DT-Master: An Ontology to Support Provenance Capturing of Global Software Development Processes

Keywords: GLOBAL SOFTWARE DEVELOPMENT, ONTOLOGY, PROVENANCE DATA, MERGE

Abstract: Global Software Development (GSD) refers to the practice of creating and maintaining software by geographically distributed teams in different parts of the world, reducing software development (SD) costs and speeding up product development. However, capturing, documenting, storing, analyzing and extracting useful information about GSD processes and their evolution is still a challenge. One way of dealing with this challenge is the use of ontologies and their ability to represent knowledge of the world or some part of it. Despite the existence of several ontologies to address SD-related issues, there is still a gap in the coverage of the technical and people management aspects required by GSD, including process provenance.This paper develops the DT-Master (Data-Master) ontology by merging two existing ontologies, aiming to address social and cultural challenges of GSD and software process provenance. Incorporating 88 classes and 96 object properties (including 7 classes and 6 equivalent properties), DT-Master allows capturing software development data, segmenting projects and managing developers regardless of location, minimizing GSD challenges.

1 INTRODUCTION

Software development methods are constantly changing and evolving, especially when considering the Global Software Development (GSD) context. Additionally, software development companies have started to use data-driven practices in parts of their business over time, aiming to reduce their costs and software creation times(Jansen, 2020).

Data provenance is the process of tracing the origins of data and how it has changed to its current state (Buneman et al., 2001). It allows us to interpret and analyze what happened during the process of creating and modifying data, even diagnosing any problems that may have occurred during the process(Lim et al., 2010).

Capturing the provenance data of software development processes allows reusing and sharing those data, even in a distributed development environment, reducing the possibility of repeated executions with previously detected failures(Bose et al., 2019).

From the analysis of the challenges and difficulties faced in the context of GSD pointed out by (Pinheiro et al., 2023), it was observed that it was necessary to improve some existing models, which however are not applied or not always used by companies and teams to complement the development and evolution of software versions.

To find solutions for sharing and analyzing software development process data, there is the possibility of using a data provenance model that supports sharing software process data in environments with different characteristics, as in GSD, where cultural problems and geographical distance become common. Based on this, this paper aims to develop a new ontology from the merge of two existing ones to address both social/cultural challenges from GSD and the aspects of software process provenance.

The remainder of this article is divided into II Background, III Methodology, IV DT-Master Validation, and V Conclusion.

2 RELATED WORKS

With the aim of obtaining a theoretical foundation of the works under development in the field, as well as identifying the main challenges highlighted in the literature, the following works were analyzed.

In ”Challenges And Problems Of Software Processes In The Context Of Global Software Development: A Systematic Mapping Study”(Pinheiro et al., 2023), a systematic mapping is conducted to understand the main challenges and issues of GSD identified in the literature, as well as the solutions used to mitigate them. After applying the filters, 51 studies were selected for full reading. From these articles, various pieces of information were extracted, including the difficulty arising from geographically distributed environments, team management problems, and the lack of a formal and standard model that meets the needs for improvement and maintenance during

the software process.

In "DKD-S: An Ontology-based Tool for Global Software Development"(Rocha et al., 2021), a tool is presented that aims to help teams working in a distributed environment to face the challenges of software development in this model. As a result, a decision support system based on an ontology called DKDOnto was developed, capable of supporting the SD process. Additionally, this tool can assist in all phases of Software Engineering and even offer recommendations for project organization and both technical and non-technical experiences.

In this context, in "DKDOnto: An Ontology to Support Software Development with Distributed Teams" (Rocha et al., 2018), the ontology used in the previously presented tool is proposed. This ontology was developed using the OWL language, and the software Protégé 4.0 [1] was used for its editing and modeling. To create this ontology, a systematic mapping was conducted to understand which tools were proposed to minimize the impacts caused by distributed software development. The proposed classes are based on classes from other previously developed ontologies, and others were created to meet the needs identified in the systematic mapping conducted.

Finally, "Supporting Software Processes Analysis And Decision-making Using Provenance Data" (Dalpra, 2020) presents the PROV-SWProcess data provenance model, an extension of the PROV model [2]. This model is capable of capturing provenance data from software processes and assisting managers in the decision-making process. PROV-SWProcess can capture prospective provenance, meaning the standard and intended process, as well as retrospective provenance, which refers to the process that has already been executed. Additionally, it handles other data from the software process, such as artifacts, activities, procedures, stakeholders, which will be further detailed in session 3.1 (Costa et al., 2021).

3 BACKGROUND

Global Software Development (GSD) is a development practice in which organizations seek professionals from other countries to be part of their teams, thus, all software development processes will be distributed globally, increasing productivity and reducing costs (Jan et al., 2016). Furthermore, one of the great benefits of this development practice, besides cost reduction, is the reduction in the time used for development, as the time zone contributes to someone always being involved in system development(b10,).

One of the models used for development management in distributed environments are data representations through ontologies. In this vein, ontology - a term of philosophical origin, which aims to describe objects of the real world - has been studied for application in the field of computing. According to (Borst, 1997), an ontology can be defined as a formal and explicit specification of a shared conceptualization. In other words, it is a formal representation of concepts and their interrelationships in certain contexts. This makes it possible to represent environments and define dependencies, relationships, processes, and other software development characteristics.

For this work, two ontologies were studied to capture provenance data: PROV-SwProcess (Dalpra, 2020) and DKDOnto (Rocha et al., 2018). Provenance is the process of tracking the origins of data and how they have changed to their current state (Buneman et al., 2001), as well as sharing and understanding software processes in a distributed and heterogeneous scenario. By creating provenance data, people can easily find, reuse, and share data; it simplifies collaboration in a GSD environment, reducing the possibility of repeated failed executions. Furthermore, provenance can be used to explain how construction failures were made (Bose et al., 2019).

3.1 PROV-SwProcess

PROV-SwProcess is defined as an extension of the W3C recommended standard PROV(W3C Working Group Note, 2013), aiming to capture and store the most relevant information about software development process provenance data properly (Dalpra, 2020). Considering this, the main aspects covered are.

- Prospective provenance: Corresponds to the software development process specification, detailing all the activities that must be carried out to generate the software. The future process;
- Retrospective provenance: Comprises the activities that were executed in the generation of the software. The past process;
- Activity: deals with the process activities used to create and/or maintain software and how they compose the software development process;
- Stakeholder: refers to organizations, persons, projects, or teams acting or interested in the software process activities;
- Resource: involve hardware equipment and software products used by the software process activities

[1] https://protege.stanford.edu/

[2] https://www.w3.org/TR/prov-primer/

- Procedure: relate to methods, techniques, and document templates adopted by the software process activities;
- Artifact: represent different types of objects produced, changed, and used in process activities;

In Figure 1, we have an example of retrospective provenance using the specified classes.

3.2 DKDOnto

The second ontology, DKDOnto (Rocha et al., 2021), presented in Figure 2 , is an ontology focused on storing information related to a project, such as members, best practices, challenges, skills, artifacts, methodology, project phase.

- Project: the main class of this knowledge base. It is responsible for storing all information regarding the Project settings, from the Allocated Members to the phases, activities, and artifacts used;
- Member: A member has skills and works in one place and participates directly in the project, reporting on best practices and challenges, using and creating the artifacts;
- BestPractices: All solutions and best practices used for any problem should be stored in this entity. The role of this class is helping to avoid challenges and problems encountered and reported by the members while performing their activities. It also helps to solve these problems and challenges;
- Challenges: All challenges and problems encountered by members should be stored in this class. A challenge can use best practices to be solved, thus, these practices that are known can be used to solve issues. This entity is fundamental because the challenges have solutions or good practices associated with some practices that can be used and made available to other members with the same problems. Can be classified into NonTechnical and Technical;
- Skills: All member's knowledge is stored in this entity. The skills of the members can be used to avoid or solve problems. This class also allows activities to be distributed to members according to their abilities;
- Artifact: a class that is used by almost all other major classes. It supports members and their activities. Tools can also use artifacts in specific activities;
- Methodology: This class stores the information about the methodology applied to the project and its particularities, allowing the team members to use a shared environment, thus reducing communication failures;
- Project Phase: class used to define the planned phases of the project, generating a sequence of activities to be followed;

3.3 Ontologies Merging

In merging, two original ontologies, are joined together to create a single merged ontology(Chatterjee et al., 2018). When two different ontologies are combined, it is usually because they address related aspects of the same domain or because one wants to extend an existing ontology with information from another. In this work, the objective of merge is the extension of an ontology with information from another. The goal of the merge is to create a single ontology that captures all the relevant information from the original ontologies, making it more comprehensive and useful.

According to (Chatterjee et al., 2018), the merge process requires some steps to complete. 1 - Analysis of the original ontologies, 2 - identification of correspondences, 3 - conflict resolution, 4 - validation testing, and 5 - documentation.

4 METHODOLOGY

First, analyze the original ontologies. It was observed that the PROV-SwProcess ontology defines and details the data that is generated in software development processes. However, in real situations, most processes are related to a project. The PROV-SwProcess ontology is unable to record and store project data, presenting a failure when proposed for use in the job market. With this in mind, the DKDOnto ontology was analyzed and it was observed that it has a similar level of detail to the PROV-SwProcess ontology, but focused on obtaining project data.

Next, identification of correspondences. An ontology has Object Properties and Classes. A class can have multiple object properties in relation to other classes. Therefore, it was also necessary to make the equivalences of classes and object properties. From there, two tables were created containing all the classes and object properties of the two ontologies, trying to find classes and properties that represented the same things. In the end, 7 classes from DKDOnto were matched with those from PROV-SwProcess as presented in Table 1.

In addition, equivalences were also made with 6 object properties presented in the Table 2.

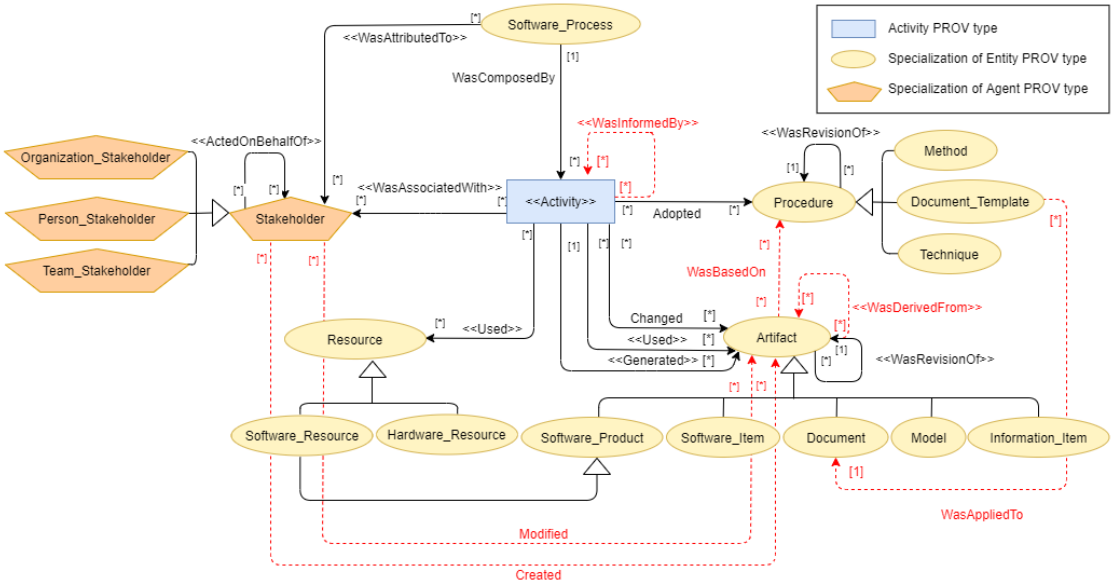

Figure 1: Example of the provenance of the PROV-SwProcess model (Dalpra, 2020).

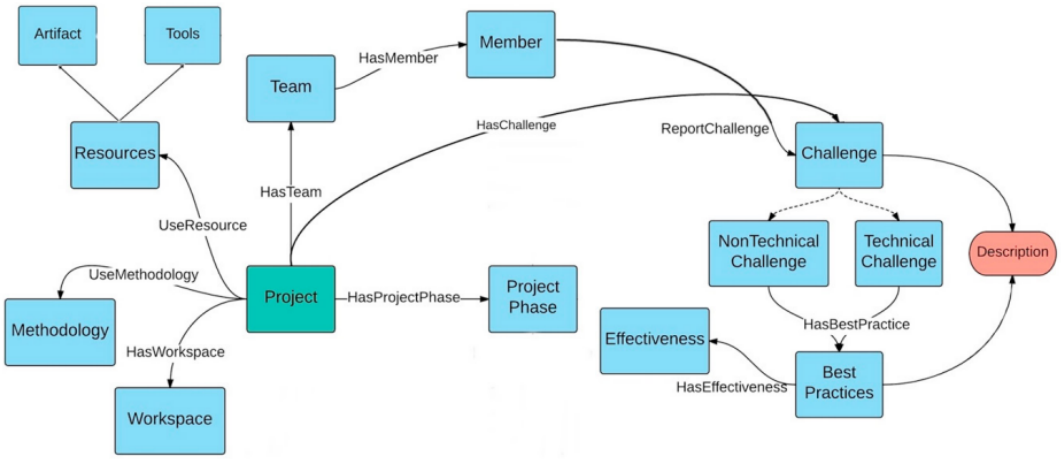

Figure 2: Relationships between the main classes defined in DKDOnto (Rocha et al., 2018).

Table 1: Equivalent Classes

PROV-SwProcess Class	DKDOnto Class
Activity	Project Phase
Activity	Project Activity
Person Stakeholder	Member
Team Stakeholder	Team
Stakeholder Role	Role
Resource	Resources
Artifact	Artifact

Table 2: Equivalent Object Properties

PROV-SwProcess Object Properties	DKDOnto Object Properties
adopts	Use Methodology
generates	Create Artifact
isSubActivity	Has Process Activity
isSubActivity	Has Task
hasRole	PlayRole
participates	inverse of Has Member

To merge the two ontologies, the Protégé application was used, a free and open-source ontology editor and structure for building intelligent systems. The third stage, is conflict resolution, after uploading the two ontologies and then performing the merge function. Reasoner was used, a Protégé functionality that checks for conflicts in the ontology, as an example. Its ontology defines that ”all vehicles have 4 wheels”

and ”there is a truck with 6 wheels” and ”the truck is a vehicle”, so one of the statements is wrong. The reasoner will warn you that this is impossible and something needs to be changed to make it possible. It is normal for this to happen when classes are equivalent. Maybe they are not equivalent.

If the ontology passes through Reasoner, it is possible. And with that, we created an example of use for the context of global software development for validation testing.

Through Protégé we have all the ontology documentation and the entire implementation process was documented during development. Additionally, the ontology can be downloaded in-app and is available for download in a link that will be made available in the final version due to the blind review procedure.

5 DT-MASTER VALIDATION

With this merge, a new robust ontology was created that covers the entire process area and now also the project area in detail. Figure 3 presents the new ontology developed in the Protégé application.

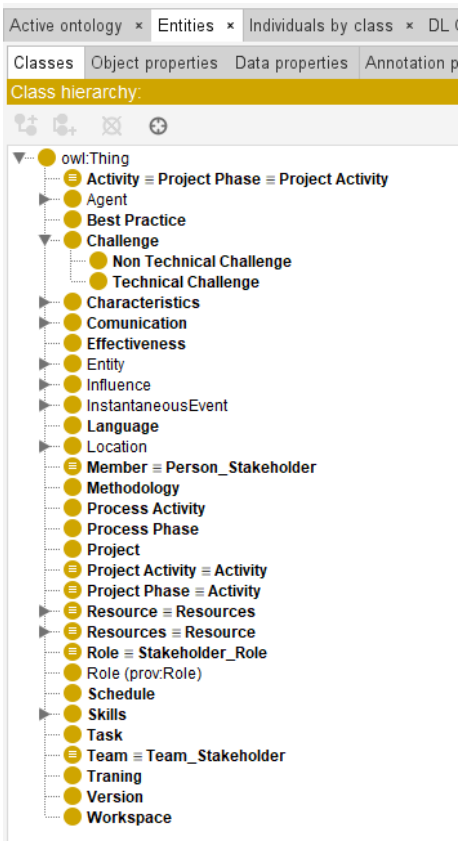

Figure 3: Merge ontology into Protégé

Before the merge, PROV-SwProcess had 20 classes and 25 object properties. And now the new ontology has 108 classes and 121 object properties.

For class equivalence we analyze the definitions of each class:

- DKDOnto Project Phase: class used to define the planned phases of the project, generating a sequence of activities to befollowed.
- DKDOnto Project Activity: Activities generated in project phase
- PROV-SwProcess Activity: represent a computational task in the software development process. It can be atomic or composite and may include the adoption of procedures, the use of resources, the modification, use and generation of artifacts, and the association with stakeholders responsible for its execution. The composite activities can be Project Phase or atomic activity can be Project Activity
- DKDOnto Member: is an individual who has access to the environment and can beallocated to projects.
- PROV-SwProcess Person Stakeholder: represent a person involved, interested or affected by the software development processes activities or results. Examples of Person Stakeholders are: a hired Programmer, an external Instructor, or a User.
- DKDOnto Team : Group related with the project
- PROV-SwProcess Team Stakeholder: represents teams involved, interested or affected by the software development processes activities or results. Examples of Teams Stakeholders are: the Software Engineering Process Group, a Quality Assurance Team, or a Testing Team.
- DKDOnto Role: function of anything.
- PROV-SwProcess Stakeholder Role: related to the function performed (or that should be performed) by a stakeholder when an activity was associated with a stakeholder.
- DKDOnto Resources: The grounds for all artifacts, frameworks, documents, applications, APIs, among others that are directly connected to the project,serving directly to all members.
- PROV-SwProcess Resource: Represents the different types of resources used by activities. It can be specialized in other two types: (i) Software Resource, and (ii) Hardware Resource.
- DKDOnto Artifact: class that is used by almost all other major classes. It supports members and their

activities. Tools can also use artifacts in specific activities;

- PROV-SwProcess Artifact: represent the objects produced, changed, or used in the software development process activities. Artifacts can be of five types: Software Product, Software Item, Document, Model, and Information Item.

All PROV-SwProcess definitions can be found at [3].

With the development of DT-Master, an example of use was created encompassing the main classes of interest of the ontologies.

In Figure 4 we have an example of using this ontology that was created and available in full size. DKDOnto aggregate classes are green and in other colors, they are PROV-SwProcess classes. The names in the arrows are the Object Properties that associate the classes.

Figure 5 shows a section of the complete example where it is possible to record initial project data.

- Project: Project under development
- Schedule: Expected duration
- NonTechnicalChallenge: New interface features
- Best Practice: Best practices typically used to solve a challenge
- Effectiveness: How efficient this best practice is for the related challenge.
- Technical Challenge: Challenges such as frontend and backend development.

Figure 6 demonstrates the planning part of the project. Practically, we can represent all of Scrum and its phases.

- Project Phase: Which phase of the project
- Schedule: Expected time for phase resolution
- Methodology: Phase approach methodology, for example Agile methodology, Scrum

Each project phase has process phases in development and their characteristics. Then we have

- Process Phase: Related to the project phase.
- Schedule: In Scrum, this is sprint times
- Process Activity: In Scrum this is Sprints.

Thinking in the context of GSD, we have people working in different places around the world. Therefore, the new ontology identifies the members responsible for each function, their location, and personal skills.

- Programmer: Role of a member
- Derek: member example
- Development Team: What is your team
- User Knowledge: Main member skills
- Continent/Country/State/Region/City/Language: All member locale and language information
- Personal Skill: Member's Hard Skills

Other information outside the example

- Email/Communication
- Cultural Characteristics.

Another section using classes from the two ontologies is shown in Figure 7.

- Activity Class: Represented by Deploy
- Member: Simon
- Artifact: Accounting System
- Personal Skill/User Skill/Email/User Knowledge: Member Related
- Version: Version of Artifact

6 CONCLUSIONS

The main contribution of this work was the development of a new ontology for people management and also for software development management based on two other existing ontologies that are currently used for different purposes, but when combined can minimize the challenges encountered in global software development. Throughout the research, it was observed that development software is usually created to help with software-related problems. This leaves a gap in the social and cultural problems encountered by the teams that develop them. However, it has not yet been possible to apply the ontology created in development environments to receive feedback on its adherence and efficiency.

ACKNOWLEDGEMENTS

I would like to thank blind for their support in carrying out this work.

[3] https://www.gabriellacastro.com.br/provswprocess/v3.html

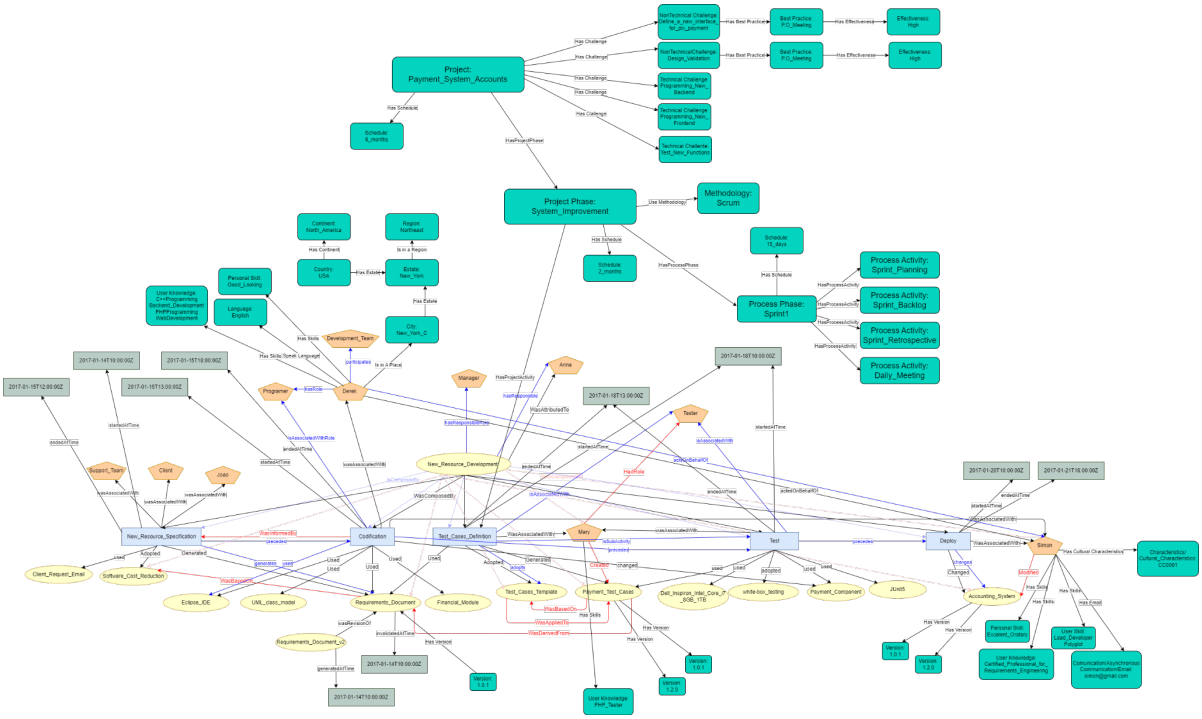

Figure 4: Example

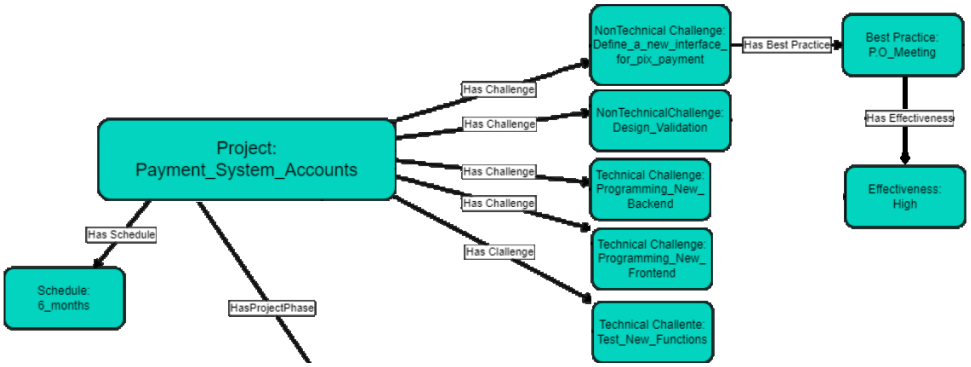

Figure 5: Project Example

REFERENCES

Guia facetado de técnicas elicitação de requisitos. https://retraining.inf.ufsc.br/guia/app/abordagens/abordagem-para-o-desenvolvimento-de-software-global. Accessed: 2024-6-21.

Borst, W. (1997). *Construction of Engineering Ontologies for Knowledge Sharing and Reuse*. Phd thesis - research ut, graduation ut, University of Twente, Netherlands.

Bose, R. J. C., Phokela, K. K., Kaulgud, V., and Podder, S. (2019). Blinker: a blockchain-enabled framework for software provenance. In *2019 26th Asia-Pacific Software Engineering Conference (APSEC)*, pages 1–8. IEEE.

Buneman, P., Khanna, S., and Wang-Chiew, T. (2001). Why and where: A characterization of data provenance. In Van den Bussche, J. and Vianu, V., editors, *Database Theory — ICDT 2001*, pages 316–330, Berlin, Heidelberg. Springer Berlin Heidelberg.

Chatterjee, N., Kaushik, N., Gupta, D., and Bhatia, R. (2018). Ontology merging: A practical perspective. In *Information and Communication Technology for Intelligent Systems (ICTIS 2017)-Volume 2 2*, pages 136–145. Springer.

Costa, G., Teixeira, E., Werner, C., and Braga, R. (2021). A proposal for sharing software process provenance data in heterogeneous environments. In *Anais do I Workshop de Práticas de Ciência Aberta para Engenharia de Software*, pages 37–42, Porto Alegre, RS, Brasil. SBC.

Dalpra, G. C. B. (2020). PROV-SwProcess: A PROV exten-

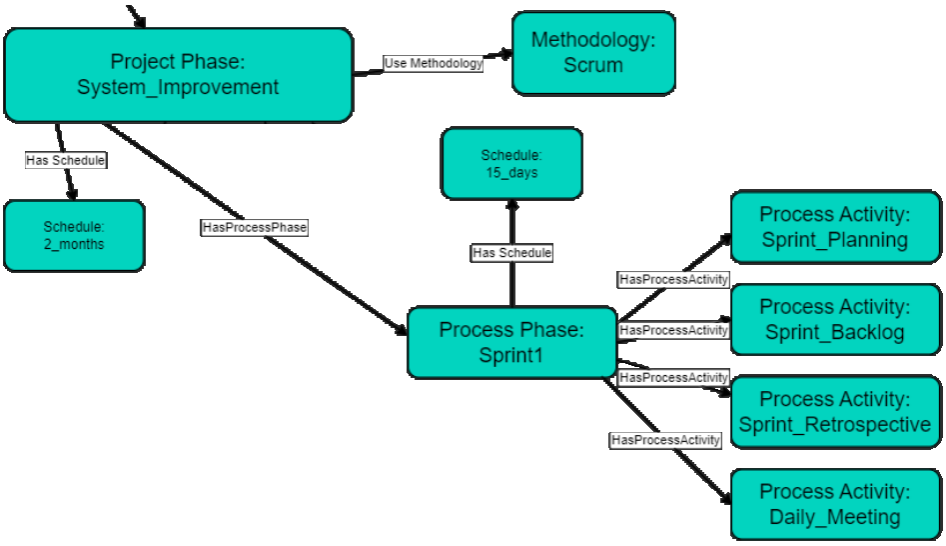

Figure 6: Project Phase Example

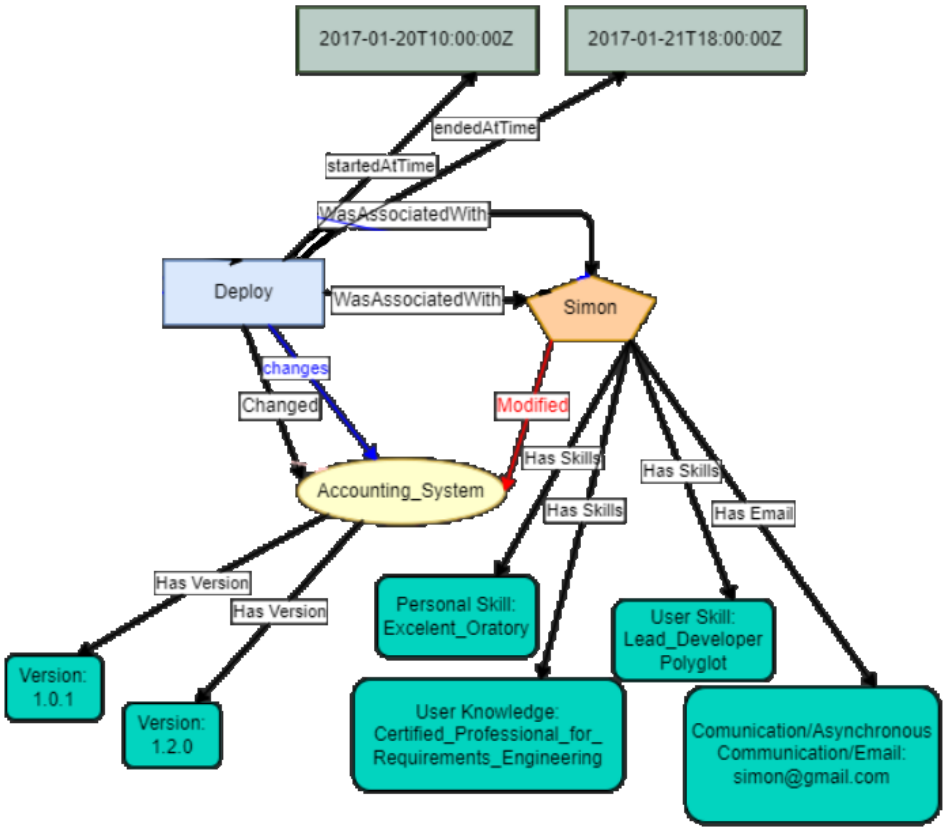

Figure 7: Activity Example

sion data model for software development processes. https://www.gabriellacastro.com.br/provswprocess/v3.html. Accessed: 2024-6-21.

Jan, S. R., Dad, F., Amin, N., Hameed, A., and Shah, S. S. A. (2016). Issues in global software development (communication, coordination and trust) a critical review. *training*, 6(7):8.

Jansen, S. (2020). A focus area maturity model for software ecosystem governance. *Information and Software Technology*, 118:106219.

Lim, C., Lu, S., Chebotko, A., and Fotouhi, F. (2010). Prospective and retrospective provenance collection in scientific workflow environments. In *2010 IEEE International Conference on Services Computing*, pages 449–456. IEEE.

Pinheiro, M. G., Macedo, G. M., and Dalpra, G. C. B. C. (2023). Principais desafios e problemas de processos de software no contexto do desenvolvimento global de software: um mapeamento sistemático. In *Proceedings of the 51 Brasilian Congress of Engineering Education*. Associação Brasileira de Educação em Engenharia.

Rocha, R., Araújo, A., Cordeiro, D., Ximenes, A., Teixeira, J., Silva, G., da Silva, D., Espinhara, D., Fernandes, R., Ambrosio, J., Duarte, M., and Azevedo, R. (2018). Dkdonto: An ontology to support software development with distributed teams. *Procedia Computer Science*, 126:373–382. Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 22nd International Conference, KES-2018, Belgrade, Serbia.

Rocha, R., Leandro, R., Silva, I., Araujo, J., Bion, D., Freitas, F., Cordeiro, D., Gomes, A., and Azevedo, R. (2021). Dkd-s: An ontology-based tool for global software development. In *2021 16th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–6.

W3C Working Group Note (2013). PROV-Overview: An overview of the PROV family of documents. http://www.w3.org/TR/prov-overview/. Accessed: 2024-6-21.