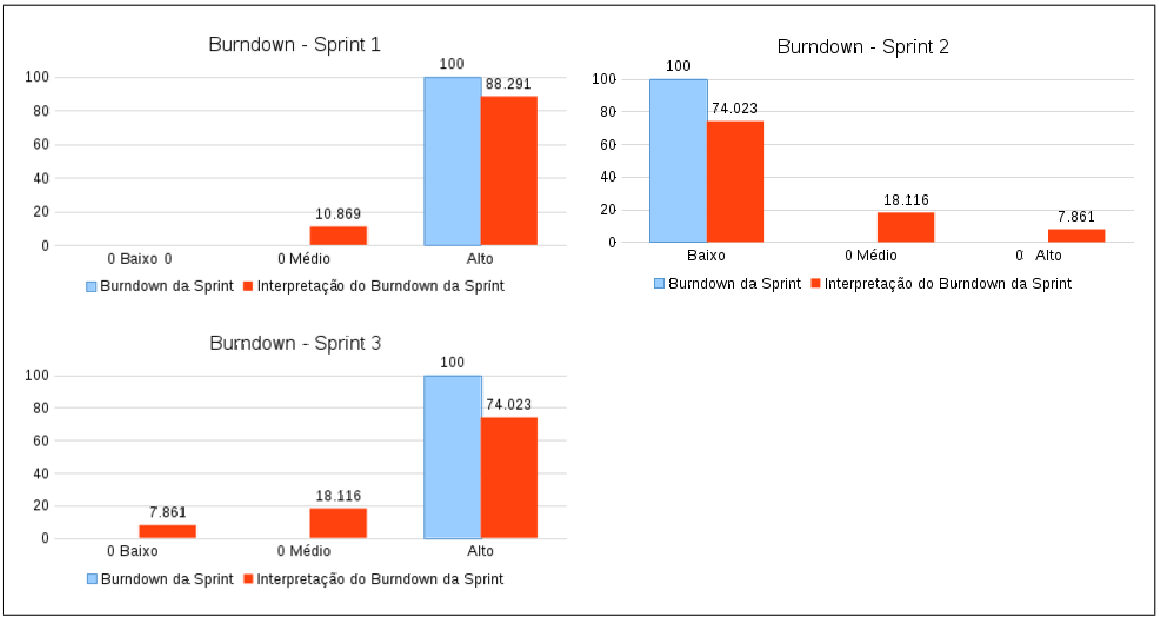

Figura B.14: Probabilidades calculadas para o grupo de métricas 4 do Projeto D