UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

CENTRO TECNOLÓGICO

COLEGIADO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO

Lucas Moraes Soares de Souza

Ramble ON: Utilizando Medição de *software* e Ontologias em Rede para prover Dados de apoio à Tomada de Decisão

Vitória, ES

2023

Lucas Moraes Soares de Souza

Ramble ON: Utilizando Medição de *software* e Ontologias em Rede para prover Dados de apoio à Tomada de Decisão

Projeto de Graduação apresentado ao Colegiado do Curso de Ciência da Computação do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito para aprovação na Disciplina Projeto de Graduação II.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Colegiado do Curso de Ciência da Computação

Orientador: Monalessa Perini Barcellos

Coorientador: Paulo Sérgio dos Santos Júnior

Vitória, ES

2023

Lucas Moraes Soares de Souza
Ramble ON: Utilizando Medição de *software* e Ontologias em Rede para prover Dados de apoio à Tomada de Decisão/ Lucas Moraes Soares de Souza. – Vitória, ES, 2023
57 p. : il. (algumas color.) ; 30 cm.

Orientador: Monalessa Perini Barcellos
Coorientador: Paulo Sérgio dos Santos Júnior

Monografia (PG) – Universidade Federal do Espírito Santo – UFES
Centro Tecnológico
Colegiado do Curso de Ciência da Computação, 2023.

1. Medição de *software*. 2. Ontologias. 3. Engenharia de *software* Contínua. 4. Goal, Question, Metric. I. de Souza, Lucas Moraes Soares. II. Universidade Federal do Espírito Santo. IV. *Ramble ON*: Utilizando Medição de *software* e Ontologias em Rede para prover Dados de apoio à Tomada de Decisão

CDU 02:141:005.7

Lucas Moraes Soares de Souza

Ramble ON: Utilizando Medição de *software* e Ontologias em Rede para prover Dados de apoio à Tomada de Decisão

Projeto de Graduação apresentado ao Colegiado do Curso de Ciência da Computação do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito para aprovação na Disciplina Projeto de Graduação II.

Trabalho aprovado. Vitória, ES, 19 de julho de 2023:

Monalessa Perini Barcellos
Orientador

João Paulo Almeida
Departamento de Informática
Universidade Federal do Espírito Santo

Glaice Kelly da S. Quirino Monfardini
Coordenadoria do Curso de Técnico em Química
Instituto Federal do Espírito Santo

Vitória, ES
2023

Dedico este trabalho a Deus, pilar máximo da minha existência. Dedico também à minha mãe, que mesmo de longe me deu forças. Aos meus amigos, que me fizeram companhia nos momentos difíceis.

Agradecimentos

Primeiramente, gostaria de agradecer a Deus por me dar forças e sabedoria para concluir este trabalho. Agradeço também à minha mãe Lucineide Vilela de Moraes pelo amor, apoio e incentivo incondicionais. Aos meus amigos, que me ajudaram a manter o equilíbrio emocional e me motivaram a continuar. Sem vocês, eu não teria conseguido.

Gostaria de agradecer também ao meu coorientador Paulo Sérgio dos Santos Júnior por compartilhar seus conhecimentos e experiências comigo e me ajudar a desenvolver este trabalho. E à minha orientadora Monalessa Perini Barcellos, que me guiou durante todo o processo e me ensinou muito sobre pesquisa científica.

Por fim, gostaria de agradecer novamente a Deus por ter colocado essas pessoas maravilhosas em minha vida.

"Sem a informação certa, você é apenas mais uma pessoa com opinião.
(Tracy O'Rourke, CEO of Allen-Bradley)

Resumo

O presente trabalho tem como objetivo apresentar um processo de apoio à medição de software baseado no método GQ(I) M e em dados organizacionais para fornecer dados relevantes para apoiar o desenvolvimento de software orientado a dados e a tomada de decisão no contexto de Engenharia de *Software* Contínua (ESC). ESC é uma abordagem que busca aprimorar a qualidade do *software* através da integração contínua, entrega contínua e feedback contínuo. O método GQM (Goal-Question-Metric) é uma técnica que permite definir objetivos, questões e métricas para medir o desempenho de um processo.

Para apoiar a excução do processo, foram utilizados componentes da solução de integração *The Band*, que é uma abordagem que utiliza ontologias em rede para integrar dados de diferentes fontes e fornecer informações relevantes para a tomada de decisão.

O trabalho foi desenvolvido a partir da revisão bibliográfica sobre os temas que fundamentam a abordagem proposta, bem como do estudo de tecnologias utilizadas, como banco de dados Postgresql, soluções de Business Intelligence e de apoio à gerência de dados. Além disso, foi realizada a definição do processo de medição proposto, que consiste em coletar dados organizacionais e representá-los graficamente para obter uma visão mais clara do processo de desenvolvimento de *software*, identificando oportunidades de melhoria.

Foi realizada uma demonstração de uso do processo proposto utilizando-se dados de uma organização real. Este trabalho teve como foco as atividades inicias do processo, anteriores a coleta de dados, e apenas um exemplo de coleta dados e construção de representação gráfica por limitações de tempo.

Algumas lacunas precisam ser preenchidas em trabalhos futuros, como a necessidade de acompanhar os resultados da aplicação do processo em uma organização e a necessidade de avaliar a abordagem em organizações de diferentes segmentos.

Palavras-chaves: Medição de *Software*. Ontologias. Engenharia de *software* Contínua. Goal, Question, Metric.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

NEMO	Núcleo de Estudos em Modelagem Conceitual e Ontologias
GQM	Goal Question Metric
GQ(I)M	Goal Question Indicator Metric
ESC	Engenharia de *Software* Contínua
SEON	*Software* Engineering Ontology Network
EO	Enterprise Ontology
SQL	Structured Query Language
SRO	Scrum Reference Ontology
OSDEF	Ontology of Software Defect Errors and Failures
BI	Business Intelligence

Sumário

1 Introdução

Este capítulo apresenta a motivação, justificativa (Seção 1.1, objetivos do trabalho (Seção 1.2), bem como o método de desenvolvimento adotado (Seção 1.3) e a organização desta monografia (Seção 1.4).

1.1 Motivação e Justificativa

Em um mercado em constante expansão como o de desenvolvimento de software, novos desafios surgem a todo momento, incluindo a necessidade de atender novos requisitos para melhor se adequar aos modelos de negócios, além da demanda por mais agilidade nas entregas das soluções contratadas. Ao enfatizar flexibilidade, eficiência e velocidade, as práticas ágeis levaram a uma mudança de paradigma na forma como o software é desenvolvido (OLSSON; ALAHYARI; BOSCH, 2012).

A ampla adoção de métodos ágeis (KURAPATI; MANYAM; PETERSEN, 2012) evidencia a necessidade de flexibilidade e rápida adaptação no atual mercado de desenvolvimento de software. Fitzgerald e Stol (2017), porém, argumentam em favor de uma abordagem mais ampla do que os métodos ágeis, a chamada Engenharia de Software Contínua (do inglês, *Continuous Software Engineering*) (ESC). ESC consiste em um conjunto de práticas e ferramentas que apoiam uma visão holística do desenvolvimento de software com o objetivo de torná-lo mais rápido, iterativo, integrado, contínuo e alinhado ao negócio (BARCELLOS, 2020).

Para que se possa realizar as atividades que caracterizam ESC são necessárias diversas aplicações[1]. Nesse sentido, organizações costumam usar diferentes aplicações para apoiar diferentes processos do desenvolvimento de software, abordando diferentes domínios da Engenharia de Software. Por exemplo, Azure Devops [2] pode ser utilizada para apoiar o planejamento das tarefas do projeto; o Github[3] pode auxiliar no versionamento de código, parte importante da atividade de integração contínua (CI); e o Clockify[4] e o Tempo[5] podem ser utilizados para gerenciar o tempo investido em cada tarefa. Os dados que são produzidos a partir do uso dessas aplicações e são nelas armazenados podem prover informações úteis para auxiliar na tomada de decisão, visando à melhoria do processo de desenvolvimento de software, do produto ou da organização que está desenvolvendo o

[1] Neste trabalho, os termos aplicação, ferramenta e sistema são usados como sinônimos
[2] <https://azure.microsoft.com/en-us/products/devops>
[3] <https://www.github.com/>
[4] <https://clockify.me/pt/>
[5] <https://www.tempo.io/>

software.

Contudo, essas diferentes aplicações apresentam dados heterogêneos, sendo necessário integrá-los para obter dados mais significativos para apoiar ações de melhoria ou tomada de decisão (FONSECA; BARCELLOS; FALBO, 2017). Além disso, é necessário identificar quais dados são realmente necessários para apoiar o desenvolvimento de software e a tomada de decisão em uma organização. Nesse sentido, é preciso identificar as necessidades de informação (do inglês, *Information Needs*) da organização que podem ser atendidas pelos dados disponíveis.

Nesse contexto, medição de software se faz muito importante. Medição de Software visa quantificar diferentes aspectos dos processos e produtos da Engenharia de Software para ajudar a entender melhor esses processos e produtos e, com isso, apoiar as atividades cotidianas do desenvolvimento de software, bem como a tomada de decisão (BARCELLOS, 2010). Para fazer essa quantificação, são usadas medidas (ou métricas)[6]. Elas são definidas para quantificar propriedades de entidades que podem ser mensuradas e, consequentemente, caracterizadas. Por exemplo, pode-se medir a produtividade de um desenvolvedor em um projeto. Nesse caso, a entidade que se quer medir e caracterizar é o desenvolvedor. A propriedade que se quer quantificar é a produtividade. E para fazer essa quantificação pode ser usada a medida quantidade média de pontos de história desenvolvidos pelo desenvolvedor por *sprint* do projeto. Ao quantificar a produtividade de vários desenvolvedores pode-se, por exemplo, identificar se há muita diferença de produtividade entre diferentes desenvolvedores, investigar as possíveis causas para produtividades mais baixas e tratá-las. Além disso, conhecendo a produtividade dos desenvolvedores, seria possível planejar melhor as *sprints* (BARCELLOS, 2023).

Medir geralmente demanda algum esforço, então é preciso medir o que realmente importa. Para isso, a medição deve ser orientada a objetivos ou necessidades de informação. Assim, antes de começar a medir, é preciso saber para que medir. Por exemplo, se uma empresa quer melhorar a qualidade dos produtos entregues, ela certamente terá a necessidade de informação acerca da quantidade de defeitos e deverá definir medidas que provejam essa informação. Existem diversos métodos que podem auxiliar na identificação das medidas que devem ser consideradas em uma organização. Dentre eles destacam-se o GQM (*Goal – Question – Metric*) (BASILI; CALDIERA; ROMBACH, 1994) e sua variação GQ(I)M (*Goal Question (Indicator) Metric*) (PARK; GOETHERT; FLORAC, 1996). O GQM diz que a partir dos objetivos, pode-se derivar necessidades de informação e, então, identificar medidas capazes de atendê-las. O GQ(I)M, por sua vez, orienta que dentre as medidas definidas devem ser identificadas as que serão utilizadas como indicadores

[6] na literatura não há consenso sobre o termos medida e métrica. Alguns autores e padrões adotam o termo métrica (e.g., (PARK; GOETHERT; FLORAC, 1996)), enquanto outros adotam medida (e.g., (ISO/IEC/IEEE, 2017)) . Neste trabalho, utiliza-se o termo medida

(BARCELLOS, 2023).

Ao auxiliar na identificação de necessidades de informação, medição de software também auxilia no uso de dados para apoiar as atividades cotidianas de desenvolvimento de software e a tomada de decisão. Como previamente discutido, as aplicações que as organizações utilizam para apoiar suas atividades armazenam muitos dados. A definição de medidas é importante, pois elas vão orientar sobre quais dados são realmente necessários e para que servem. Para isso, é preciso considerar quais são os objetivos e necessidades de informação da organização, identificar as medidas necessárias para atendê-las e verificar se há dados suficientes para calculá-las. Caso não haja, pode ser necessário esforço adicional de coleta de dados (BARCELLOS, 2023).

Este trabalho apresenta uma abordagem chamada *Ramble ON*[7], que combina GQ(I)M (*Goal Question (Indicator) Metric*) (PARK; GOETHERT; FLORAC, 1996) e *Immigrant* (SANTOS JúNIOR, 2023) para auxiliar na identificação de necessidades de informação, extração de dados integrados e implementação indicadores que atendam as necessidades de informação, utilizando os dados presentes nas aplicações utilizadas no processo de desenvolvimento de software da organização. Ele foi desenvolvido no contexto de uma pesquisa de doutorado realizada no NEMO[8] (Núcleo de Estudos em Modelagem Conceitual e Ontologias) pelo doutorando Paulo Sérgio dos Santos Júnior, a qual aborda o problema da integração semântica no contexto de ESC. A pesquisa de doutorado visa à criação de *Immigrant*, uma abordagem de integração semântica baseada em redes de ontologias. Um dos componentes de *Immigrant* é *The Band* (SANTOS JúNIOR, 2023), uma arquitetura de integração de dados baseada em redes de ontologias que fornece dados integrados a partir de dados armazenados em aplicações usadas por uma organização para atender suas necessidades de informação. Uma rede de ontologias é uma coleção de ontologias relacionadas através de uma variedade de relacionamentos, como alinhamento e dependência (SUÁREZ-FIGUEROA et al., 2012). Em *The Band* ontologias são utilizadas para atribuir semântica aos dados das aplicações. Além disso, a arquitetura da rede de ontologias serve como base para a arquitetura da solução de integração e cada ontologia da rede é usada para a construção de serviços para lidar com os dados das aplicações e repositórios para armazená-los de acordo com uma semântica comum.

Em *Immigrant* (SANTOS JúNIOR, 2023), a partir das necessidades de informação da organização e dos dados que ela tem disponíveis nas aplicações usadas para apoiar o processo de software, é feita a integração semântica dos dados das aplicações e, assim, dados

[7] Ramble ON é uma música da banda Led Zeppelin <https://www.youtube.com/watch?v=EAmIuTI4wRg> que possui o seguinte trecho: "*As folhas estão caindo por toda parte* (dados armazenados em diferentes aplicações), *é hora de pegar o meu caminho* (caminho da melhoria organizacional baseada em dados)

[8] <https://nemo.inf.ufes.br/>

integrados e informações são apresentados em *dashboards* para apoiar o desenvolvimento de software e a tomada de decisão. Neste trabalho, é proposto um processo para auxiliar na identificação das necessidades de informação e na definição de medidas alinhadas a elas e também aos dados disponíveis na organização. *The Band*, o componente de *Immigrant* responsável pela integração de dados, é utilizado para prover os dados considerados no processo proposto.

1.2 Objetivos

Este trabalho tem como objetivo desenvolver um processo de medição baseado em GQ(I)M e *The Band* para auxiliar organizações de software desde a definição de suas necessidades de informação até o atendimento destas utilizando dados disponíveis na organização. Esse objetivo geral é decomposto nos seguintes objetivos específicos:

i. Definir um processo de medição;

ii. Projetar uma arquitetura de acesso a dados para apoiar o processo proposto;

iii. Demonstração do processo proposto utilizando dados disponibilizados por uma organização real.

1.3 Método de Desenvolvimento do Trabalho

Este trabalho foi conduzido de acordo com as seguintes atividades:

i. Revisão da Literatura: consistiu na revisão bibliográfica sobre os temas que fundamentam este trabalho: ontologias e redes de ontologias, ontologias aplicadas à integração de dados, *Immigrant*, *The Band* e medição de software. Para isso, foi realizada a leitura de materiais (livros, teses, dissertações e artigos científicos) pertinentes ao assunto;

ii. Estudo de Tecnologias: nesta atividade foram estudadas as tecnologias utilizadas neste trabalho. A solução foi desenvolvida utilizando tecnologias que viabilizam armazenamento de dados (banco de dados Postgresql[9]), soluções de *Business Inteligence* (Metabase) [10] e de acesso aos dados de forma integrada (Dremio) [11].

iii. Definição do processo de medição: consistiu na definição do processo proposto neste trabalho. O processo consiste, de forma resumida, no levantamento de objetivos, definição de necessidades de informação, coleta de dados organizacionais e representação

[9] <https://www.postgresql.org/>
[10] <https://www.metabase.com/>
[11] <https://www.dremio.com/>

gráfica dos mesmos para obter uma visão mais clara do processo de desenvolvimento de software, identificando pontos de melhoria e oportunidades de otimização.

iv. Implementação da solução para execução do processo proposto: incluiu projetar e arquitetar uma solução computacional para apoiar a extração, armazenamento, integração e apresentação de dados integrados de ferramentas utilizadas no processo de produção de software. Nesta atividade foram utilizados componentes de *The Band*, o banco de dados Postgresql, a solução de *Business Inteligence* Metabase e o mecanismo de *data lake* Dremio para apoio ao acesso aos dados de forma integrada.

v. Utilização do processo de medição proposto: nesta atividade o processo de medição proposto e a solução computacional de apoio foram utilizados considerando-se dados reais de uma *startup*.

vi. Escrita da Monografia: Consiste na escrita da monografia.

1.4 Organização deste Texto

Além desta introdução, esta monografia conta também com os seguintes capítulos:

- **Capítulo 2 - Fundamentação Teórica e Tecnologias Utilizadas**: Apresenta o conteúdo teórico necessário para o entendimento do trabalho proposto;
- **Capítulo 3 - *Ramble ON***: Apresenta o processo proposto e sua utilização considerando dados de uma organização real;
- **Capítulo 4 - Conclusão**: Apresenta as considerações finais deste trabalho e apontamentos para trabalhos futuros.

2 Fundamentação Teórica e Tecnologias Utilizadas

Este capítulo apresenta os principais fundamentos teóricos e as tecnologias relacionadas a este trabalho. Primeiramente, são apresentados os fundamentos teóricos: Engenharia de *Software* Contínua (Seção 2.1, Ontologias e redes de ontologias (Seção 2.3) e a abordagem de integração *Immigrant* (Seção 2.4); em seguida são apresentados as tecnologias utilizadas (Seção 2.5); e por fim, é apresentada as considerações finais do capítulo (Seção 2.6).

2.1 Engenharia de Software Contínua

Engenharia de Software Contínua (ESC) consiste em um conjunto de práticas e aplicações que apoiam uma visão holística do desenvolvimento de software com o objetivo de torná-lo mais rápido, iterativo, integrado, contínuo e alinhado ao negócio (BARCELLOS, 2020).

ESC envolve práticas e aplicações que visam estabelecer um fluxo de ponta a ponta entre a demanda do cliente e a entrega rápida de um produto ou serviço. O quadro geral pelo qual isso pode ser alcançado vai além dos princípios ágeis e traz à tona um conjunto holístico de atividades contínuas (FITZGERALD; STOL, 2017). De acordo com Johanssen et al. (2019), em ESC, os clientes são proativos, e os usuários e outras partes interessadas estão envolvidas no processo, aprendendo com o uso de dados e *feedback*. O planejamento é contínuo, assim como engenharia de requisitos, que se concentra em recursos, arquitetura modularizada e design, e rápida realização de mudanças.

Existem vários trabalhos que dão uma visão geral de ESC (e.g., (OLSSON; ALAHYARI; BOSCH, 2012), (FITZGERALD; STOL, 2017), (JOHANSSEN et al., 2019) (BARCELLOS, 2020)). O modelo *Stairway to Heaven* (StH) (OLSSON; ALAHYARI; BOSCH, 2012) define cinco estágios pelos quais organizações passam na evolução de um desenvolvimento de software tradicional para um desenvolvimento de ESC e orientado a dados:

i. *Desenvolvimento Tradicional* (do inglês, *Traditional Development*): o primeiro estágio consiste no desenvolvimento tradicional. Trata-se de uma abordagem para o desenvolvimento de software caracterizado por ciclos de desenvolvimento lentos, fases sequenciais e uma fase de planejamento rigoroso onde requisitos são estabelecidos antecipadamente. Normalmente, a abordagem é caracterizada por uma interação

em cascata entre gerenciamento de produto, desenvolvimento, teste do sistema e o cliente. Projetos que adotam essa abordagem sofrem de longos ciclos de *feedback* e dificuldades para integrar o *feedback* do cliente no processo de desenvolvimento do produto.

ii. Organização Ágil (do inglês, *Agile Organization*): para superar as limitações impostas pelo modelo tradicional, emprega-se o desenvolvimento ágil. As práticas ágeis são caracterizadas por pequenas equipes multifuncionais de desenvolvimento, *sprints* curtos de desenvolvimento resultando em software funcional e planejamento contínuo no qual o cliente está envolvido para permitir *feedback* do cliente. Em organizações ágeis, no entanto, o gerenciamento de produtos e verificação do sistema ainda funcionam de acordo com a abordagem tradicional.

iii. Integração Contínua (do inglês, *Continuous Integration*): neste estágio a empresa consegue estabelecer práticas que permitem integração frequente de trabalho, *builds* diários (compilação dos artefatos de software) e *commit* de alterações (elaboração por escrito das mudanças feitas em uma aplicação), por exemplo, *builds* automatizados e testes automatizados. Neste ponto, tanto a equipe de desenvolvimento de produtos quanto a equipe de teste e verificação trabalham de acordo com práticas ágeis com ciclos curtos de *feedback* e integração do trabalho.

iv. Entrega Contínua (do inglês, *Continuous Deployment*): a entrega contínua implica que se envie continuamente as alterações feitas em código para o produto ao invés de planejar lançamentos de grandes funcionalidades. Isso permite o *feedback* contínuo, a capacidade de aprender com os dados de uso e elimina o trabalho que não produz valor para o cliente. Neste ponto, P&D (Pesquisa e Desenvolvimento), gerenciamento de produtos e clientes estão todos envolvidos em um ciclo de desenvolvimento rápido e ágil no qual o tempo de resposta é menor se comparado ao tradicional.

v. P&D como um Sistema de Inovação (do inglês, *R&D as an Innovation System*): no estágio final, a organização coleta dados de seus clientes e usa a base de clientes instalada para executar experimentos frequentes de recursos para apoiar o desenvolvimento de software orientado a dados do cliente.

2.2 GQM - *(Goal, Question, Metric)*

O método GQM tem como objetivo fornecer suporte para a definição de medidas que estejam alinhadas com os objetivos de negócio da organização. GQM se baseia na premissa de que, para medir de forma eficaz, uma organização deve primeiro estabelecer seus objetivos e os objetivos de seus projetos e, em seguida, relacioná-los aos dados necessários. Finalmente, deve fornecer um *framework* para análise e interpretação dos

dados em relação aos objetivos estabelecidos (BASILI; CALDIERA; ROMBACH, 1994).

A aplicação do GQM resulta em uma estrutura hierárquica de três níveis: Nível Conceitual (Objetivo), Nível Operacional (Questão) e Nível Quantitativo (Medida). Através de uma abordagem *top-down* orientada a objetivos, são definidos objetivos que são refinados em questões. Medidas são então definidas de forma a serem adequadas para responder às questões. A análise das medidas permite verificar o grau de alcance dos objetivos (SOLINGEN, 1999). No contexto deste trabalho o conceito "Questão" abordado por Basili, Caldiera e Rombach (1994) é referido como "Necessidade de Informação" pois diversos dos trabalhos usados como base para este trabalho utilizam esta nomenclatura ao se referir a perguntas que precisam ser respondidas afim auxiliar na realização dos objetivos traçados.

Para exemplificar o uso do método GQM: suponhamos que uma equipe de desenvolvimento de software deseje melhorar seu processo de revisão de código. Assim, ela poderia ter como objetivo "melhorar a eficácia das revisões de código". A partir desse objetivo, a equipe poderia identificar necessidades de informação tais como "Quantos defeitos são encontrados durante as revisões de código?" e "Quanto tempo leva para concluir uma revisão de código?". Para prover dados que permitam responder a essas questões, a equipe poderia definir medidas como "Número de defeitos encontrados por revisão de código" e "Tempo médio gasto na revisão de código por solicitação pull".

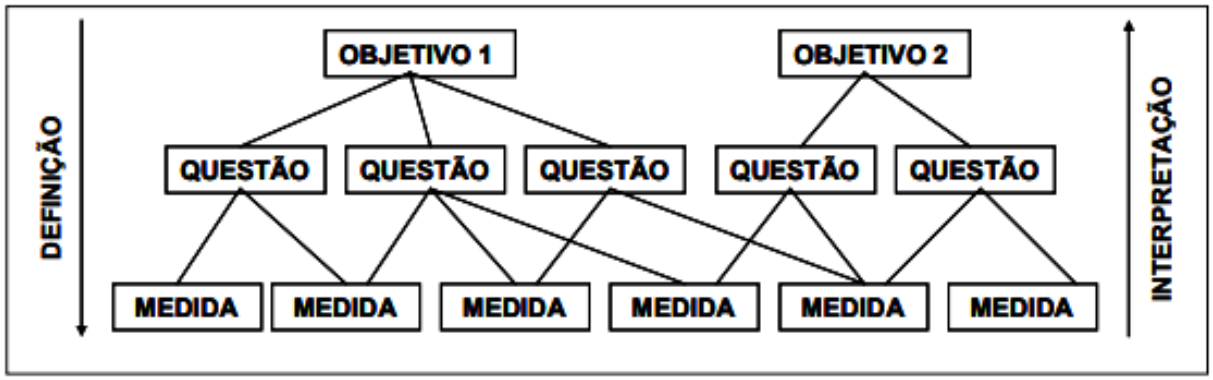

Figura 1 – GQM (BASILI; CALDIERA; ROMBACH, 1994).

Assim, de forma geral, um objetivo necessita de informações para ser monitorado /alcançado e essas informações são representadas através de questões que, por sua vez, precisam de dados fornecidos por medidas para serem respondidas.

GQM foi usado com base para novos métodos de medição mais complexos, tais como o GQM*Stratagies, que fornece mecanismos para relacionar explicitamente os objetivos de medição de software aos objetivos de mais alto nível da organização de software e aos objetivos e estratégias do negócio como um todo (BASILI et al., 2007); e o GQ(I)M, que propõe o alinhamento de medidas e indicadores com os objetivos. Esse método está baseado no entendimento de que identificar questões e medidas sem estabelecer um indicador muitas

vezes não é suficiente (PARK; GOETHERT; FLORAC, 1996). A seção a seguir detalha alguns aspectos do GQ(I)M.

2.2.1 GQ(I)M - *(Goal, Question, Indicator, Metric)*

(PARK; GOETHERT; FLORAC, 1996) concluíram que identificar questões e medidas sem visualizar um indicador geralmente não é suficiente para iniciar um programa de medição bem-sucedido. As exibições ou relatórios usados para comunicar os dados (chamados de indicadores no método GQ(I)M) são um elo fundamental que pode determinar o sucesso ou o fracasso de um programa de medição. Esses indicadores servem como especificação de requisitos para os dados que devem ser coletados, o processamento e análise dos dados de medição que devem ocorrer e o cronograma pelo qual essas atividades ocorrem.

Ao identificar questões e definir indicadores, é importante ter em mente o(s) objetivo(s) sendo abordados e como os resultados da medição serão usados. Deve-se desenvolver perguntas relacionadas a cada objetivo, ou sub-objetivos derivados, que ajudariam a determinar se os critérios de sucesso foram atendidos. Tais perguntas são formuladas de maneira a obter uma resposta quantitativa. Usando as respostas a essas perguntas, alguns indicadores gerais podem ser descritos. Uma vez conhecidas as questões ou problemas, pode-se desenhar um indicador específico. Recomenda-se que seja desenvolvido um indicador para cada pergunta. Goethert e Fisher (2003) sugerem os seguintes passos para a aplicação do método GQ(i)M:

i. **Identificar os objetivos de negócio**: os objetivos organizacionais devem ser identificados para garantir o alinhamento com questões importantes;

ii. **Identificar o que se quer saber ou aprender**: as atividades de medição devem estar alinhadas com os objetivos, vinculadas às estratégias e processos de negócios;

iii. **Identificar sub-objetivos**: os sub-objetivos fornecem um refinamento do objetivo e fornecem respostas às perguntas;

iv. **Identificar Entidades e Atributos**: os sub-objetivos e questões relacionadas definem o foco das medidas;

v. **Formalizar seus objetivos de medição**: os objetivos de medição combinam propósito, perspectiva e fatores ambientais e contextuais para informar as atividades de projeto e análise;

vi. **Identificar questões quantificáveis e indicadores relacionados**: esboçar medidas ou exibições ajuda a garantir que os requisitos de medição sejam atendidos;

vii. **Identificar os Elementos de Dados**: nesta etapa, são identificados os elementos de dados necessários para a construção dos indicadores. A existência e as fontes dos dados necessários são avaliadas;

viii. **Definir as medidas**: as definições devem ser criadas com o objetivo do indicador em mente e fornecer uma resposta à pergunta;

ix. **Identificar ações para implementação**: conhecendo os dados necessários e definindo-os, analisa-se a situação existente na organização em relação às necessidades de medição;

x. **Preparar um plano de ação**: uma vez concluída a análise de lacunas entre os dados necessários e as atividades de medição existentes, um plano de ação pode ser preparado.

A Figura 2 ilustra os passos do GQ(I)M.

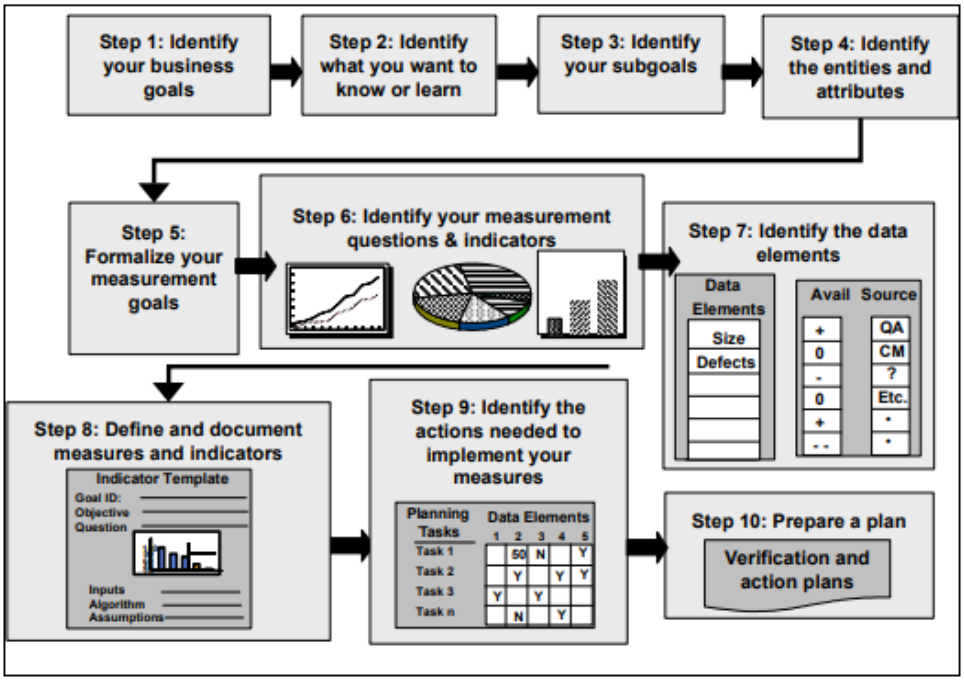

Figura 2 – Passo a passo do GQ(I)M (GOETHERT; FISHER, 2003).

2.3 Ontologias

Ontologias são reconhecidas como instrumentos adequados para lidar com questões de interoperabilidade semântica (IZZA, 2009). Uma ontologia é uma especificação formal e explícita de uma conceituação compartilhada (STUDER; BENJAMINS; FENSEL, 1998). Para lidar com a questão da interoperabilidade semântica, ontologias podem ser utilizadas para estabelecer um entendimento comum sobre o domínio de interesse, servindo como

uma interlíngua para a comunicação. Ao explicitar e formalmente definir os conceitos, relações, propriedades e restrições em um domínio ou tarefa de interesse, uma ontologia pode ser usada para apoiar a comunicação entre pessoas trabalhando nesse universo de discurso, para facilitar a integração de dados, modelos e sistemas desenvolvidos de forma independente e como uma especificação reutilizável para a construção de sistemas no domínio (JASPER; USCHOLD, 1998).

Porém, o uso de uma única ontologia para apoiar a integração de diversas aplicações, cobrindo diferentes domínios, pode ser inviável, uma vez que tal ontologia pode ser difícil de manipular, usar e manter. Por isso, em casos envolvendo diferentes domínios, o uso de redes de ontologias (do inglês *Ontology Network* - ON) é uma abordagem mais promissora. Uma ON é uma coleção de ontologias relacionadas através de uma variedade de relacionamentos, como alinhamento e dependência. Um ontologia em rede é, então, uma ontologia integrada a uma rede de ontologias (SUÁREZ-FIGUEROA et al., 2012). Neste trabalho, foram utilizadas ontologias de *Software Engineering Ontology Network* (SEON) (RUY et al., 2016), que é brevemente apresentada a seguir.

2.3.1 SEON - *Software Engineering Ontology Network*

SEON (RUY et al., 2016) é uma rede de ontologias que descreve vários subdomínios do domínio de Engenharia de Software. SEON organiza suas ontologias de acordo com as camadas definidas por Scherp et al. (2011), a saber: *Foundational*, *Core* e *Domain*. A Figura 3 apresenta a arquitetura SEON. Na *Foundational Layer*, há a *Unified Foundational Ontology* (UFO) (GUIZZARDI, 2005), cujas distinções são usadas para classificar conceitos de SEON como, por exemplo, objetos, ações, compromissos, agentes, papéis, objetivos e assim por diante. UFO fornece o embasamento necessário para os conceitos e relações de todas as ontologias em rede. A *Core Layer* contém ontologias de núcleo, que fornece conhecimento central relativo a Engenharia de Software para a rede. Nesta camada há duas ontologias relacionadas a Engenharia de Software: a *software Process Ontology* (SPO) (BRINGUENTE; FALBO; GUIZZARDI, 2011) (RUY, 2017) e a *System and software Ontology* (SysSwO) (DUARTE et al., 2018) (COSTA et al., 2022). SPO estabelece uma conceitualização comum sobre processos de software, enquanto SysSwO trata da natureza do sistema e do software, incluindo artefatos de software, constituição de software, execução de software, sistema computacional e equipamentos de hardware. Existem também duas ontologias de núcleo mais gerais: a *Enterprise Ontology* (EO) (FALBO et al., 2014) e a *Ontology on Measurements* (COM) (BARCELLOS; FALBO; FRAUCHES, 2014). EO lida com aspectos relacionados às organizações, como membros da equipe, enquanto COM define conceitos relacionados ao domínio da medição. Na *Domain-specific Layer* aparecem as ontologias de domínio, fundamentadas em ontologias de núcleo e em UFO e abrangendo vários subdomínios de Engenharia de Software (por exemplo, requisitos de software, design,

gerenciamento de configuração e medição de software).

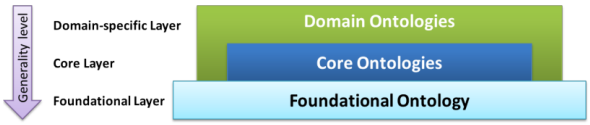

Figura 3 – Arquitetura de SEON (RUY et al., 2016).

A seguir são apresentados fragmentos da *Quality Assurance Process Ontology* (RUY, 2017) e *Enterprise Ontology* (FALBO et al., 2014) de SEON relevantes para este trabalho.

2.3.2 EO - Enterprise Ontology

EO (FALBO et al., 2014) é uma ontologia de núcleo que lida com os aspectos relacionados organizações. A Figura 4, extraído de Santos Júnior (2023) apresenta um fragmento incluindo conceitos relevantes para este trabalho.

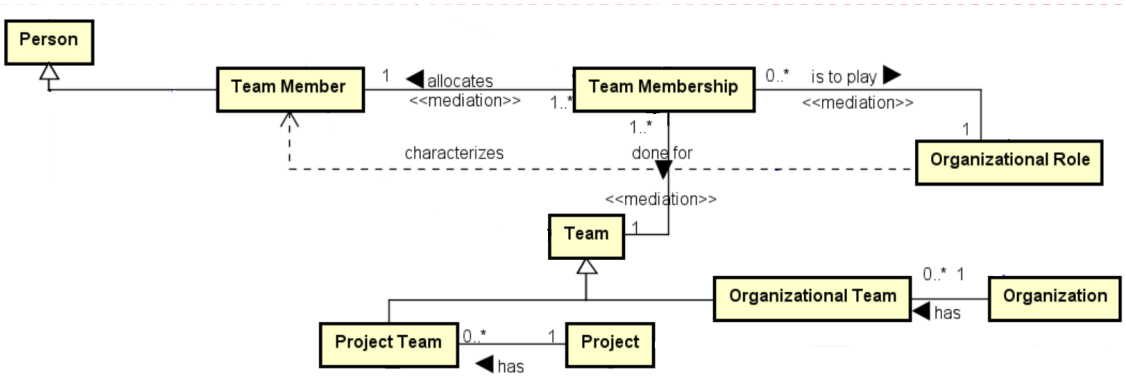

Figura 4 – Fragmento de EO (SANTOS JúNIOR, 2023).

Dentre os conceitos abordados há *Organizational Role*, que consiste em um papel social reconhecido por uma *Organization*, que é designado a agentes quando são contratados, incluídoso em uma equipe, alocação ou simplesmente participam de uma atividade junto à organização. Gerente de projeto, designer, e programador são exemplos de *Organization Roles*.

Um *Team Member* é uma *Person* que ocupa um *Organizational Role* em um *Team*. Um *Team* pode estar relacionado a um *Project (Project Team)*, por exemplo, a equipe de desenvolvimento de um determinado projeto de software de uma *Software Organization*, ou para uma *Organization (Organizational Team)*, por exemplo, a equipe de marketing de uma *Software Organization*. A alocação de um *Team Member* para ocupar um *Organizational Role* em uma *Team* é feita através de uma relação de *Team Membership*. Por exemplo, um *Team Membership* pode alocar um funcionário como *Team Member* para desempenhar o *Organizational Role* de programador em uma *Team* de desenvolvimento de um projeto de especifica de uma dada *Software Organization*.

2.3.3 *Continuum*: (sub)Rede de Ontologias sobre Engenharia de Software Continua.

ESC é um domínio complexo que envolve domínios de Negócios, Desenvolvimento, Operações e Inovação para entregar produtos ou serviços que atendam à demanda dos clientes. Com o objetivo de fornecer conhecimento sobre ESC, Santos Júnior (2023) desenvolveu uma rede de ontologias, denominada *Continuum*, que visa representar a conceituação relacionada aos processos envolvidos em ESC. Na visão geral da Engenharia de Software, ESC aparece como um (grande) subdomínio envolvendo outros subdomínios. Assim, Santos Júnior (2023) desenvolveu *Continuum* como uma sub-rede de SEON (RUY et al., 2016), reutilizando alguns elementos de SEON (como sua arquitetura, mecanismos de integração e ontologias em rede) para desenvolver *Continuum*.

Continuum pode ser usada como um modelo conceitual de referência para dar suporte a diferentes soluções relacionadas ao conhecimento e à interoperabilidade, como comunicação, aprendizado, harmonização de padrões, documentação semântica e integração de aplicações, entre outros. Neste trabalho, *Continuum* é utilizada para auxiliar a integração de dados. Ela fornece a conceitualização necessária para dar suporte à integração semântica. Mais especificamente, *Continuum* possui os modelos conceituais e axiomas que são usados para criar alguns componentes presentes na solução de integração *The Band* (por exemplo, bancos de dados e serviços federados).

Uma vez que *Continuum* foi desenvolvida para apoiar a integração de dados, suas ontologias foram desenvolvidas para definir eventos (por exemplo, data de início e término das atividades realizadas em um projeto, pessoas que realizaram tais atividades e resultados produzidos nas atividades que ocorreram no passado). Dessa forma, os verbos usados para descrever relacionamentos nos modelos conceituais das ontologias de *Continuum* estão no pretérito.

As seções a seguir apresentam a arquitetura de *Continuum* e uma ontologias de domínio relevante a este trabalho: *Scrum Reference Ontology (SRO)*, que aborda aspectos relacionados ao desenvolvimento ágil de software com Scrum (SANTOS JÚNIOR et al., 2021).

A Figura 5 mostra a arquitetura da versão atual da *Continuum*. Sendo uma sub-rede da SEON, *Continuum* tem a mesma arquitetura de três camadas usada em SEON. Assim, UFO (GUIZZARDI, 2005) (GUIZZARDI et al., 2022) fundamenta ontologias de núcleo que, por sua vez, são usadas para definir ontologias específicas de domínio. Na Figura 5, cada círculo (nó da rede) representa uma ontologia. O círculo pontilhado representa uma ontologia em desenvolvimento. As setas denotam as relações de dependência entre as ontologias em rede, indicando que os conceitos da ontologia de destino são reutilizados pela ontologia de origem.

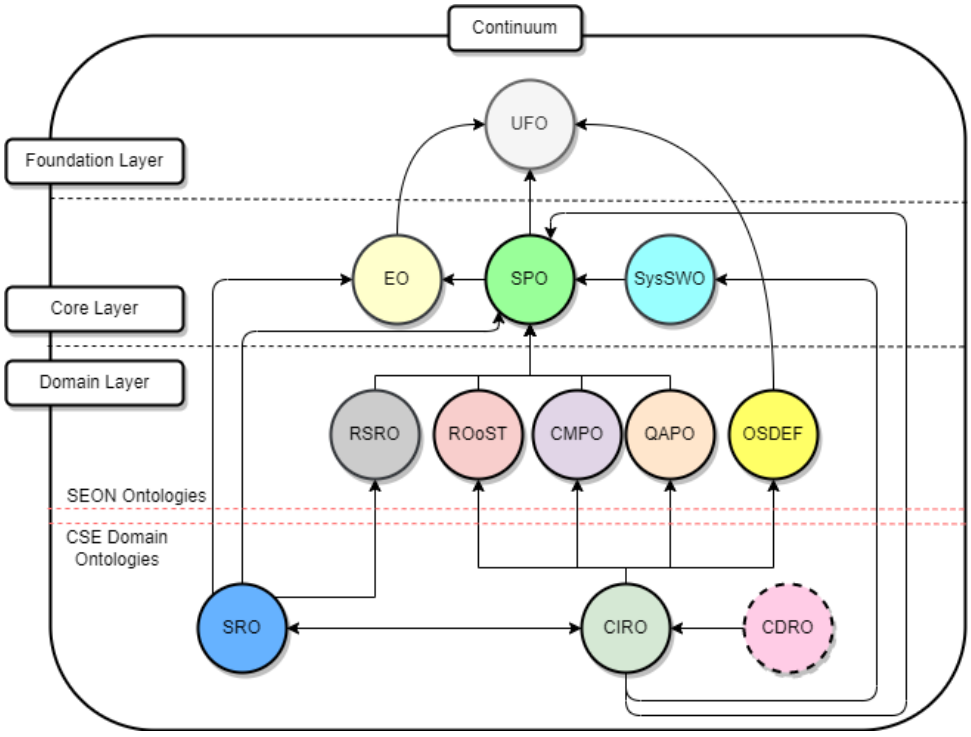

Figura 5 – Arquitetura de *Continuum* (SANTOS JúNIOR, 2023).

2.3.4 SRO - *Scrum Ontology Reference*

Scrum é um método ágil criado a partir da observação de que o desenvolvimento de software é muito complexo e imprevisível para ser totalmente planejado no início de um projeto. Portanto, ele emprega uma abordagem iterativa e incremental para otimizar a previsibilidade e controlar o risco (SCHWABER; SUTHERLAND, 2013). O processo Scrum começa com um planejamento inicial para estabelecer os requisitos do produto e registrá-los ordenados no *backlog* do produto, que está em constante evolução ao longo do processo de desenvolvimento (RISING; JANOFF, 2000). Segundo o Scrum, um projeto é desenvolvido por meio de ciclos incrementais (geralmente com duração de um mês ou menos) chamados *sprints*. Para cada *sprint*, há uma reunião de planejamento do *sprint*, quando a equipe seleciona do *backlog* do produto os itens a serem abordados no *sprint* e planeja o trabalho a ser feito. O resultado do planejamento é registrado no *sprint backlog*. Um *sprint* produz um produto visível, utilizável e entregável que implementa uma ou mais interações do usuário com o sistema. A ideia chave por trás de um *sprint* é entregar uma funcionalidade que represente valor. Cada incremento de produto se baseia em incrementos anteriores. O objetivo é completar as tarefas definidas no *sprint backlog* até a data de entrega do *sprint* e entregar um incremento de um produto pronto. Um incremento é dito pronto se estiver em conformidade com os critérios de aceitação estabelecidos e, assim, puder ser entregue ao cliente.

A Scrum Reference Ontology (SRO), fornece uma conceituação abrangente sobre Scrum no contexto de desenvolvimento de software, detalhando o processo Scrum, os

vários papéis desempenhados no processo e os artefatos manipulados (SANTOS JÚNIOR et al., 2021). SRO foi desenvolvida integrada a SEON. Assim, uma vez que SEON descreve conceitos gerais em Engenharia de Software, SRO se concentra em especificidades do processo de desenvolvimento usando Scrum, enquanto reutiliza noções gerais de Engenharia de Software como "projeto de software", "equipe do projeto" e "requisito". A SRO está organizada em cinco sub-ontologias (SANTOS JÚNIOR et al., 2021):

i. *Scrum Process*: aborda o eventos que ocorrem em um projeto que adota o Scrum, como as cerimônias do Scrum;

ii. *Scrum Stakeholders*: diz respeito às equipes, agentes e papéis envolvidos em um projeto Scrum;

iii. *Scrum Stakeholder Participation*: trata da participação dos *stakeholders* nos eventos de um projeto Scrum;

iv. *Product and Sprint Backlog*: aborda aspectos relacionados aos requisitos estabelecidos em um Scrum projeto e atividades planejadas para concretizá-los;

v. *Scrum Deliverables*: foca sobre os resultados produzidos durante um projeto Scrum.

A Figura 6 mostra uma visão geral de SRO. Na figura, cada pacote dentro do pacote SRO representa uma sub-ontologia de SRO. A relação de dependência indica que uma (sub)ontologia utiliza conceitos de outra. Já a Figura 7 representa um fragmento de SRO referente a *Product and Sprint Backlog Subongology*.

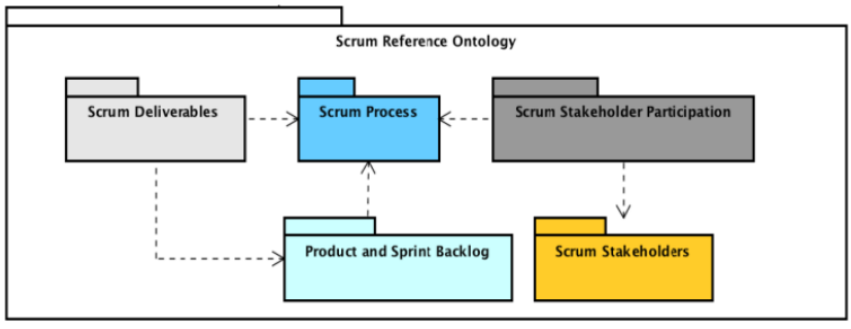

Figura 6 – Arquitetura de SRO (SANTOS JÚNIOR et al., 2021).

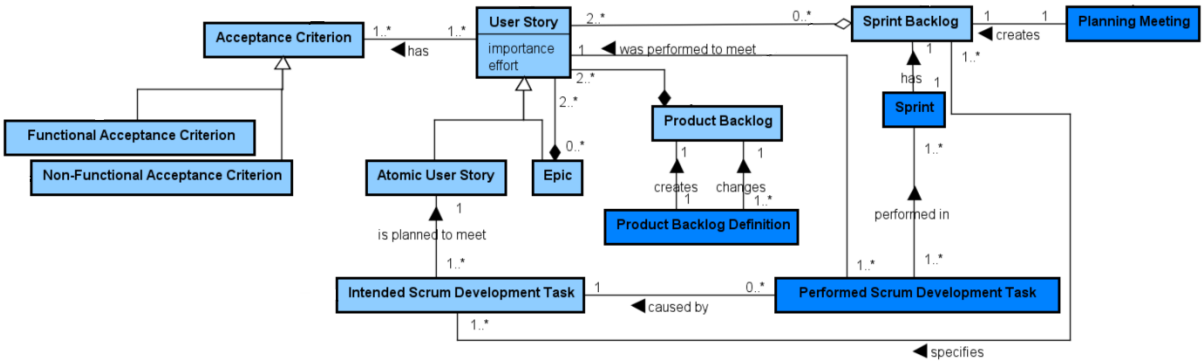

Figura 7 – Fragmento da *Product and Sprint Backlog Subongology (SANTOS JÚNIOR et al., 2021)*.

2.4 *Immigrant*: Uma abordagem Baseada em Redes de Ontologias para Integração de Dados

Immigrant é uma abordagem baseada em rede de ontologia para integrar dados de aplicações para apoiar o desenvolvimento de software orientado a dados no contexto de ESC. A abordagem considera, em uma perspetiva *top-down*, as necessidades de informações da organização a serem obtidas a partir do estado atual da organização e das ações de melhoria e implementação de práticas de ESC necessárias. Para isso, *Immigrant* provê dois componentes: *California* (SANTOS; BARCELLOS; CALHAU, 2020), um processo baseado na teoria de sistemas para implementação de práticas de ESC, e *Zeppelin* (SANTOS JÚNIOR et al., 2021), um instrumento de diagnóstico de práticas de ESC. Em uma perspetiva *bottom-up*, a abordagem considera os dados disponíveis nas aplicações utilizadas pela organização para apoiar seu processo de desenvolvimento. Considerando as necessidades de informação da organização e os dados disponíveis, a abordagem provê o componente *The Band* (SANTOS JúNIOR, 2023), que utiliza ontologias de *Continuum* como interlíngua para integrar dados de diferentes aplicações e apresentar dados integrados em *dashboards* para permitir o monitoramento e *insights* visando à tomada de decisão orientada a dados. Assim, *The Band* permite que *Immigrant* aborde questões semânticas envolvidas na integração de dados para fornecer dados integrados e significativos considerando as necessidades de informação da organização. A Figura 8 apresenta uma visão geral de *Immigrant*.

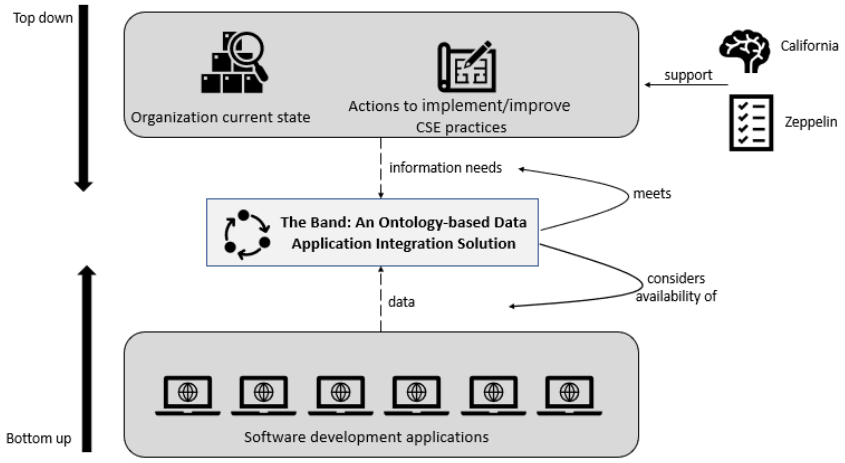

Figura 8 – Visão Geral de *Immigrant* (SANTOS JúNIOR, 2023).

2.4.1 *The Band*

The Band (SANTOS JúNIOR, 2023) é uma solução de software de integração de dados baseada em redes de ontologias e *Federated Information System*[1] (FIS. A rede de ontologias fornece as camadas de abstração a serem consideradas (ou seja, *fundamental, core e domain*), os subdomínios e respectivos conceitos, relacionamentos e axiomas a serem abordados. A arquitetura FIS, por sua vez, fornece meios para criar sistemas que compartilham, trocam e combinam dados e uma interface para um cliente acessar dados na federação de sistemas. Em *The Band*, cada ontologia em rede é usada como base para um *ontology-based service* (OBS) que é um sistema da federação *The Band* e possui seu próprio repositório, denominado *ontology-based data repository* (OBDR) (SANTOS JúNIOR, 2023). Portanto, cada OBS captura, armazena e compartilha dados relacionados à porção de domínio endereçada pela referida ontologia em rede.

Ao criar OBSs baseados em ontologias em rede, é possível observar os relacionamentos entre as ontologias em rede e identificar quais dados precisam ser trocados entre diferentes OBSs. Ao organizar OBSs em um FIS, os critérios FIS relevantes podem ser considerados para contribuir para a definição da arquitetura da solução. Por exemplo, ao aplicar o critério de transparência, *The Band* deve permitir que um cliente pesquise dados sem saber onde eles estão armazenados e usando uma linguagem de consulta baseada em uma conceituação comum (ou seja, conceitos da ON). Ao aplicar o critério de autonomia, os OBSs devem ser capazes de trabalhar independentemente de outros OBSs e lidar com todos os dados necessários para tratar o domínio de interesse.

A Figura 9 ilustra como ontologias da rede dão origem a OBSs e OBDRs na

[1] Um Sistema de Informação Federado (FIS) é um conjunto de componentes de sistemas de informação distintos e autônomos, os participantes de uma federação. Os participantes em primeiro lugar operam de forma independente, mas possivelmente abriram mão de alguma autonomia para participar de uma federação. (BUSSE et al., 1999)

arquitetura de *The Band*. A Figura 10, por sua vez, mostra um Fragmento de diagrama Entidade Relacionamento da OBDR baseada na SRO.

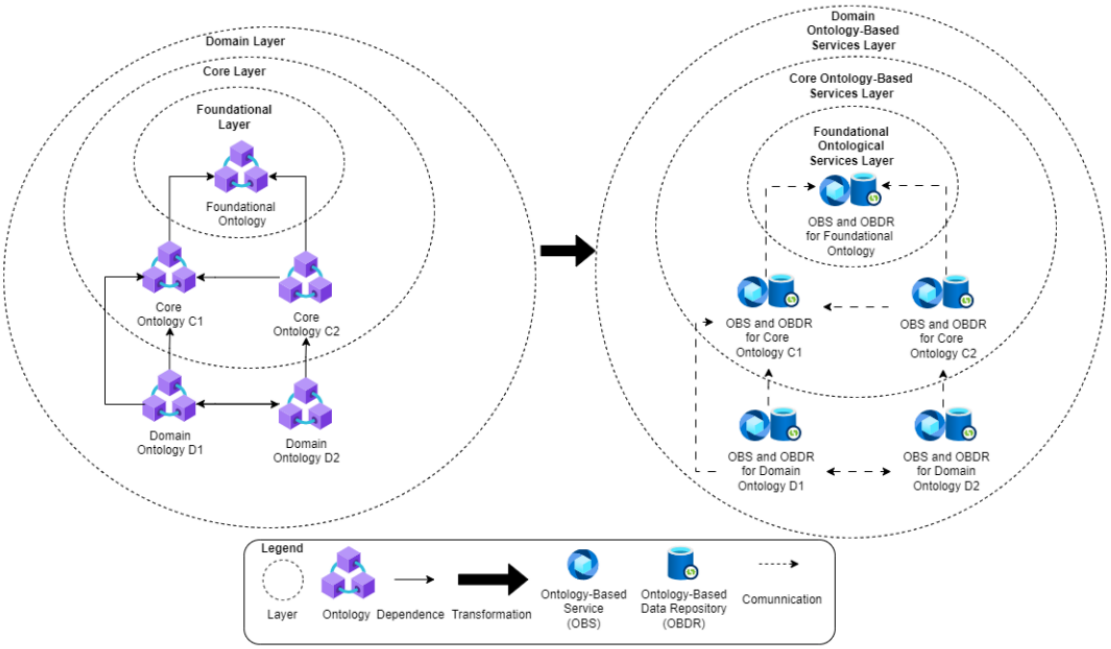

Figura 9 – Transformação de ON em FIS (SANTOS JúNIOR, 2023).

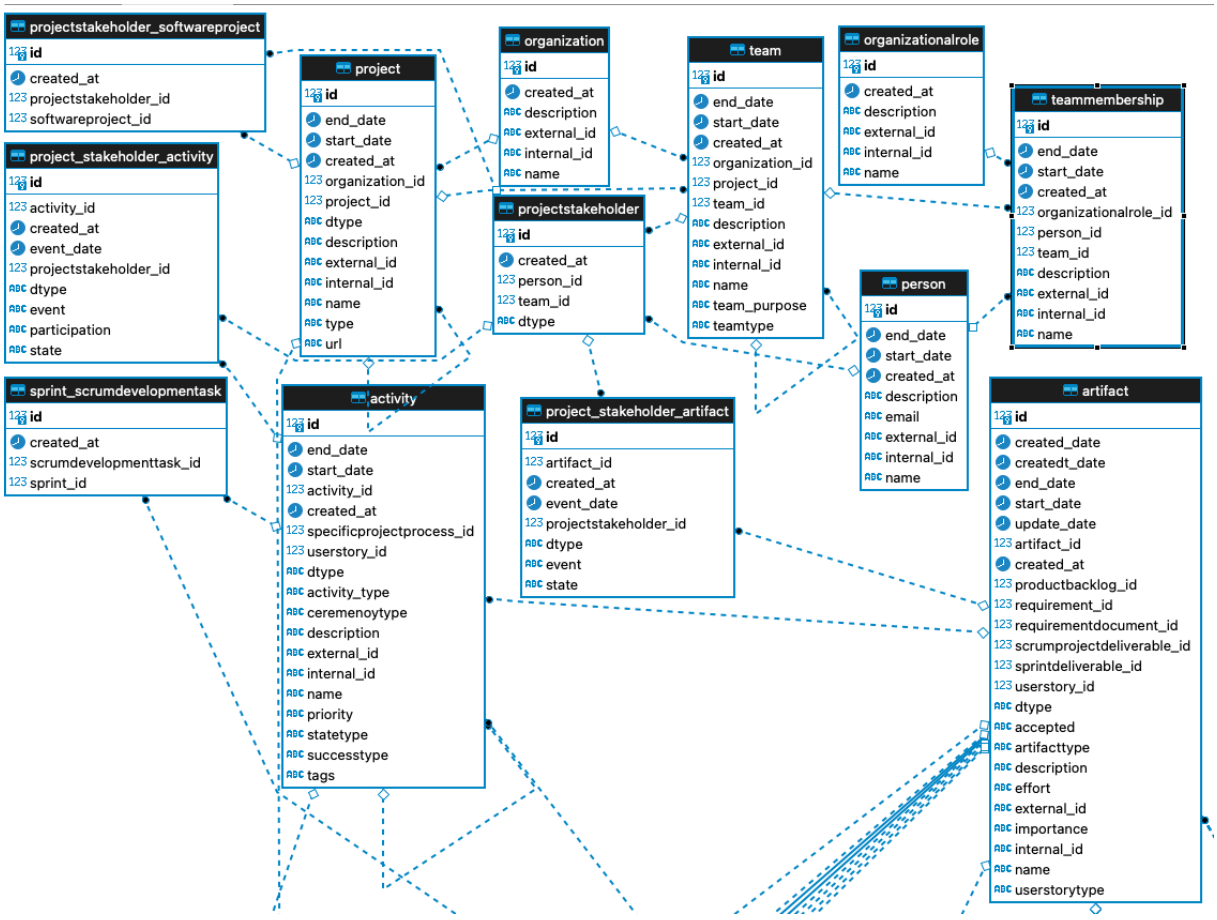

Figura 10 – Fragmento de diagrama Entidade Relacionamento da OBDR baseada na SRO.

2.5 Tecnologias Utilizadas

A seguir são apresentadas algumas informações sobre as tecnologias utilizadas neste trabalho.

2.5.1 Dremio

Dremio[2] é um mecanismo de *data lake*[3] que permite acesso rápido e fácil a dados de várias fontes, como armazenamento em nuvem, bancos de dados relacionais, bancos de dados NoSQL e APIs (DREMIO, 2023). Dremio permite que os usuários consultem e analisem dados em seu formato nativo, sem a necessidade de movimentação ou transformação de dados. Também fornece recursos como catalogação de dados, governança de dados e segurança de dados.

2.5.2 PostgreSQL

O PostgreSQL é um sistema de gerenciamento de banco de dados relacional *open source* (The PostgreSQL Global Development Group, 2023)[4]. Ele é conhecido por sua confiabilidade e recursos avançados. Com suporte para consultas complexas e grande volume de dados, o PostgreSQL é capaz de armazenar e manipular informações de forma eficiente. É uma escolha popular para projetos devido à sua estabilidade e comunidade ativa de desenvolvedores.

2.5.3 Metabase

Metabase[5] é uma aplicação *open source* de *business intelligence* que permite formulação de consultas a dados e visualização de respostas em gráficos, tabelas ou painéis interativos (*dashboards*) (METABASE, 2023). Com o Metabase, é possível conectar-se a diversas fontes de dados, desde bancos de dados relacionais até repositórios de dados em nuvem, e explorar seus dados com uma interface intuitiva e amigável. Também é possível criar modelos semânticos que enriquecem os dados coletados com metadados, segmentos e métricas, e facilitam a análise por pessoas não técnicas.

2.5.4 Arquitetura de acesso a dados

Para apoiar a coleta de dados das OBDRs, este trabalho propõe uma solução baseada em *The Band* (SANTOS JúNIOR, 2023) e nas soluções de *Business Inteligence*

[2] <https://www.dremio.com/>

[3] Um *data lake* é um repositório centralizado que permite armazenar todos os seus dados estruturados e não estruturados em qualquer escala (REDHAT, 2019).

[4] <https://www.postgresql.org/>

[5] <https://www.metabase.com/>

(Metabase) e de apoio à acesso a dados (Dremio). A Figura 11 provê uma visão geral da solução proposta. Ela define a estrutura geral da solução, representando seus elementos e como eles se relacionam entre si.

A solução foi desenvolvida utilizando-se as tecnologias apresentadas na Seção 2.5 com apoio da solução de integração semântica *The Band* (SANTOS JúNIOR, 2023). *The Band* captura dados armazenados nas aplicações utilizadas pela organização e os armazena, sem intervenção humana, em OBDRs, seguindo a arquitetura da rede de ontologias *Continuum*. Dessa forma, os dados são armazenados em repositórios referentes ao subdomínio ao qual pertencem. Os OBDRs são, então, as fontes de dados consideradas nas atividades Identificar dados a coletar e Coletar valores para as medidas. Esses dados são disponibilizados em bancos de dados Postgres que são acessados pelo Dremio, possibilitando assim que diferentes OBDRs possam ser acessados em uma só consulta SQL. Por fim, a aplicação Metabase é usada para acessar os dados através do Dremio e Gerar *dashboards*.

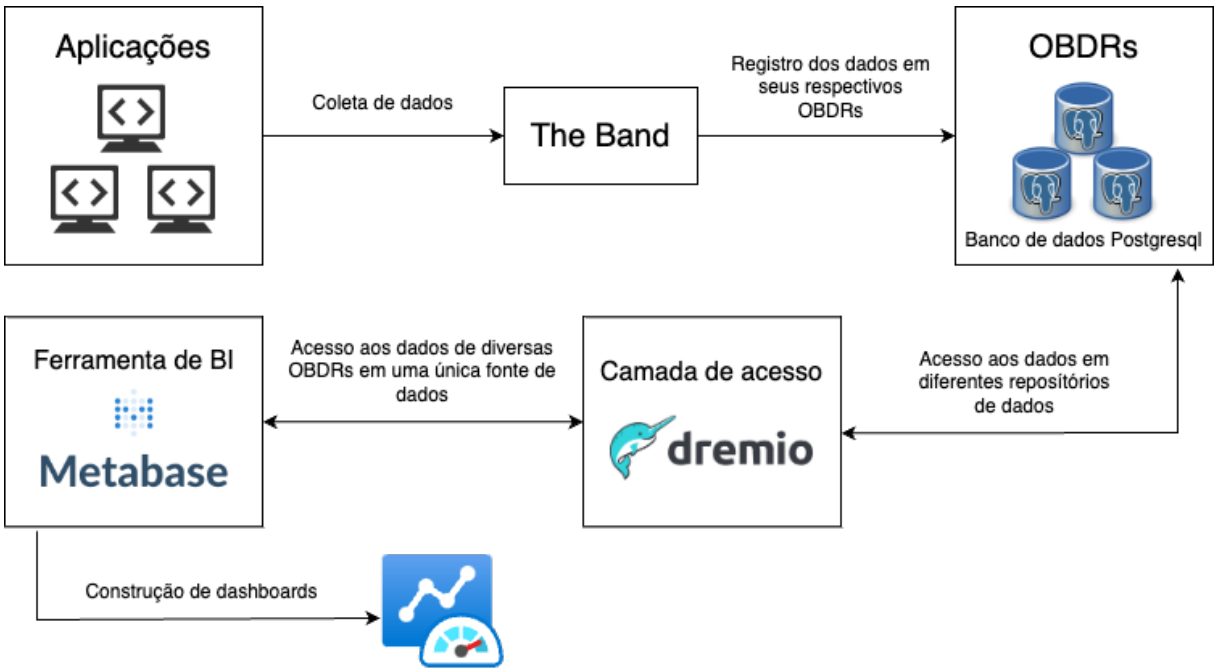

Figura 11 – Solução proposta para apoiar atividades de *Ramble ON*.

2.6 Considerações Finais do Capítulo

Neste capítulo foi apresentada a fundamentação teórica e os recursos tecnológicos necessários para o desenvolvimento deste trabalho. Quanto à fundamentação teórica, foi discorrido sobre Engenharia de Software Contínua, GQM, GQ(I)M, ontologias, SEON, *Continuum*, *Immigrant* e *The Band*. Além disso, também foram apresentadas as tecnologias utilizadas no desenvolvimento da solução proposta.

3 *Ramble ON*: Um Processo de Medição baseado em GQ(I)M e Dados Organizacionais

Este capítulo tem como objetivo apresentar Ramble ON e o seu uso considerando dados reais de uma organização de software (startup). O capítulo está organizado como segue: a seção 3.1 apresenta o propcesso proposto, a seção 3.2 apresenta o uso do processo proposto e de uma solução computacional para atender necessidades de informação de uma startup utilizando dados que ela tem disponíveis e, por fim, a seção 3.3 apresenta as considerações finais deste capítulo.

3.1 Abordagem Proposta

Ramble ON é um processo de medição baseado no GQ(I)M que visa à definição de medidas alinhadas às necessidades de informação e à utilização de dados disponíveis nas aplicações usadas pela organização para prover informações que apoiem o desenvolvimento de software orientado a dados (i.e., o uso de dados para a realização de atividades do processo de software e para a tomada de decisão). *Ramble ON* consiste em oito atividades que guiam o engenheiro de software (ou outro *stakeholder*), na definição das necessidades de informação da organização, medidas e identificação de dados necessários, passando pela análise de viabilidade de responder as necessidades de informação até chegar à apresentação de dados integrados que atendam às necessidades de informação identificadas. A Figura 12 apresenta uma visão geral da *Ramble ON*. Assim como o GQ(I)M, *Ramble ON* prevê a definição de objetivos, necessidades de informação, medidas, indicadores e suas respectivas representações gráficas. No entanto, *Ramble ON* vai além do escopo do GQ(I)M. Enquanto o GQ(I)M foca no planejamento da medição, *Ramble ON* também inclui a coleta dos dados para as medidas (i.e., valores medidos), a apresentação dos dados e análise dos resultados.

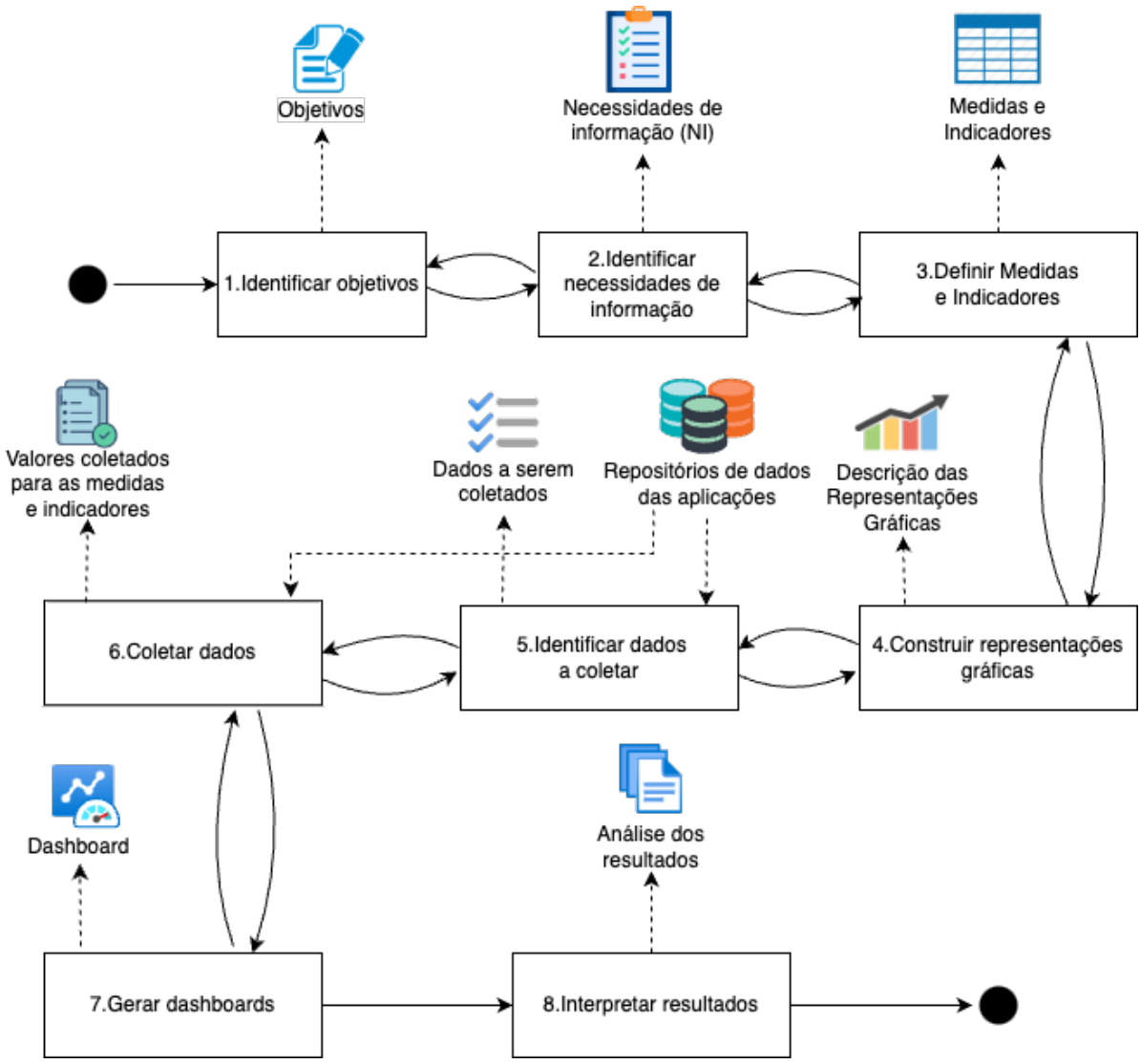

Figura 12 – Visão Geral de *Ramble ON*.

De forma resumida, a primeira atividade, Identificar objetivos define os objetivos que se quer monitorar/alcançar considerando a análise dos dados. A segunda atividade, Identificar necessidades de informação, define as questões que se quer responder com os dados. A terceira atividade, Definir Medidas e Indicadores, define as medidas que serão utilizadas para atender às necessidades de informação e quais delas servirão como indicadores do alcance aos objetivos definidos. Indicadores são medidas que, associadas a critérios de avaliação ou decisão pré-definidos, são capazes de fornecer informações que descrevem o alcance dos objetivos estabelecidos (BARCELLOS; FALBO; ROCHA, 2009). A quarta atividade, Construir representações gráficas, tem como objetivo construir representações gráficas que serão usadas para representar os dados e auxiliar a avaliar o alcance aos objetivos. A quinta atividade, Identificar dados a coletar, visa identificar os dados que devem ser coletados para obter valores para as medidas definidas e popular as representações gráficas. A sexta atividade, Coletar dados, visa acessar os repositórios de dados da organização e extrair os dados para construir as representações gráficas

previamente estabelecidas. A sétima atividade, Gerar *dashboard*, consolida as representações gráficas contendo os dados coletados em um *dashboard* que provê informações integradas e significativas para os *stakehorlders*. Por fim, a oitava atividade, Interpretar resultados, consiste na análise dos dados para monitoramento do alcance aos objetivos definidos, obtenção de informações úteis à tomada de decisão, extração de *insights* e recomendações de melhorias acerca dos processos e produtos de software.

As atividades de *Ramble ON* têm uma natureza iterativa, possibilitando que ao longo de seu uso se retorne a passos anteriores para ajustes considerando-se o aprendizado ou novas informações obtidas em passos seguintes. Os resultados produzidos em uma atividade são usados como entrada para a atividade subsequente - para não poluir a figura, isso não está representado explicitamente. As próximas seções descrevem e exemplificam cada atividade da *Ramble ON*.

3.1.1 Identificar Objetivos

Nesta atividade são identificados os objetivos da organização. Geralmente, identifica-se os objetivos de negócio e a partir deles deriva-se objetivos técnicos. Objetivos de negócio de uma organização de software visam melhorar a produtividade, flexibilidade e satisfação do cliente, norteando as medições que devem ser feitas em seus processos, produtos e recursos (BASILI; CALDIERA; ROMBACH, 1994). Objetivos técnicos, por sua vez, são objetivos mais diretamente relacionados às áreas técnicas da organização (e.g., a área de testes pode ter o objetivo de "melhorar a eficácia dos testes", o qual contribui com o objetivo de negócio "melhorar a satisfação dos clientes", uma vez que influencia na entrega de produtos de qualidade). Seguindo Basili, Caldiera e Rombach (1994) podemos definir:

- *Produtos*: artefatos, entregáveis e documentos que são produzidos durante o ciclo de vida do sistema; por exemplo, especificações, projetos, programas, conjuntos de testes.
- *Processos*: atividades que devem ser realizadas para produzir o software ; por exemplo, especificação, projeto, teste, entrevista.
- *Recursos*: itens usados pelos processos para produzir suas saídas; por exemplo, hardware, software e espaço físico.

Não existe uma única forma de se levantar os objetivos. Por exemplo, Goethert e Fisher (2003) argumentam que é necessário que antes mesmo de se escrever os objetivos de uma dada organização, ela tenha estabelecido sua Missão e sua Visão. A Missão descreve a declaração de propósito da organização (e.g., "Desenvolver e manter tecnologias de informação e sistemas intensivos de software integrados para o governo"); enquanto a Visão descreve o que a organização deseja aspirar, a tarefa especial da organização e qual

motivação específica une as partes interessadas da organização - incluindo membros, líderes e qualquer outra pessoa afetada pelo problema que uma organização busca solucionar através de seu produto ou serviço (e.g., "Somos reconhecidos como a principal organização governamental no desenvolvimento e manutenção de tecnologias de informação superiores e sistemas intensivos de software integrado"). Exemplos de objetivos que poderiam ser levantados a partir da visão e missão citadas: (i) "Fornecer o melhor ambiente e processos de engenharia de sistema/software", (ii) "Melhorar a postura financeira da organização", (iii) "Entregar sistemas de alta qualidade e no prazo" e (iv) "Liderar o governo no desenvolvimento de tecnologia" (GOETHERT; FISHER, 2003). Tanto Missão quanto Visão precisam ser traçadas junstamente aos *stakeholders* da organização, caso ainda não tenham sido feitas antes da inicialização do processo.

Outra abordagem possível é se guiar pelos problemas que se apresentam no processo de desenvolvimento de uma organização. Uma reunião deve ser feita juntamente com os *stakeholders* afim de levantar os problemas apresentados no processo de produção de software. Uma vez que uma lista de problemas é levantada, pode-se traçar objetivos para resolver tais problemas. Exemplos de problemas: "muitas tarefas necessitando de retrabalho", "reclamações por parte dos desenvolvedores de que os requisitos não estão claros o bastante" ou "dificuldade de comunicação entre os *stakeholders* e os desenvolvedores". Esses problemas evidenciam possíveis objetivos tais como: "melhorar a comunicação entre *stakeholders* e desenvolvedores", "diminuir o retrabalho nas tarefas de desenvolvimento" e "melhorar a especificação das tarefas".

A definição de objetivos pode ser realizada seguindo-se uma abordagem de detalhamento, na qual objetivos mais gerais são definidos primeiro e, então, são detalhados em objetivos mais específicos. A abordagem inversa também pode ser aplicada: definir objetivos específicos e agrupá-los em objetivos mais gerais.

Ao final desta atividade deve-se obter uma lista de objetivos a serem alçados pela organização.

3.1.2 Identificar necessidades de informação

A necessidade de informação é um conceito central no paradigma GQM, que visa à melhoria contínua dos processos informacionais e do conhecimento organizacional. Uma necessidade de informação é definida como a lacuna entre o estado atual e o estado desejado de conhecimento de um indivíduo ou grupo, que motiva a busca e o uso de informações relevantes. Uma vez que os objetivos tenham sido estabelecidos, deve-se formular perguntas para serem respondidas a fim de preencher as lacunas de conhecimento identificadas. A formulação destas perguntas deve ser feita, ou no mínimo, validadas pelo líder técnico da organização ou com os *stakeholders* através de entrevistas ou reuniões que visem identificar quais entidades e relações entre entidades se afetam os objetivos a serem alcançados.

Goethert e Fisher (2003) exemplifica uma possível necessidade de informação relacionada ao objetivo de negócio "Fornecer entrega pontual de software de alta qualidade" com a seguinte pergunta: "Qual é a conformidade dos sistemas com os requisitos do cliente?". Nem sempre será possível traçar perguntas precisas o bastante para realizar os passos seguintes de levantamento de medidas/indicadores e produção de representações gráficas em um primeiro momento. É importante ter em mente a possibilidade de, a partir de perguntas mais gerais, elaborar perguntas mais facilmente mensuráveis.

Ao final desta atividade se espera a produção de um conjunto de questões que, uma vez respondidas, satisfaçam as necessidades de informação levantadas para os objetivos definidos em Identificar Objetivos. Para se manter a rastreabilidade é importante relacionar as necessidades de informação aos seus respectivos objetivos. Um exemplo é apresentado na Tabela 1.

Tabela 1 – Exemplos de Objetivos e Necessidades de Informação.

Objetivos	**Necessidades de informação**
Aumentar qualidade dos produtos entregues	Qual é a conformidade dos sistemas com os requisitos do cliente? Qual é o percentual de sistemas em total conformidade com os requisitos do cliente?
Melhorar aderência aos prazos de entrega acordados	Quais a taxa de aderência ao prazo dos projetos concluídos?

Ao final desta atividade deve-se obter uma lista de necessidades de informação, ou perguntas, a serem respondidas.

3.1.3 Definir Medidas e Indicadores

Uma vez conhecidas as necessidades de informação, deve-se definir as medidas/indicadores que serão usadas para atendê-las e fornecerão informação quantitativa para apoiar a tomada de decisões (FENTON; NEIL, 2000). Dentre as medidas definidas, é preciso identificar quais servirão como indicadores, ou seja, medidas capazes de fornecer informações que diretamente indicam o alcance (ou não) dos objetivos estabelecidos (BARCELLOS; FALBO; ROCHA, 2009). Após identificar as medidas, é preciso estabelecer sua definição operacional. Este aspecto não foi abordado neste trabalho. Orientações sobre como estabelecer a definição operacional de uma medida podem ser encontradas em Rocha, Santos e Barcellos (2012).

As medidas/indicadores definidos devem estar relacionadas às respectivas necessidades de informação e objetivos, como ilustrado na Tabela 2 .

Ao final desta atividade deve-se obter uma tabela que relacione as necessidades de informação definidas no passo anterior e medidas ou indicadores que as respondam.

Tabela 2 – Exemplos de Medidas.

Objetivo	Aumentar qualidade dos produtos entregues.
Necessidade de Informação	Qual é a conformidade dos sistemas com os requisitos do cliente?
Medidas	Número de requisitos homologados junto ao cliente; Número de requisitos atendidos pelo sistema; Taxa de conformidade de requisitos (dada pela razão entre o Número de requisitos atendidos pelo sistema e o Número de requisitos homologados junto ao cliente).

3.1.4 Construir Representações Gráficas

Goethert e Fisher (2003) argumentam que identificar objetivos, questões, medidas e indicadores sem visualizar uma representação gráfica geralmente não é suficiente para iniciar um programa de medição bem-sucedido. A representação gráfica visa apresentar dados coletados para um indicador (ou combinação de indicadores) de forma a facilitar sua análise. Uma representação gráfica geralmente é um gráfico ou tabela associado às necessidades da organização. As representações gráficas frequentemente fazem uma comparação entre valores, como, por exemplo, valores planejados e reais. Esses representações gráficas servem como especificação de requisitos para os dados que devem ser coletados e sua análise.

As seguintes orientações devem ser lembradas para gerar representações gráficas adequadas (Practical Software and Systems Measurement, 2000):

- Usar convenções consistentes;
- Manter a simplicidade, visando a uma mensagem clara;
- Títulos únicos devem refletir *insight* e escopo;
- Rotular cada eixo e fornecer marcadores de escala;
- Anotar com marcos e eventos significativos;
- Usar os mesmos eixos e escalas se as representações gráficas forem comparadas.

A Figura 13 apresenta um exemplo de representação gráfica usada para responder a seguinte necessidade de informação: “Qual é a conformidade dos sistemas com os requisitos do cliente?”:

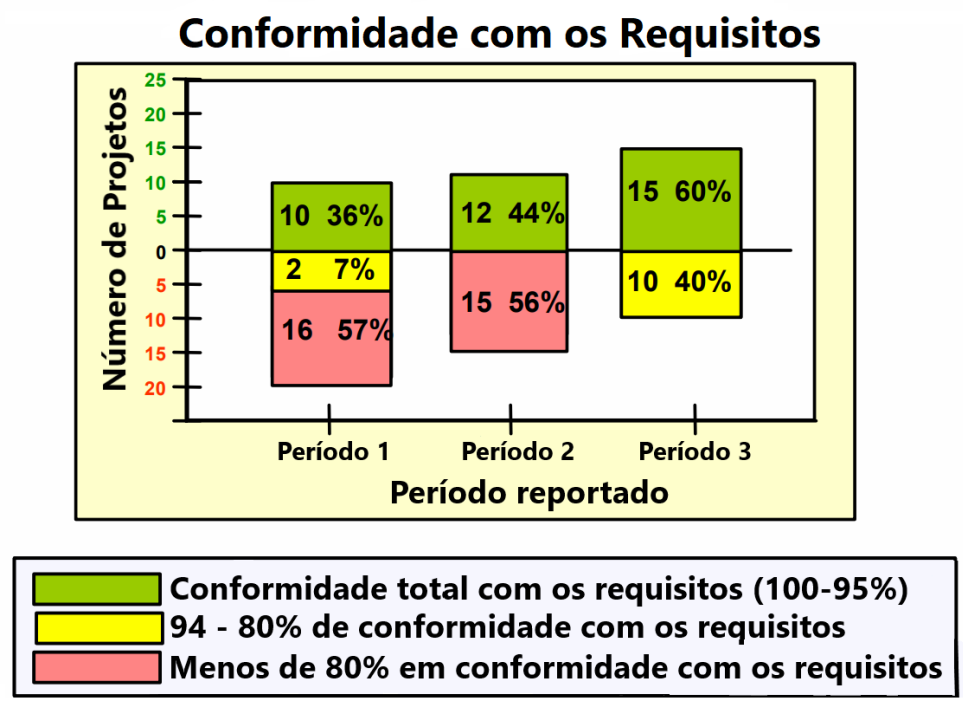

Figura 13 – Representação gráfica para indicador de conformidade com os requisitos (GOETHERT; FISHER, 2003).

A representação gráfica indicada na Figura 13 representa o nível de conformidade dos sistemas com os requisitos especificados pelo cliente para projetos concluídos por período. Os projetos concluídos em um período de tempo específico são classificados usando os valores de avaliação especificados pela organização, conforme mostrado na legenda. A conformidade total com os requisitos do cliente é definida como 100-95% (i.e., de 95 a 100% dos requisitos homologados foram atendidos pelo sistema). Para cada período, o gráfico indicam o número de projetos concluídos em cada categoria. Por exemplo, no período 1, 10 projetos estavam em total conformidade com os requisitos. Isso representa 36% dos projetos concluídos durante o período considerado.

A representação gráfica apresentado acima omitiu quais seriam os períodos em questão, mas a presença desses marcadores temporais apontam um exemplo no qual um cronograma de coleta dos dados seria necessário, orientando desta forma a coleta de dados.

Cabe ressaltar que as representações gráficas apresentadas anteriormente é apenas um exemplo. Outros tipos de representação, mais elaboradas ou mais simples (e.g., gráficos de barras, gráficos pizza) podem ser criadas. Também é importante considerar que quando a organização está iniciando a prática de medição, é possível que ela não tenha dados suficientes para estabelecer os valores das faixas e precise de um tempo coletando e analisando dados para chegar aos valores adequados.

Ao fim desta atividade, espera-se que cada necessidade de informação tenha uma representação gráfica associada, considerando-se as medidas a ela relacionadas.

3.1.5 Identificar Dados a Coletar

Nesta atividade, considerando-se os dados disponíveis nas aplicações usadas pela organização, são identificados aqueles que são necessários para a coleta de valores para as medidas definidas e posterior elaboração das representações gráficas. Esta atividade deve ser realizada iterativamente com a anterior, a fim que de que as medidas definidas possam ser obtidas a partir dos dados disponíveis. Caso não haja dados disponíveis para alguma das medidas definidas, a medida pode ser substituída por outra que possa ser obtida a partir dos dados (quando pertinente) ou a necessidade de se coletar os dados associados à medida deve ser registrada para que sejam realizadas ações para tornar tais dados disponíveis (quando possível).

As atividades realizadas até aqui visam garantir que os dados coletados pela organização sejam úteis, i.e., estejam associados a objetivos e necessidades de informação claramente definidos.

A Figura 14 adaptada de (PARK; GOETHERT; FLORAC, 1996), ilustra a rastreabilidade entre os dados e os objetivos e necessidades de informação.

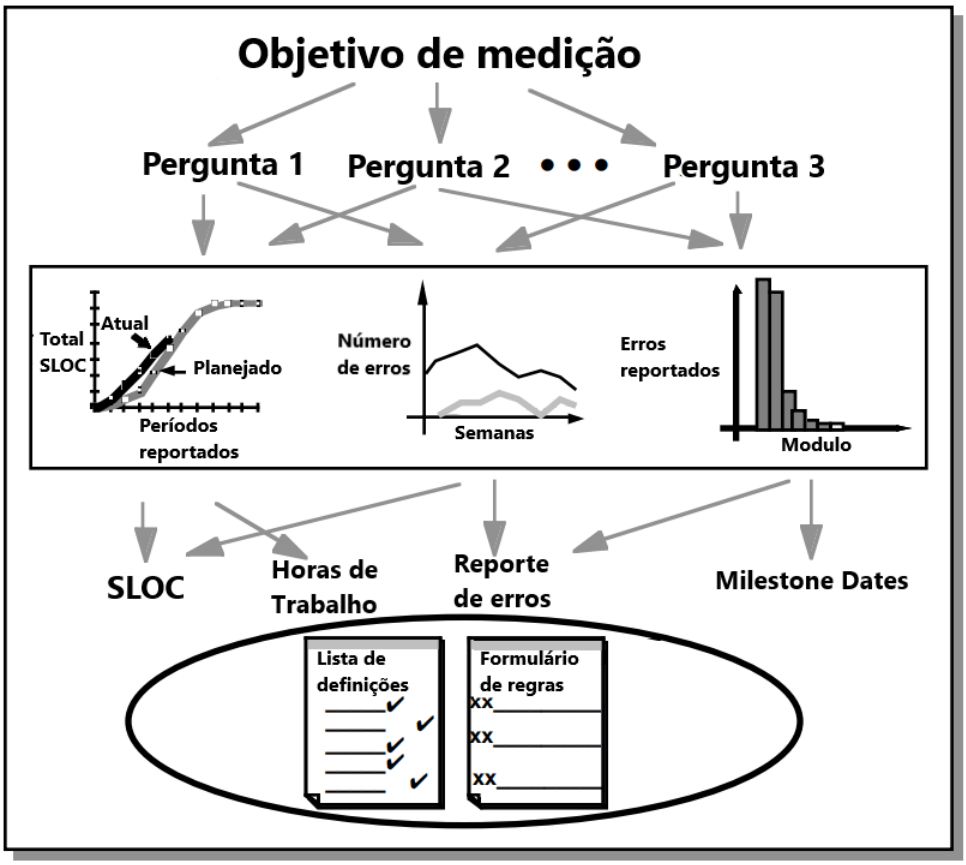

Figura 14 – Demonstração de rastreabilidade do processo GQM (PARK; GOETHERT; FLORAC, 1996).

Ao final desta atividade deve-se obter uma lista de dados a serem coletados para atender às medidas e indicadores definidos no passo anterior.

3.1.6 Coletar Valores para as Medidas

Uma vez que os dados necessários para as medidas sejam identificados, é necessário extraí-los de seus respectivos repositórios de dados. O resultado desta atividade são os dados coletados para as medidas e estruturados de forma que possam ser usados como fonte para as representações gráficas previamente estabelecidas.

Vale ressaltar que para que os dados sejam utilizados adequadamente, eles devem estar disponíveis e serem corretos. Caso seja necessário, devem ser realizados tratamento dos dados (e.g., verificação de consistência, correção, integração, etc.). Além disso, conforme destacado por Santos Júnior (2023), uma organização pode coletar os dados gerados por diferentes aplicações e armazená-los em diferentes repositórios de dados, sendo de suma importância se atentar às aplicações usadas, quantos repositórios de dados existem e se certificar de que, caso existam diversos repositórios de dados, questões de interoperabilidade semântica sejam devidamente tratadas.

Outro ponto a ser considerado nesta atividade, conforme destacado por Santos Júnior (2023), é a existência da quantidade de dados necessária. A ausência de dados pode levar a valores medidos e representações incompletos. Por exemplo, suponha que a organização deseje melhorar o tempo de entrega de funcionalidades do produto ao cliente (objetivo) e que, para isso, ela precisa saber o tempo médio para conclusão de uma história de usuário, a partir do momento em que ela é adicionada ao backlog do produto (necessidade de informação). Para isso, a organização poderia usar a medida/indicador *Lead Time* (JONES et al., 2020), que calcula a média da diferença em unidade de tempo (e.g., hora, dia, semana) entre a criação das histórias de usuário e sua conclusão. No entanto, caso a organização não tenha registrado o momento de criação das histórias de usuário ou quando foram concluídas, ou caso tenha registrado para algumas histórias de usuário apenas, o valor obtido para a medida não será realista e a representação não será adequada. Dessa forma, será necessário coletar os dados adequadamente para gerar valores medidos e representações corretas. Caso isso não seja possível, pode ser necessário definir novas medidas/indicadores, ou até mesmo, rever as necessidades de informações para que sejam atendíveis pelos dados existentes.

Ao fim desta atividade espera-se que se tenha os dados coletados para as medidas e acessíveis para as respectivas representações gráficas.

3.1.7 Gerar Dashboards

Nesta atividade, as representações gráficas geradas são consolidadas em um ou vários *dashboards*, possibilitando assim uma visão centralizada dos dados coletados. Essa centralização facilita o monitoramento e análise das medidas e indicadores, ajudando na tomada de decisão pelos *stakeholders* da organização. A produção das representações

gráficas consiste na preparação de consultas SQL para acesso aos dados seguido de manipulação dos dados acessados para a produção de gráficos. Esta atividade exige que se tenha conhecimento de consultas SQL, caso o Engenheiro de Software que esteja executando esse processo não possua tal conhecimento é necessário que um terceiro membro da organização (ou profissional da área de banco de dados não associado a organização) com conhecimento de SQL seja acionado para a preparação dessas consultas.

3.1.8 Interpretar Resultados

Esta é a atividade final do processo, na qual é feita a interpretação dos resultados e são feitas recomendações a partir dos *dashboards* gerados.

Ao fim do processo, espera-se que tenham sido definidos: um conjunto de objetivos, necessidades de informação, medidas e indicadores com respectivas representações gráficas, e dados coletados e representados em um ou vários *dashboards* para apoiar a análise dos dados e tomada de decisão. Conforme novos objetivos surjam ao longo do tempo, novas iterações deste processo devem ser executadas, gerando novas necessidades de informação e com isso novas medidas, indicadores e representações gráficas, evoluindo os *dashboards* gerados gerando novos.

3.2 Demonstração de uso de *Ramble ON*

Para demonstrar o uso de *Ramble ON*, o processo e a solução de apoio propostos foram utilizados considerando-se dados disponibilizados por uma *startup*. A *startup* considerada é uma *fintech* que utiliza práticas de Engenharia de Software Contínua (e.g., *time-box*, integração contínua, entrega contínua) e possui uma única equipe de desenvolvimento cujo tamanho pode variar. Na demonstração realizada neste trabalho, havia onze desenvolvedores listados nas bases de dados da organização, um Líder Técnico e um Líder de Produto. Para apoiar o processo de desenvolvimento de software, a organização utiliza o MS DevOps[1] e o GitLab[2] para auxiliar, respectivamente, na gestão de tarefas e no controle de versão de código e na implementação do processo de integração contínua e implantação contínua. As atividades de *Ramble ON* foram realizadas em colaboração com o Líder Técnico da organização.

Como informado na seção anterior, para apoiar a atividades Identificar dados a coletar, Coletar dados e Gerar dashboards de *Ramble ON* foi utilizado *The Band* (SANTOS JúNIOR, 2023). *The Band* tem como objetivo automatizar a coleta de dados nas aplicações e armazená-los em OBDRs, permitindo, assim, a criação de representações gráficas e *dashboards*, baseados em dados da organização.

[1] <https://azure.microsoft.com/en-us/products/devops>
[2] <https://gitlab.com/>

Na atividade Identificar objetivos, foram definidos com a organização os seguintes objetivos: **Melhorar o tempo de entrega de novas funcionalidades de seus produtos** e **Melhorar o planejamento das *sprints* dos projetos**.

Tendo como base os objetivos, a atividade Identificar necessidades de informação foi realizada por meio de uma entrevista com o Líder Técnico. A Tabela 3 apresenta as necessidades de informação identificadas para cada objetivo.

Tabela 3 – Objetivos e Necessidades de Informação.

#	**Objetivos**	**Necessidade de Informação**	**Descrição**
1	Melhorar o tempo de entrega de novas funcionalidades dos produtos	(1.a) Qual o tempo médio de espera de uma história de usuário até que seu desenvolvimento seja iniciado? (1.b) Qual o tempo médio de desenvolvimento de uma história de usuário (desde o início de seu desenvolvimento até ser concluída com sucesso)? (1.c) Qual o tempo médio de entrega de uma história de usuário (desde sua criação)? (1.d) Qual a quantidade média de retrabalho durante o desenvolvimento de uma versão do produto?	(1.a), (1.b) e (1.c) auxiliam no entendimento do tempo que tem sido despendido para entregar novas funcionalidades. Uma melhora no valor de (1.c) é um indicativo direto do alcance ao objetivo de melhora no tempo de entrega. (1.d) auxilia a identificar se tem havido quantidade significativa de retrabalho que esteja impactando no tempo de entrega de novas funcionalidades.
2	Melhorar o planejamento das *sprints* dos projetos	(2.a) Qual é o tempo médio de conclusão com sucesso de um item do backlog em umo *sprint*? (2.b) Qual a taxa média de itens planejados para o *sprint* que não são concluídos em umo *sprint*? (2.c) Qual a taxa média de itens planejados para o *sprint* que são iniciados mas não concluídos em umo *sprint*?	(2.a) auxilia no entendimento de quanto tempo leva para realizar um item do backlog. O gerente do projeto pode usar essa informação para estimar quantos itens podem ser inclusos no backlog de umo *sprint*. (2.b) e (2.c) ajudam a identificar quanto trabalho tem sido planejado para o *sprint* e não concluído. Uma melhora nos valores de (2.b) e (2.c) - particularmente (2.b) - é um indicativo direto do alcance ao objetivo de melhora no planejamento do *sprint*.

Uma vez identificadas as Necessidades de Informação, a próxima atividade é Definir

Medidas e Indicadores necessários para atender as necessidades de informação. As Tabelas 4 e 5 apresentam fragmentos dos conjuntos de Objetivos, Necessidades de Informação e Medidas/Indicadores definidos junto com o Líder Técnico. As medidas que desempenham papel de indicador (i.e., aquelas diretamente usadas para verificar o alcance ao objetivo) são identificadas com *.

Tabela 4 – Necessidades de Informação, Medidas e Indicadores relacionados ao objetivo "Melhorar o tempo de entrega de novas funcionalidades de seus produtos".

#	**Necessidade de Informação**	**Medida/Indicador**	**Descrição**
1.a	Qual o tempo médio de espera de uma história de usuário até que seu desenvolvimento seja iniciado?	*Wait Time*	Informa quanto tempo uma história de usuário fica no *backlog* até ser selecionada para ser desenvolvida (momento do início de desenvolvimento da história de usuário - momento da inclusão da história de usuário no *backlog* do produto).
1.b	Qual o tempo médio de desenvolvimento de uma história de usuário (desde o início de seu desenvolvimento até ser concluída com sucesso)?	*Cycle Time*	Indica quanto tempo uma história de usuário demora para ser concluída, a partir do momento em que tem início seu desenvolvimento (momento da conclusão da história de usuário - momento do início de desenvolvimento da história de usuário).
1.c	Qual o tempo médio de entrega de uma história de usuário (desde sua criação)?	*Lead Time**	Indica quanto tempo, a partir de sua criação, uma história de usuário leva para ser entregue (momento da entrega da história de usuário - momento de adição da história de usuário ao *backlog* do produto).
1.d	Qual a quantidade média de retrabalho durante o desenvolvimento de uma versão do produto?	*Time to Repair*	Identifica quanto tempo um defeito leva para ser resolvido após ser identificado no ambiente de desenvolvimento e/ou homologação (momento de conclusão do defeito - momento de registro do defeito).

Tabela 5 – Necessidades de Informação, Medidas e Indicadores relacionados ao objetivo "Melhorar o planejamento das sprints dos projetos".

#	**Necessidade de Informação**	**Medida/Indicador**	**Descrição**
2.a	Qual é o tempo médio de conclusão com sucesso de um item do *backlog* em umo *sprint*?	Tempo de conclusão de item do *backlog*.	Identifica o tempo despendido para concluir um item do *backlog* do *sprint* desde sua adição ao *backlog* do *sprint* até sua conclusão com sucesso (momento de conclusão do item - momento de adição do item ao backlog do *sprint*).
2.b	Qual a taxa média de itens planejados para o *sprint* que não são concluídos em umo *sprint*?	Taxa de não conclusão*	Indica a quantidade de itens inclusos no backlog do *sprint* que não foram concluídos na respectivo *sprint* (quantidade de itens não concluídos no *sprint*/quantidade de itens planejados no backlog do *sprint*).
2.c	Qual a taxa média de itens planejados para o *sprint* que são iniciados mas não concluídos em umo *sprint*?	Taxa final de *Work in Progress*	Indica a quantidade de itens do backlog do *sprint* que estavam em desenvolvimento (i.e., foram iniciados mas não concluídos) no momento do término do *sprint* (quantidade de itens iniciados e não concluídos no término do *sprint*/quantidade de itens planejados no *backlog* do *sprint*).

Após a definição das medidas/indicadores, se segue para a atividade Construir representações gráficas, cujo resultado é apresentado na Tabela 6.

Tabela 6 – Necessidades de Informação, Indicadores e Representação Gráfica.

#	**Necessidade de Informação**	**Indicador**	**Representação Gráfica**
1.a	Qual o tempo médio de espera de uma história de usuário até que seu desenvolvimento seja iniciado?	*Wait Time*	Gráfico de linha indicando a média dos valores coletados (em ordem cronológica), por período de medição.
1.b	Qual o tempo médio de desenvolvimento de uma história de usuário (desde o início de seu desenvolvimento até ser concluída com sucesso)?	*Cycle Time*	Gráfico de linha indicando a média dos valores coletados por *sprint*.
1.c	Qual o tempo médio de entrega de uma história de usuário (desde sua criação)?	*Lead Time**	Gráfico de linha indicando dos valores médios coletados no projeto por período de medição, em ordem cronológica. No gráfico deve ser representada uma reta (limite) com o valor alvo almejado pela organização(meta), para permitir comparação dos valores realizados com o valor desejado.

Tabela 6 – *Continuação da página anterior*

#	Necessidade de Informação	Medida /Indicador	Representação Gráfica
1.d	Qual a quantidade média de retrabalho durante o desenvolvimento de uma versão do produto?	*Time to Repair*	Gráfico de linha indicando a média dos valores coletados por *sprint* do projeto.
2.a	Qual é o tempo médio de conclusão com sucesso de um item do backlog em um *sprint*?	Tempo de conclusão de item do backlog.	Gráfico de linha indicando a média dos valores coletados por *sprint* do projeto.
2.b	Qual a taxa média de itens planejados para o *sprint* que não são concluídos em umo *sprint*?	Taxa de não conclusão *	Gráfico de linha indicando os valores médios coletados no projeto por *sprint*, em ordem cronológica. No gráfico deve ser representada uma reta (limite) com o valor alvo almejado pela organização (meta), para permitir comparação dos valores realizados com o valor desejado.
2.c	Qual a taxa média de itens planejados para o *sprint* que são iniciados mas não concluídos em umo *sprint*?	Taxa final de *Work in Progress*	Gráfico de linha indicando dos valores médios coletados no projeto por *sprint*, em ordem cronológica.

A próxima atividade consiste em Identificar dados a coletar. Para obter dados para as medidas definidas são necessários dados sobre planejamento e execução de histórias de usuário e outros itens do *backlog* (e.g., tarefas) e sobre defeitos e execução de tarefas para resolvê-los, entre outros. Esses dados estão armazenados no MS DevOps.

Para auxiliar nessa etapa, foram utilizados OBDRs providos por *The Band*. Como os OBDRs foram desenvolvidos baseados em ontologias de domínio (e.g., SRO), é possível identificar os dados a serem coletados utilizando conceitos do domínio. Dessa forma, com base na ontologia pode-se identificar quais conceitos são relacionados a cada medida e acessar os dados armazenados nas respectivas tabelas do OBDR (uma vez que um OBDR é desenvolvido com base em uma ontologia em rede, os conceitos dessa ontologia são espelhados na estrutura do repositório). Por exemplo, se para calcular o valor da medida "quantidade de tarefas alocadas a um membro da equipe" são necessários os conceitos *Team*, *Person*, *Project* e *Task*, os dados necessários para coletar a medida estarão armazenados nas tabelas do OBDR referentes a esses conceitos.

O principal benefício de utilizar OBDRs é que não é necessário mudar o repositório quando as aplicações são alteradas, visto que os OBDRs são independentes de aplicação. Assim, a organização pode mudar de aplicação (e.g., de DevOps para Jira) e, não precisa mudar a identificação de dados, pois *The Band* é capaz de capturar os dados da nova aplicação e armazená-los no OBDR correspondente, sem perder os dados anteriormente capturados e sem precisar alterar o OBDR.

A Tabela 7 apresenta conceitos de SRO necessários para a obtenção de valores para

as medidas definidas.

Tabela 7 – Medidas e Conceitos relacionados.

#	**Medida**	**Conceitos**
1.a	*Wait Time*	Diferença de tempo entre a adição de uma User Story ao Product Backlog de um Scrum Project e sua adição a um Sprint Backlog do projeto.
1.b	*Cycle Time*	Diferença de tempo entre o início da primeira Performed Scrum Development Task relacionada a uma User Story e o fim da última Successfuly Performed Scrum Development Task relacionada àquela *User Story* em um Scrum Project.
1.c	*Lead Time*	Diferença de tempo entre a adição de uma User Story ao Product Backlog de um Scrum Project e o fim da última Successfuly Performed Scrum Development Task relacionada àquela *User Story* no projeto.
1.d	*Time to Repair*	Diferença de tempo entre o início de uma Performed Scrum Development Task realizada para tratar um Defect em um Scrum Project e o fim da última Successfuly Performed Scrum Development Task relacionada àquele *Defect*[3] no projeto.
2.a	Tempo de conclusão de item do *backlog*	Diferença de tempo entre a adição de uma User Story ao Sprint Backlog de um Scrum Project e o fim da última Successfuly Performed Scrum Development Task relacionada àquela *User Story* no projeto.
2.b	Taxa de não conclusão	Razão entre a quantidade de Performed Scrum Development Task com data de início e sem data de fim em uma Sprint + quantidade de Intended Scrum Developement Tasks que não causaram Succesfully Performed Scrum Development Task e a quantidade de Intended Scrum Developement Tasks definidas para aquela Sprint em um Scrum Project.
2.c	Taxa final de *Work in Progress*	Razão entre a quantidade de Performed Scrum Development Task com data de início e sem data de fim em uma Sprint e a quantidade de Intended Scrum Developement Tasks definidas para aquela Sprint em um Scrum Project.

A atividade seguinte consiste em Coletar dados. Idealmente os dados capturados das aplicações seriam salvos no OBDR correspondente ao domínio da aplicação. Por exemplo, dados do MS Azure DevOps seriam armazenados em um OBDR criado baseado em SRO. Esses dados seriam disponibilizados em bancos de dados Postgres e poderiam ser acessados pelo Dremio através de consultas SQL envolvendo um ou mais OBDRs,

[3] O conceito *Defect* pertence à OSDEF (Ontology of Software Defect Errors and Failures) (DUARTE, 2022) e não foi explorado neste trabalho

i.e., é possível fazer consultas integrando os dados. Dremio é uma aplicação que permite criar uma interface de acesso a múltiplos OBDRs em uma única consulta. É importante comentar que criar esta interface de acesso aos OBDRs traz uma vantagem para quem está realizando consultas que precisam acessar mais de um base de dados: a estrutura de bases de dados segue o mesmo padrão da rede de ontologias. Como os OBDRs foram criados baseados em ontologias em redes, a estrutura desses repositórios segue a estrutura da ontologias. Por exemplo, em EO há os conceitos *Person* e *Team*. Uma vez que SRO foi criada reutilizando esses conceitos, haverão entidades com o mesmo nomes nos OBDRs criados a partir de EO e SRO.

Apesar das OBDRs terem sido criadas, dados necessários para as medidas e indicadores identificados não puderam ser coletados, e por isso, os passos subsequentes se viram comprometidos. Os conceitos utilizados para atender as medias 1.b, 1.c, 1.d, 2.a, 2.b e 2.c dependiam do acesso a data de inicio ou data de fim de uma tarefa (*task*) e esses dados não estavam presentes na base de dados, impossibilitando coleta e a construção das representações gráficas. *Ramble ON* permite o retorno à passos anteriores, possibilitando que novas medidas e indicadores possam ser atendidos, porém o tempo limitado em que este trabalho foi desenvolvido impossibilitou que atividades anteriores fossem refeitas.

Contudo, foi possível realizar a coleta referente a medida *Wait Time*. Esta medida está associada ao objetivo "Melhorar o tempo de entrega de novas funcionalidades dos produtos" e à necessidade de informação "Qual o tempo médio de espera de uma história de usuário até que seu desenvolvimento seja iniciado?". A Listagem 3.1 apresenta a consulta que seria realizada no OBDR de SRO para capturar os dados. A Figura 15 apresenta uma representação gráfica de *Wait Time*.

Listagem 3.1 – Consulta SQL Total de tarefas.

```
select
    sub_query.project_name as project,
    sub_query.sprint_name as sprint,
    sub_query.sprint_start_date as sprint_start_date,
    floor(avg(sub_query.wait_time)) avg_wait_time_days
 from
    (
    select
        distinct user_story.id as user_story_id,
        sprint.name as sprint_name,
        sprint.start_date as sprint_start_date,
        scrumproject.name as project_name,
        user_story.name as user_story_name,
        user_story.dtype as tipo,
        user_story.created_date as criacao,
        sprint_backlog.dtype as tipo3,
        sprint_backlog.id as sb_id,
        aa_sprint_backlog.event_date as add_to_sprint_backlog,
        date_part('days', aa_sprint_backlog.event_date - user_story.created_date)
            as wait_time
```

```
    from "sro.public".specificprojectprocess_activity sa
        -- sprint
        inner join "sro.public".specificprojectprocess sprint on sa.
            specificproject_id = sprint.id
        inner join "sro.public".activity a on a.id = sa.activity_id
        inner join "sro.public".activity_artifact aa on aa.activity_id = a.id

        -- user_story
        inner join "sro.public".artifact user_story on user_story.id = aa.
            artifact_id

        -- scrumproject
        inner join "sro.public".generalprojectprocess scrum_process on sprint.
            generalprojectprocess_id = scrum_process.id
        inner join "sro.public".project scrumproject on scrum_process.
            softwareproject_id = scrumproject.id


        -- relacao user_story e sprint_backlog
        INNER JOIN "sro.public".artifact_artifact AS aa_sprint_backlog
            ON aa_sprint_backlog.artifact_to_id = user_story.id
        INNER JOIN "sro.public".artifact sprint_backlog
            on sprint_backlog.id = aa_sprint_backlog.artifact_from_id
    where
        user_story.dtype = 'UserStory'
        and sprint_backlog.dtype = 'SprintBacklog'
        and sprint.dtype = 'Sprint'
    ) as sub_query

group by project_name, sprint, sprint_start_date
```

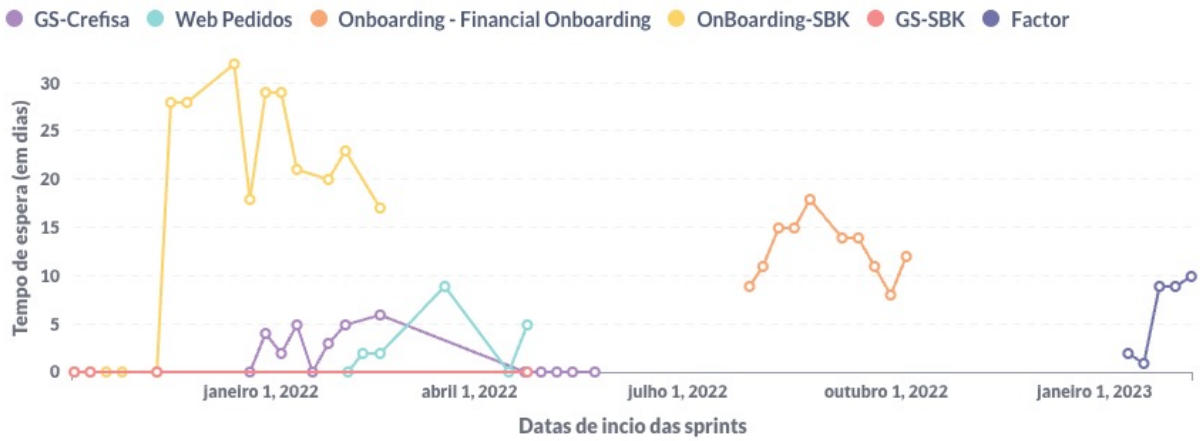

Figura 15 – Gráfico relacionado à medida *Wait Time*.

Por fim, as atividades finais, Gerar *dashboards* e Interpretar resultados não puderam ser realizadas por conta dos problemas anteriormente reltados na atividade de Coleta de dados. Contudo, o levantamento dos objetivos da organização, juntamente com a identificação das necessidades de informação, a definição de medidas, construção de representações gráficas e identificação dos dados a coletar já permite que o processo seja iniciado em

uma etapa mais avançada em uma tentativa futura. Desta forma o processo não pôde ser realizado em sua completude, sendo necessário uma correção na coleta de dados antes que o processo possa ser de fato concluído e análises relevantes possam ser geradas.

3.3 Considerações Finais do Capítulo

Este capítulo apresentou a abordagem chamada *Ramble ON* que auxilia no planejamento e execução da medição de software alinhada à disponibilidade de dados, incluindo desde a definição dos objetivos que se deseja alcançar até apresentação de representações gráficas utilizando dados organizacionais para apoiar a análise de dados e tomada de decisão. Tal abordagem utiliza como base GQ(I)M (PARK; GOETHERT; FLORAC, 1996) e a solução de Integração *The Band* (SANTOS JúNIOR, 2023).

Uma demonstração de uso foi realizada, porém não pode ser finalizada por falhas no processo de coleta de dados por parte da *statup* participante. Como resultado, foi possível realizar um levantamento dos objetivos da organização, juntamente com a identificação das necessidades de informação, a definição de medidas, construção de representações gráficas e identificação dos dados a coletar. Desta forma o processo *Ramble ON* serviu como uma demonstração do planejamento para medição sem contudo realizar a coleta de dados.

4 Conclusão e Trabalhos Futuros

Neste capítulo são realizadas as considerações finais deste trabalho, sendo apresentadas suas principais contribuições e perspectivas de trabalhos futuros.

4.1 Conclusão

A Engenharia de Software Contínua envolve práticas e ferramentas que visam estabelecer um fluxo *end-to-end* entre a demanda do cliente e a entrega rápida de um produto ou serviço (BARCELLOS, 2020). Nesse contexto, medição de software se faz muito importante pois visa quantificar diferentes aspectos dos processos e produtos da Engenharia de Software para ajudar a entender melhor esses processos e produtos e, com isso, apoiar as atividades cotidianas do desenvolvimento de software, bem como a tomada de decisão (BARCELLOS, 2010).

Neste trabalho, foi desenvolvido *Ramble ON*, um processo de medição baseado no GQ(I)M que visa à definição de medidas alinhadas às necessidades de informação e à utilização de dados disponíveis nas aplicações usadas pela organização para prover informações que apoiem o desenvolvimento de software orientado a dados. Na seção 3.2, foi apresentada uma demonstração de uso de *Ramble ON* com o apoio de componentes de *The Band* (SANTOS JúNIOR, 2023) (particularmente OBDRs) e das ferramentas Dremio e Metabase .

Ao longo do desenvolvimento deste trabalho, foram utilizados conhecimentos diversos obtidos durante a graduação em diferentes disciplinas e em na Iniciação Científica realizada pelo autor do trabalho. Os conhecimentos mais utilizados foram obtidos através das seguintes temas de estudo:

- **Engenharia de Software**: é o motivador central deste trabalho: entender e melhorar o processo de desenvolvimento de software. Em especial, forneceu a base para o estudo de Engenharia de Software Contínua, um dos principais objetos de estudo deste trabalho.
- **Banco de Dados**: proveu conhecimento necessário para a compreensão do funcionamento de bancos de dados relacionais, tanto para o registro de dados quanto para a consulta dos mesmos através da linguagem SQL. Graças a esses conhecimentos foi possível desenvolver a arquitetura da solução apresentada neste trabalho, com o armazenamento de dados no banco relacional Postgresql, e acesso ao mesmo através da ferramenta Dremio.

- **Engenharia de Ontologias** : Forneceu conhecimentos necessários para o entendimento da solução de integração *The Band*, central para o estudo de caso realizado neste trabalho.

No Capítulo 1 desta monografia foram apresentados os objetivos a serem alcançados no desenvolvimento deste trabalho. Na Tabela 8 são apresentados novamente cada um dos objetivos, assim como a indicação de atendimento e os resultados alcançados que evidenciam o atendimento dos mesmos.

Tabela 8 – Objetivos e seus resultados.

Objetivos do trabalho	**Status**	**Resultado**
Definir um processo de medição.	Atendido	Processos Ramble ON (vide seção 3.1).
Projetar uma arquitetura de acesso a dados para apoiar o processo a dados.	Atendido	Elaboração de uma arquitetura de acesso a dados (vide Seção 3.2).
Demonstração do processo proposto utilizando dados disponibilizados por uma organização.	Atendido	Demonstração de uso de Ramble ON (vide seção 3.3).

Entre as dificuldades para o desenvolvimento deste trabalho, destaca-se o aprendizado acerca dos processos de medição utilizados, a saber GQM e GQ(I)M e o estudo da rede de ontologia *Continuum*, com foco na ontologia SRO, com a qual o autor nunca havia tido contato antes de iniciar os estudos para o desenvolvimento deste projeto. A escolha dos métodos GQM e GQ(I)M foi devido ao fato de serem métodos de medição estabelecidos na literatura e de uso intuitivo (particularmente o GQM). Já a escolha das ontologias citadas foi devido ao desenvolvimento deste trabalho ser associado a uma pesquisa de doutorado realizada no NEMO na qual tais ontologias são parte crucial, tendo algumas delas sido desenvolvidas pelo doutorando, que é coorientador deste trabalho.

Outro obstáculo relevante foi a integração entre Dremio e Metabase. A escolha do Metabase se deu pela experiência previa do autor com esta ferramenta, já o Dremio se deu pela experiência do coorientador deste trabalho. Apesar do conhecimento prévio do autor com Metabase, ele nunca havia utilizado Dremio, o que exigiu um estudo sobre a ferramenta. A tarefa de integrar ambas se mostrou intuitiva e simples, sendo necessário apenas a instalação de um *driver* de acesso no Metabase que permitisse conexão com o Dremio. Porém, após a conexão o Metabase não identificava os bancos de dados internos ao Dremio pela interface usual de escrita das consultas, e não havia documentação no site oficial da ferramenta acerca do uso junto ao Dremio (inclusive o próprio *driver* de acesso usado é de autoria da comunidade, não sendo oficialmente adotado pelo Metabase). Após uma serie de tentativas e erros foi possível realizar acesso aos bancos de dados internos

ao Dremio especificando qual banco na consulat SQL via a seguinte sintaxe: "<nome do banco>.public".<nome da tabela>. Ao fim do processo, foi possível realizar a integração de forma apropriada.

Apesar das dificuldades encontradas, a elaboração deste trabalho possibilitou que o autor vivenciasse experiências que permitiram a agregação de conhecimento em sua formação, como: entendimento acerca de processos de medição de software, aprendizado sobre ESC, aprofundamento do saber acerca de ontologias relacionadas ao domínio de engenharia de software, experiência com integração de dados através do uso da *The Band* e experiência com ferramenta de *data lake* (Dremio).

Por fim, vale ressaltar que a demonstração de uso de *Ramble ON* não pode ser concluída por falhas na coleta de dados por parte da *startup*. A não conclusão do processo impossibilita a análise do resultado do mesmo na produção de software da empresa, sendo necessários trabalhos futuros para execução apropriada do processo e análise de seus resultados em uma organização. Contudo, a demonstração do planejamento para medição de software usando *Ramble ON* auxiliou a empresa a definir melhor seus objetivos e quais necessidades de informação precisam ser satisfeitas para alcança-los.

4.2 Trabalhos Futuros

Como trabalhos futuros, os seguintes pontos foram identificados como melhorias no processo:

- Concluir a aplicação do processo e aplicar os resultados obtidos, para entender melhor as vantagens em utilizar o mesmo.
- Avaliar o processo proposto em empresas de diferentes segmentos, a fim de verificar sua eficácia em diferentes contextos.
- Realizar estudos para aprimorar a coleta de dados.
- Comparar a abordagem proposta com outras abordagens existentes, a fim de identificar suas vantagens e desvantagens em relação a outras soluções.
- Integrar o processo proposto à abordagem *Immigrant*, permitindo que o mesmo auxilie no levantamento de objetivos e necessidades de informação. (SANTOS JúNIOR, 2023).

Referências

BARCELLOS, M. Medição de software - um importante pilar da melhoria de processos de software. *Engenharia de Software Magazine*, p. 55–60, 05 2010. Citado 2 vezes nas páginas 14 e 51.

BARCELLOS, M.; FALBO, R.; FRAUCHES, V. Towards a measurement ontology pattern language. *CEUR Workshop Proceedings*, v. 1301, 01 2014. Citado na página 23.

BARCELLOS, M. P. Towards a framework for continuous software engineering. In: *Proceedings of the 34th Brazilian Symposium on Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2020. (SBES '20), p. 626–631. ISBN 9781450387538. Disponível em: <https://doi.org/10.1145/3422392.3422469>. Citado 3 vezes nas páginas 13, 18 e 51.

BARCELLOS, M. P. *Medição de Software, Fronteiras da Engenharia de Software, Episódio 37*. 2023. <https://fronteirases.github.io>. [Accessed: 2023-07-07]. Citado 2 vezes nas páginas 14 e 15.

BARCELLOS, M. P.; FALBO, R. A.; ROCHA, A. R. An Ontology-based Approach for Software Measurement and Suitability Measures Basis Evaluation to Apply Statistical Software Process Controle in High Maturity Organizations. In: *PhD Colloquium - 28th International Conference on Conceptual Modeling (ER 2009)*. [S.l.: s.n.], 2009. v. 597, p. 31. Citado 2 vezes nas páginas 34 e 37.

BASILI, V. et al. $GQM^{+}Strategies - -AligningBusinessStrategieswithSoftwareMeasurement.2007.Citadonapágina20.$

BASILI, V. R.; CALDIERA, G.; ROMBACH, H. D. The goal question metric approach. In: . [S.l.: s.n.], 1994. Citado 4 vezes nas páginas 8, 14, 20 e 35.

BRINGUENTE, A.; FALBO, R.; GUIZZARDI, G. Using a foundational ontology for reengineering a software process ontology. *JIDM*, v. 2, p. 511–526, 01 2011. Citado na página 23.

BUSSE, S. et al. Federated information systems: Concepts, terminology and architectures. 06 1999. Citado na página 29.

COSTA, S. D. et al. A core ontology on the human–computer interaction phenomenon. *Data Knowledge Engineering*, v. 138, p. 101977, 2022. ISSN 0169-023X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0169023X21000951>. Citado na página 23.

DREMIO. *Homepage | Dremio*. 2023. <https://www.dremio.com/>. [Accessed 11-Apr-2023]. Citado na página 31.

DUARTE, B. B. *An Ontology-based Reference Model for the Software Systems Domain with a focus on Requirements Traceability*. Tese (Doutorado), 2022. Citado na página 47.

DUARTE, B. B. et al. Ontological foundations for software requirements with a focus on

requirements at runtime. *APPLIED ONTOLOGY (ONLINE)*, v. 13, p. 73–105, 2018. ISSN 18758533. Citado na página 23.

FALBO, R. et al. Towards an enterprise ontology pattern language. In: . [S.l.: s.n.], 2014. Citado 2 vezes nas páginas 23 e 24.

FENTON, N. E.; NEIL, M. Software metrics: roadmap. In: *Proceedings of the Conference on the Future of Software Engineering*. [S.l.: s.n.], 2000. p. 357–370. Citado na página 37.

FITZGERALD, B.; STOL, K.-J. Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*, Elsevier BV, v. 123, p. 176–189, jan 2017. Disponível em: <https://doi.org/10.1016%2Fj.jss.2015.06.063>. Citado 2 vezes nas páginas 13 e 18.

FONSECA, V. S.; BARCELLOS, M. P.; FALBO, R. d. A. An ontology-based approach for integrating tools supporting the software measurement process. *Science of Computer Programming*, Elsevier BV, v. 135, p. 20–44, feb 2017. Disponível em: <https://doi.org/10.1016%2Fj.scico.2016.10.004>. Citado na página 14.

GOETHERT, W.; FISHER, M. *Deriving Enterprise-Based Measures Using the Balanced Scorecard and Goal-Driven Measurement Techniques*. Pittsburgh, PA, 2003. Disponível em: <http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6507>. Citado 8 vezes nas páginas 8, 21, 22, 35, 36, 37, 38 e 39.

GUIZZARDI, G. *Ontological foundations for structural conceptual models*. 2005. <https://research.utwente.nl/en/publications/ontological-foundations-for-structural-conceptual-models>. [Accessed 30-Jul-2022]. Citado 2 vezes nas páginas 23 e 25.

GUIZZARDI, G. et al. Ufo: Unified foundational ontology. *Applied ontology*, v. 1, n. 17, p. 167–210, 2022. Citado na página 25.

ISO/IEC/IEEE. Iso/iec/ieee international standard - systems and software engineering–measurement process. *ISO/IEC/IEEE 15939:2017(E)*, p. 1–49, 2017. Citado na página 14.

IZZA, S. Integration of industrial information systems: from syntactic to semantic integration approaches. *Enterprise Information Systems*, Informa UK Limited, v. 3, n. 1, p. 1–57, feb 2009. Disponível em: <https://doi.org/10.1080%2F17517570802521163>. Citado na página 22.

JASPER, R.; USCHOLD, M. A framework for understanding and classifying ontology applications. *IJCAI-99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends*, 1998. Disponível em: <http://www.cs.man.ac.uk/~horrocks/Teaching/cs646/Papers/uschold99.pdf>. Citado na página 23.

JOHANSSEN, J. O. et al. Continuous software engineering and its support by usage and decision knowledge: An interview study with practitioners. *Journal of Software: Evolution and Process*, Wiley, v. 31, n. 5, p. e2169, may 2019. Disponível em: <https://doi.org/10.1002%2Fsmr.2169>. Citado na página 18.

JONES, C. L. et al. Practical software and systems measurement continuous iterative development measurement framework. *Version*, v. 1, p. 15, 2020. Citado na página 41.

KURAPATI, N.; MANYAM, V.; PETERSEN, K. Agile software development practice adoption survey. In: . [S.l.: s.n.], 2012. v. 111, p. 16–30. ISBN 978-3-642-30349-4. Citado na página 13.

METABASE. *Metabase documentation*. 2023. <https://www.metabase.com/docs/latest/>. [Accessed 11-Apr-2023]. Citado na página 31.

OLSSON, H. H.; ALAHYARI, H.; BOSCH, J. Climbing the "stairway to heaven-- a mulitiple-case study exploring barriers in the transition from agile development towards continuous deployment of software. In: *2012 38th Euromicro Conference on Software Engineering and Advanced Applications*. [S.l.: s.n.], 2012. p. 392–399. Citado 2 vezes nas páginas 13 e 18.

PARK, R. E.; GOETHERT, W. B.; FLORAC, W. A. Goal-driven software measurement. a guidebook. In: . [S.l.: s.n.], 1996. Citado 6 vezes nas páginas 8, 14, 15, 21, 40 e 50.

Practical Software and Systems Measurement. *Practical Software and Systems Measurement: A Foundation for Objective Project Management*. 2000. Disponível em: <https://www.psmsc.com/PSMGuide.asp>. Citado na página 38.

REDHAT. *O que é um data lake?* 2019. <https://www.redhat.com/pt-br/topics/data-storage/what-is-a-data-lake>. [Accessed 11-Apr-2023]. Citado na página 31.

RISING, L.; JANOFF, N. The scrum software development process for small teams. *IEEE Software*, Institute of Electrical and Electronics Engineers (IEEE), v. 17, n. 4, p. 26–32, 2000. Disponível em: <https://doi.org/10.1109%2F52.854065>. Citado na página 26.

ROCHA, A.; SANTOS, G.; BARCELLOS, M. *Medição de Software e Controle Estatístico de Processos*. [S.l.]: Ministério da Ciência, Tecnologia e Inovação (MCTI), SEPIN/PBQP Software, 2012. vol. 8. p. Citado na página 37.

RUY, F. et al. Seon: A software engineering ontology network. In: . [S.l.: s.n.], 2016. p. 527–542. ISBN 978-3-319-49003-8. Citado 4 vezes nas páginas 8, 23, 24 e 25.

RUY, F. B. *Software Engineering Standards Harmonization: An Ontology-based Approach*. Tese (Doutorado), 2017. Citado 2 vezes nas páginas 23 e 24.

SANTOS JÚNIOR, P. S. et al. From a scrum reference ontology to the integration of applications for data-driven software development. *Information and Software Technology*, v. 136, p. 106570, 2021. ISSN 0950-5849. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950584921000537>. Citado 4 vezes nas páginas 8, 25, 27 e 28.

SANTOS JúNIOR, P. S. *From Continuous Software Engineering Reference Ontologies to the Integration of Data for Data-Driven Software Development*. Tese (Doutorado), 2023. Citado 15 vezes nas páginas 8, 15, 24, 25, 26, 28, 29, 30, 31, 32, 41, 42, 50, 51 e 53.

SANTOS, P. S. d.; BARCELLOS, M. P.; CALHAU, R. F. Am i going to heaven? first step climbing the stairway to heaven model results from a case study in industry. In: *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2020. (SBES '20), p. 309–318. ISBN 9781450387538. Disponível em: <https://doi.org/10.1145/3422392.3422406>. Citado na página 28.

SCHERP, A. et al. Designing core ontologies. *Applied Ontology*, IOS Press, NLD, v. 6, n. 3, p. 177–221, aug 2011. ISSN 1570-5838. Citado na página 23.

SCHWABER, K.; SUTHERLAND, J. *The Definitive Guide to Scrum: The Rules of the Game*. [S.l.]: Scrum.org, 2013. Citado na página 26.

SOLINGEN, E. B. Rini van. The goal/question/metric method: a practical guide for quality improvement of softwaredevelopment. McGraw-Hill Publishing Company, jan 1999. Disponível em: <https://www.researchgate.net/publication/243765439_The_GoalQuestionMetric_Method_A_Practical_Guide_for_Quality_Improvement_of_Software_Development>. Citado na página 20.

STUDER, R.; BENJAMINS, V. R.; FENSEL, D. Knowledge engineering: Principles and methods. *Data &amp Knowledge Engineering*, Elsevier BV, v. 25, n. 1-2, p. 161–197, mar 1998. Disponível em: <https://doi.org/10.1016%2Fs0169-023x%2897%2900056-6>. Citado na página 22.

SUÁREZ-FIGUEROA, M. C. et al. *Ontology Engineering in a Networked World*. Springer Berlin Heidelberg, 2012. 2 p. Disponível em: <https://doi.org/10.1007%2F978-3-642-24794-1>. Citado 2 vezes nas páginas 15 e 23.

The PostgreSQL Global Development Group. *PostgreSQL Database Management System*. 2023. <https://www.postgresql.org/>. [Accessed: 2023-07-08]. Citado na página 31.