Rumo a um catálogo semântico de qualidades de processos de desenvolvimento de software ágeis

Bruno Lopes Machado[1,†] e João Paulo A. Almeida[1,*,†]

[1] *Núcleo de Estudos em Modelagem Conceitual & Ontologias (NEMO), Universidade Federal do Espírito Santo, Vitória, Brasil*

Abstract

This short paper presents the initial results in developing a semantic catalog of the main quality types concerning agile software development processes based on Scrum. Each module of this catalog consists of a reference ontology that clarifies the semantics of a quality type (such as velocity, lead time, cycle time, etc.). This work presents the first developed module, which covers the type of quality commonly visualized by so-called "burndown charts." The ontologies in the catalog are aligned with the Software Engineering Ontology Network (SEON), particularly specializing the Scrum Reference Ontology (SRO) and the Core Ontology on Measurement (COM).

Resumo

Este artigo curto apresenta os resultados iniciais do desenvolvimento de um catálogo semântico dos principais tipos de qualidades de processos de desenvolvimento de software ágeis baseados em Scrum. Cada módulo deste catálogo consiste em uma ontologia de referência que esclarece a semântica de um tipo de qualidade (como velocity, lead time, cycle time, etc.). Este artigo apresenta o primeiro módulo desenvolvido, que cobre um tipo de qualidade de processo comumente visualizado nos burndown charts. O catálogo tem ontologias alinhadas à rede SEON (Software Engineering Ontology Network), especializando em particular a Scrum Reference Ontology (SRO) e a Core Ontology on Measurement (COM).

Keywords

Agile software development process, Scrum, process qualities, reference ontology

1. Introdução

Nos últimos anos, a adoção de métodos ágeis [1] para o gerenciamento de projetos de software tem aumentado significativamente. Métodos Ágeis como Scrum e Kanban são valorizados por melhorar a flexibilidade, eficiência e capacidade de resposta das equipes de projeto. De acordo com o relatório da Digital.ai (2023) [2], 71% das organizações pesquisadas utilizam metodologias ágeis. Para gerenciar projetos ágeis de forma eficaz, a maioria das organizações utiliza ferramentas como Jira, Trello e Azure DevOps [2], que permitem planejar, monitorar e relatar o progresso do projeto. Estas ferramentas aumentam a visibilidade do andamento do trabalho e ajudam as equipes a identificar problemas e tomar decisões informadas. A capacidade de visualizar e acompanhar o progresso das tarefas e realização de histórias de usuários ("*user stories*") em tempo real é essencial para manter o foco da equipe e garantir que as metas sejam alcançadas.

Neste contexto, as métricas de acompanhamento desempenham um papel crucial no sucesso dos projetos ágeis, fornecendo dados objetivos sobre qualidade, desempenho e a eficácia das práticas ao longo do tempo. Métricas como velocidade, tempo de ciclo, *lead time* e taxa de defeitos são usadas para medir o progresso e a qualidade do trabalho, permitindo que as equipes ajustem processos e melhorem continuamente.

Proceedings of the 17th Seminar on Ontology Research in Brazil (ONTOBRAS 2024) and 8th Doctoral and Masters Consortium on Ontologies (WTDO 2024), Vitória, Brazil, October 07-10, 2024.

* Corresponding author.

† These authors contributed equally.

blopesmachado@gmail.com (B.L. Machado); jpalmeida@ieee.org (J.P.A. Almeida)

0009-0004-6260-5686 (B.L. Machado); 0000-0002-9819-3781 (J.P.A. Almeida);

Projetos ágeis podem utilizar múltiplas ferramentas para gerenciar tarefas, bugs, repositórios de código, entre outros, e cada uma pode ter seu próprio conjunto de métricas e formatos de dados. Neste contexto, ontologias de referência podem proporcionar um artefato comum para mapear e integrar dados de diferentes ferramentas [3].

Este artigo apresenta os resultados iniciais do desenvolvimento de um catálogo semântico com ontologias de referência para as principais qualidades de processos de desenvolvimento ágil baseados em Scrum. O catálogo visa harmonizar termos e definições, facilitando a comunicação e a comparação de dados entre ferramentas e projetos. O primeiro módulo, focado nas qualidades de processo tipicamente visualizadas em *burndown charts*, é apresentado como parte da *Agile Measurement Reference Ontology* (AMRO). O catálogo está encorado na rede de ontologias SEON (Software Engineering Ontology Network) [4] (https://dev.nemo.inf.ufes.br/seon), reutilizando desta rede conceitos básicos do domínio de engenharia de software.

2. Referencial Teórico

2.1. Métodos ágeis, Scrum e a Ontologia de Referência do Scrum (SRO)

Os métodos ágeis surgiram como uma resposta às limitações dos métodos tradicionais de gerenciamento de projetos e desenvolvimento de software, que frequentemente se mostravam inflexíveis e inadequados para lidar com a complexidade e a incerteza do cenário moderno. Neste trabalho adota-se o método ágil Scrum [5], levando em conta sua grande adoção. O método define dentre outros elementos papéis, eventos e artefatos. Os papéis centrais são o Product Owner, que define a visão do produto; o Scrum Master, que facilita as práticas do Scrum; e o Time de Desenvolvimento, que executa o trabalho de forma colaborativa. Os eventos principais incluem os Sprint, ciclos de 2 a 4 semanas de desenvolvimento, e reuniões como *Sprint Planning, Daily Scrum, Sprint Review e Sprint Retrospective*, que visam planejar, monitorar e melhorar o processo. Os artefatos incluem *User Stories, Product Backlog, Sprint Backlog* e *Tasks,* que ajudam a organizar e priorizar o trabalho. Além disso, métricas como *Velocity, Lead Time, Cycle Time, Cumulative Flow, Burnup e Burndown* são usadas para monitorar e otimizar o progresso do projeto.

A Ontologia de Referência do Scrum (SRO) [3] reusada neste trabalho está estruturada em cinco subontologias. AMRO reusa conceitos de SRO em especial da sua subontologia *Product and Sprint Backlog Subontology,* apresentada na Figura 1. A subontologia foca em aspectos relacionados aos requisitos estabelecidos em um projeto Scrum, tais como: *user stories* definidas no *product backlog*, critérios de aceitação, seleção de *user stories* para composição de um sprint, planejamento e execução de tarefas de desenvolvimento (*tasks*). Estes conceitos serão essenciais para caracterizar as qualidades do processo ágil que são associadas à execução de tarefas que visam implementar as *user stories.*

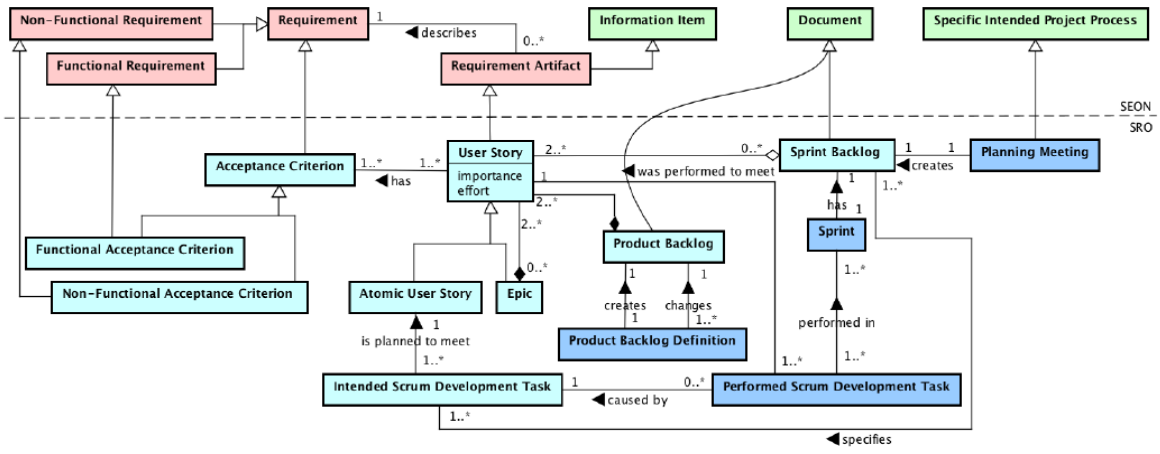

Figura 1: SRO - Product and Sprint Backlog Subontology

2.2. Core Ontology on Measurement (COM) e M-OPL

Barcellos, Falbo e Frauches [6] definiram um conjunto de padrões ontológicos para ontologias de medição que formam a *Measurement Ontology Pattern Language* (M-OPL), posteriormente incorporados à rede de ontologias SEON na ontologia de núcleo *Core Ontology on Measurement* (COM). Os padrões são organizados em seis grupos: *Measurable Entities, Measures, Measurement Units & Scales, Measurement Procedures, Measurement Planning and Measurement & Analysis*.

Neste trabalho, ancoramos AMRO nos elementos do primeiro grupo, *Measurable Entities* (Entidades Mensuráveis). A Figura 2 mostra os padrões deste grupo com o fundo azul claro, relevando também os conceitos da UFO (em branco) que fundamentam os padrões. Todo Elemento Mensurável (*Measurable Element*) [6] é um Tipo de Qualidade Mensurável (*Measurable Quality Type*) que caracteriza uma Entidade Mensurável (*Measurable Entity*). Volume e produtividade são exemplos de elementos mensuráveis. Tipos de Qualidades podem classificar qualidades simples (no caso de *Simple Quality Type*, um *powertype* cujo base type é *Simple Q*uality) ou complexas (*Complex Quality Type*, um *powertype* cujo base type é *Complex Quality*).

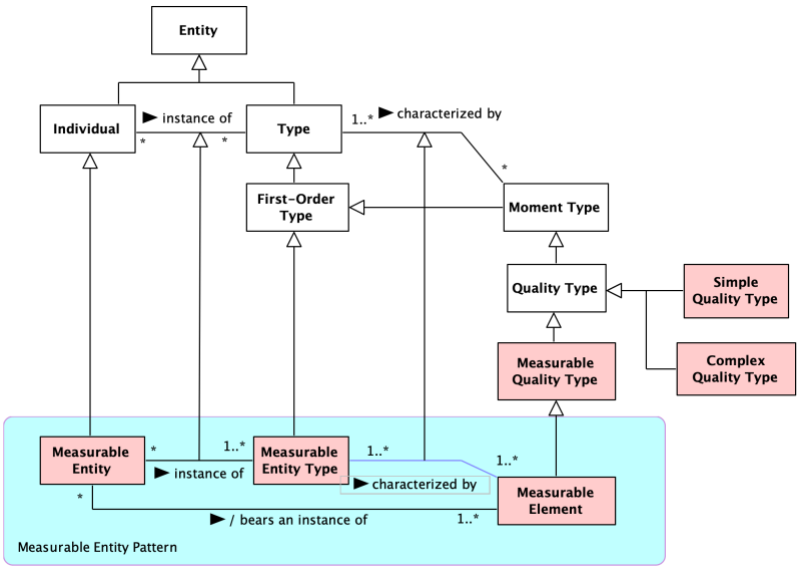

Figura 2: Padrão ontológico incorporado na Core Ontology on Measurement (adaptado de [6])

Entidades Mensuráveis (*Measurable Entities*) são classificadas de acordo com Tipos de Entidades Mensuráveis (Measurable Entity Types), tipos cujas instâncias são também tipos. Por exemplo, João (uma instância de Measurable Entity) é uma instância de Pessoa (uma instância de M*easurable Entity* Type). Measurable Entity Types são caracterizados por Measurable Elements (por exemplo, o tipo "Pessoa" pode ser caracterizado por Measurable Elements como "peso", "altura").

Barcellos, Falbo e Rocha [6] afirmam ainda que uma Medida (omitida no fragmento apresentado na Figura 2) é um instrumento que permite associar Elementos Mensuráveis com Valores de uma Escala. Por exemplo, o número de requisitos pode ser usado como medida para associar um valor ao elemento mensurável "tamanho", que caracteriza o tipo de entidade mensurável "projeto". Assim, uma Medida quantifica um Elemento Mensurável e possui uma Escala composta por Valores de Escala. Além disso, uma Escala pertence a um Tipo de Escala (por exemplo, absoluta, nominal).

3. Burndown

Um dos grandes desafios no gerenciamento de projetos de software é cumprir os prazos acordados, muitas vezes prejudicados pelo monitoramento insuficiente do progresso diário. O sucesso desse monitoramento depende de fatores como a qualidade do planejamento, a priorização de tarefas e a gestão de riscos e mudanças. Em ambientes ágeis, o *burndown* se destaca como um elemento de

medição essencial para acompanhar o progresso das *user stories* ou tarefas durante um sprint, promovendo transparência e colaboração na equipe, como destaca Rubin [7].

O *burndown*, conforme descrito por Ambler [8], é a relação entre o trabalho restante e o tempo disponível para concluí-lo, sendo especialmente útil em equipes que operam em sprints. Representado graficamente, o *burndown* auxilia na identificação de atrasos e no ajuste de estratégias para cumprir os prazos. Esse gráfico utiliza o eixo X para representar o tempo e o eixo Y para mostrar o trabalho a ser feito (por exemplo em número de user stories) e uma projeção linear do trabalho a ser feito ao longo do tempo. A Figura 3 apresenta um exemplo de *burndown chart* de um sprint, no período de 02/01/2023 a 11/01/2023, com 80 *user stories* priorizadas em seu *backlog*. Há baixo progresso no início da iteração, e aumento do progresso a partir da metade de iteração.

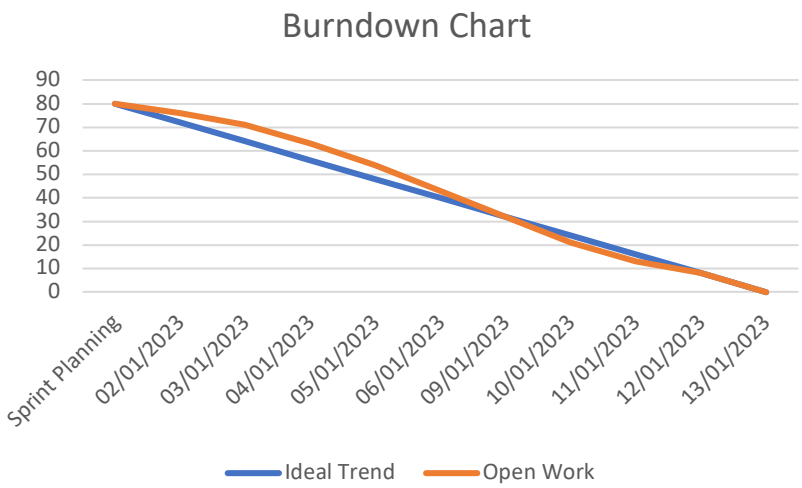

Figura 3: Exemplo de um *Burndown Chart*

4. Agile Measurement Reference Ontology

A AMRO está sendo projetada em seis módulos: (i) a subontologia de Process Burndown, que foca na quantidade de trabalho planejado remanescente ao longo do processo; (ii) a subontologia de Process Burnup, que aborda a quantidade de trabalho já concluído; (iii) a subontologia de *Velocity*, que indica a quantidade de trabalho que um time completa em certas unidades de tempo; (iv) a subontologia *Cumulative Flow*, que se concentra na quantidade de trabalho em diferentes estágios (Exemplo: “To Do”, “Doing” e “Done”) ao longo do tempo; (v) a subontologia de Lead Time; e a subontologia de Cycle Time. A Figura 4 mostra uma visão geral do AMRO com base nesses módulos. Cada pacote dentro do pacote AMRO representa uma subontologia referente a um tipo de qualidade de processo. A relação de dependência indica que as ontologias reusam elementos de SRO (e indiretamente SPO) e COM.

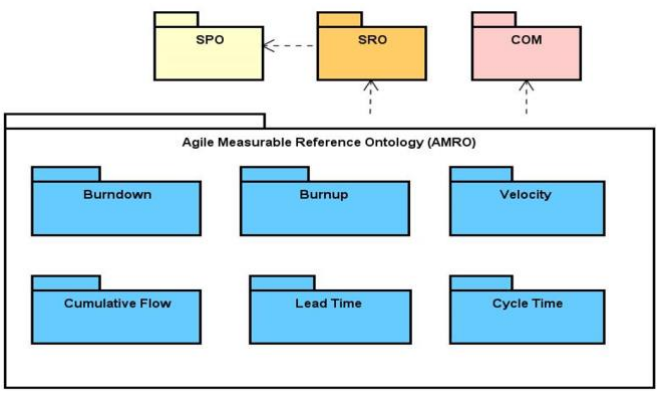

Figura 4: Arquitetura de AMRO

4.1. Burndown Sub-Ontology

O módulo de *burndown* visa responder às seguintes questões de competência:

QC01. Qual é o andamento atual do projeto/sprint?

QC02. A equipe está concluindo os itens do trabalho planejado?

QC03. A iteração atual indica atraso?

QC04. Quanto trabalho ainda precisa ser feito?

QC05. A equipe está melhorando a execução ao longo do tempo?

QC06. Considerando o progresso histórico do projeto, o trabalho será concluído como previsto?

A Figura 5 apresenta a sub-ontologia de *burndown*. Os elementos em azul se somam àqueles de SRO (amarelo claro), COM (rosa claro) e SRO (laranja). Três níveis de *burndown* são considerados: (i) *Process Burndown*, no nível de processo; (ii) *Sprint Burndown*, em cada sprint do process; e (iii) *Sprint Segment Burndown*, em segmentos de um sprint. Os dois primeiros são tipos de qualidades complexas, que se ancoram na qualidade simples *Sprint Segment Burdown*. Todas são instâncias de *Measurable Element*. *Sprint Segments* foram introduzidos para identificar segmentos (partes "construídas" [9] de Sprints). Estas partes são relacionadas entre si através de uma relação *has next segment* correspondendo à relação *meets* de Allen [10]. Note que Scrum Processes, Sprints e Sprint Segments são eventos. Seguindo Guarino [11], estamos estendendo o uso de qualidades também para eventos (para além de *endurants*). Em [11], Guarino introduz o conceito de "local quality" para denotar uma qualidade inerente a uma parte de uma entidade. Um dos exemplos fornecidos para "local quality" é a largura de um rio. Em cada segmento de um rio, ao longo da dimensão da distância para a foz, por exemplo, há potencialmente uma largura diferente. Desta forma, uma expressão como "largura de um rio" se refere de fato a uma qualidade complexa formada a partir das "local qualities" de cada segmento do rio. Guarino classifica essa qualidade complexa (a "largura do rio") como um "quality field". Já o comprimento do rio da nascente até a foz é considerado uma "global quality" (inerente ao todo integralmente). Estes conceitos são relevantes para eventos usando a dimensão temporal: qualidades de partes temporais são "local qualities", que podem ser agregadas em "quality fields". Por exemplo, ao caracterizar um episódio de chuva, é possível falar de um "campo de intensidade" composto de diferentes "intensidades locais" em partes temporais do episódio de chuva ao longo do tempo [11]. A duração da chuva pode ser compreendida como uma "global quality". Desta forma, entendemos nesta ontologia que o Sprint Segment Burdown é uma "local quality", e Sprint Burndown e Process Burndown, "quality fields" relativos a um Sprint ou ao processo como um todo.

Cada tipo de qualidade deve ser associado a um espaço de valores [12]. No caso de *Sprint Segment Burn*down esse valor é um inteiro não negativo que se refere à quantidade de *user stories* completadas no *Sprint Segment* correspondente, o que é definido formalmente pela seguinte regra OCL:

```
context SprintSegmentBurndown::value : Integer
derive: self.segment.doneStoriesInSegment->size()
```

Como *Sprint Burndown* é uma "quality field", usamos um espaço de valores formado por sequências de inteiros não negativos, se referindo à quantidade de *user stories* completadas em cada *Sprint Segment* ao longo do *Sprint*. Este valor é definido formalmente pela seguinte regra OCL:

```
context SprintBurndown::value : Sequence(Integer)
derive: segmentBurnDowns-> collect(value)
```

Seria possível também conceituar uma "global quality" alternativa (S*printBurndown'*) a *Sprint Burndown* no nível de cada sprint, cujo valor "agregado" de trabalho realizado seria dado por:

```
context SprintBurndown'::aggregatedValue : Integer
derive: segmentBurndowns-> collect(value)->sum()
```

No caso de *Process Burndown*, como "quality field", seu valor seria dado pela sequência formada pela concatenação de todas as sequências de valores dos segment burndowns (note que assumimos que a ordem de sprint burndowns e segment burndowns é mantida, ou seja, os fins de associação *sprintBurdowns* e *segmentBurndowns* são ordenados refletindo as relações temporais entre os eventos correspondentes):

```
context ProjectBurndown::value : Sequence(Integer)
derive: sprintBurndowns-> collect(segmentBurndowns)->flatten()
```

Novamente, uma variante como “global quality” seria possível, com um valor agregado para Project Burndown:

```
context ProjectBurndown::value : Integer
derive: sprintBurnDowns-> collect(segmentBurndowns)->flatten()->sum()
```

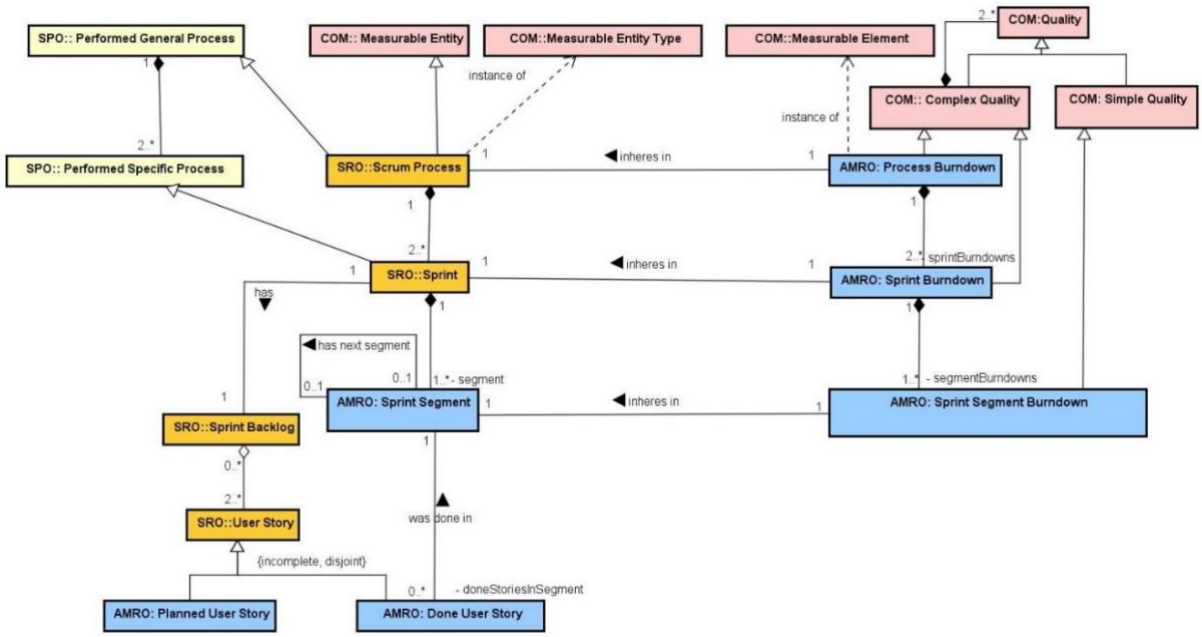

Figura 5: Burndown Subontology

5. Conclusão e Trabalhos Futuros

Este trabalho apresentou o primeiro módulo de um catálogo de tipos de qualidades de processos ágeis. O desenvolvimento desta ontologia teve como objetivo fornecer um modelo estruturado ancorado em SEON (e UFO), viabilizando responder questões de competência sobre qualidades de processos. Foi possível identificar a utilidade dos conceitos de “*local quality*”, “*global quality*” e “*quality field*” para definir (formalmente) diferentes aspectos do processo de software que podem ser compreendidos de forma geral sob o termo “burndown”. Desta forma, é possível desambiguar a noção de “burndown” em contextos específicos. Não foram abordados ainda aspectos relativos ao conceito de “Medida” em si. Na presente ontologia, *burndown* foi conceituado em diferentes granularidades de elementos do processo de desenvolvimento de software, porém apenas em termos de quantidades de *user stories*. Outras variações incluem *story points* ou ainda outras unidades de esforço. Ademais, algumas questões de competência (QC3, QC6) remetem a restrições de processos (como prazos definidos) ainda não capturadas em SRO. Desta forma, estes aspectos ainda serão examinados para caracterizar todos os elementos semânticos subjacentes a um *burndown chart*.

A criação de uma ontologia de referência para métricas ágeis é um projeto contínuo e em evolução. O próximo passo envolve o desenvolvimento de outras ontologias de referência para diferentes qualidades de processos ágeis, como *velocity*, o *lead time*, *cycle time*, entre outras. Cada uma dessas ontologias será projetada e avaliada, garantindo que o conjunto completo de ontologias ofereça uma visão holística e integrada das qualidades de processos ágeis para acompanhar a evolução de um projeto. A avaliação deve ser dar com a aplicação dos vários módulos em projetos realistas de integração de dados, a exemplo do relatado em [3] para SRO.

Agradecimentos

Este trabalho contou com o financiamento do CNPq (313412/2023-5) e da FAPES (1022/2022).

Referências

[1] Beck, K., Beedle, M., van Bennekum, A., et al. Manifesto for Agile Software Development. Agile Alliance, 2001. URL: https://www.agilealliance.org/agile101/the-agile-manifesto/

[2] Digital.ai. 17th Annual State of Agile Report, 2023. URL: https://digital.ai/resource-center/analyst-reports/state-of-agile-report/

[3] Santos Júnior, P. S., Almeida, J. P. A., Falbo, R. A., Barcellos, M. P. From a Scrum Reference Ontology to the Integration of Applications for Data-Driven Software Development, Information and Software Technology, 136 (2021).

[4] Borges Ruy, F., Falbo, R. A., Barcellos, M. P., Costa, S. D., Guizzardi, G. SEON: A software engineering ontology network, Proc. 20th International Conf. Knowledge Engineering and Knowledge Management (EKAW 2016), Springer, 2016, pp. 527-542.

[5] Schwaber, K., Sutherland, J., The Scrum Guide. Scrum.org, 2020, URL: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf

[6] Barcellos, M. P., Falbo, R. A., Frauches, V. Towards a Measurement Ontology Pattern Language, Proceedings of ONTO.COM/ODISE@ FOIS, 2014. URL: https://ceur-ws.org/Vol-1301/ontocomodise2014_9.pdf

[7] Rubin, K. S. Essential Scrum: A Practical Guide to the Most Popular Agile Process. Addison-Wesley Professional, 2012.

[8] Ambler, S. W., Lines, M. Agile Modeling: Effective Practices for Extreme Programming and the Unified Process. Wiley, 2002.

[9] Pribbenow, S. Parts and wholes and their relations, Mental Models in Discourse Processing and Reasoning, Advances in Psychology, vol. 128, North-Holland (1999), pp. 359–382.

[10] Allen, J. F. Maintaining knowledge about temporal intervals, Communications of the ACM 26(11) (1983), pp.832-843.

[11] Guarino, N. Local Qualities, Quality Fields, and Quality Patterns: A Preliminary Investigation, Proceedings of the Second Interdisciplinary Workshop The Shape of Things (SHAPES), 2013. URL: https://ceur-ws.org/Vol-1007/

[12] Guizzardi, G. Ontological Foundations for Structural Conceptual Models, Ph.D. Thesis, University of Twente, The Netherlands, 2005.