UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE CRATEÚS

CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

SAORI PEREIRA DA COSTA

GQM+PA: UM FRAMEWORK BASEADO EM GOAL-QUESTION-METRIC E PESQUISA-AÇÃO PARA ENSINO DE MEDIÇÃO DE SOFTWARE

CRATEÚS

2019

SAORI PEREIRA DA COSTA

GQM+PA: UM FRAMEWORK BASEADO EM GOAL-QUESTION-METRIC E PESQUISA-AÇÃO PARA ENSINO DE MEDIÇÃO DE SOFTWARE

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação do Campus de Crateús da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas de Informação.

Orientador: Prof. Me. Allysson Allex de Paula Araújo

CRATEÚS

2019

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

C875g Costa, Saori Pereira da.
GQM+PA: Um Framework baseado em Goal-Question-Metric e Pesquisa-Ação para Ensino de Medição de Software / Saori Pereira da Costa. – 2019.
85 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Crateús, Curso de Sistemas de Informação, Crateús, 2019.
Orientação: Prof. Me. Allysson Allex de Paula Araújo.

1. Qualidade de Software. 2. Medição de Software. 3. GQM. 4. Ensino. 5. Pesquisa-ação. I. Título.
CDD 005

SAORI PEREIRA DA COSTA

GQM+PA: UM FRAMEWORK BASEADO EM GOAL-QUESTION-METRIC E PESQUISA-AÇÃO PARA ENSINO DE MEDIÇÃO DE SOFTWARE

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação do Campus de Crateús da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas de Informação.

Aprovada em:

BANCA EXAMINADORA

Prof. Me. Allysson Allex de Paula Araújo (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Me. André Meireles de Andrade
Universidade Federal do Ceará (UFC)

Prof. Me. Nécio de Lima Veras
Instituto Federal do Ceará (IFCE)

AGRADECIMENTOS

Agradeço, primeiramente a minha família, em especial, Francisca Pereira Cardoso da Costa, Sirliane Pereira da Costa e Valdemir Pontes Costa por estarem sempre ao meu lado e acreditarem em mim.

Agradeço ao meu orientador Allysson Allex de Paula Araújo pela contribuição, compreensão e paciência na elaboração desta pesquisa.

Agradeço a todos o professores que contribuíram na minha formação.

Agradeço aos meus amigos que fizeram parte dessa jornada e me apoiaram, em especial, Ayrton Sousa Marinho pelo incentivo e apoio.

Por fim, agradeço a todos que contribuíram de alguma forma.

“Sua constelação sempre irá te proteger. Supera a dor e dá forças pra lutar.”

(Edu Falaschi)

RESUMO

A Engenharia de Software surgiu devido a necessidade de construir sistemas de software com alta qualidade e com mínimo de recursos. Dessa forma, identifica-se a Qualidade de Software como uma importante área da Engenharia de Software por, justamente, preocupar-se em assegurar que produtos e processos de software tenham qualidade. Para compreender melhor se um processo ou produto de software é de qualidade ou não, é necessário avaliá-lo. Uma prática recorrente para tal processo é a medição de software, a qual está intrinsecamente relacionada ao conceito de métricas de software. Entretanto, a definição de quais métricas devem ser avaliadas em um projeto de software demonstra-se uma atividade complexa, necessitando, assim, de métodos de apoio a decisão. Dentre os métodos comumente adotados em projetos de software, pode-se destacar o *Goal-Question-Metric* (GQM). O GQM tem como objetivo alinhar a definição de métricas aos objetivos da organização. Todavia, sabe-se que a compreensão e execução dos processos que envolvem medição de software está naturalmente ligado ao nível de capacitação do profissional que a realiza. Nesse sentido, existem diversos desafios no ensino de qualidade de software o que torna necessário investigar diferentes abordagens que promovam aos alunos alinhar a teoria e prática e, consequentemente, os tornem profissionais mais preparados para o prosseguimento de suas carreiras. Diante de tais motivações, o presente trabalho objetiva propor um *framework* baseado na integração do método Pesquisa-ação ao GQM, denominado GQM+PA. Quanto a avaliação empírica do GQM+PA, investigou-se múltiplos casos de estudo no Núcleo de Práticas de Desenvolvimento de Sistemas (NPDS), estabelecido na Universidade Federal do Ceará (Campus de Crateús). Em relação às contribuições atingidas por este trabalho, destaca-se i) fornecer um mecanismo integrado para o aprendizado técnico e prático sobre medição de software, ii) viabilizar empiricamente aos discentes uma reflexão sobre a relevância quanto a constante acompanhamento de métricas de software e iii) todas as rotinas, *dashboards* e planilhas desenvolvidas podem ser adaptados em outros projetos e usufruídos como um ativo organizacional.

Palavras-chave: Qualidade de Software. Medição de Software. GQM. Ensino. Pesquisa-ação.

ABSTRACT

Software Engineering emerged from the need to build software systems with high quality and with minimal resources. Hence, Software Quality is identified as an important area of Software Engineering, precisely because it is concerned with ensuring that software products and processes have quality. In order to better understand whether a process or software product is of quality or not, it is necessary to evaluate it. A often practice for such a process is software measurement, which is intrinsically related to the concept of software metrics. However, the definition of which metrics should be evaluated in a software project demonstrates a complex activity, thus requiring methods for support decision. Among the methods usually adopted in software projects, one can highlight the Goal-Question-Metric (GQM). GQM aims to align the definition of metrics with the objectives of the organization. However, it is known that the understanding and execution of processes adressing software measurement is naturally linked to the level of qualification of the professional that performs it. Therefore, there are several challenges in teaching software quality, which makes it necessary to investigate different approaches that encourage students to align theory and practice and, consequently, make them professionals better prepared to continue their careers. Faced with such motivations, the present work aims to propose a framework, called GQM+PA, based on the integration between the research-action method and GQM. Concerning to the empirical evaluation of GQM+PA, multiple cases of study were investigated in the Practical Center of Systems Development (NPDS), headed at the Federal University of Ceará (Campus Crateús). Regarding the contributions, one can highlight i) provide a integrated framework for enabling both theoretical and practical learning about software measuring, ii) empirically empower the students to the process of thinking about the relevance in carrying out software metrics analyses and iii) all the routines, dashboards and sheets be available to be adapted for other projects ans exploited as an organizational asset.

Keywords: Software Quality. Software Measurement. GQM. Teaching. Research-action.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo serão apresentados o contexto e motivação do presente estudo, bem como os objetivos e a estrutura do mesmo.

1.0.1 Contextualização

A tecnologia está cada vez mais ganhando espaço na rotina das organizações. Como consequência, softwares são recorrentemente desenvolvidos como soluções para atender as expectativas e problemas enfrentados pelas empresas. No entanto, construir um sistema de qualidade não é uma tarefa trivial. É necessário, por exemplo, identificar adequadamente as especificações requeridas pelo cliente, qual linguagem de programação será adotada, qual a melhor plataforma, quais padrões serão utilizados, entre outras decisões. Segundo Bartié (2002), os aspectos que envolvem o desenvolvimento de um software tem um nível de complexidade crescente. Identifica-se, então, que a construção de um sistema envolve uma definição de processos para que se tenha um produto final, ou seja, um produto de software.

Para Jalote (2012), um processo de software envolve um conjunto de atividades que precisam estar interligadas por padrões e, se as atividades atuarem corretamente conforme os padrões, o resultado desejado é atingido. Logo, a produção de um produto de software está atrelado a realização de um conjunto de etapas. Visando apoiar tais atividades, surge a Engenharia de Software, a qual tem como premissa ser uma disciplina que envolve todos os aspectos da produção de um software, desde a especificação do sistema até sua manutenção (SOMMERVILLE, 2011). Diante de tal conceituação, percebe-se a relevância em prover uma compreensão aprimorada dos processos que abrangem a construção de sistemas. O objetivo de tal entendimento refere-se a possibilidade de aprimorar a qualidade do processo e, consequentemente, do produto final.

A garantia de qualidade, no contexto da Engenharia de Software, compreende todos os processos de desenvolvimento de software (BARTIÉ, 2002). Pressman e Maxim (2016) definem qualidade de software como a conformidade aos requisitos funcionais e de desempenhos que já foram declarados, aos padrões de desenvolvimento documentados e às características que são esperadas de todo software. A garantia de qualidade consiste em procedimentos, técnicas e ferramentas aplicadas por profissionais para assegurar que um produto atinja padrões que já tenham sido especificados durante o ciclo de desenvolvimento do produto (BUENO; CAMPELO, 2010).

Para compreender melhor se um processo ou produto de software é de qualidade ou não, faz-se necessário que o profissional de software saiba, de fato, como avaliá-los. Nesse contexto, tem-se o processo de medição de software como uma prática recorrente para avaliação de sistemas. Através da medição de software, torna-se possível ter uma compreensão adequada e maior controle sobre o projeto, ou seja, viabiliza-se verificar sua qualidade com base em dados representados. Logo, ao internalizar o papel da medição de software, possibilita-se um maior controle sobre a qualidade de processos e produtos desenvolvidos (ROCHA *et al.*, 2012).

1.0.2 Motivação

O processo de medição de software tem uma intrínseca relação com o uso de métricas de software. Segundo Pressman e Maxim (2016), as métricas de software são medidas quantitativas que possibilitam aos envolvidos em um projeto de software ter um entendimento sobre a eficácia do processo de software. Porém, o processo de definição de quais métricas devem ser avaliadas demonstra-se complexo, visto que uma escolha inadequada das métricas pode levar a um entendimento errado do processo, o que dificulta a análise, contribui para decisões equivocadas e gera esforço desnecessário (GOMES *et al.*, 2001). Como consequência a uma má especificação, os resultados adquiridos contribuirão para que as tomadas de decisões sejam equivocadas. Dessa forma, fica claro que uma estrutura inadequada para medição pode afetar todo um projeto e, consequentemente, interferir na qualidade do produto final.

Assim, faz-se necessário o apoio adequado de métodos que apoiem a definição de quais métricas devem ser utilizadas, dentre os quais pode-se destacar o *Goal-Question-Metric (GQM)*. O GQM demonstra-se amplamente adotado em projetos de software e tem como propósito alinhar a definição de medidas aos objetivos de uma organização (BASILI, 1992). Ou seja, a organização deve definir primeiramente os seus objetivos e de seus projetos e, posteriormente, relacioná-los com a definição de métricas.

Entretanto, a compreensão e execução dos processos que envolvem a medição de software como, por exemplo, o uso de GQM, está intimamente relacionada ao nível de capacitação do profissional que a realiza. Nesse sentido, o ensino de qualidade de software desempenha um papel fundamental na formação dos alunos vinculados a graduações que contemplam Engenharia de Software na grade curricular. Sob tal ótica, Wangenheim *et al.* (2009) enfatizam o "desafio de ensinar medição de uma forma concentrada e atraente proporcionando o entendimento dos conceitos-chaves e capacitando o educando para aplicação de medição na prática".

Dentre as razões que ampliam a lacuna entre o desejado e o existente sobre a prática de medição de software, pode-se mencionar, em particular, a recorrente forma expositiva pela qual a mesma é geralmente ensinada (HOCK; HUI, 2004). Segundo Choi e Hannafin (1995), há de se haver a transferência de experiências para situações do mundo real para que o aluno atinja níveis mais altos de cognição. Sob o ponto de vista educacional, Chanin *et al.* (2018) consideram crucial implementar uma abordagem colaborativa e engajadora para que os estudantes entendam o que se pretende ensinar. Considerando essas premissas, o presente trabalho hipotetiza quão benéfico pode ser investigar a adoção de práticas que contribua para alinhar a teoria e prática no mercado de trabalho, sob o contexto de ensino de medição de software.

Nesse sentido, tem-se a perspectiva teórica contemplada pela literatura de medição e qualidade de software, enquanto a perspectiva teórica é proveniente da junção do GQM às práticas da Pesquisa-Ação (PA). A PA é um método de pesquisa amplamente difundido em trabalhos educacionais e se caracteriza por envolver estratégias de ação planejada que são implementadas e, a seguir, sistematicamente submetidas a observação, "reflexão e mudança" (GRUNDY; KEMMIS, 1982). A motivação pelo uso da PA diz respeito ao fato da mesma ter como principal característica ser iterativa, ou seja, envolve um processo que segue um ciclo em que se aprimora a prática através da oscilação sistemática entre agir no campo da prática e investigar a respeito dela (TRIPP, 2005).

Assim, o presente trabalho pretende integrar as etapas que constituem o GQM às práticas da PA, mesclando ambos em um *framework* que oriente o ensino de medição de software. Justifica-se, portanto, a relevância em investigar a adoção de tal concepção, haja vista que se torna factível promover aos alunos uma compreensão integrada, atraente e prática do processo de medição de software.

Em relação ao processo de avaliação, este trabalho será realizado no Núcleo de Práticas de Desenvolvimento de Software (NPDS) estabelecido na Universidade Federal do Ceará - Campus de Crateús. O objetivo do NPDS é fornecer aos estudantes de Tecnologia da Informação um ambiente para realização de projetos de software e hardware que possa prepará-los para o mercado de trabalho. Além de contribuir na formação dos alunos, esta pesquisa espera contribuir para uma gestão mais eficiente do NPDS, tendo em vista o maior controle sobre os processos de software que será possível atingir. Consequentemente, almeja-se que o conhecimento e processo (rotinas, planilhas, *dashboards*, etc) implantados permaneçam como ativos organizacionais no NPDS e/ou sejam adaptados por outras instituições.

1.0.3 Objetivo Geral

Propor um *framework*, chamado GQM+PA, baseado na integração de Goal-Question-Metric e Pesquisa-Ação para ensino de medição de software.

1.0.3.1 Objetivos Específicos

- Definir o plano GQM e práticas de aprendizado a serem implantados;
- Avaliar a aprendizagem dos alunos em relação aos principais conceitos que envolvem o processo de medição de software;
- Avaliar o efeito, em termos de produto e processo, decorrente do uso de GQM+PA.

1.0.4 Estrutura do Trabalho

Neste capítulo inicial foi introduzida uma compreensão geral deste trabalho, descrevendo o contexto, motivação e objetivo do mesmo. Além da Introdução, a atual versão desta pesquisa está estruturada em mais sete capítulos:

- Capítulo 2 - Fundamentação Teórica: são discutidos os principais conceitos acerca das áreas que constituem este trabalho: Qualidade de Software, Modelos de Maturidade, Medição de Software, Métricas de Software, *Goal-Question-Metric (GQM)* e Ensino em Engenharia de Software.
- Capítulo 3 - Trabalhos Relacionados: são analisados os principais trabalhos que ajudaram na construção desta pesquisa.
- Capítulo 4 - Procedimentos Metodológicos: tem por objetivo apresentar a metodologia empregada neste trabalho.
- Capítulo 5 - *Framework* GQM+PA: tem por objetivo descrever a proposta deste trabalho.
- Capítulo 6 - Planejamento do Estudo: é apresentado o planejamento da abordagem GQM+PA.
- Capítulo 7 - Resultados e Análises: são apresentados os resultados da presente pesquisa.
- Capítulo 8 - Considerações Finais: são discutidas as contribuições, limitações e trabalhos futuros desta pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo é apresentada uma revisão bibliográfica dos principais tópicos contemplados pela presente pesquisa que servem como base para o entendimento deste trabalho. A Seção 2.1 apresenta as principais definições e características na área de qualidade de software. A Seção 2.2 apresenta os modelos de maturidade. A Seção 2.3 apresenta os conceitos relacionados à medição de software. A Seção 2.4 apresenta as diversas definições e classificações referentes as métricas de software e, finalmente, a Seção 2.5 apresenta os fundamentos de GQM.

2.1 Qualidade de Software

O termo "qualidade" é um termo recorrente no vocabulário das pessoas e organizações, tendo sua formalização definida por diferentes autores. Segundo Carvalho e Paladini (2013), na década de 1930, o termo controle de qualidade evoluiu bastante com o desenvolvimento de sistema de medidas, ferramentas de controle estatístico do processo e do surgimento de normas específicas. Nesse contexto, Crosby (1992), define qualidade como conformidade aos requisitos. Já Feigenbaum (1983), conceitua que qualidade é o melhor para certas condições do cliente. Para Ishikawa (1985), o controle de qualidade busca produzir produtos com qualidade que satisfaçam exigências dos consumidores. Resumidamente, pode-se perceber que, diante dos conceitos mencionados, para que um produto seja considerado de qualidade é preciso, principalmente, atender as expectativas dos clientes, ou seja, o que foi especificado pelos mesmos.

Ao se ter um melhor entendimento sobre o conceito de qualidade, entendê-lo aplicado ao contexto de desenvolvimento software, torna-se fundamental para a presente pesquisa. O conceito de qualidade de software está relacionado à área de Engenharia de Software cuja definição, de acordo com Sommerville (2011), caracteriza-se como uma disciplina de engenharia que se preocupa em todos os aspectos da produção de um software, desde as fases iniciais da especificação do sistema até sua manutenção, quando o sistema já está sendo usado.

Como a demanda por produtos de software é crescente, garantir a qualidade desses produtos demonstra-se um fator preponderante para as empresas de desenvolvimento de software. Com isso, identifica-se o papel da Engenharia de Software na compreensão dos processos de desenvolvimento de software e, consequentemente, numa constante busca de melhoria nos processos para garantir a qualidade do produto. Logo, para realizar a garantia de qualidade é fundamental ter conhecimento sobre os processos que envolvem a construção de um software.

Para Pressman e Maxim (2016), a qualidade de software está relacionada a conformidade com requisitos funcionais e de desempenho claramente declarados, normas de desenvolvimento explicitamente documentadas e características implícitas, que são esperadas em todo software desenvolvido profissionalmente. Koscianski e Soares (2007) complementam ao expor que, para se ter qualidade em um produto de software, é preciso fazer uso de boas práticas no processo de desenvolvimento. Com isso, fica claro que dependendo do contexto e de diferentes concepções um produto pode ser considerado de qualidade ou não.

Portanto, em síntese, a qualidade de software está relacionada com a conformidade do que foi especificado e da necessidade de satisfazer as expectativas dos clientes. Sommerville (2011) conclui que a qualidade do software está relacionada diretamente à qualidade do processo de desenvolvimento de software. Para este autor, a qualidade do produto final está profundamente relacionada ao processo de produção de software, pois ao se ter uma preocupação com a qualidade das etapas de desenvolvimento, como consequência haverá um produto de software com qualidade.

Uma vez compreendido os diferentes conceitos sobre qualidade de software, faz-se necessário esclarecer os principais fatores que a influenciam, conforme discutido na seção a seguir.

2.1.1 Fatores de Qualidade

Para a avaliar a qualidade de um produto de software, é necessário definir um modelo de qualidade e que seja usado na definição das metas de qualidade para os produtos de software final e intermediários (TÉCNICAS, 2003). No modelo, os atributos ou fatores são definidos como características para avaliar os produtos de software. Pressman e Maxim (2016) definem que os fatores de qualidade podem ser usados como uma sugestão genérica da qualidade de uma aplicação e que se concentram no software como um todo. Tais autores exemplificam duas classificações de fatores que afetam a qualidade, são elas: Fatores de Qualidade de McCall e Fatores de Qualidade ISO 9126.

2.1.1.1 Fatores de qualidade de MacCall

Pressman e Maxim (2016) descrevem que o modelo de qualidade proposto por MacCall, Richards e Walters refere-se a categorização dos fatores que afetam a qualidade de software. Conforme pode-se observar na Figura 1, essa classificação é composta por fatores

que enfatizam três aspectos de um produto de software, que são as características operacionais (Operação do Produto), habilidade de suportar mudanças (Revisão do Produto) e adaptabilidade a novos ambientes (Transição do Produto).

Figura 1 – Fatores de qualidade de software de MacCall, Richard e Walters

Fonte: Pressman e Maxim (2016).

A Operação do Produto envolve as características operacionais do produto e é decomposta em cinco fatores:

1. Correção: o produto atende as especificações e aos objetivos do cliente.
2. Confiabilidade: cumpre a função pretendida com a precisão exigida.
3. Usabilidade: esforço fundamental para utilizar, aprender, preparar as entradas e interpretar as saídas de um programa.
4. Integridade: apenas pessoas autorizadas podem ter controle de acesso ao software e aos dados.
5. Eficiência: o quanto de recursos computacionais e códigos são exigido para uma função.

A Revisão do Produto está relacionada a habilidade de suportar mudanças e é decomposta em três fatores:

1. Facilidade de manutenção: esforço necessário para identificar e corrigir uma falha em um programa.
2. Flexibilidade: esforço necessário para realizar mudanças em um programa.
3. Testabilidade: esforço necessário para testar um programa e garantir que não contenha erros e que atenda ao que foi especificado.

A transição do produto refere-se a adaptabilidade a novos ambientes e é decomposta em três fatores:

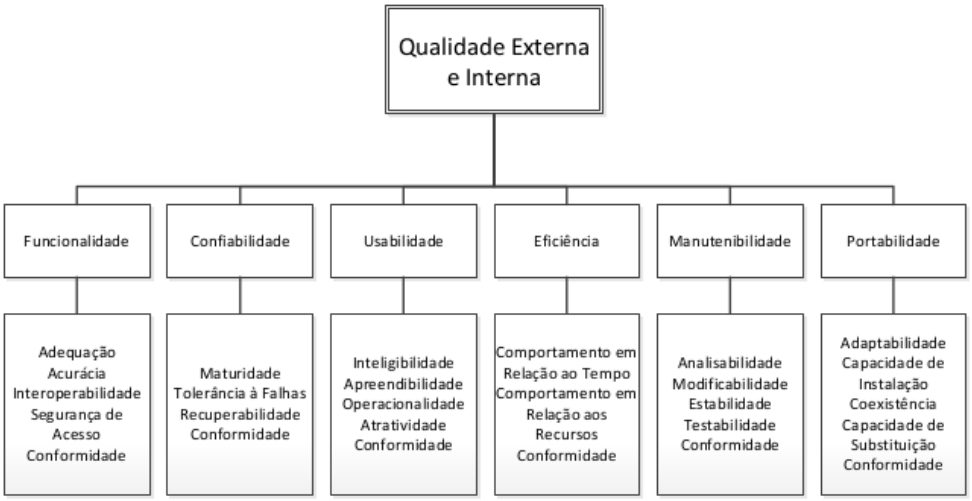
1. Portabilidade: esforço necessário para transferir um programa de um ambiente para outro.
2. Reusabilidade: o quanto um programa ou partes pode ser reutilizado em um outro contexto.
3. Interoperabilidade: esforço necessário para integrar um sistema com outro.

2.1.1.2 Fatores de qualidade ISO 9126

Desenvolvido como forma de identificar os atributos fundamentais de qualidade para software de computador, o padrão ISO 9126 que é definido como um modelo de qualidade externa e interna, apresenta atributos que são categorizados em seis fatores de qualidade, conforme ilustrado na Figura 2.

1. Funcionalidade: grau com que o software atenda as necessidades apresentadas, conforme os seguintes atributos: adequabilidade, exatidão, interoperabilidade, conformidade e segurança.
2. Confiabilidade: capacidade do software de ficar disponível para uso, conforme os seguintes atributos: maturidade, tolerância a falhas, facilidade de recuperação.
3. Usabilidade: capacidade do software de ter facilidade de compreensão, facilidade de aprendizagem e operabilidade.
4. Eficiência: capacidade do software de apresentar desempenho adequado com relação ao tempo e aos recursos.
5. Facilidade de manutenção: capacidade do software de permitir uma correção, conforme a facilidade de análise, de mudanças, estabilidade e testabilidade.
6. Portabilidade: capacidade do software de poder ser transferido de um ambiente para outro conforme a adaptabilidade, facilidade de instalação, conformidade e facilidade de substituição.

Figura 2 – Fatores de qualidade ISO 9126



Fonte: Santos (2014).

Nota-se, portanto, que os modelos apresentados anteriormente possuem uma interseção de fatores que contribuem para analisar a qualidade de um produto de software. Para Pressman e Maxim (2016), esses modelos são uma excelente lista de verificação para avaliar a qualidade de um produto de software e ao usar esses fatores haverá uma sólida indicação de qualidade de um software.

Os modelos de qualidade MacCall e ISO 9126, assemelham-se ao terem a característica de apresentar fatores que podem ser avaliados, ou seja, podem passar pelo processo de medição com o propósito de revelar algum indicador que demostre se há ou não qualidade em um determinado aspecto de um software. Além de se mostrarem ter fatores em comum como, portabilidade, confiabilidade, usabilidade e facilidade de manutenção. Apesar dos modelos partilharem alguns atributos, nota-se que no ISO 9126, além dos fatores há subcaracterísticas que também podem ser medidas.

2.2 Modelos de Maturidade

No atual cenário competitivo em que as organizações estão inseridas, a busca por melhorias em seus produtos e processos são fundamentais para se atingir um diferencial. Para o desenvolvimento de software com qualidade, dentro de prazos e custos controlados e compatíveis com o mercado, é essencial a melhoria dos processos. Para tanto, abordagens e experiências para a melhoria de processo de software baseadas em modelos têm sido utilizadas com sucesso pelas organizações de software (CRESPO *et al.*, 2004). Diante de tal contexto, a adoção de modelos

de maturidade vem se consolidando cada vez mais nas empresas. Os modelos de maturidade fornecem orientações para as empresas na definição de seu plano de melhoria da qualidade e produtividade (MAGNO *et al.*, 2011).

Existem diversos modelos de maturidade os quais podem ser adotados como, por exemplo, *CMM (Capability Maturity Model)*, *CMMI (Capability Maturity Model Integration)*, *PMMM (Project Management Maturity Model)*, MPS-BR (Melhoria do Processo de Software Brasileiro) e ISO/IEC 12207. Em tais modelos há uma definição recorrente que existem diferentes níveis de maturidade que os indicam o nível de evolução de organizações e qual grau de qualidade apresentam. Assim, um conjunto de boas práticas para o desenvolvimento de produtos, serviços e projetos são definidos pelos modelos.

Será apresentado abaixo o MPS.BR e o CMMI, visto que o primeiro é amplamente difundido no contexto brasileiro, enquanto o segundo é um dos mais utilizados no mundo. De acordo com ROCHA *et al.* (2006), o MPS-BR está sendo implementado em diversas empresas do Brasil por Instituições Implementadoras, através do Fórum de Credenciamento e Controle do MR-MPS. Já o CMMI tem sido adotado por muitas organizações no mundo com o objetivo de possibilitar a elevação da maturidade da capacidade de suas equipes nas atividades relacionadas ao software (MORGADO *et al.*, 2007). A motivação em discutir tais modelos diz respeito à oportunidade de compreender a relação entre os modelos de maturidade e o processo de desenvolvimento de software, em especial com o processo de medição.

2.2.1 MPS-BR (Melhoria de Processos do Software Brasileiro)

O programa MPS-BR é um modelo de qualidade criado pela Softex (Associação para Promoção da Excelência do Software Brasileiro) com o objetivo de melhorar a performance do desenvolvimento de software nas organizações brasileiras, focando nas micros, pequenas e médias empresas. Para a construção do MPS-BR, foi usado como referência o CMMI, ISO/IEC 12207 e ISO/IEC 15504, fazendo uso de boas práticas de Engenharia de Software e sendo adaptado ao contexto brasileiro. Sendo constituído de três componentes : um Modelo de Referência de Melhoria de Processo de Software (MR-MPS) que define níveis de maturidade; Método de Avaliação (MA-MPS) que integra o processo de avaliação e o Modelo de Negócios (MN-MPS) que descreve as regras para implantar o MR-MPS pelas empresas de consultoria (SOFTEX, 2012).

Koscianski e Soares (2007) destacam três guias do modelo MPS-BR, que são:

1. Guia geral: descrição geral do MPS-BR e detalhamento do modelo de referência (MR-MPS) e as definições necessárias para seu entendimento e aplicação.
2. Guia de aquisição: recomendações para conduzir as compras de software e serviços.
3. Guia de avaliação: descreve o processo de avaliação, os requisitos para avaliação, o método e formulário para encaminhar a avaliação.

Segundo Maretto (2013), tanto o CMMI e o MR-MPS-SW (Modelo de Referência MPS para Software) estruturam os processos de software e guiam a melhoria de processos em níveis de maturidade. Os níveis de maturidades apresentam um conjunto de processos que possuem um determinado nível de capacidade. Com isso, é possível saber a evolução de uma organização ao implementar processos com diferentes fases de melhoria. Os níveis de maturidade do MPS-BR são:

1. Nível A (Em Otimização) : análise de causas, resolução, inovação e implantação na organização.
2. Nível B (Gerenciado): gerência quantitativa do projeto, desempenho do processo organizacional.
3. Nível C (Definido): processo de gerência de riscos, análise de decisão e resolução.
4. Nível D (Largamente Definido): desenvolvimento de requisitos, solução técnica, validação, verificação, integração do produto, instalação do produto e liberação do produto.
5. Nível E (Parcialmente Definido): adaptação do processo para gerência de projeto, definição do processo organizacional, avaliação e melhoria do processo organizacional e treinamento.
6. Nível F (Gerenciado): processos de garantia de qualidade, aquisição, gerência de aquisição e medição.
7. Nível G (Parcialmente Gerenciado): processos de gerência de projetos e de requisitos.

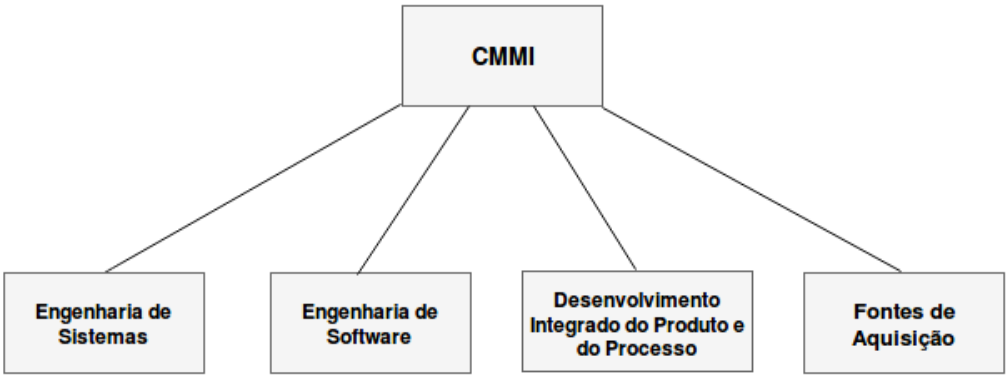
2.2.2 CMMI (Capability Maturity Model Integration)

O modelo CMMI constitui-se das melhores práticas voltadas ao desenvolvimento e à manutenção de produtos e dos serviços, englobando todo o ciclo de vida do produto, desde sua concepção até a sua entrega e manutenção (CHRISSIS *et al.*, 2003). Koscianski e Soares (2007) complementam descrevendo que o CMMI tem como meta servir de guia para melhoria de processos na organização e da habilidade dos profissionais em gerenciar o desenvolvimento, aquisição e manutenção de serviços ou produto. A seguir serão discutidos os principais conceitos sobre CMMI.

2.2.2.1 Disciplinas do CMMI

De acordo com Koscianski e Soares (2007), no modelo CMMI, são apresentados quatro disciplinas ou corpos de conhecimento: Engenharia de sistemas, Engenharia de software, Desenvolvimento integrado do produto e do processo e Fontes de aquisição, conforme é apresentado na Figura 3.

Figura 3 – Disciplinas do CMMI



Fonte: (KOSCIANSKI; SOARES, 2007).

1. Engenharia de sistemas: tem como propósito obter sistemas de forma bem-sucedida. Conforme as necessidades, restrições e expectativas declarados pelos clientes. Engenheiros de sistemas propõem produtos e soluções através de análise, projeto, validação, teste, implementação, treinamento e suporte.
2. Engenharia de software: Engenharia de software se dedicando às atividades de gerenciamento de projetos, desenvolvimento de ferramentas, métodos e teorias que dêem apoio a produção de software e não se preocupando apenas com os processos técnicos de desenvolvimento.
3. Desenvolvimento integrado do produto e do processo: abordagem que utiliza a colaboração de *stakeholders* (envolvidos no projeto) para atender melhor às expectativas e requisitos dos clientes.
4. Fontes de aquisição: atua na aquisição de produtos.

2.2.2.2 Representação por estágios x contínua

O CMMI apresenta duas representações: por estágio ou contínua (KOSCIANSKI; SOARES, 2007). A primeira destaca-se por expor uma estrutura caracterizada por níveis de

maturidade. Na contínua, há a possibilidade de escolher a sequência de melhorias que convém aos objetivos da organização (KOSCIANSKI; SOARES, 2007). A seguir é descrito os níveis de cada representação, conforme as perspectivas de Koscianski e Soares (2007).

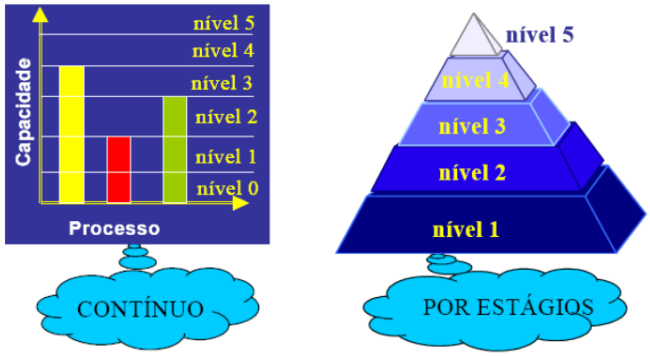
1. Representação por estágios: são definidos cinco níveis de maturidade que são estabelecidos para guiar a melhoria de processos e apresentam um caminho de melhoria de processos para a organização. São eles:
    - Nível 1 (Inicial): a organização não apresenta um ambiente estável de desenvolvimento de software, não há padrões. Projetos apresentam problemas relacionados a prazos e no cumprimento de requisitos.
    - Nível 2 (Gerenciado): nos projetos da organização, contém requisitos gerenciados e processos medidos, controlados e planejados.
    - Nível 3 (Definido): processos bem caracterizados e entendidos, possibilitando maior consistência nos produtos gerados.
    - Nível 4 (Gerenciado quantitativamente): processos são escolhidos para apoiar o desempenho geral de outros processos. Faz-se uso de métodos estatísticos e outras técnicas quantitativas.
    - Nível 5 (Otimizado): melhoria contínua dos processos com base em um entendimento quantitativo das causas comuns de alterações de desempenho.
2. Representação contínua: são apresentados seis níveis de capacitação que representam caminhos de melhoria e indicam a evolução para cada área de processo (conjunto de práticas que executadas coletivamente satisfaz um conjunto de objetivos que é importante para haver melhoria nessa área).
    - Nível 0 (Incompleto): corresponde a não realização de um processo.
    - Nível 1 (Realizado): cada processo cumpre todos os objetivos específicos para satisfazer um conjunto de objetivo.
    - Nível 2 (Gerenciado): os processos cumprem os requisitos de nível 1 e, são planejados e executados de acordo com uma determinada política. Além disso, processos são monitorados, controlados e revisados, assim como os produtos.
    - Nível 3 (Definido): cumpre os requisitos de nível 2. Além de estar adaptado a um conjunto de processos padronizados seguindo as diretivas da organização. Processos são melhorados continuamente.
    - Nível 4 (Gerenciado quantitativamente): os processos são definidos e controlados

quantitativamente através de técnicas estatísticas.

- Nível 5 (Otimizado): cumpre os requisitos de nível 4 e é moldado para realizar os objetivos da organização e tem como foco melhoria contínua de desempenho.

Para Rincon (2009), a representação contínua proporciona que uma organização tenha sua capacidade avaliada por processos e que cada um tenha níveis diferentes de capacidade e, na representação por estágios, uma organização deve implementar os processos conforme uma sequência pré-estabelecida. Diferentemente da representação contínua, a organização é avaliada por um conjunto de processos de acordo com o nível que ela estiver pretendendo alcançar (RINCON, 2009). A representação contínua e por estágio é ilustrada na Figura 4.

Figura 4 – Representações do CMMI



Fonte: Rincon (2009).

Em ambos os modelos, nota-se que há a presença de um processo relacionado a medição de software, onde dados dos processos são coletados e em seguida analisados com o propósito de extrair informações que ajudem na tomada de decisões das organizações e na melhoria dos processos. Diante disso, na seção a seguir será discutido os principais conceitos sobre medição de software.

2.3 Medição de Software

A atividade de fazer medição é recorrente há muito tempo para a sociedade. Medir produtos, o comprimento de um objeto, o peso de uma pessoa, a área de um retângulo, entre outros são indispensáveis para revelar informações que possam ajudar em tomadas de decisões. Fenton (1994) conclui que medição é o processo pelo qual números ou símbolos são associados a atributos de entidades no mundo real, com a finalidade de descrevê-la de acordo com regras bem definidas. Assim como o termo medição está presente em várias áreas, na Engenharia de Software não é diferente. O desenvolvimento de software requer um procedimento para a

obtenção de informações sobre processos de software e, assim, ter um adequado entendimento e avaliação (BASILI *et al.*, 1994). As medições fornecem descrições quantitativas dos processos e produtos, proporcionando entendimento de comportamento e de resultados (PFLEEGER *et al.*, 1997).

A medição de software pode ser compreendida como o processo contínuo de definição, coleta e análise de dados sobre o processo de desenvolvimento de software e seus produtos, com objetivo de entender e controlar o processo e seus produtos para fornecer informações significativas e, assim, melhorar esse processo e seus produtos (SOLINGEN; BERGHOUT, 1999). Segundo Ambler (1999), a medição, no desenvolvimento de software, pode ser vista como o processo de definir, coletar, analisar e agir sobre medidas que possam melhorar tanto a qualidade do software desenvolvido por uma organização quanto o processo de desenvolvimento utilizado. Conforme Franca *et al.* (1998b), o principal foco da medição é o produto final, ajudando a verificar a robustez do produto e sua aderência em relação à especificação.

Solingen e Berghout (1999) e Ambler (1999) descrevem que a medição de software envolve definição, coleta e análise de dados para que se possa ter um entendimento adequado sobre o processo de desenvolvimento e, então, identificar pontos de melhorias. Sob outro ponto de vista, Franca *et al.* (1998a) expõem que a medição de software está relacionada diretamente a melhoria de um processo de desenvolvimento de software, devido à necessidade de realizar medições controladas e as várias abordagens de desenvolvimento de software, pois é importante criar mecanismos que permitam aos laboratórios de software estabelecer, avaliar e desenvolver programas de medição. Já Pfleeger *et al.* (1997) complementam relatando que a medição de software está se estabelecendo como uma importante prática para suportar as iniciativas de melhorias no processo de software realizadas por muitas empresas.

Através da aplicação de medição, torna-se possível obter dados ou informações sobre o que é medido e, consequentemente, tomar decisões com base nessas informações a fim de melhorar os processos e produtos. Logo, ao se realizar o processo de medição, medidas serão aplicadas. Medida é uma variável na qual é atribuído um valor como resultado de uma medição (IEC, 2005). Salviano *et al.* (2006) descrevem que uma medida básica inclui um atributo mensurável de uma entidade, um procedimento para quantificar um atributo e um valor derivado da aplicação do procedimento. Referente a uma medida básica, contém os conceitos de escala de medição, unidade de medição, observação (como o ato de designar um valor) e unidade de observação.

Nesse contexto, existem diferentes tipos de medidas que englobam o produto ou processo. Medidas de produto são obtidas através de características de um produto ou artefato em qualquer etapa do desenvolvimento, como o código-fonte ou uma especificação de requisitos (MILLS, 1998). Já as medidas de processo são obtidas através das atividades incluídas no desenvolvimento dos produtos, como o esforço dedicado a cada atividade e a quantidade de defeitos injetados ou detectados em cada uma (MILLS, 1998).

Fica evidente, que as medições são indispensáveis para melhorias tanto de produtos como de processos. Diante da relevância de processos apresentarem melhorias, Florac e Carleton (1999) ressaltam três objetivos os quais devem ser atendidos ao se realizar a medição de processos de software:

1. Coletar dados para medir o desempenho do processo;
2. Analisar desempenho do processo;
3. Armazenar e utilizar os dados para interpretar os resultados de observações e análises, predizer custos e desempenho futuros, fornecer *baselines*, identificar tendências e avaliar a estabilidade e capacidade do processo.

Diante da abordagem de Florac e Carleton (1999), é notório que com aplicação destas três etapas pode-se obter, inicialmente, um plano para melhoria nos processos de forma adequada e ajudar na identificação de problemas e de pontos de melhorias nos processos.

Para ter um melhor entendimento sobre produtos ou processos de software é necessário medi-los, por isso, várias métricas são aplicadas para a realização de medição. Portanto, torna-se relevante conhecer os principais conceitos relacionados às métricas de software.

2.4 Métricas de Software

Métrica é um método para determinar se um sistema, componente ou processo possuem um certo atributo (COMMITTEE *et al.*, 1990). De acordo com Tsuruta (2011), as métricas de software são formas de mensurar uma característica da qualidade. As métricas de software proporcionam uma base quantitativa para planejar e prever processos de desenvolvimento de software (RAWAT *et al.*, 2012). Esposte (2014) destaca que as métricas, na Engenharia de Software, fornecem uma forma de medir quantitativamente atributos relacionados às entidades do software e do processo de desenvolvimento. Diante disso, percebe-se que as métricas envolvem um conjunto de medidas que oferecem uma atribuição quantitativa em decorrência da medição de um produto ou processo.

Perkins *et al.* (2003) ressaltam a importância de métricas nos projetos de uma organização. Para ele, métricas de diferentes aspectos do desenvolvimento podem ajudar a estabelecer o progresso do projeto, e serem usadas no gerenciamento para prover base para tomada de decisão. Se forem usadas da maneira certa, são essenciais para o sucesso de um projeto. Através das métricas, torna-se possível ter um melhor acompanhamento sobre o desempenho dos projetos. Ao se ter uma compreensão sobre os projetos, identificar as falhas torna-se mais fácil e, assim, pode-se tomar decisões ágeis para saná-las.

2.4.1 Classificação das Métricas de Software

Assim como foram identificadas diferentes medidas, as métricas podem ser categorizadas de diversas formas. De acordo com Honglei *et al.* (2009), as métricas são classificadas em três tipos:

1. Métricas do processo: destacam o processo de desenvolvimento de software, visando a duração do processo, custo incorrido e tipo de metodologia utilizada. Além de poderem ser usadas para aumentar o desenvolvimento e manutenção de software.
2. Métricas do projeto: são usadas para monitorar a situação do projeto. Métricas do projeto impedem os problemas ou riscos potenciais calibrando o projeto e ajudando a otimizar o software. Métricas do projeto descrevem o projeto, características e execução do mesmo.
3. Métricas do produto: descrevem os atributos do produto de software em qualquer fase do seu desenvolvimento. Podem medir o tamanho do programa, a complexidade do software, desempenho, portabilidade e facilidade de manutenção. As métricas de produto são usadas para medir o meio ou o produto final.

Além da classificação quanto ao tipo, as métricas também podem ser definidas sob diferentes critérios: métricas objetivas, subjetivas, qualitativas, quantitativas, organizacionais e de acompanhamento (SATO, 2007). Conforme são descritas a seguir.

1. Métrica Objetiva: o valor da métrica objetiva depende apenas do objeto em questão e não de quem está interpretando como, por exemplo, número de *commits* no repositório, pois é obtida diretamente da ferramenta.
2. Métrica Subjetiva: depende do objeto em questão e do ponto de vista de quem a está interpretando. Um exemplo é a qualidade do código.
3. Métrica Quantitativa: o valor desse tipo de métrica pertence a um intervalo de uma certa magnitude e é representado por um número, permitindo assim que medidas quantitativas

sejam comparadas entre si. Um exemplo é o número de linha de código.

4. Métrica Qualitativa: os valores dessa métrica são caracterizados por palavras, símbolos ou figuras ao invés de números como, por exemplo, o humor da equipe.
5. Métrica Organizacional: medem a quantidade de valor de negócio que foi entregue ao cliente (HARTMANN; DYMOND, 2006).
6. Métrica de Acompanhamento: prover informações que ajudam o time na compreensão e melhoria do processo que produz valor de negócio, ou seja, a equipe precisa de medições locais para ajudar a atingir os objetivos traçados (HARTMANN; DYMOND, 2006).

Conforme apresentado anteriormente, existem diferentes métricas que ajudam a melhorar um desempenho de um projeto. Entretanto, definir um conjunto de métricas a ser avaliado demonstra-se um desafio considerável demandando, assim, existem abordagens que auxiliem nesse processo de definição. Diante disso, é essencial executar uma abordagem que ajude na definição de métricas. Perante o exposto, a seguir será apresentado uma das principais abordagens utilizadas para tal fim, o GQM.

2.5 *Goal-Question-Metric (GQM)*

O GQM foi definida, originalmente para avaliar defeitos em um conjunto de projetos no Centro Espacial da NASA (BASILI, 1992). Apesar de ter sido usada para definir e avaliar metas para um projeto particular, a abordagem teve seu uso expandido para um contexto maior, como em programas de avaliação de qualidade de software (BASILI; ROMBACH, 1996). Essa abordagem caracteriza-se por ser baseada em objetivos. Allen e Davis (2010) descrevem que abordagens orientadas a objetivos visam: identificação de objetivos de negócio ou uma questão que necessita ser respondida; identificação das necessidades de informação essenciais para determinar se o objetivo foi atingido ou para responder a questão; quantificação das necessidades de informação sob a forma de medidas e a análise de medidas para estabelecer se os objetivos foram alcançados ou se a questão foi respondida de forma adequada.

Para Basili (1992), o GQM é um mecanismo para definir e avaliar um conjunto de objetivos operacionais, através de medição. Isso representa uma abordagem sistêmica para integrar objetivos com modelos de processo de software, produtos e perspectivas de interesse baseada em necessidades específicas do projeto e da organização. Rocha *et al.* (2012) concluem que o GQM, fundamenta-se na compreensão de que, para realizar medições de forma adequada, uma organização precisa primeiramente determinar os seus objetivos e os objetivos de seus

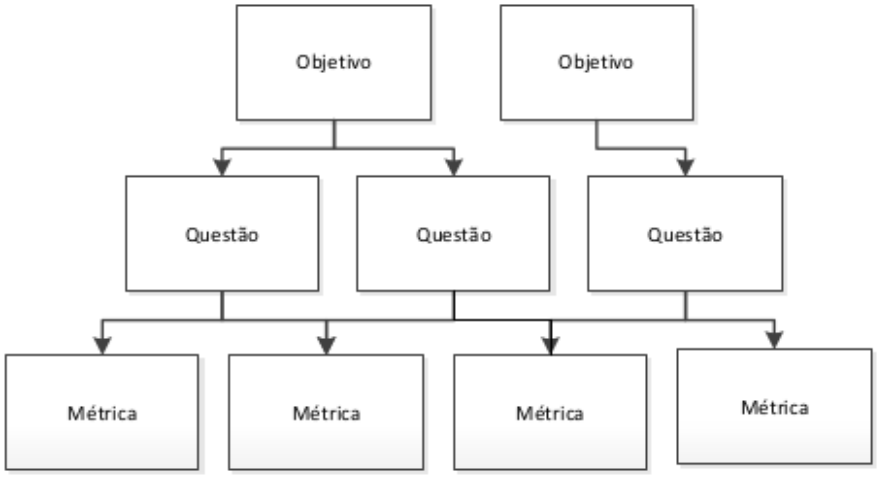
projetos e depois relacioná-los aos dados necessários e, no final, deve-se fornecer um *framework* para analisar e interpretar os dados com relação aos objetivos estabelecidos. Segundo Borges (2003), a principal característica do GQM é a utilização de uma abordagem *top-down* para definir as medidas. Além disso, complementa que o GQM difundi que o processo de mensuração não deve ser guiado pelas medidas em si, mas pelos objetivos que se pretende atingir com sua coleta. Logo, as medições só devem ser realizadas se estiverem apoiadas por uma meta claramente definida.

De acordo com Sato (2007), ao aplicar o GQM tem-se como consequência uma estrutura dividida em três níveis:

1. Nível conceitual (Objetivo): um objetivo é determinado para um objeto relativo a algum modelo de qualidade, com base em diversos pontos de vista e para um ambiente especıfico. Um objeto pode ser um produto (documento, código-fonte, testes, etc.), um processo (especificação, entrevista, codificação, etc.) ou um recurso (pessoas, hardware, espaço fisico, etc.).
2. Nível Operacional (Questão): um conjunto de perguntas é empregado para caracterizar a forma de avaliação e cumprimento do objetivo escolhido. As perguntas tentam associar o objeto de estudo com as características de qualidade desejáveis a partir do ponto de vista definido.
3. Nível Quantitativo (Métrica): um conjunto de dados é associado às perguntas com o propósito de encontrar uma forma quantitativa de respondê-las.

Conforme a Figura 5, Solingen e Berghout (1999) demostram que o modelo GQM começa de cima para baixo com a definição de um objetivo de medição. Este objetivo é refinado em várias perguntas que as dividem em seus principais componentes. Cada pergunta é então refinada em métricas que fornecem informações para responder as perguntas. Os dados de medição são interpretados de baixo para cima. Como as métricas foram definidas com um objetivo, as informações fornecidas pelas métricas devem ser interpretadas e analisadas em relação ao objetivo, para concluir se é ou não alcançado.

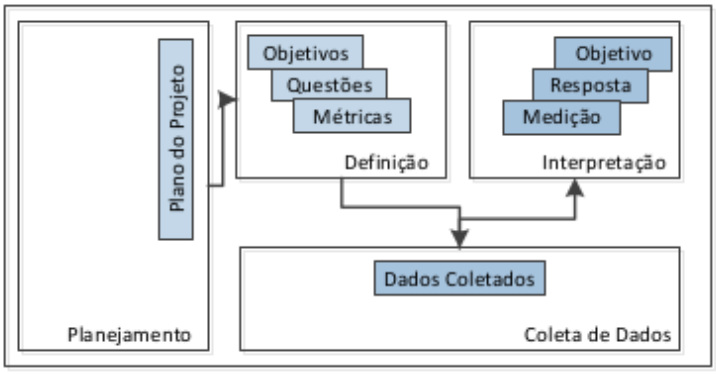
Figura 5 – Estrutura da Abordagem GQM



Fonte: Santos (2014).

Além da sua estrutura hierárquica, Solingen e Berghout (1999) destacam que GQM apresenta quatro fases distintas conforme demonstrado através da Figura 6:

Figura 6 – Estrutura das Fases do GQM



Fonte: Santos (2014).

1. Planejamento: um projeto de medição é selecionado, definido, caracterizado e planejado, resultando em um plano de projeto.
2. Definição: um programa de medição é definido, onde objetivos, perguntas, métricas são definidas e documentadas, como ilustrado na Figura 6. Em que uma métrica é utilizada para responder diferentes perguntas de diferentes objetivos.
3. Coleta de Dados: dados são coletados com base no plano de projeto e nas métricas.
4. Interpretação: os dados coletados são avaliados com relação às métricas definidas que fornecem respostas para as perguntas definidas, após respondidas as questões, o objetivo pode ser avaliado.

Dessa forma, pode-se constatar que uma vantagem nessa abordagem é que ela sustenta a identificação das métricas apropriadas, conforme o contexto e os objetivos da avaliação tanto quanto sustenta a análise e legitimidade dos dados coletados, assim como a interpretação e armazenamento desses dados (SARAIVA, 2006).

2.6 Ensino em Engenharia de Software

O mercado de trabalho exige cada vez mais profissionais com habilidades que vai desde o conhecimento sobre uma determinada tecnologia, ou seja, habilidades técnicas até habilidades relacionadas a comunicação, trabalho em equipe e autogerenciamento. A fim de diminuir a lacuna entre o meio acadêmico e o mercado de trabalho, surge a Engenharia de Software (ES) que trata dos processos relacionados ao desenvolvimento de sistemas. Logo, o ensino de Engenharia de Software torna-se para a formação profissional um processo fundamental para qualificar profissionais na indústria de software, tendo em vista o crescimento de desenvolvimento de software e sua importância na sociedade.

O currículo de ES deve corresponder às necessidades da indústria, e só assim as Universidades podem produzir profissionais altamente qualificados, podendo atender às necessidades da indústria de software (MISHRA; MISHRA, 2012). Como a preocupação com a qualidade do produto é um dos aspectos principais da ES, é preciso que a qualidade seja incluída ao longo do processo de desenvolvimento do produto. Diante disso, a ES busca tornar sistemático o desenvolvimento do software através do ensino de métodos e técnicas que objetivam entregar o produto com qualidade, dentro do prazo e orçamento (ANDRADE *et al.*, 2015).

Entretanto, percebe-se que ainda é um desafio determinar métodos de aprendizagem que correspondem as demandas do mercado. Cera *et al.* (2012) destacam que as metodologias de ensino-aprendizagem empregadas no ensino superior, em sua grande maioria, estão centradas no ensino. Diante disso, o estímulo a habilidades ligadas ao trabalho em equipe, proatividade e interdisciplinariedade figuram como um complemento a formação do perfil do egresso (CERA *et al.*, 2012). Contudo, constata-se que já há uma preocupação em organizar o processo de ensino e aprendizagem de uma forma que torne o ambiente acadêmico próximo das situações que ocorrem no mercado de trabalho (SCHOTS *et al.*, 2009).

3 TRABALHOS RELACIONADOS

Com o objetivo de discutir os trabalhos relacionados na presente pesquisa, apresenta-se abaixo uma seção focada nas pesquisas que tratam sobre Medição de Software, em seguida, os trabalhos que abordam o tema Ensino em Engenharia de Software e, finalmente, realiza-se uma síntese comparativa em relação ao estado da arte.

3.0.1 Medição de Software

Gomes *et al.* (2001) descrevem uma abordagem para avaliação de processo de software que especifica como selecionar métricas conforme a abordagem GQM, além de estabelecer a realização de medição como parte integrante do processo de desenvolvimento. A proposta do trabalho não é só medir, mas levantar dados sobre características do processo com objetivo de avaliá-lo. Em relação a seleção e definição de métricas, foram determinadas segundo a abordagem GQM, onde foram inicialmente definidos os objetivos a serem alcançados com a medição. Os objetivos determinados foram: Melhorar a precisão das estimativas de projeto, Aumentar a qualidade dos produtos liberados para uso e Diminuir o custo final dos projetos. Tais objetivos foram baseados a partir da identificação dos principais problemas enfrentados pelas empresas de desenvolvimento no Brasil, conforme a visão de especialistas nas áreas de gerência, qualidade de software, processo e melhoria de processo.

O trabalho de Wangenheim (2000) apresenta com detalhes o planejamento e execução de um programa de medição com base no GQM, destacando as orientações necessárias para sua execução na prática. O autor expõe as fases do processo GQM, sendo constituído de seis etapas: estudo prévio, identificação de metas GQM e desenvolvimento de plano GQM, desenvolvimento do plano de mensuração, coleta de dados, análise e interpretação e capturar experiências. A abordagem GQM é analisada através do conhecimento da relação entre custo e benefício da introdução de programas de medição baseados em GQM em vários ambientes industriais.

Tahir *et al.* (2016) apresentam uma revisão da literatura de estudos sobre programas de medição de software em busca de modelos de planejamento de medição existentes e ferramentas usadas para implementar os programas de medição, os fatores de sucesso e fracasso dos programas de medição e estratégias para enfrentar seus desafios. Nessa pesquisa, 65 estudos foram revisados e analisados. Sendo que foram identificados 35 modelos de planejamento de medição e 11 ferramentas associadas, a maioria recomenda extensões ou melhorias para

abordagens baseadas em objetivos. Dentre os fatores de sucesso encontrados destaca-se a adoção organizacional de programa de medição. Já as estratégias mencionadas para enfrentar os desafios destacam-se: gerenciamento efetivo de mudanças e gerenciamento de partes interessadas de medição, suporte de ferramentas automatizadas e incorporação de mecanismos de engenharia para projetar programas de medição sustentáveis, eficazes, escaláveis e extensíveis, conhecimento de medição e desenvolvimento de padrões.

3.0.2 Ensino em Engenharia de Software

Wangenheim *et al.* (2009) propõem um jogo educacional para medição de software a fim de ampliar o conhecimento de conceitos básicos e aprender sobre execução de medição na prática. Através do jogo, será possível exercitar a aplicação de medição na área de gerência de projetos alinhando-se ao nível dois de maturidade do CMMI, tendo como processo de medição a abordagem GQM. O jogo serviu como complemento as aulas tradicionais de alunos de pós-graduação fornecendo um ambiente para exercitar os conceitos repassados. Como resultado do projeto, muitos alunos relataram, de forma subjetiva, que o jogo os ajudou a aprender sobre conceitos, processos e aplicação de medição.

Cera *et al.* (2012) apresentam a utilização do método de Aprendizagem Baseada em Problemas, que se caracteriza por apoiar a aproximação entre teoria e prática, para aprendizagem em Engenharia de Software e para estimular habilidades profissionais nos alunos como, por exemplo, comunicação e proatividade. Em tal trabalho é apresentado meios para fornecer uma base de aprendizado mais próximo do mercado de trabalho, através de desafios nas resoluções de problemas reais e de forma coletiva. Como resultado, foi identificado pelos alunos a importância de práticas e trabalho em equipe para a formação profissional dos mesmos.

Andrade *et al.* (2015) demostram uma metodologia de ensino teórica e prática utilizada na disciplina de Engenharia de Software. A metodologia apresentada por Andrade *et al.* (2015) caracteriza-se por dividir as atividades em duas categorias: as intra-classe que são atividades realizadas em sala de aula e envolvem aulas expositivas, exercícios e discussões em grupo e as extra-classe em que as são atividades desenvolvidas fora do ambiente físico da sala de aula, sendo que é desenvolvido um projeto de software. Além dessas duas atividades, tem-se também atividades híbridas em que os alunos podem iniciar uma atividade em sala e estendê-la para fora do ambiente de aula. Como consequência da aplicação da metodologia, obteve-se resultados favoráveis que demostram que a metodologia está adequada com o objetivo

de envolver a teoria e prática no ensino de Engenharia de Software.

3.0.3 Síntese Comparativa

Os trabalhos apresentados sobre medição de software serviram como base para a presente pesquisa tendo em vista a necessidade de uma melhor compreensão sobre como ocorre o processo de medição e sua relevância no desenvolvimento de software para a garantia de qualidade, além de demostrarem a definição e como são executadas as etapas do GQM. Já os trabalhos sobre ensino em Engenharia de Software serviram para identificar os esforços que estão sendo adotados para se ter um melhor processo de aprendizagem a fim de conciliar a teoria com a prática em prol de uma formação acadêmica e profissional mais sólida.

Conforme verificou-se previamente, apesar de já existirem iniciativas em relação ao ensino de medição de software, não constatou-se nenhum trabalho que se orientasse pelo GQM, apesar de tal ferramenta ser amplamente adotada no mercado. Avaliando a relevância em investigar tal lacuna, o presente trabalho busca incentivar e integrar as atividades do GQM como práticas pedagógicas em um processo de aprendizagem.

4 PROCEDIMENTOS METODOLÓGICOS

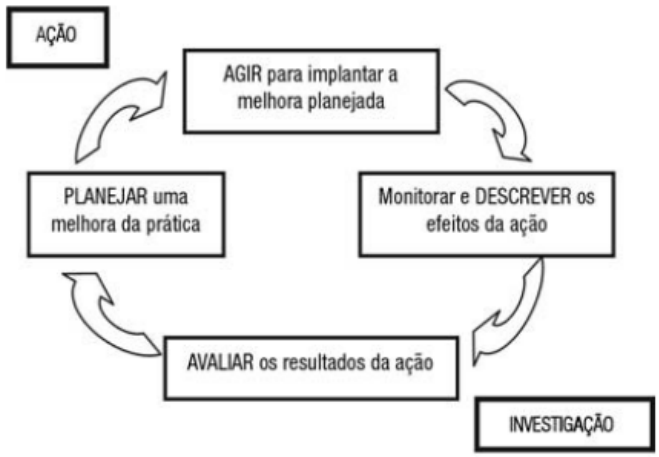
O presente trabalho será desenvolvido com base no tipo de pesquisa exploratória, que tem como objetivo possibilitar maior familiaridade com o problema, com vistas a tomá-lo mais explícito ou a constituir hipóteses, em que há aprimoramento de ideias ou descoberta de intuições (GIL, 2002). Diante disso, a pesquisa exploratória neste trabalho assume a forma de estudo de casos múltiplos sobre a adoção de um *framework* denominado GQM+PA como ferramenta de ensino de medição de software. Carneiro (2018) salienta que através de estudo de caso possibilita-se compreender fenômenos sociais e psicológicos complexos, em que múltiplas variáveis intervêm.

Em relação aos tipos de pesquisa quanto a sua abordagem, existem dois tipos: quantitativa e qualitativa. A quantitativa preocupa-se com a medição objetiva e a quantificação dos resultados. Busca a clareza, evitando distorções na fase de análise e interpretação dos dados, garantindo segurança em relação às inferências obtidas (GODOY, 1995). Já a qualitativa não procura enumerar ou medir os fenômenos estudados, nem emprega instrumental estatístico na análise de dados. Abrange obter dados descritivos sobre pessoas, lugares e processos interativos com a presença do pesquisador com a situação estudada para compreender os eventos segundo a visão dos participantes no estudo (GODOY, 1995). No contexto desta pesquisa, as duas abordagens serão empregadas. Com a abordagem quantitativa, dados empíricos sobre processos e produtos de software serão obtidos através da medição dos mesmos. Em termos qualitativos, será possível investigar a compreensão que os alunos obtiverem sobre a aprendizagem de medição de software através da realização de questionários abertos e da observação via pesquisador.

Conforme mencionado anteriormente, o método de pesquisa que norteia o presente trabalho é a pesquisa-ação. A pesquisa-ação tem por pressuposto que os que nela se envolvem compõem um grupo com objetivos e metas comuns, interessados em um problema que se manifesta num dado contexto (PIMENTA, 2005). Engel (2000) enfatiza que neste tipo de pesquisa procura-se interferir na prática de modo inovador no decorrer do processo de pesquisa e não apenas como consequência de uma recomendação no período final do projeto. Além disso, a pesquisa-ação pode ser adotada em qualquer cenário em que haja interação social que se caracterize por um problema em que há o envolvimento de pessoas, tarefas e procedimentos (ENGEL, 2000). Dessa forma, a adoção do referido método demonstra-se pertinente a presente pesquisa devido a necessidade de implantar e interferir no aprendizado de medição de software em um cenário prático orientado pelas etapas do GQM.

Quanto a se pensar na natureza de um projeto de pesquisa-ação, Tripp (2005) destaca cinco modalidades: pesquisa-ação técnica, pesquisa-ação prática, pesquisa-ação política, pesquisa-ação socialmente crítica e pesquisa-ação emancipatória. Para este trabalho, identifica-se a modalidade "pesquisa-ação técnica"que se caracteriza pelo pesquisador tomar uma prática existente de algum lugar e a implementa em seu próprio ambiente de prática para realizar uma melhoria. A pesquisa-ação tem como principal característica ser interativa, ou seja, envolve um processo que segue um ciclo em que se aprimora a prática através da oscilação sistemática entre agir no campo da prática e investigar a respeito dela (TRIPP, 2005). Conforme ilustrado na Figura 7, o ciclo se inicia, após a identificação do problema, com o planejamento de uma solução, em seguida ocorre a implementação do que foi planejado, o monitoramento e a avaliação de sua eficácia.

Figura 7 – Representação em quatro fases do ciclo básico da investigação-ação



Fonte: Tripp (2005).

De acordo com Tripp (2005), o ciclo básico da pesquisa-ação é constituído de quatro fases:

1. **Planejar**: nesta etapa, planeja-se uma mudança a fim de haver uma melhoria de sua prática, aprendendo mais durante o processo.
2. **Agir**: nesta etapa, o que foi planejado é posto em execução.
3. **Descrever**: realiza-se um monitoramento e descreve-se os impactos devido a ação.
4. **Avaliar**: nesta etapa, é avaliado os resultados da ação e em seguida é reiniciado um novo ciclo, promovendo a busca por melhoria constante.

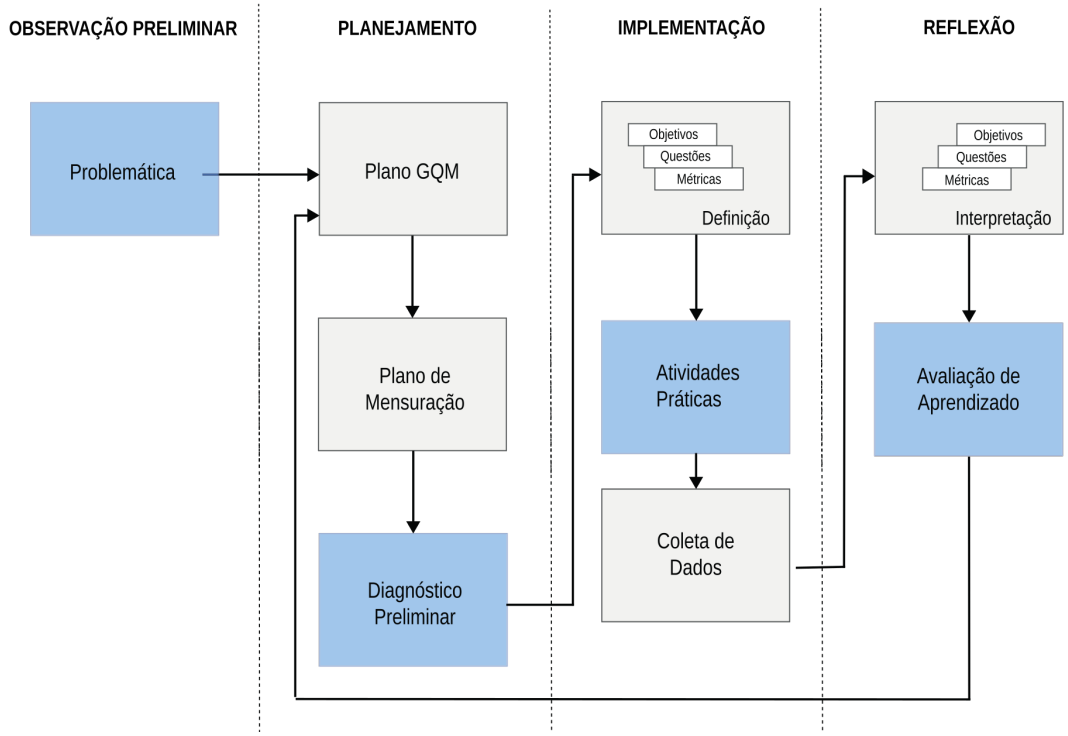
5 FRAMEWORK GQM+PA

A proposta de pesquisa-ação adaptado para este estudo foi inspirada pela obra de Kemmis *et al.* (2013), o qual é dividido em quatro diferentes etapas. A primeira etapa envolve a ação de um grupo de pessoas ou um pesquisador com o intuito de identificar um problema. Uma vez identificado a problemática, busca-se fazer algo para resolvê-la. Em seguida, analisa-se o quão bem sucedida foram os esforços realizados para solucionar tal problema. Caso os resultados não sejam considerados bem sucedidos, o grupo envolvido busca empregar novos esforços na próxima iteração com o propósito de melhorar os resultados obtidos.

A partir do *framework* para pesquisa-ação definido por Kemmis *et al.* (2013), o presente trabalho propõe a integração do processo de GQM ao seu contexto, formando assim, uma visão única e integrada. Em síntese, conforme pode ser visto na Figura 8, a pesquisa inicia com um estudo preliminar para determinar qual é o tipo de problema a ser resolvido. A segunda etapa caracteriza-se por realizar o planejamento sobre a abordagem GQM e sobre as práticas de aprendizagem que será adotado para capacitação dos alunos. Na terceira etapa, é definido a ação para concretizar o que foi planejado na etapa anterior, bem como a observação sobre o que foi implantado. Com base na observação, em que se constitui a quarta etapa, é feita uma reflexão sobre o que ocorreu ao executar a ação e avalia-se os resultados atingidos.

Ao integrar a pesquisa-ação com GQM, proporciona-se que as etapas que constituem o GQM também se façam presentes no processo de pesquisa-ação. Em termos de atividades a serem realizadas, verifica-se que a pesquisa-ação e o GQM tem suas particularidades e objetivos. Objetivando facilitar tal distinção, distingue-se as atividades de ensino via pesquisa-ação (azul) e do GQM (cinza) na Figura 8. Dessa forma, tem-se que o GQM+PA é constituído de 4 etapas: 1) Observação Preliminar; 2) Planejamento; 3) Implementação e Observação e 4) Reflexão. Acoplado a tais etapas, têm-se 9 atividades específicas, sendo 5 específicas do GQM e 4 oriundas da pesquisa-ação. A seguir serão detalhadas cada uma dessas etapas, bem como as respectivas atividades.

Figura 8 – *Framework* GQM+PA - Visão Geral



Fonte: Autoria própria.

Além disso, existem três papéis a serem desempenhados no GQM+PA:

1. Monitor: será responsável por repassar os conhecimentos necessários sobre medição de software e GQM, as instruções para a coleta dos dados das métricas, além de observar e intervir durante a implantação do GQM+PA.
2. Responsável (gerente): responsável por gerenciar os projetos, definindo as atividades que deverão ser feitas pelo alunos e aprovando as mesmas. O gerente de projeto tem como formação acadêmica mestrado em Ciência da Computação. Já na indústria de software apresenta experiência há oito anos e é gerente de projetos no NPDS há dois anos.
3. Time de discentes: serão responsáveis por realizar a coleta das métricas e receberão a capacitação necessária para realizar tal processo.

5.1 Observação Preliminar

Antes de prosseguir para a fase de planejamento, sugere-se a realização de uma análise em busca de identificar a problemática a ser investigada. Conforme salientado anteriormente, o principal problema que motiva o presente estudo é o ensino de medição de software. Nesse momento, necessita-se averiguar o processo de desenvolvimento e avaliar qual o nível de compreensão que os alunos têm sobre medição de software. Isto é importante para comparar o

entendimento deles antes e depois de implantar um processo de aprendizado. Além disso, nessa etapa, objetiva-se obter as informações preliminares que sejam pertinentes ao planejamento.

5.2 Planejamento

Uma vez feita uma avaliação prévia e identificado o problema, torna-se indispensável realizar um planejamento para aplicar as mudanças necessárias para solucionar os problemas identificados. Nesta etapa, planeja-se tanto a execução do GQM quanto as práticas de aprendizagem a serem empregadas. Para concretizar o planejamento, as seguintes atividades são fundamentais:

- **Definição do Plano GQM**: conforme sugerido no GQM, inicialmente, faz-se necessário definir os objetivos que precisam se alcançados (GOMES *et al.*, 2001). Uma vez tendo identificado os objetivos, planeja-se como ocorrerá a medição. De acordo com o que foi especificado com o gerente ou responsável pelo projeto, e a partir de questões atreladas aos objetivos, definem-se as métricas para alcançar as metas previamente determinadas. Assim, o artefato Plano GQM é caracterizado.
- **Definição de Plano de Mensuração**: nesta atividade será definido pela associação das medidas especificadas no plano GQM e do plano de projeto de software, mediante a definição de quando, como e por quem os dados requisitados podem ser coletados (WANGENHEIM, 2000).
- **Diagnóstico Preliminar**: após estabelecido como será a mensuração e coleta de dados é necessário definir como será a metodologia de aprendizagem do alunos estagiários em relação a medição de software e GQM a fim de torná-los capazes de executar as etapas do GQM e compreenderem a importância de implantar um processo de medição. Tais práticas podem ser representadas através de rotinas que garantam a compreensão dos conceitos envolvidos.

5.3 Implementação e Observação

Nesta etapa é colocado em prática o que foi planejado na etapa anterior (Planejamento), onde as práticas de aprendizagem são incorporadas ao projeto sob avaliação. Através das atividades práticas de aprendizado serão executadas as fases do GQM pelos alunos. Dessa forma, faz-se necessário definir de que forma e quais dados serão coletados em termos de pro-

cesso, produto e aprendizagem dos alunos. Além disso, o monitor analisará como os alunos se comportam em relação a realização dessas práticas a fim de identificar, por exemplo, dificuldades dos alunos em implantá-las e também ajudar a intervir em problemas que surgirem.

5.4 Reflexão

Finalmente, na Reflexão, objetiva-se interpretar os resultados obtidos e se as práticas de aprendizagem foram bem sucedidas. Diante de tal análise, será determinado se houve melhorias significativas na compreensão do alunos sobre medição de software e se os objetivos definidos pela gerência foram alcançados. Com base nas reflexões sobre as informações obtidas, deve-se identificar quais as principais dificuldades enfrentadas pelos alunos e o que ainda pode ser melhorado no programa de medição. Caso os resultados não sejam satisfatórios, pode-se iniciar um novo ciclo a fim de corrigir ou melhorar pontos de falha. Assim como uma metodologia iterativa e incremental de desenvolvimento em que há ciclos onde cada um pode ser melhorado, o GQM+PA busca melhoria contínua a cada ciclo.

6 PLANEJAMENTO DO ESTUDO

Este capítulo tem por objetivo descrever as etapas necessárias para a execução desta pesquisa, onde será apresentado a observação preliminar e contextualização do estudo, o plano GQM, plano de mensuração e diagnóstico preliminar.

6.1 Observação Preliminar e Contextualização do Estudo Empírico

Conforme mencionado na Introdução, este trabalho envolveu o Núcleo de Práticas de Desenvolvimento de Sistemas (NPDS) o qual encontra-se estabelecido na Universidade Federal do Ceará (Campus de Crateús). Atualmente, o NPDS possui quatro projetos em andamento e é composto por onze alunos estagiários, um gerente de projetos e um analista de sistemas. Destaca-se que todos os alunos estagiários cursaram a disciplina de Engenharia de Software. Considerou-se cada um dos projetos como um estudo de caso diferente, possibilitando assim, criar linhas de convergência e divergência sobre o material. No NPDS é utilizado a metodologia ágil *Scrum* para a gestão dos projetos, sendo estabelecido 10 dias como duração de cada *sprint*. A alocação das atividades a serem realizadas durante a *sprint* é definida pelo gerente de projetos em conjunto com os alunos.

Para avaliação do GQM+PA foram selecionados quatro projetos, identificados como **Delta**, **Alfa**, **Beta** e **Gama**. Optou-se por omitir maiores informações e os nomes reais dos projetos devido à restrições de confidencialidade. Para cada projeto estão alocados dois alunos estagiários. Todos os projetos são supervisionados pelo gerente de projetos e o analista de sistemas. Conforme será discutido posteriormente, o projeto **Delta** foi utilizado apenas como pré-teste durante uma única *sprint*. Através dos resultados deste pré-teste tornou-se possível identificar melhorias para avaliação dos projetos **Alfa**, **Beta** e **Gama**. Quem desempenhou o papel de Monitor foi a autora principal deste trabalho, a qual é concludente em Sistemas de Informação e já havia sido concluído o estágio supervisionado. Dessa forma, a mesma dedicou-se exclusivamente ao acompanhamento dos referidos projetos. Na Tabela 1, apresenta-se uma caracterização geral dos projetos e o respectivo time associado que participaram da implantação do GQM+PA:

Tabela 1 – Descrição dos Projetos

Projeto	Tempo em execução	Tecnologias	Time
Delta	7 meses	Angular 7 (JavaScript/Typescript), Spring Framework (Java) e MySQL	Aluno G: sexo masculino, 22 anos e está no estágio II Aluno H: sexo feminino, 22 anos e está no estágio I Aluno I: sexo masculino, 23 anos e está no estágio II Aluno J: sexo masculino e está no estágio II
Alfa	4 meses	Python com Django Rest Framework, JavaScript com Angular, MySQL e Clarity	Aluno A: sexo masculino, 23 anos e está no estágio II Aluno B: sexo masculino, 20 anos e está no estágio I
Beta	4 meses	Java com Spring, Angular, JavaScript, Clarity, React Native e GitLab	Aluno C: sexo masculino, 22 anos e está no estágio II Aluno D: sexo masculino, 21 anos e está no estágio I
Gama	4 meses	React Native, Spring, Angular, Angular material e Api do Google para mapas	Aluno E: sexo masculino, 26 anos e está no estágio II Aluno F: sexo masculino, 21 anos e está no estágio I

Fonte: Autoria própia.

6.2 Plano GQM

Para realizar a avaliação foi estabelecido como objetivo principal definir e analisar, no contexto do NPDS, métricas relacionadas ao processo e produto de software. Assim, almeja-se identificar, através das medições, características do projeto escolhido que possam contribuir para um melhor controle em termos de processo e produto, além de, por intermédio dos dados obtidos, detectar pontos de melhorias.

Para a definição dos objetivos da avaliação, conforme sugerido na Seção 5, considerou-se a perspectiva do gerente de projetos haja vista que é necessário o alinhamento da definição das métricas aos objetivos do projeto. Através de uma entrevista semi-estruturada realizada com o gerente do NPDS, identificou-se como objetivos: i) melhorar a precisão das estimativas de projeto e ii) aumentar a qualidade dos produtos liberados para uso. Assim, o primeiro objetivo relaciona-se ao processo de desenvolvimento, enquanto o segundo refere-se ao produto propriamente dito. O roteiro e uma síntese das respostas oriundas da entrevista encontram-se disponíveis no Apêndice D.

A partir da definição dos objetivos, torna-se necessário vincular a cada um desses

objetivos um conjunto de questões a serem respondidas através da avaliação de determinadas métricas. Portanto, tem-se uma relação entre os Objetivos (*Goal*), Questões (*Question*) e Métricas (*Metrics*). Vale destacar que as questões, bem como as métricas investigadas no presente trabalho, foram validadas junto ao gerente. Na entrevista, quando o gerente foi questionado sobre o que considerava como sendo as metas estratégicas do NPDS, disponível no Questionário 4 (Apêndice D), o mesmo considerou como metas fornecer um ambiente para formação técnica complementar aos alunos de Ciência da Computação e Sistemas de Informação através de estágios, desenvolver software qualidade para instituições parceiras e garantir satisfação dos clientes. Diante disso, percebe-se que há uma preocupação em proporcionar aos alunos uma formação adequada para o mercado de trabalho. Destacou-se também a importância de fornecer produtos de qualidade que satisfaçam os clientes. Além disso, dentre as possíveis estratégias citadas pelo gerente referem-se a vivência de metodologia ágil e boas práticas de Engenharia de Software. Já no que diz respeito as principais preocupações do gerente no NPDS, observa-se que foi respondido falta de métricas e práticas de Engenharia de Software que não foram implantadas.

A seguir é detalhado cada objetivo com suas determinadas questões e respectivas métricas.

6.2.0.1 Objetivo 1: Melhorar a precisão das estimativas de projeto

O primeiro objetivo está relacionado ao processo de software: melhorar a precisão das estimativas do projeto. No contexto do NPDS, as estimativas estão relacionadas ao tempo execução de uma atividade, ou seja, se o tempo estimado para uma determinada atividade foi realmente o tempo gasto para concluí-la. A partir desse objetivo foram definidas duas questões:

1. **Qual a precisão das estimativas do escopo do projeto?** Baseado nessa questão é possível analisar as estimativas relacionadas ao processo de desenvolvimento e se o objetivo foi alcançado. Para a Questão 1 foram especificadas as métricas *Efetividade de Pontos*, *Velocidade de Pontos* e *Entrega Prometida*. A Métrica 1.1 (*Efetividade de Pontos*) é obtida através da relação entre a pontuação executada sobre a pontuação estimada, onde o ponto executado refere-se ao ponto que foi realmente gasto para realizar a atividade. Já o ponto estimado refere-se ao ponto planejado para realizar atividade. A Métrica 1.2 (*Velocidade de Pontos*) refere-se a quão rápido foi executada uma determinada atividade, ou seja, se o valor da *Velocidade de Pontos* tiver aumentado de uma *sprint* para outra, significa que houve aumento de produtividade. A Métrica 1.3 (*Entrega Prometida*) analisa se a atividade

foi entregue na referida *sprint*.

2. **Qual a precisão das estimativas de esforço do projeto?** Em relação a tal questão avaliou-se as métricas *Efetividade de Horas* e *Velocidade de Horas*. Tais métricas apresentam concepção semelhante as métricas da Questão 1, com a diferença de que, ao invés de pontos executados e estimados, considera-se as horas executadas e estimadas. Justifica-se a opção de avaliar as métricas da Questão 2, tendo em vista que ao longo da *sprint* os alunos não dispõem dos dados referentes a pontuação executada.

Inspirado em Gomes *et al.* (2001), a Figura 9 sintetiza as informações previamente mencionadas sob o contexto do Objetivo 1:

Figura 9 – Síntese de Objetivo 1.

OBJETIVO 1: Melhorar precisão das estimativas do projeto
1) Qual a precisão das estimativas do escopo projeto?
1.1) Efetividade de Pontos **Pontuação executada: refere-se aos pontos que foram realmente gastos para realizar a atividade.** **Pontuação planejada: refere-se aos pontos estimados para realizar atividade.** **Como obter: pontuação executada / pontuação planejada**
1.2) Velocidade de Pontos **Refere-se a quão rápido foi executada uma determinada atividade** **Como obter: pontuação executada**
1.3) Entrega prometida **Analisar se a atividade foi realizada e entregue na sprint.** **Como obter: manualmente.**
2) Qual a precisão das estimativas de esforço do projeto?
2.1) Efetividade de Horas **Tempo executado: refere-se ao tempo que foi realmente gasto para realizar a atividade.** **Tempo planejado: refere-se ao tempo estimado para realizar atividade.** **Como obter: tempo executado / tempo planejado**
2.2) Velocidade de Horas **Refere-se a quão rápido foi executada uma determinada atividade** **Como obter: tempo executado**

Fonte: Autoria própria.

6.2.0.2 Objetivo 2: Aumentar a qualidade dos produtos liberados para uso

Conforme mencionado anteriormente, o segundo objetivo está relacionado a qualidade do produto liberado para uso. Para tal objetivo, foram definidas duas questões, conforme também sintetizado através da Figura 10:

1. **Qual a qualidade dos produtos antes de liberação para uso?** Para tal questão foi deter-

minada a adoção da métrica *Complexidade Ciclomática* a qual mensura a complexidade de um determinado módulo a partir da contagem do número de caminhos independentes que ele pode executar até o seu fim.

2. **Qual foi a aceitação das funcionalidades entregues?** Em relação a essa questão, especificou-se a Métrica 2.1 (*Issues Aprovadas*) a qual reflete a quantidade de atividades aprovadas pelo Product Owner (PO). No contexto do NPDS, o cargo de PO é exercido pelo próprio gerente de projetos.

Figura 10 – Síntese do Objetivo 2.

OBJETIVO 2: Aumentar a qualidade dos produtos liberados para uso
1) Qual a qualidade dos produtos antes de liberação para uso?
1.1) Complexidade Ciclomática **Mensura a complexidade de um determinado módulo (uma classe, um método, uma função etc), a partir da contagem do número de caminhos independentes que ele pode executar até o seu fim.**
2) Qual foi a aceitação das funcionalidades entregues?
2.1) Issues aprovados pelo PO

Fonte: Autoria própria.

6.3 Plano de Mensuração

Para uma integração apropriada de medidas de plano de projeto determinada pelo GQM, faz-se necessário o desenvolvimento de um Plano de Mensuração (WANGENHEIM, 2000). O Plano de Mensuração é definido pela integração das medidas definidas no Plano GQM e do plano de projeto de software através da determinação de quando, como, e por quem os dados requeridos podem ser coletados (WANGENHEIM, 2000). Diante disso, neste trabalho, para a realização de coletas de dados de forma confiável, adotou-se a estratégia de periodicidade que se caracteriza, nesse contexto, por rotinas de aprendizado em que os desenvolvedores preencherão planilhas a fim de coletar, internalizar e monitorar constantemente os dados das métricas a serem obtidas. Duas rotinas foram estabelecidas para cada objetivo: Rotinas Diárias e Rotinas de Retrospectiva. As Rotinas Diárias caracterizam-se por planilhas que possuem um conjunto de métricas refentes aos objetivos especificados pelo gerente. As mesmas serão preenchidas diariamente pelos alunos que coletarão as métricas determinadas durante uma *sprint*. Já nas Rotinas de Retrospectiva, os dados serão coletados pelos alunos apenas no final de cada *sprint* e incluídos na respectiva planilha. Adicionalmente, disponibilizou-se um guia para orientar os alunos sobre cada métrica, incluindo uma explicação sobre cada uma e, inclusive, como obtê-las.

Estes guias encontram-se disponíveis no Apêndice E.

Na Rotina Diária referente ao Objetivo 1, ilustrada na Figura 11, os alunos preencherão a cada dia dados relacionados ao tempo executado e planejado da atividade desenvolvido. Conforme ilustrado na Figura 12, ao final de cada *sprint*, os alunos realizarão a Rotina de Retrospectiva a qual diz respeito ao preenchimento dos dados relacionados a pontuação executada e planejada para cada atividade da *sprint*.

Figura 11 – Rotina Diária referente ao Objetivo 1.

ROTINA DIÁRIA: Melhorar precisão das estimativas do projeto	
Preencha a planilha diariamente	**Respostas**
1) Atividade: *Atividade a ser realizada pelo aluno.*	
2) Tempo executado na atividade: *Quanto tempo foi gasto naquela atividade por dia.*	
3) Tempo planejado para atividade: *Quanto tempo foi estimado para realizar a atividade por dia.*	

Fonte: Autoria própria.

Figura 12 – Rotina de Retrospectiva referente ao Objetivo 1.

ROTINA DE RETROSPECTIVA: Melhorar precisão das estimativas do projeto	
Seção 1: Precisão das estimativas do escopo e esforço do projeto	**Respostas**
1) Atividade: *Atividade a ser realizada pelo aluno.*	
2) Pontuação executada: *Total de pontos gastos para realizar a tarefa.*	
3) Pontuação planejada: *Total de pontos que foi estimado para realizar a tarefa.*	

Fonte: Autoria própria.

Na Rotina Diária referente ao Objetivo 2, a qual encontra-se descrita na Figura 13, os alunos preencherão a cada dia dados relacionados a atividade desenvolvida a fim de verificar a qualidade do produto desenvolvido. Neste caso, através da métrica *Complexidade Ciclomática*. No fim da *sprint*, quando as atividades tiverem sido encerradas, é realizada a Rotina de Retrospectiva (Figura 14) onde se analisa a quantidade de *issues* aprovadas pelo PO e se a atividade foi entregue ou não.

Figura 13 – Rotina Diária referente ao Objetivo 2.

ROTINA DIÁRIA: Aumentar a qualidade dos produtos liberados para uso	
Preencha a planilha diariamente	**Respostas**
1) Atividade: *Atividade a ser realizada pelo aluno.*	
2) Complexidade Ciclomática: *Mensura a complexidade de um determinado módulo (uma classe, um método, uma função etc), a partir da contagem do número de caminhos independentes que ele pode executar até o seu fim.*	

Fonte: Autoria própria.

Figura 14 – Rotina de Retrospectiva referente ao Objetivo 2.

ROTINA DE RETROSPECTIVA: Aumentar a qualidade dos produtos liberados para uso	
Seção 1: Qualidade dos produtos	**Respostas**
1) Atividade: *Atividade a ser realizada pelo aluno.*	
2) Quantidade de issues aprovadas pelo PO:	
3) A atividade foi entregue?	

Fonte: Autoria própria.

6.4 Diagnóstico Preliminar

Antes de iniciar a etapa de Implementação e Observação, faz-se necessário a realização da atividade Diagnóstico Preliminar com o propósito de identificar os conhecimentos gerais dos alunos e prover uma breve explicação do processo a ser realizado, incluindo o que é GQM, as rotinas de aprendizado, preenchimento da planilhas, etc. Para a realização dessa aprendizagem, buscou-se repassar os conceitos fundamentais de medição de software por meio de um *workshop* de 2 horas. Dessa forma, o propósito inicial foi apenas de fornecer uma base suficiente para que os alunos aprendam e exerçam na prática as rotinas estabelecidas pelo GQM+PA e, consequentemente, compreendam o processo de medição de software. Também, foi destacado a importância do GQM para a formação profissional, para a qualidade dos produtos e para um melhor processo de desenvolvimento. Salienta-se também o papel fundamental do monitor em acompanhar a equipe e intervir quando necessário.

Antes dos alunos iniciarem a implantação das etapas do GQM foi aplicado dois questionários. No primeiro questionário, disponível no Apêndice A, busca-se ter uma compreensão sobre o conhecimento dos alunos sobre medição de software e da abordagem GQM. Com base nas respostas dos alunos será possível identificar em que aspecto pode-se melhorar a aprendizagem dos alunos em relação a medição de software. Tendo em vista que os alunos serão responsáveis por aplicar o GQM, torna-se imprescindível que tenham uma boa compreensão de

como executar as etapas de medição de software e sua relevância para garantia da qualidade dos produtos desenvolvidos. Dessa forma, após implantar no ambiente de desenvolvimento um processo de medição, espera-se que os alunos compreendam empiricamente a importância da medição para desenvolver produtos de qualidade, além da contribuição profissional para os mesmos, capacitando-os ainda mais para o mercado de trabalho.

Já no segundo questionário, disponível no Apêndice B, procura-se extrair informações sobre o ambiente de desenvolvimento. Destaca-se no questionário assuntos relacionados principalmente a medição e a qualidade dos produtos a fim de identificar se realizam medição e as possíveis dificuldades associadas. Além disso, busca-se verificar se consideram viável e importante para os mesmos a adoção de práticas de medição.

Após o processo de capacitação inicial previamente contextualizado, as planilhas foram disponibilizadas para que os alunos pudessem realizar as Rotinas Diárias e Rotinas de Retrospectiva. Caso verifique-se dificuldades em efetuar alguma tarefa, o monitor intervém a fim de esclarecer as dúvidas.

Através das rotinas de aprendizado, espera-se que, em um contexto prático de aprendizado, os alunos apresentem uma melhor compreensão das métricas e de qual impacto elas têm na qualidade dos produtos. Com base nos resultados, torna-se possível verificar em que aspectos relacionados a produção de software precisam melhorar. Como as métricas estão relacionadas a melhorar a precisão das estimativas do projeto e aumentar qualidade dos produtos, espera-se que, com os dados obtidos, os alunos sejam capazes de verificar o impacto que a medição tem em identificar o que pode ser melhorado e como um processo de medição traz um melhor entendimento sobre o processo de desenvolvimento de software.

Enquanto os alunos obtêm as métricas referentes aos objetivos, o monitor usufrui do método de Pesquisa-Ação visando acompanhar e intervir conforme necessário na realização das etapas do GQM. Logo, o monitor terá a responsabilidade de acompanhar comportamento dos alunos a fim de lidar em conjunto com as problemáticas identificadas.

Para analisar se o processo de aprendizagem dos alunos foi bem sucedido, ou seja, se os alunos compreenderam como se caracteriza um processo de medição através da implantação do GQM, o monitor, além da observação, utilizará de um questionário semi-estruturado, disponível no Apêndice C, com o objetivo de verificar o nível dos alunos em relação ao conhecimento sobre medição de software e, assim, tornar possível comparar o conhecimento antes e após adoção do GQM+PA. Portanto, através da Pesquisa Ação em conjunto com o GQM, viabiliza-se um

contexto prático de ensino de engenharia de software com o propósito de tornar as práticas relativas a medição de software inerentes ao dia a dia dos alunos no processo de desenvolvimento de sistemas.

7 RESULTADOS E ANÁLISES

A seguir é descrito e analisados os resultados deste trabalho. Inicialmente é apresentado os resultados do pré-teste realizado com o projeto **Delta**. Nas seções seguintes são analisados os resultados dos demais projetos, incluindo as questões técnicas e de aprendizado dos alunos.

7.1 Lições aprendidas no Projeto Delta (Pré-teste)

Antes de iniciar a implementação das práticas do GQM+PA, optou-se por realizar um pré-teste durante duas *sprints*. Durante a etapa de pré-teste, os alunos eram acompanhados pelo monitor, diariamente, e caso tivessem dúvidas o monitor os ajudariam. O monitor além de observar ficou responsável por apresentar como seria a coleta das métricas, apresentado as planilhas e explicando como deviam ser preenchidas.

As principais dificuldades apresentadas pelos alunos corresponderam a como preencher as planilhas, sobre o que é determinada métrica, como utilizar os *plugins* para obter as métricas. Além disso, foi observado falta de engajamento em realizar a coleta por alguns. Percebe-se que, no início, era necessário lembrá-los de preencher as planilhas até que se tornasse um processo rotineiro.

A maioria das atividades estavam relacionadas a correção de *bugs*. Isso acabava confundindo os alunos, pois tinham dificuldade de relacionar as métricas a essas atividades, por consequência, alguns não preenchiam devido acharem que não era necessário medir. Às vezes, dois alunos trabalhavam em uma mesma atividade e isso ocasionava que só um aluno preenchia as planilhas.

A fim de melhorar as práticas relacionadas ao implantar o GQM+PA, notou-se a necessidade em formalizar previamente e explicitar a importância de medição de software, o funcionamento e a definição de cada métrica para que os alunos tenham a capacidade de identificar pontos de melhorias em relação ao produto desenvolvido pelos mesmos.

7.2 Resultado do teste

7.2.1 Avaliação técnica

7.2.1.1 Objetivo 1: Melhorar a precisão das estimativas do projeto

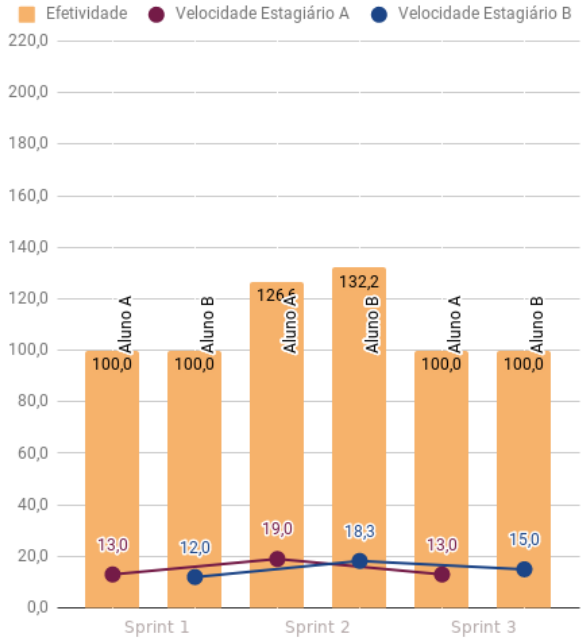
Determinou-se o Objetivo 1 como forma de analisar as estimativas relacionadas ao tempo de execução de uma atividade, sendo o mesmo dividido em duas questões: Qual a precisão das estimativas de escopo do projeto? E qual a precisão das estimativas de esforço do projeto? Na primeira questão são analisadas métricas referentes a pontuação estimada e a pontuação executada e se uma atividade foi entregue no final de uma *sprint*. Já na segunda questão analisa-se as métricas referentes ao tempo executado e estimado de uma atividade.

7.2.1.1.1 Qual a precisão das estimativas de escopo do projeto?

Para melhorar a precisão de estimativas do projeto foi determinada a questão: Qual a precisão das estimativas de escopo do projeto? Através de tal questão busca-se verificar as estimativas associadas ao processo de desenvolvimento.

Conforme demonstrado na Figura 15, referente ao **Projeto Alfa**, o Aluno A apresentou um aumento na *Velocidade de Pontos* da primeira para a segunda *sprint*, porém, na terceira *sprint* ocorreu uma queda desse valor, sendo que o resultado ideal para a referida métrica seja igual ou próximo a 100. Em relação a *Efetividade de Pontos*, pode-se verificar que na primeira e terceira *sprint*, a pontuação foi de acordo com o planejado. Na segunda *sprint*, por sua vez, o aluno apresentou dificuldades e demonstrou ter sido necessário mais pontos para realizar as atividades. Em relação ao Aluno B, constata-se um comportamento semelhante ao Aluno A em ambas métricas *Efetividade de Pontos* e *Velocidade de Pontos*.

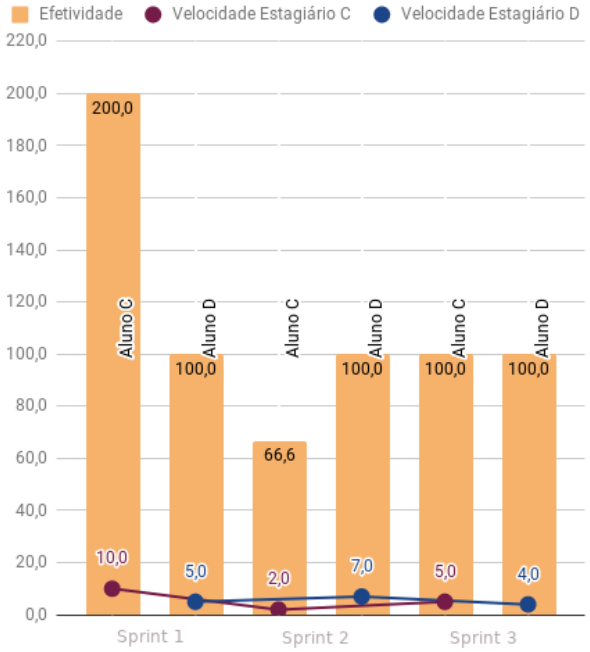
Figura 15 – Efetividade x Velocidade - Projeto Alfa



Fonte: Autoria própria.

Ao analisar a Figura 16 referente ao **Projeto Beta**, percebe-se que o Aluno C apresentou uma queda significativa na *Velocidade de Pontos* da primeira para a segunda *sprint*. Já na terceira *sprint*, verifica-se que o mesmo apresentou aumento na *Velocidade de Pontos* devido ter recebido mais atividades. No que se refere a *Efetividade de Pontos*, nota-se que o Aluno C apresentou dificuldades em estimar suas atividades, sendo que na primeira *sprint* o mesmo teve a pontuação planejada bem menor que a executada. Na segunda *sprint* a pontuação executada foi maior que a planejada e, apenas na terceira *sprint*, o Aluno C estimou corretamente, ou seja, a pontuação planejada foi igual a executada. Em relação ao Aluno D, observa-se que a *Velocidade de Pontos* da primeira para a segunda *sprint* aumentou, pois o mesmo recebeu mais atividades na segunda *sprint*. Na terceira *sprint* houve uma queda nesse valor devido o aluno ter recebido menos atividade comparada a *sprint* anterior. Já o resultado de *Efetividade de Pontos*, percebe-se que o aluno não apresentou dificuldades em estimar suas atividade diferentemente do Aluno C, pois a pontuação planejada do Aluno D nas três *sprints* foi igual a pontuação executada.

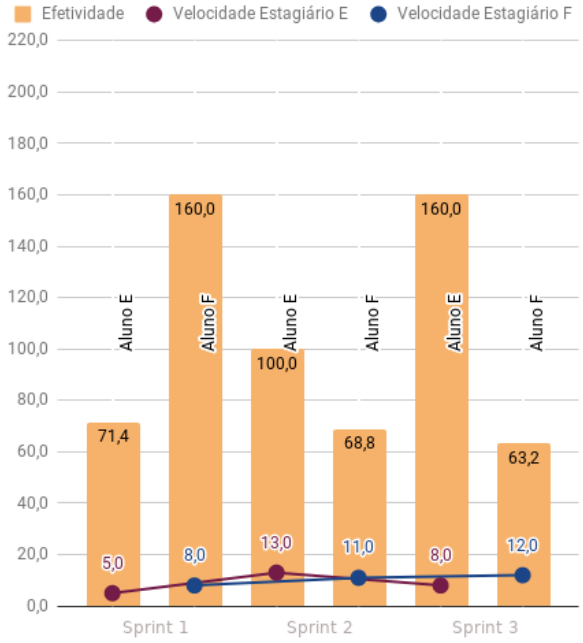
Figura 16 – Efetividade x Velocidade - Projeto Beta



Fonte: Autoria própria.

No que se refere ao **Projeto Gama**, representado na Figura 17, nota-se que o Aluno E apresentou aumento significativo na *Velocidade de Pontos* da primeira sprint para a segunda, demonstrando um aumento na produtividade nesse período, já na terceira sprint ocorreu uma queda nesse valor e como consequência disso houve uma redução na produtividade. Com relação a *Efetividade de Pontos*, ocorreu variações no resultado de tal métrica. Na primeira *sprint*, a pontuação executada do Aluno E foi menor que a estimada, já na segunda o mesmo estimou de acordo com o valor executado, mas na terceira a pontuação executada foi bem maior que a planejada. Assim, nota-se que o aluno apresentou dificuldades em estimar suas tarefas. Já o resultado do Aluno F demonstrou crescimento gradativo em cada *sprint* na *Velocidade de Pontos*, ou seja, houve um aumento de produtividade tal qual o Aluno E apresentou na primeira e segunda *sprint*. Quanto a *Efetividade de Pontos*, o Aluno F apresentou também variações conforme o Aluno E. Na primeira *sprint*, a pontuação executada foi maior que a planejada pelo aluno, já na segunda *sprint* e na terceira a pontuação executada foi bem menor que a planejada. Observa-se, então, que o aluno também apresentou dificuldades em estimar corretamente suas atividades.

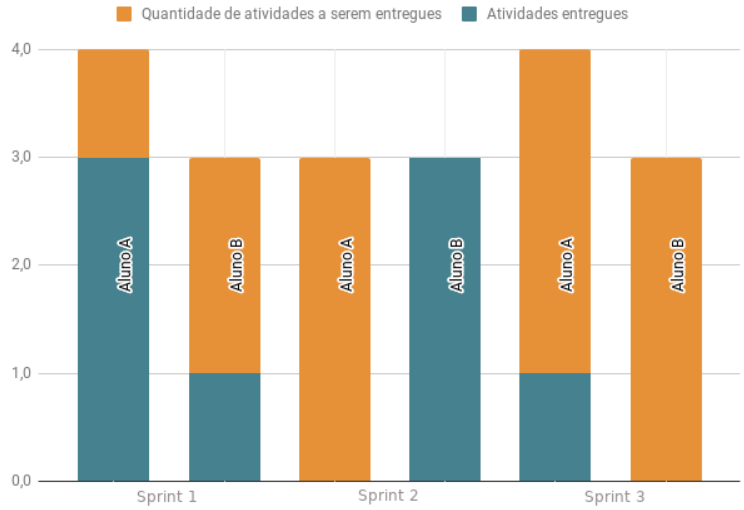
Figura 17 – Efetividade x Velocidade - Projeto Gama



Fonte: Autoria própria.

Adicionalmente, como forma de verificar se uma determinada atividade foi realizada e entregue em uma *sprint* foi adotada a análise da métrica *Entrega Prometida*. No **Projeto Alfa**, conforme ilustrado a Figura 18, tinha sido atribuído ao Aluno A na primeira *sprint* um total de quatro tarefas, mas o mesmo entregou três atividades. Na segunda *sprint* foram atribuídas três atividades, só que nenhuma atividade foi entregue no final da *sprint*. Na terceira *sprint* foram determinadas quatro tarefas e apenas uma foi entregue. Já o Aluno B, na primeira *sprint* recebeu três atividade e entregou somente uma. Na segunda *sprint*, o aluno também recebeu três atividade e as três foram entregues. Na terceira *sprint* foi alocado para o mesmo três atividades, porém nenhuma foi entregue.

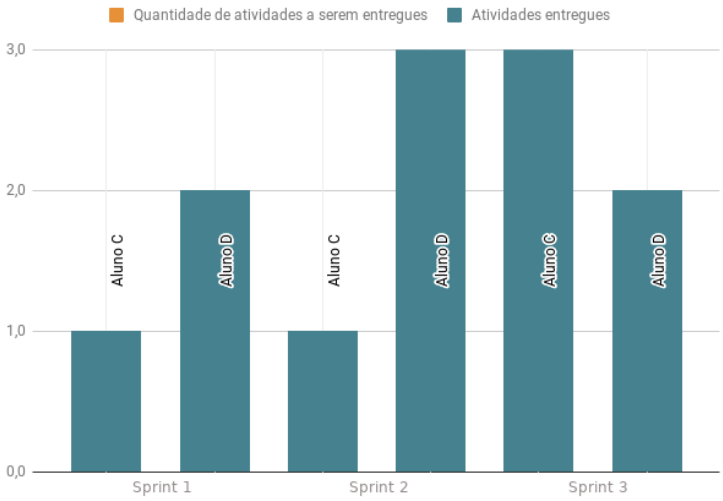
Figura 18 – Quantidade de atividades a serem entregues x Atividades entregues - Projeto Alfa



Fonte: Autoria própria.

No **Projeto Beta**, representado na Figura 19, o Aluno C recebeu uma atividade na primeira *sprint* a qual foi entregue. Na segunda *sprint* foi atribuída ao mesmo também uma tarefa, que, novamente, foi devidamente entregue. Na terceira *sprint*, o Aluno C já recebeu três atividades, sendo que todas foram entregues no final da *sprint*. Ao Aluno D, na primeira *sprint*, foram estabelecidas duas atividades que foram entregues. Na segunda *sprint* o mesmo recebeu três atividades que também foram entregues. Na terceira *sprint* o aluno ficou responsável por duas atividades as quais também foram entregues no final da *sprint*.

Figura 19 – Quantidade de atividades a serem entregues x Atividades entregues - Projeto Beta

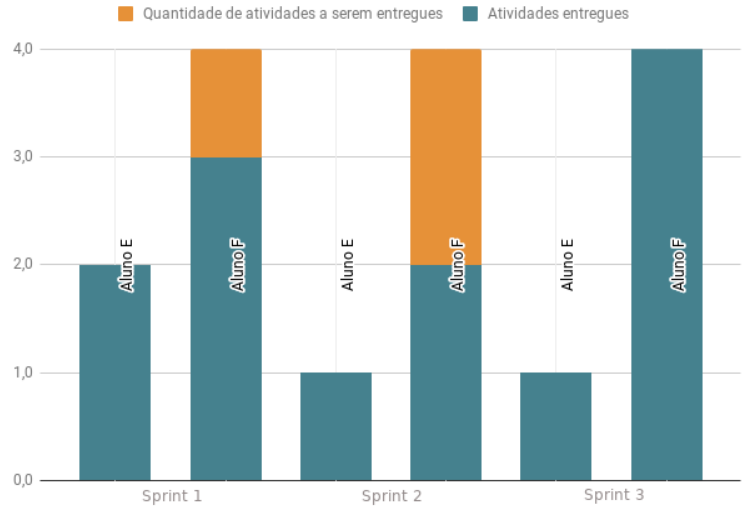


Fonte: Autoria própria.

No **Projeto Gama**, conforme a Figura 20, o Aluno E recebeu duas atividades na primeira *sprint* e ambas foram entregues. Na segunda *sprint*, por sua vez, foi determinada uma

tarefa a qual foi devidamente entregue. Na terceira *sprint* também foi especificada uma única tarefa a qual foi concretizada. Já o Aluno F, na primeira *sprint*, recebeu quatro tarefas das quais três foram entregues. Na segunda *sprint* foram definidas quatro atividades, sendo que apenas duas foram entregues. Na terceira *sprint* foram estabelecidas quatro atividades e todas foram entregues no final da *sprint*.

Figura 20 – Quantidade de atividades a serem entregues x Atividades entregues - Projeto Gama



Fonte: Autoria própria.

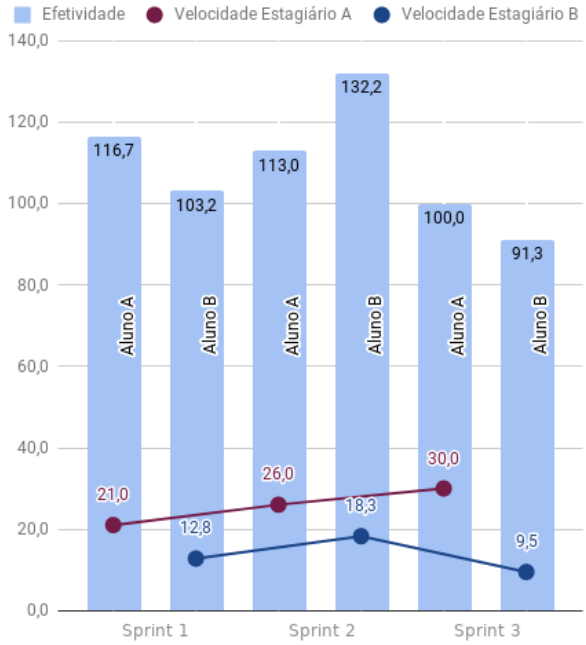
7.2.1.1.2 Qual a precisão das estimativas de esforço do projeto?

A fim de melhorar as estimativas de projeto especificou-se outra questão: Qual a precisão das estimativas de esforço do projeto? Através desta será possível também analisar as estimativas acerca do processo de desenvolvimento. Diferentemente da Questão 1, onde analisou-se a pontuação, a partir de agora serão analisadas as métricas relacionadas as horas estimadas e executadas por meio das métricas *Efetividade de Horas* e *Velocidade de Horas.*

Conforme pode-se verificar na Figura 21 referente ao **Projeto Alfa**, o Aluno A apresentou um aumento gradativo na *Velocidade de Horas* em cada *sprint*, ou seja, em cada *sprint* houve um aumento na produtividade atingiu de suas atividades. No que se refere a *Efetividade de Horas* houve variações em cada *sprint*. O ideal é que a efetividade fique próxima ou igual a 100. Nota-se que o aluno A atingiu na terceira *sprint Efetividade de Horas* igual a 100. Logo, percebe-se que o aluno teve inicialmente dificuldades em estimar suas atividades. O Aluno B apresentou um desempenho diferente do Aluno A, como pode-se analisar no referido gráfico, pois houve um aumento de *Velocidade de Horas* da primeira *sprint* para a segunda *sprint*, porém na terceira *sprint* houve um decréscimo. Na segunda *sprint*, foi a que mais atividades o Aluno B

recebeu e na terceira *sprint* a demanda foi menor, havendo assim variações na velocidade. Já a *Efetividade de Horas* na primeira *sprint* apresentou um resultado adequado tendo em vista que ficou próximo a 100. Já na segunda *sprint* o Aluno B executou um valor acima do que era desejado, enquanto na terceira *sprint* foi executado abaixo do ideal. Portanto, através dos resultados obtidos, constata-se a dificuldade nas estimativas concernentes as tarefas a serem concretizadas.

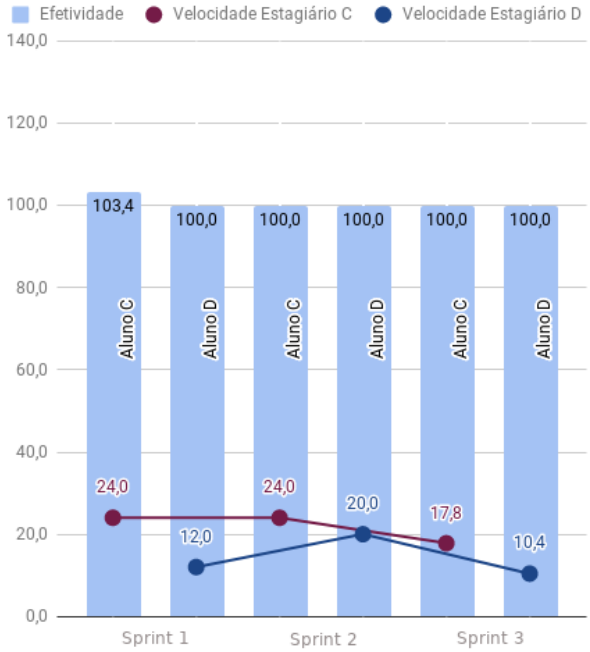
Figura 21 – Efetividade x Velocidade - Projeto Alfa



Fonte: Autoria própria.

De acordo com a Figura 22 do **Projeto Beta**, observa-se que o Aluno C apresentou *Velocidade de Horas* constante nas duas primeiras *sprints* enquanto na terceira *sprint* houve uma queda desse valor, contribuindo, assim, para uma produtividade baixa comparada às anteriores devido a execução de menos atividades. Já a *Efetividade de Horas* foi constante durante as três *sprints* com valor próximo ou igual a 100, refletindo uma estimativa adequada, ou seja, o tempo que foi estimado correspondeu ao tempo executado para realizar determinada tarefa. Em relação ao Aluno D o resultado do mesmo apresentou variações quanto *Velocidade de Horas*. Da primeira para a segunda *sprint* houve um aumento significativo, contribuindo assim para um aumento na produtividade. Entretanto, na terceira *sprint*, ocorreu uma queda desse valor em virtude de haver menos atividades executadas e também pelas faltas do Aluno D. Já a *Efetividade de Horas* durante as três *sprints* foi precisa, isto é, nas três o valor foi igual a 100. Logo, o aluno não apresentou dificuldade em estimar o tempo de suas atividades.

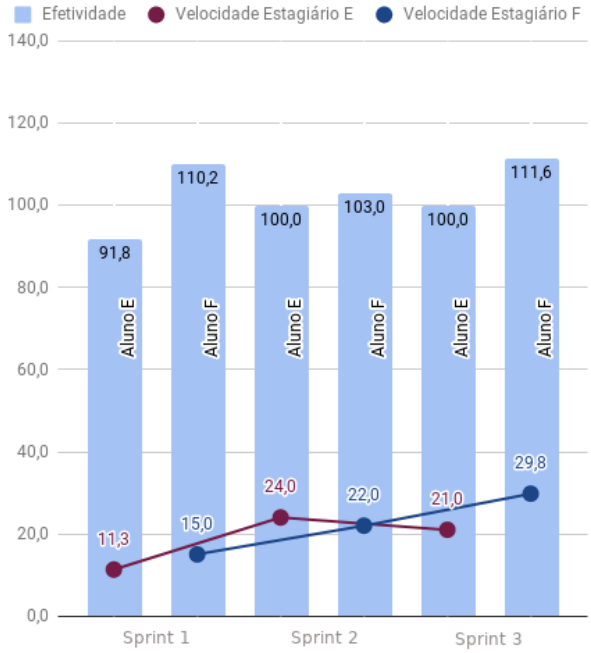
Figura 22 – Efetividade x Velocidade - Projeto Beta



Fonte: Autoria própria.

Ao analisar a Figura 23 referente ao **Projeto Gama**, verifica-se que o Aluno E apresentou aumento significativo da *Velocidade de Horas* da primeira para a segunda *sprint*. No entanto, na terceira *sprint* houve uma pequena queda devido o aluno receber uma quantidade menor de tarefas nessa *sprint*. Os resultados da *Efetividade de Horas* nas três *sprints* foi bem próximo de 100 na primeira *sprint*, já nas outras *sprints* foi precisa (igual a 100), ou seja, o aluno conseguiu melhorar suas estimativas. O Aluno F apresentou *Velocidade de Horas* crescente em cada *sprint*, ou seja, em cada *sprint* sua produtividade foi aumentando significativamente. Já os valores de *Efetividade de Horas* apresentaram variações. Na primeira *sprint* o Aluno F teve tempo gasto acima do desejado, enquanto na segunda *sprint* foi mais próximo de 100 e, finalmente, na terceira *sprint* apresentou o tempo gasto acima do valor desejado. Tal resultado demonstra que o Aluno F apresentou dificuldades em fazer suas estimativas.

Figura 23 – Efetividade x Velocidade - Projeto Gama



Fonte: Autoria própria.

Além dessas análises, apresenta-se as percepções oriundas da monitora. Através de tal observação nota-se que alguns alunos apresentaram uma variação ao estimar as atividades, pois alguns alunos alegaram que, inicialmente, apresentavam dificuldades em estimar suas tarefas. Concluí-se, assim, a presença de dificuldades no processo de estimativas de atividades conforme foi demonstrado nos resultados anteriormente. Além disso, no questionário 2 (Apêndice B) os alunos responderam que tinham dificuldade alta ou média em estimar suas atividades contra apenas um aluno que colocou baixa dificuldade. Portanto, pode-se observar que de fato que há uma concordância entre tal relato e as respostas do questionário. Já outros alunos estimaram corretamente contradizendo, assim, ao que foi respondido no questionário. Diante disso, quando indagados no questionário 3 (Apêndice C) o porquê que no questionário 2 (Apêndice B) alguns responderam que haviam dificuldade de estimar as atividades (pontuar as atividades), mas nas planilhas as estimativas das atividades estavam próximas ou iguais a pontuação real. O Aluno A justificou que "Uma adequação no decorrer do trabalho. Inicialmente e por certo tempo a pontuação das tarefas permaneceu confusa entre projetos. Faltou uma melhor definição por parte de cada grupo. À medida que os projetos andaram, cada grupo encontrou a melhor forma de estimar cada atividade". Já o Aluno D afirmou que "No começo, foi um chute aleatório de pontuações, que se moldou ao longo das *sprints* e fomos atingindo maturidade e consequentemente estimando melhor as pontuações".

Um outro fator que contribui para os resultados apresentados anteriormente que foi observado pelo monitor durante a coleta de métricas na qual dificultou tal processo, refere-se aos feriados que ocorreram durantes as *sprints* que acabaram influenciando nos resultados dos alunos como a *Velocidade de Pontos*, além de faltas devido provas ou trabalhos, e em algumas *sprints* ocorreu de alunos receberem menos atividades comparada as *sprints* anteriores que acabou influenciando os resultados dos alunos.

7.2.1.2 Objetivo 2: Melhorar a qualidade dos produtos liberados para uso

O Objetivo 2 foi especificado com a finalidade de desenvolver produtos com um nível de qualidade maior para os clientes. Por meio das métricas determinadas para esse objetivo será possível verificar a qualidade do código produzido pelos alunos estagiários.

7.2.1.2.1 Qual a qualidade dos produtos antes de liberação para uso?

A questão Qual a qualidade dos produtos antes de liberação para uso? foi determinada com o objetivo de verificar a qualidade do que é desenvolvido. Para tal questão, especificou-se a métrica *Complexidade Ciclomática*.

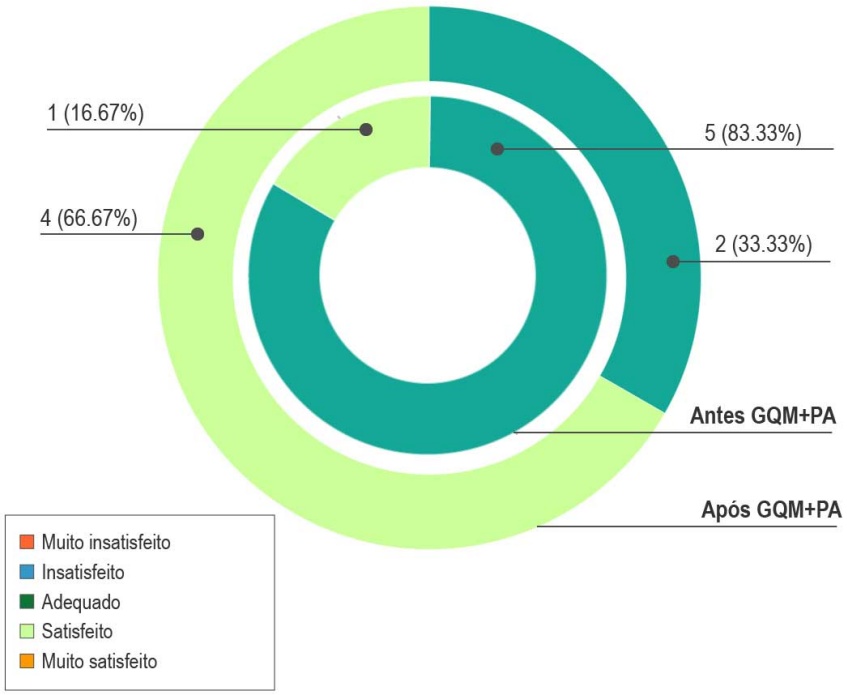
Para análise de tal métrica foi considerada a avaliação do monitor e as respostas do Questionário 3 (Apêndice C). Através da percepção do monitor, notou-se que quando os alunos foram questionados sobre o resultado da *Complexidade Ciclomática*, alguns discentes descreveram que se preocupavam com o valor da métrica, pois se indagavam sobre o que poderia ser feito para melhorar tais resultados. O Aluno A, por exemplo, relatou: "Comecei a observar melhor os procedimentos que produzi e passei a tentar diminuir a quantidade tarefas que cada um resolvia. Consegui, com isso, separar melhor as responsabilidades de cada procedimento diminuindo a complexidade num geral". Já o Aluno F, destacou que "Me prendi mais ao detalhe da complexidade e tentei não extrapolar os valores medianos, fazendo métodos menores, dividindo bem as funções". Logo, conclui-se que, ao contextualizar sobre dados obtidos via *Complexidade Ciclomática* na rotina de desenvolvimento, os alunos começaram a se preocupar com a melhoria do código desenvolvido.

Foi observado também que no início foi apresentado dúvida relacionada ao *plugin* para obtenção da *Complexidade Ciclomática* sobre, por exemplo, como deveriam analisar o valor de tal métrica. Para instalação foi necessário o acompanhamento do monitor, onde foi explicado sobre o funcionamento do *plugin*, como são exibidos os dados da métrica além de

ser ressaltado o que é *Complexidade Ciclomática*. Os alunos relataram também dificuldades sobre como incluir os resultados da *Complexidade Ciclomática* na planilha, tendo em vista que trabalharam em diversos arquivos e com isso teriam muitos dados. Então, foi decidido em conjunto com o monitor que colocassem os nomes dos arquivos trabalhos com seus respectivos dados de *Complexidade Ciclomática*.

Além disso, como resultado dos Questionários 2 e 3 (Apêndice B e C) sobre a percepção dos alunos quanto ao nível de satisfação em relação ao que era produzido, foi conduzido um comparativo antes e após a execução do GQM+PA, conforme ilustrado na Figura 24. Observa-se que antes de aplicar GQM+PA, 5 (83%) alunos responderam que achavam adequado o que desenvolvia e apenas 1 (16%) considerou que estava satisfeito com o que desenvolvia. Após a experiência de adoção do GQM+PA, 4 (66%) responderam que estavam satisfeitos e 2 (33%) declararam como adequado. Percebe-se que houve uma mudança na percepção dos alunos em relação ao que é produzido por eles, favorecendo a internalização sobre a importância em medir software para produzir um produto melhor.

Figura 24 – Quão satisfeito você está com a qualidade do produto que você entrega?



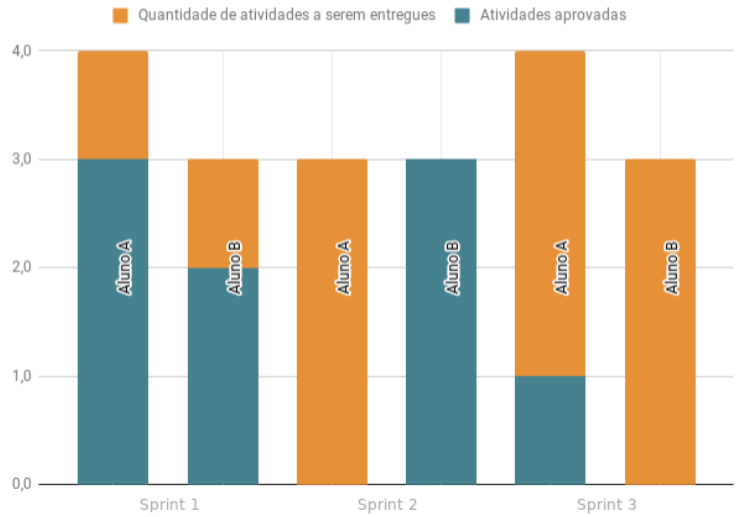
Fonte: Autoria própria.

7.2.1.2.2 Qual foi a aceitação das funcionalidades entregues?

Com o propósito de melhorar a qualidade dos produtos desenvolvidos foi avaliada outra questão: Qual foi a aceitação das funcionalidades entregues? Através da mesma será possível verificar se atividades entregues no final de uma *sprint* foram aprovadas pelo PO.

No **Projeto Alfa**, conforme a Figura 25, o Aluno A recebeu quatro atividades e três foram aprovadas pelo PO no final da primeira *sprint*. Na segunda , das três atividades que foi atribuída ao Aluno A, nenhuma foi entregue. Na terceira *sprint* foi aprovada apenas uma tarefa, sendo que foram alocadas quatro atividades. Já o Aluno B, na primeira *sprint* recebeu três atividades e duas foram entregues e aprovadas pelo PO. Na segunda *sprint* o aluno ficou responsável por três atividades e todas foram entregues e aceitas. Na terceira *sprint* o aluno também recebeu três tarefas, porém nenhuma foi entregue no final da *sprint*. No geral, na primeira *sprint* das sete atividades alocadas para os alunos cinco foram entregues e aprovadas. Na segunda *sprint* foram alocadas seis tarefas e apenas três entregues e aprovadas. Na terceira *sprint* das sete atividades alocadas apenas uma foi entregue e aprovada.

Figura 25 – Quantidade de atividades a serem entregues x Atividades aprovadas

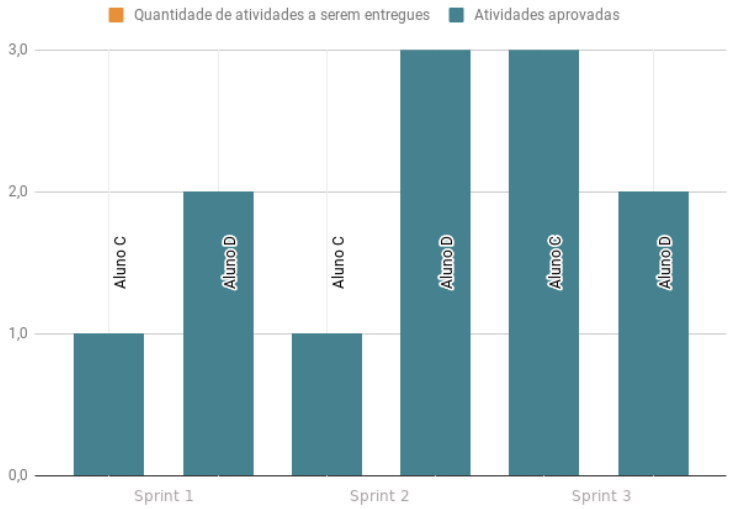


Fonte: Autoria própria.

No **Projeto Beta**, de acordo com a Figura 26, o Aluno C recebeu uma atividade que foi entregue e aceita pelo PO ao final da primeira *sprint*. Na segunda *sprint* o Aluno C também entregou uma tarefa na qual foi aprovada no final da *sprint*. Na terceira *sprint*, o aluno recebeu três atividades as quais foram todas aprovadas. Já o Aluno D, na primeira *sprint*, tinha duas atividades alocadas as quais foram aceitas. Na segunda *sprint*, o Aluno D recebeu três

tarefas, sendo todas entregues e aprovadas. Na terceira *sprint*, foram atribuídas ao Aluno D duas atividades que também foram entregues e aceitas pelo PO. No contexto geral, das onze atividades alocadas para os alunos no decorrer das três *sprints* todas foram entregues e aprovadas.

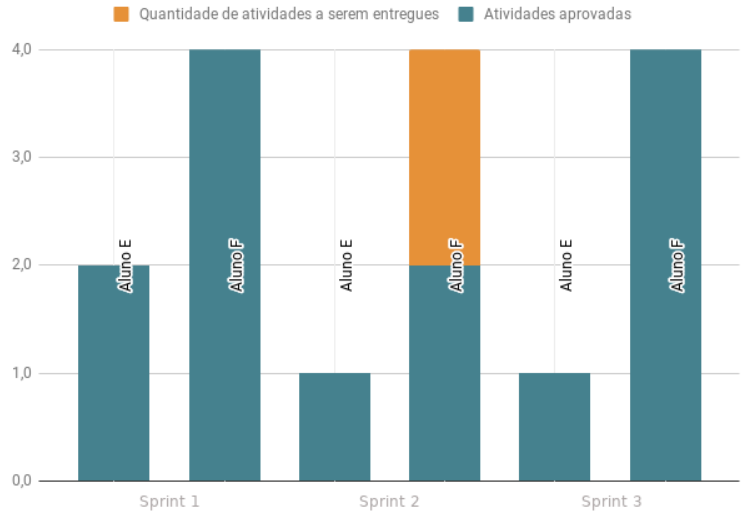
Figura 26 – Quantidade de atividades a serem entregues x Atividades aprovadas



Fonte: Autoria própria.

Em relação ao **Projeto Gama**, conforme Figura 27, o Aluno E na primeira *sprint* recebeu duas atividades e ambas foram aprovadas pelo PO. Na segunda *sprint* foi estabelecida uma atividade para o mesmo que foi entregue e aprovada. Na terceira *sprint* o Aluno E também recebeu uma atividade que foi entregue e aprovada . Já o Aluno F, na primeira *sprint*, das quatro atividades que recebeu, todas foram aceitas pelo PO. Na segunda *sprint* também foram especificadas quatro tarefas, mas apenas duas foram entregues e aprovadas. Na terceira *sprint* foram alocadas quatro tarefas, sendo todas devidamente aprovadas. Em geral, na primeira *sprint* das seis atividades demandadas aos alunos todas foram entregues e aceitas. Na segunda *sprint* das cinco atividades alocadas três foram entregues e aprovadas. Já na terceira *sprint* das cinco tarefas alocadas todas foram entregues e aceitas.

Figura 27 – Quantidade de atividades a serem entregues x Atividades aprovadas



Fonte: Autoria própria.

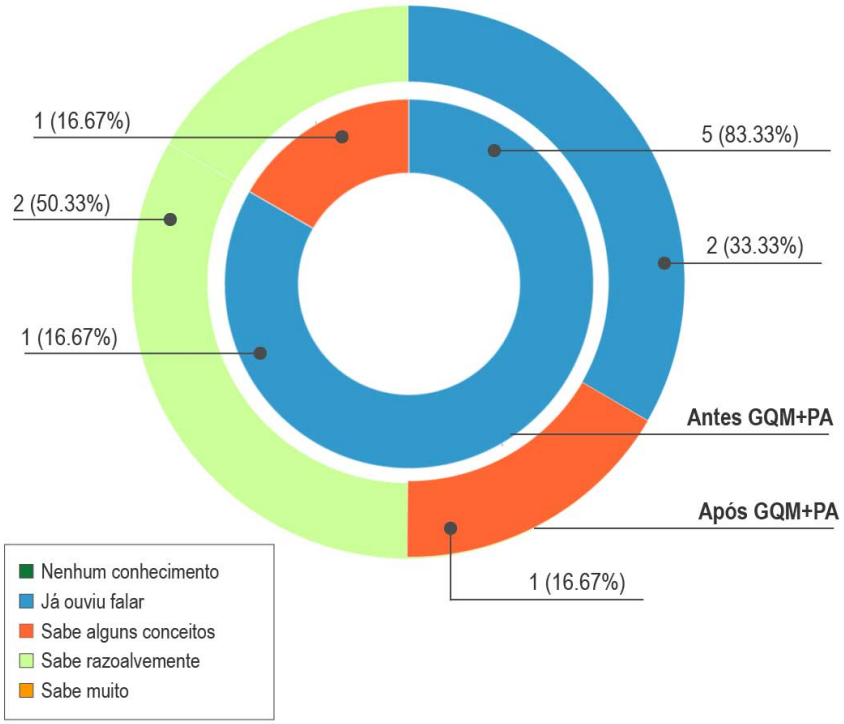
7.2.2 Reflexão de Aprendizado dos Discentes

Para avaliar o nível de aprendizado dos alunos em relação a medição de software, métricas e GQM foram aplicados três questionários. O Questionário 1 (disponível no Apêndice A), trata de questões relacionadas aos conhecimentos gerais sobre medição de software. O Questionário 2 (disponível no Apêndice B) visa compreender a percepção dos alunos quanto a qualidade de software no contexto do processo de desenvolvimento. Finalmente, o Questionário 3 (disponível no Apêndice C) visa estabelecer um diagnóstico do aprendizado dos alunos após a adoção do GQM+PA. Dessa forma, os Questionários 1 e 2 foram aplicados antes da adoção do GQM+PA, enquanto o Questionário 3 foi aplicado no fim.

Ao analisar as respostas oriundas do Questionário 1 (Apêndice A) referente a relevância da adoção de processo de medição de software para um projeto de sucesso, nota-se que 4 (66%) alunos consideram importante adotar tal processo, 1 (16%) acha razoavelmente importante e 1 (16%) considera importante. Além disso, todos os alunos consideram necessário ter uma compreensão sobre medição para suas carreiras. O Aluno F destacou que "Um profissional completo e competitivo para o mercado tem que dispor de conhecimentos para medir seu software, seus códigos, para buscar torná-lo mais otimizado e eficiente". Já o Aluno A expõe que "Conhecimentos que regem o funcionamento de um projeto e que podem melhorar os processos que estruturam esse funcionamento são importantes pois tornam a produção mais eficiente e todas as áreas da produção do software precisam saber o que acontece quando tais medições são aplicadas para se adequarem a elas e seu uso mais rapidamente".

Dos resultados acerca do nível de conhecimento sobre medição de software disponíveis nos Questionários 1 e 3 (Apêndices A e C), conforme a Figura 28, verifica-se que, antes de utilizar o GQM+PA, 5 (83%) alunos já haviam ouvido falar sobre medição de software, enquanto apenas 1 (16%) aluno alegou saber alguns conceitos. Ao monitor, os alunos alegaram que já haviam ouvido falar sobre medição de software na disciplina de Engenharia de Software. Após aplicação do GQM+PA, nota-se que três alunos indicaram que agora sabem razoavelmente, dois já ouviram falar e um alegou sabe alguns conceitos. Conclui-se, portanto, uma evolução significativa no conhecimento de medição de software, visto que quatro (66%) alunos apresentaram saber alguns conceitos e saber razoavelmente ao final do processo. Apesar de alguns alunos terem evoluído, ainda registra-se dois alunos (33%) que responderam que já ouviram falar, mesmo participando de um processo de medição onde diariamente coletava-se dados de métricas. Isso reflete uma dificuldade de alguns alunos em assimilar certas práticas de medição. Além disso, através da análise das respostas dos alunos quando questionados se os mesmos adotavam alguma atividade relacionada a medição de software antes da adoção do GQM+PA, nota-se que todos afirmaram que não utilizavam. O Aluno C respondeu "Não saber qual atividade usar". O Aluno A afirmou "que não saberia como fazer ou qual adotar". Já os demais alunos responderam que não sabem ou qual utilizar. Diante disso, observa-se que por meio da experiência com GQM+PA os alunos passaram a identificar o que é uma atividade de medição.

Figura 28 – Nível de conhecimento sobre medição de software

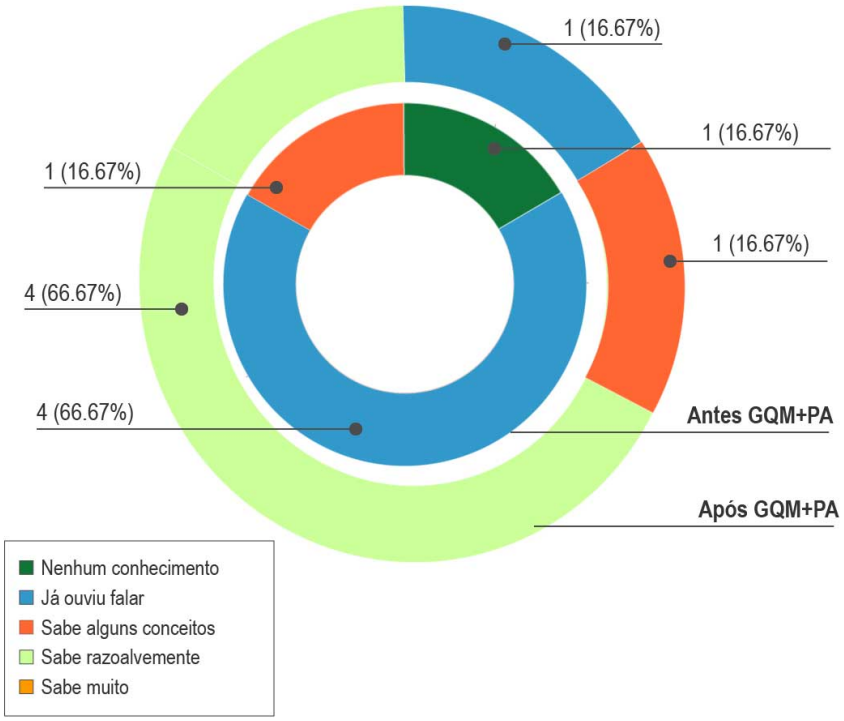


Fonte: Autoria própria.

A Figura 29 reflete as respostas dos alunos referente ao nível de conhecimento sobre métricas de software, disponíveis nos Questionários 1 e 3 (Apêndice A e C). Nota-se que, antes de ser aplicado o GQM+PA, 4 (66%) alunos responderam que já ouviram falar sobre métricas, enquanto somente 1 (16%) relatou que não tinha nenhum conhecimento e 1 (16%) declarou que sabe alguns conceitos. Após aplicação do GQM-PA, 4 (66%) responderam que sabem razoavelmente sobre métricas, sendo que antes 4 (66%) apenas tinham ouvido falar e um não havia nenhum conhecimento e outro apenas sabia alguns conceitos. Depois de GQM-PA, além de 4 (66%) passarem a saber razoavelmente, tem-se 1 (16%) aluno declarando que agora sabe alguns conceitos e 1 (16%) aluno que respondeu que já ouviu falar. Isso revela que a maioria dos alunos conseguiram adquirir mais conhecimentos sobre métricas através da prática diária orientada pelo GQM+PA. Apesar disso, ainda nota-se que um aluno respondeu que apenas ouviu falar sobre métricas, apesar de que diariamente o mesmo fazia coleta das métricas. Todos os alunos também relataram no questionário 3 (Apêndice C) a importância quanto ao uso da métrica *Complexidade Ciclomática*, demostrando assim, um entendimento sobre a importância sobre métricas. Nota-se, portanto, que mesmo todos os alunos descrevendo que é possível melhorar a qualidade do que é produzido e que as métricas os ajudaram a acompanhar o desempenho dos mesmos houve uma

discordância na resposta de um aluno que respondeu que apenas tinha ouvido falar sobre medição. Além disso, como resposta da pergunta referente a adoção de métricas no NPDS após uso do GQM+PA, disponível no questionário 3 (Apêndice C), os alunos destacaram a importância das métricas no processo de desenvolvimento. O Aluno E ressaltou "A possibilidade de acompanhar o rendimento de forma mais intrínseca no processo de desenvolvimento". Enquanto o Aluno A frisou que "Houve uma mudança de atitude nas equipes que passarão a observar um pouco mais o andamento do seu trabalho".

Figura 29 – Nível de conhecimento sobre métricas de software



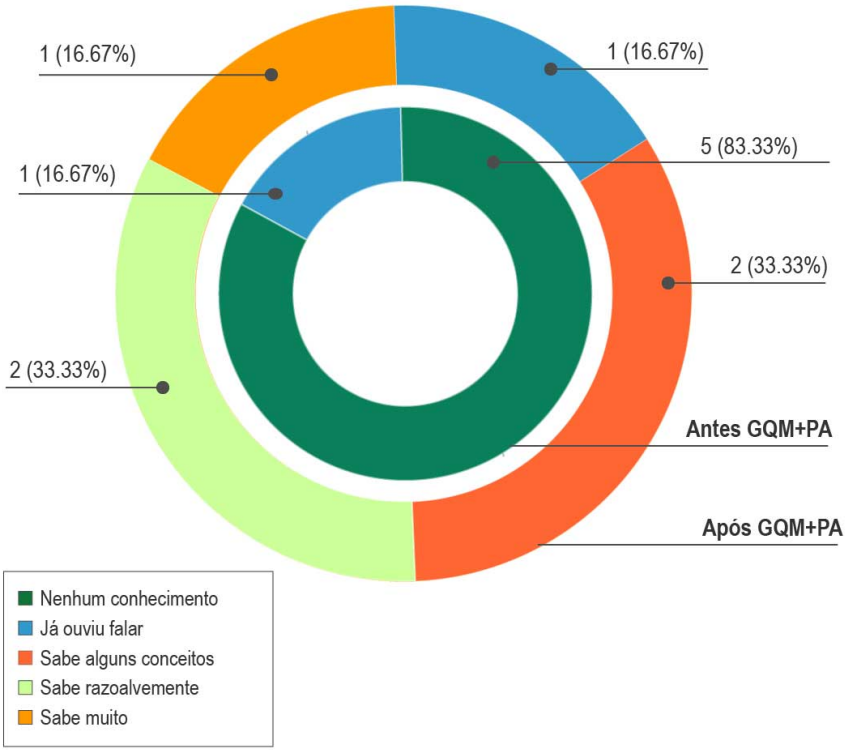
Fonte: Autoria própria.

Através da Figura 30 torna-se possível analisar as respostas dos alunos acerca do nível de conhecimento dos mesmos sobre GQM, disponíveis nos questionários 1 e 3 (Apêndice A e C). Antes de aplicar GQM+PA 5 (83%) alunos responderam que não tinham nenhum conhecimento sobre GQM e apenas 1 (16%) relatou que já ouviu falar. Após a aplicação do GQM+PA verifica-se que 2 (33%) alunos sabem razoavelmente sobre GQM, enquanto 2 (33%) sabem alguns conceitos, apenas 1 (16%) sabe muito e somente 1 (16%) ainda alega que somente ouviu falar. Nota-se, portanto, que através da aplicação do GQM-PA, a maioria dos alunos conseguiram ter compreensão sobre GQM.

Como resposta dos alunos a pergunta relacionada a prática de adotar GQM+PA

para a experiência dos mesmos com medição, disponível no questionário 3 (Apêndice C), 5 (83%) consideraram importante e 1 (16%) respondeu muito importante tal experiência. Ainda no questionário 3 (Apêndice C), os alunos destacaram que gostariam de continuar adotando GQM+PA nos projetos do NPDS. O Aluno A respondeu que "Sim, principalmente com relação a complexidade ciclomática que nos ajudou a tentar separar melhor as responsabilidade dos procedimento dentro do sistema". Já o Aluno C considerou que "Sim. Para processo de melhoria contínua dos projetos desenvolvidos".

Figura 30 – Nível de conhecimento sobre GQM



Fonte: Autoria própria.

7.2.3 Reflexão do Monitor

Para avaliar a atuação dos alunos durante a implantação do GQM+PA, tem-se a análise do monitor que esteve presente no decorrer desse processo. Antes de iniciar a coleta das métricas houve um momento em que o monitor realizou uma apresentação sobre os principais conceitos de medição de software, como funciona a abordagem GQM+PA e o papel dos alunos na coleta de métricas. Também contextualizou-se como seria realizada a coleta dos dados e como se deve proceder quanto ao preenchimento das planilhas referentes as rotina adotadas. Ao longo

do processo, caso os alunos apresentassem alguma dúvida, o monitor estava presente durante a implantação do GQM-PA para, juntos, buscarem uma solução.

Dentre as dificuldades identificadas pelo monitor, destaca-se a incorporação da rotina de preenchimento das planilhas no dia a dia. Diante disso, o monitor buscou explicar a melhor forma de tornar esse processo mais simples como, por exemplo, nas próprias planilhas haviam explicações sobre as métricas, inclusive como poderiam ser obtidas, seja manualmente ou através de um *plugin* (vide a *Complexidade Ciclomática*). Observou-se que muitas vezes os alunos esqueciam de preencher as planilhas, logo, era necessário que o monitor ficasse acompanhando tal processo. Os alunos constantemente relatavam que esqueciam de preencher as planilhas e, por isso, era necessário lembrá-los diariamente. Registra-se, entretanto, que alguns alunos não apresentavam dificuldades em lembrar de tal processo, mas haviam outros os quais necessitavam de notificações constantes.

Uma dúvida apresentada no início do processo refere-se ao uso do *plugin* (Code Metrics) para obtenção da *Complexidade Ciclomática*. Para instalação do *plugin* foi necessário o acompanhamento do monitor de tal modo que fosse explicado sobre o funcionamento do mesmo, como os dados são exibidos e, mais importante, como interpretar a *Complexidade Ciclomática*. Percebeu-se que essa constante reflexão sobre os resultados da *Complexidade Ciclomática* providenciou aos alunos um senso de responsabilidade quanto a melhoria do código desenvolvido. Todavia, os alunos ainda apresentaram dificuldades em como colocar os resultados de tal métrica na planilha tendo em vista que trabalhariam em diversos arquivos e com isso teriam muitos dados. Para lidar com este desafio, sugeriu-se que colocassem os nomes dos arquivos trabalhos com seus respectivos resultados.

Outro questionamento foi sobre como descrever na planilha a atividade que estava sendo trabalhada. Como a maioria dos discentes estava com dificuldade em fazer tal descrição, o monitor sugeriu que utilizassem a definição que continha nas *issues* presentes no *backlog* da *sprint*. Ou seja, os alunos deveriam olhar a atividade que trabalhariam em um determinado dia e a descrever conforme apresentado no *backlog*.

Além disso, como algumas atividades não estavam relacionadas ao processo de desenvolvimento, não era possível obter a *Complexidade Ciclomática*. Por exemplo, algumas tarefas eram refentes a fazer alguma configuração ou resoluções de conflito no *git*. Nesse cenário, o time decidiu inserir as demais métricas, mesmo não tendo a *Complexidade Ciclomática*.

Uma questão discutida pelo time foi a situação relativa às faltas dos alunos. Decidiu-

se que os mesmos informassem nas planilhas que haviam faltado. No entanto, percebia-se que, após a sugestão, a maioria ainda não informava quando faltava. Nesse sentido, os projetos tiveram que lidar com diversas ausências em decorrência de feriados ou solicitações de falta devido a demandas de outras disciplinas.

Além das dúvidas sobre como preencher as planilhas referentes à Rotina Diária, alguns alunos apresentaram dificuldades em preencher a planilha concernente ao final da *sprint*, ou seja, da Rotina de Retrospectiva. Os mesmos perguntaram se era preciso informar todas as atividades em uma *sprint*. Além disso, os alunos demonstraram dificuldades em converter as horas trabalhadas em uma atividade em pontos. Um detalhe a ser salientado é que em cada projeto as atividades eram pontuadas de forma diferente, pois a equipe que decidia como seria o critério de pontuação.

Como alguns alunos trabalhavam em uma mesma atividade ou cooperavam com o outro para resolver um problema, os mesmos se questionavam se deveria considerar que trabalharam na mesma atividade ou se não informava nada. Foi decidido que colocassem a mesma atividade com os dados das outras métricas nas respectivas planilhas.

8 CONSIDERAÇÕES FINAIS

A Qualidade de Software é uma área da Engenharia de Software que se preocupa em garantir que o processo de desenvolvimento tenha um nível de qualidade adequado às demandas dos clientes. Nesse sentido, o processo de medição de software exerce um papel fundamental tendo em vista que viabiliza-se ao time de desenvolvimento uma compreensão respaldada por dados e, consequentemente, maior controle sobre o projeto. No contexto da medição de software destaca-se a importância quanto a internalização das rotinas de análise de métricas referentes ao processo de desenvolvimento ou concernente ao próprio produto de software. Existem diversos métodos que orientam o processo de medição de software, dentre os quais pode-se destacar o *Goal-Question-Metric* (GQM) devido sua adoção mercadológica.

Todavia, a adoção de processos que envolvem medição de software está intrinsecamente atrelada a experiência do profissional que a realiza. Apesar da referida relevância, verifica-se diversos desafios quanto a adoção de metodologias de ensino que alinhem a teoria e prática de mercado sob o contexto de medição de software. Considerando tal premissa, o presente trabalho apresentou e avaliou a adoção de um *framework* denominado GQM+PA o qual integra as etapas que constituem o *Goal-Question-Metric* às práticas da Pesquisa-Ação.

Quanto a avaliação empírica conduzida, investigou-se múltiplos casos de estudo no Núcleo de Práticas de Desenvolvimento de Sistemas (NPDS) o qual encontra-se estabelecido na Universidade Federal do Ceará (Campus de Crateús). Ao todo, foram investigados 4 projetos (sendo um utilizado para pré-teste) constituídos por 11 alunos estagiários. Alinhado às demandas estratégicas do NPDS, avaliou-se dois objetivos: 1) Melhorar a precisão das estimativas do projeto e 2) Aumentar a qualidade dos produtos liberados para uso. Nesse sentido, possibilitou-se aos estagiários um acompanhamento constante quanto às métricas em ambas as questões. Conforme percebido na avaliação do aprendizado dos alunos após a adoção do GQM+PA, constata-se uma sensível evolução quanto ao nível de conhecimento e conscientização sobre medição de software, métricas de software e o processo de GQM.

Destaca-se a seguir as principais contribuições decorrente da presente pesquisa: i) através do GQM+PA torna-se possível fornecer um mecanismo integrado para o aprendizado teórico e prático sobre medição de software; ii) viabilizou-se empiricamente ao discentes uma reflexão sobre a relevância quanto ao acompanhamento de métricas para o processo de desenvolvimento de software, contribuindo, assim, para ampliar a qualidade dos produtos desenvolvidos no estágio; iii) todas as rotinas, *dashboards* e planilhas desenvolvidas podem ser

adaptados em outros projetos e usufruídos como um ativo organizacional, inclusive por outras instituições.

Em relação às limitações, salienta-se que o período de análise do GQM+PA e a quantidade de discentes avaliados pode ser considerado reduzido. Entretanto, verifica-se que os desafios identificados demonstraram-se saturados conforme aproximou-se do final do processo. Outra dificuldade identificada foi a ocorrência excessiva de feriados e ausência dos estagiários, influenciando assim, nos resultado obtidos. Em relação à avaliação concernente a qualidade do produto, abordou-se somente a *Complexidade Ciclomática*. Tal limitação ocorreu em decorrência das restrições dos projetos analisados.

Em termos de trabalhos futuros, almeja-se desenvolver um sistema de apoio que facilite a coleta, visualização e análise referente aos resultados obtidos ao invés do uso de planilhas. Adicionalmente, pretende-se investigar a adoção do GQM+PA em outros ambientes de estágios.

REFERÊNCIAS

ALLEN, J. H.; DAVIS, N. Measuring operational resilience using the cert resilience management model. 2010.

AMBLER, S. W. **More process patterns: delivering large-scale systems using object technology**. [S.l.]: Cambridge University Press, 1999.

ANDRADE, R. M.; SANTOS, I. S.; ARAÚJO, I. L.; CARVALHO, R. M. Uma metodologia para o ensino teorico e pratico da engenharia de software. In: **Fórum de Educação em Engenharia de Software (FEES). In VI Congresso Brasileiro de Software: Teoria e Prática (CBSoft)**. [S.l.: s.n.], 2015.

BARTIÉ, A. **Garantia da qualidade de software**. [S.l.]: Gulf Professional Publishing, 2002.

BASILI, G. C. V. R.; ROMBACH, H. D. The goal question metric approach. **Encyclopedia of Software Engineering**, 1996.

BASILI, V.; CALDIERA, G.; ROMBACH, D. **The Goal Question Metric Paradigm: Encyclopedia of software engineering**. [S.l.]: John Wiley & Sons, 1994.

BASILI, V. R. **Software modeling and measurement: the Goal/Question/Metric paradigm**. [S.l.], 1992.

BORGES, E. P. Um modelo de medição para processos de desenvolvimento de software. **Departamento de Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais. Belo Horizonte**, 2003.

BUENO, C. d. F. dos S.; CAMPELO, G. B. Qualidade de software. 2010.

CARNEIRO, C. Multiple case studies: research strategy in psychoanalysis and education. **Psicologia USP**, SciELO Brasil, v. 29, n. 2, p. 314–321, 2018.

CARVALHO, M.; PALADINI, E. **Gestão da qualidade: teoria e casos**. [S.l.]: Elsevier Brasil, 2013.

CERA, M. C.; FORNO, M.; VIEIRA, V. G. Uma proposta para o ensino de engenharia de software a partir da resolução de problemas. **Revista Brasileira de Informática na Educação**, v. 20, n. 3, p. 116–132, 2012.

CHANIN, R.; SALES, A.; SANTOS, A.; POMPERMAIER, L.; PRIKLADNICKI, R. A collaborative approach to teaching software startups: findings from a study using challenge based learning. In: ACM. **Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering**. [S.l.], 2018. p. 9–12.

CHOI, J.-I.; HANNAFIN, M. Situated cognition and learning environments: Roles, structures, and implications for design. **Educational technology research and development**, Springer, v. 43, n. 2, p. 53–69, 1995.

CHRISSIS, M. B.; KONRAD, M.; SHRUM, S. **CMMI guidlines for process integration and product improvement**. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., 2003.

COMMITTEE, I. S. C. *et al.* Ieee standard glossary of software engineering terminology (ieee std 610.12-1990). los alamitos. **CA: IEEE Computer Society**, v. 169, 1990.

CRESPO, A. N.; SILVA, O. J.; BORGES, C. A.; SALVIANO, C. F.; ARGOLLO, M.; JINO, M. Uma metodologia para teste de software no contexto da melhoria de processo. **Simpósio Brasileiro de Qualidade de Software**, p. 271–285, 2004.

CROSBY, P. B. Qualy is free: the art of making quality certain. 1992.

ENGEL, G. I. Pesquisa-ação. **Educar em Revista**, SciELO Brasil, n. 16, p. 181–191, 2000.

ESPOSTE, A. d. M. D. Tomada de decisões orientadas a métricas de software: observações de métricas de produto e vulnerabilidades de software via dw e plataforma de monitoramente de código-fonte. **Monografia. Universidade de Brasília**, 2014.

FEIGENBAUM, A. V. **Total quality control**. [S.l.: s.n.], 1983.

FENTON, N. Software measurement: A necessary scientific basis. **IEEE Transactions on software engineering**, IEEE, v. 20, n. 3, p. 199–206, 1994.

FLORAC, W. A.; CARLETON, A. D. **Measuring the software process: statistical process control for software process improvement**. [S.l.]: Addison-Wesley Professional, 1999.

FRANCA, L. P.; STAA, A. V. *et al.* Software measurement for small organizations. In: CITESEER. **Proceedings of Conference on Advanced Information Systems Engineering**. [S.l.], 1998.

FRANCA, L. P. A.; STAA, A. von; LUCENA, C. J. P. de; MCC32, P.-R. **Medição de software para pequenas empresas: uma solução baseada na Web**. [S.l.]: PUC, 1998.

GIL, A. C. Como classificar as pesquisas. **Como elaborar projetos de pesquisa**, Atlas São Paulo, v. 4, p. 44–45, 2002.

GODOY, A. S. Introdução à pesquisa qualitativa e suas possibilidades. **Revista de administração de empresas**, SciELO Brasil, v. 35, n. 2, p. 57–63, 1995.

GOMES, A.; OLIVEIRA, K.; ROCHA, A. R. Avaliação de processos de software baseada em medições. **Simpósio Brasileiro de Engenharia de Software. Rio de Janeiro, RJ**, 2001.

GRUNDY, S.; KEMMIS, S. Educational action research in australia: The state of the art (an overview). **The action research reader**, Deakin University Press Victoria, v. 3, p. 321–335, 1982.

HARTMANN, D.; DYMOND, R. Appropriate agile measurement: using metrics and diagnostics to deliver business value. In: IEEE. **Agile Conference, 2006**. [S.l.], 2006. p. 6–pp.

HOCK, G. T.; HUI, G. L. S. A study of the problems and challenges of applying software metrics in software development industry. In: **Proceedings of the M2USIC-MMU International Symposium on Information and Communication Technologies, Putrajaya, Malaysia**. [S.l.: s.n.], 2004. p. 8–11.

HONGLEI, T.; WEI, S.; YANAN, Z. The research on software metrics and software complexity metrics. In: IEEE. **Computer Science-Technology and Applications, 2009. IFCSTA'09. International Forum on**. [S.l.], 2009. v. 1, p. 131–136.

IEC, I. Iso/iec 25000 software engineering software product quality requirements and evaluation (square) guide to square. **Systems Engineering**, v. 41, 2005.

ISHIKAWA, K. **What is total quality control? The Japanese way**. [S.l.]: Prentice Hall, 1985.

JALOTE, P. **An integrated approach to software engineering**. [S.l.]: Springer Science & Business Media, 2012.

KEMMIS, S.; MCTAGGART, R.; NIXON, R. **The action research planner: Doing critical participatory action research**. [S.l.]: Springer Science & Business Media, 2013.

KOSCIANSKI, A.; SOARES, M. d. S. Qualidade de software. **São Paulo, SP. Editora: Novatec**, 2007.

MAGNO, A.; REJANE, C.; SIMÕES, F.; PEREIRA, Í.; SIMÕES, L. **Modelos de Maturidade**. [S.l.]: UNEB, 2011.

MARETTO, C. X. Uma arquitetura de referência para medição de software. Universidade Federal do Espírito Santo, 2013.

MILLS, E. E. **Software Metrics (CMU/SEI-CM-12-1.1).)**. 1998. <http://www.sei.cmu.edu/publications/documents/cms/cm.012.html>. Acessado em: 28/04/2018.

MISHRA, A.; MISHRA, D. Industry oriented advanced software engineering education curriculum. **Croatian Journal of Education**, Citeseer, v. 14, n. 3, p. 595–624, 2012.

MORGADO, G. P.; GESSER, I.; SILVEIRA, D. S.; MANSO, F. S.; LIMA, P. M.; SCHMITZ, E. A. Práticas do cmmi® como regras de negócio. **Production**, SciELO Brasil, v. 17, n. 2, p. 383–394, 2007.

PERKINS, T.; PETERSON, R.; SMITH, L. Back to the basics: Measurement and metrics. In: CITESEER. **The Journal of Defense Software Engineering**. [S.l.], 2003.

PFLEEGER, S. L.; JEFFERY, R.; CURTIS, B.; KITCHENHAM, B. Status report on software measurement. **IEEE software**, IEEE, v. 14, n. 2, 1997.

PIMENTA, S. G. Pesquisa-ação crítico-colaborativa: construindo seu significado a partir de experiências com a formação docente. **Educação e pesquisa**, Universidade de São Paulo, v. 31, n. 3, 2005.

PRESSMAN, R.; MAXIM, B. **Engenharia de Software-8ª Edição**. [S.l.]: McGraw Hill Brasil, 2016.

RAWAT, M. S.; MITTAL, A.; DUBEY, S. K. Survey on impact of software metrics on software quality. **IJACSA) International Journal of Advanced Computer Science and Applications**, Citeseer, v. 3, n. 1, 2012.

RINCON, A. M. Qualidade de software. **XI Encontro de Estudantes de Informática do Tocantins**, p. 75–86, 2009.

ROCHA, A.; MONTONI, M.; SANTOS, G.; OLIVEIRA, K.; NATALI, A. C.; MIAN, P.; CONTE, T.; MAFRA, S.; BARRETO, A.; ALBUQUERQUE, A. *et al.* Dificuldade e fatores de sucesso na implementação de processos de software utilizando o mr-mps e o cmmi. In: SN. **I Workshop de Implementadores (W2-MPS. BR)**. [S.l.], 2006.

ROCHA, A. d.; SOUZA, G. d. S.; BARCELLOS, M. Medição de software e controle estatístico de processos. **Ministério da Ciência, Tecnologia e Inovação; Secretaria de Política de Informática**, p. 21, 2012.

SALVIANO, C. F. *et al.* Uma proposta orientada a perfis de capacidade de processo para evolução da melhoria de processo de software. [sn], 2006.

SANTOS, R. M. Características e medidas de software para avaliação da qualidade da interação humano-computador em sistemas ubíquos. **Masters Report, Federal Universy of Ceará**, 2014.

SARAIVA, A. V. **Utlização da Abordagem Goal-Question-Metrics(GQM) na Elaboração e Execução de Planos de Avaliação de Usabilidade de Software: Um Estudo Empírico sobre um Software Agropecuário**. [S.l.], 2006.

SATO, D. T. Uso eficaz de métricas em métodos ágeis de desenvolvimento de software. **Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo**, v. 139, 2007.

SCHOTS, M.; SANTOS, R.; MENDONÇA, A.; WERNER, C. Elaboração de um survey para a caracterização do cenário de educação em engenharia de software no brasil. **Anais do II Fórum de Educação em Engenharia de Software, XXIII Simpósio Brasileiro de Engenharia de Software**, p. 57–60, 2009.

SOFTEX. **MPS-SW-Guia Geral MPS de Software**). 2012. <http://www.softex.br/mpsbr/guias>. Acessado em: 20/04/2018.

SOLINGEN, R. V.; BERGHOUT, E. **The Goal/Question/Metric Method: a practical guide for quality improvement of software development**. [S.l.]: McGraw-Hill, 1999.

SOMMERVILLE, I. **Engenharia de Software, Tradução de André Maurício de Andrade Ribeiro; Revisão técnica de Kechi Hirama**. [S.l.]: São Paulo, Addison Wesley, 2011.

TAHIR, T.; RASOOL, G.; GENCEL, C. A systematic literature review on software measurement programs. **Information and Software Technology**, Elsevier, v. 73, p. 101–121, 2016.

TÉCNICAS, A. B. de N. **NBR ISO/IEC 9126-1: Engenharia de software: qualidade de produto: parte 1: modelo de qualidade**. [S.l.]: ABNT Rio de Janeiro, 2003.

TRIPP, D. Pesquisa-ação: uma introdução metodológica. **Educação e pesquisa**, SciELO Brasil, v. 31, n. 3, 2005.

TSURUTA, M. **Um estudo sobre a relação entre qualidade e arquitetura de software**. Tese (Doutorado) — Universidade de São Paulo, 2011.

WANGENHEIM, C. G. von; THIRY, M.; KOCHANSKI, D.; STEIL, L.; SILVA, D.; LINO, J. Desenvolvimento de um jogo para ensino de medição de software. 2009.

WANGENHEIM, C. v. Utilização do gqm no desenvolvimento de software. **Laboratório de Qualidade de Software, Instituto de Informática, Universidade do Vale do Rio dos Sinos. São Leopoldo**, p. 62, 2000.

APÊNDICE A – QUESTIONÁRIO 1: CONHECIMENTO GERAIS SOBRE MEDIÇÃO DE SOFTWARE

1. Nível de conhecimento sobre medição de software?
    a) Nenhum conhecimento
    b) Já ouviu falar
    c) Sabe alguns conceitos
    d) Sabe razoavelmente
    e) Sabe muito
2. Nível de conhecimento sobre métricas de software?
    a) Nenhum conhecimento
    b) Já ouviu falar
    c) Sabe alguns conceitos
    d) Sabe razoavelmente
    e) Sabe muito
3. Nível de conhecimento sobre a abordagem Goal-Question-Metric (GQM)?
    a) Nenhum conhecimento
    b) Já ouviu falar
    c) Sabe alguns conceitos
    d) Sabe razoavelmente
    e) Sabe muito
4. Quão relevante você acredita ser a adoção de um processo de medição de software para um projeto de sucesso?
    a) Sem importância
    b) Pouco importante
    c) Razoavelmente importante
    d) Muito Importante
    e) Extremamente importante
5. Você considera necessário ter uma compreensão sobre medição de software para sua carreira? Por quê?

APÊNDICE B – QUESTIONÁRIO 2: PROCESSO DE DESENVOLVIMENTO

1. Você encontra dificuldades em compreender o que significa atingir qualidade de software?
    a) Nenhuma dificuldade
    b) Baixa dificuldade
    c) Média dificuldade
    d) Alta dificuldade
    e) Extrema dificuldade
2. Na sua visão, como se pode verificar a qualidade de um produto de software?
3. Você adota alguma atividade relacionada a medição de software? Se sim, qual? Se não, por quê?
4. Você utiliza alguma métrica no processo de desenvolvimento? Se sim, qual? Se não, por quê?
5. Você encontra dificuldades em estimar pontuação para as atividades a serem desenvolvidas?
    a) Nenhuma dificuldade
    b) Baixa dificuldade
    c) Média dificuldade
    d) Alta dificuldade
    e) Extrema dificuldade
6. Você encontra dificuldades na execução de suas atividades relacionadas a garantia da qualidade do produto? Se sim, quais?
7. Quanto você concorda sobre a adoção de atividades específicas para realização de medição de software?
    a) Discordo totalmente
    b) Discordo parcialmente
    c) Indiferente
    d) Concordo parcialmente
    e) Concordo totalmente
8. Quão satisfeito você está com a qualidade do produto que você entrega?
    a) Muito insatisfeito
    b) Insatisfeito
    c) Adequado
    d) Satisfeito

e) Muito satisfeito

APÊNDICE C – QUESTIONÁRIO 3

1. Após aplicação do GQM+PA, qual seu nível de conhecimento sobre medição de software
   a) Nenhum conhecimento
   b) Já ouviu falar
   c) Sabe alguns conceitos
   d) Sabe razoavelmente
   e) Sabe muito
2. Após aplicação do GQM+PA, qual seu nível de conhecimento sobre métricas de software?
   a) Nenhum conhecimento
   b) Já ouviu falar
   c) Sabe alguns conceitos
   d) Sabe razoavelmente
   e) Sabe muito
3. Após aplicação do GQM+PA, qual seu nível de conhecimento sobre a abordagem Goal-Question-Metric (GQM)?
   a) Nenhum conhecimento
   b) Já ouviu falar
   c) Sabe alguns conceitos
   d) Sabe razoavelmente
   e) Sabe muito
4. Qual sua percepção sobre a adoção de métricas no NPDS após uso do GQM+PA?
5. Ao se deparar com os resultados oriundos da complexidade ciclomática, quais foram suas atitudes?
6. Você apresentou dificuldades em preencher e lembrar de preenchê-las diariamente? Por quê?
7. Após aplicar GQM+PA quão relevante você acredita que a experiência de adotar GQM+PA contribuiu para sua experiência com medição?
   a) Sem importância
   b) Pouco importante
   c) Razoavelmente importante
   d) Muito importante
   e) Extremamente importante

8. Gostaria de continuar adotando GQM+PA? Por quê?
9. Como resultado do questionário anterior, observou-se que alguns responderam que haviam dificuldade de estimar as atividades (pontuar as atividades). No entanto, nas planilhas as estimativas das atividades estavam próximas ou iguais a pontuação real. Logo, percebe-se uma discordância ao que foi respondido no questionário, como você explica isso?

APÊNDICE D – SÍNTESE DA ENTREVISTA COM GERENTE DE PROJETOS

1. Qual sua formação acadêmica (graduação e pós-graduação)? Em qual área? Há quanto tempo?

   Mestrado em Ciência da Computação , 3 anos.
2. Quanto tempo de experiência você possui na indústria de software?

   8 anos.
3. Há quanto tempo você exerce o cargo de gerente de projetos no NPDS?

   2 anos.
4. Há quantos projetos em desenvolvimento no NPDS?

   4.
5. Há quanto tempo o NPDS desenvolve projetos de software?

   3 anos.
6. Quantas pessoas contribuem no NPDS?

   19 pessoas.
7. O que são as metas estratégicas do NPDS?
   - *Ambiente de para formação técnica complementar aos alunos de Ciência da Computação e Sistemas de Informação através de estágios;*
   - *Desenvolver software qualidade para instituições parceiras;*
   - *Garantir satisfação dos clientes.*
8. Que forças têm impacto nas metas estratégicas?
   - *Nível de competência – qualidade técnica;*
   - *Apoio da gestão do NPDS – professores e analista (servidores).*
9. Quais seriam possíveis estratégias para alcançar os objetivos do NPDS?
   - *Qualidade de estagiários;*
   - *Disponibilidade da gerência;*
   - *Boas práticas de Engenharia de Software;*
   - *Relacionamento com cliente (atender expectativas do cliente);*
   - *Vivência de metodologia ágil, de Engenharia de Software, além de treinamento.*
10. Atualmente, quais são as suas principais preocupações (problemas) em relação ao NPDS?
    - *Diferente nível entre equipes;*
    - *Dificuldade nivelamento de equipe;*
    - *Práticas de Engenharia de Software que não foram implantadas e automação;*

- *Falta de métricas;*
- *Participação do clientes;*
- *Gerência competente para atingir.*

11. Atualmente, é realizado algum procedimento de mensuração de software no NPDS? Se sim, qual?
    - *Scrum – pontos, com relação a performance: velocidade da equipe e eficiência;*
    - *Pontos entregues;*
    - *Quantidade de bugs abertos e fechados (descobertos a cada versão.*
12. Na sua visão, quão benéfico seria a adoção de um processo de medição no NPDS?
    - *Necessário, se não mede não há decisão direita;*
    - *Importante, ao membro e gerência sobre qualidade do produto e equipe conheça (performance, como está).*

APÊNDICE E – GUIA GERAL PARA OBTENÇÃO DE MÉTRICAS

- OBJETIVO 1: Melhorar precisão das estimativas do projeto
  1. Qual a precisão das estimativas de escopo do projeto?
     - Efetividade de Pontos

       Pontuação executada: refere-se aos pontos que foram realmente gastos para realizar a atividade. Pontuação planejada: refere-se aos pontos estimados para realizar atividade. Como obter: pontuação executada / pontuação planejada
     - Velocidade de Pontos

       Refere-se a quão rápido foi executada uma determinada atividade. Como obter: pontuação executada.
     - Entrega prometida

       Analisar se a atividade foi realizada e entregue na sprint. Como obter: manualmente.
  2. Qual a precisão das estimativas de esforço do projeto?
     - Efetividade de Horas

       Tempo executado: refere-se ao tempo que foi realmente gasto para realizar a atividade. Tempo planejado: refere-se ao tempo estimado para realizar atividade. Como obter: tempo executado / tempo planejado.
     - Velocidade de Horas

       Refere-se a quão rápido foi executada uma determinada atividade. Como obter: tempo executado.
- OBJETIVO 2: Aumentar a qualidade dos produtos liberados para uso
  1. Qual a qualidade dos produtos antes de liberação para uso?
     - Complexidade Ciclomática

       Mensura a complexidade de um determinado módulo (uma classe, um método, uma função etc), a partir da contagem do número de caminhos independentes que ele pode executar até o seu fim. Como obter: através de *plugin*.
  2. Qual foi a aceitação das funcionalidades entregues?
     - *Issues* aprovados pelo PO.