Software Process Improvement through the Combination of Data Provenance, Ontologies and Complex Networks

Maria Luiza Falci, Regina Braga, Victor Stroële and José Maria N. David
Postgraduate Program in Computer Science, Federal University of Juiz de Fora, Juiz de Fora, Minas Gerais, Brazil

1 INTRODUCTION

Software development is a complex activity that is usually followed by unexpected software behaviour (Fuggetta, 2000). Monitoring different development phases can be a good tactic for dealing with the software unpredictability, improving also its quality. One way of monitoring software properly is through software process control. Software process control is a critical factor for developing a software with good quality.

Software process can be characterized as a set of necessary activities for developing and implementing software, executed by a group of organized people according to an organizational structure and supported by conceptual and technical tools (Acuna et al., 2000). To be able to control software process, the support of suitable tools and techniques is essential. Monitoring and continuously improve processes are important mechanisms to be implemented. This monitoring and improvement go through the information knowledge related to the process, considering that, based on process model information, as well as based on previous or similar process executions, it is possible to refine new executions and provide process quality and software improvements. Therefore, with the monitoring and analysis of models and execution data, it's possible to bring previous knowledge to improve new processes. One way of saving this previous knowledge is through the use of data provenance techniques.

Data provenance describes the origin and the history of use of this data (Cuevas-Vicenttín et al., 2015). It is a relevant metadata, which can facilitate a better understanding about data, and provide information about where data came from, which process have generated it, what have influenced it, its context, how it was produced and its path until it was saved.

Considering the software process context, data provenance can improve it through provenance models and previous executions analysis. In order to facilitate data provenance capture and integration, there are some models such as OPM (Moreau et al., 2011) and PROV (Groth and Moreau, 2013), this last one the most widely used in academic community nowadays.

The PROV model has some specific extensions, such as ProvONE (Cuevas-Vicenttín et al., 2015), which is specific to be used in scientific workflows context and can be extended to be used with software processes. Furthermore, we believe that in this context, it is possible to extract implicit knowledge from stored provenance data, which can help future executions, as, for example, to find out obsolete tasks, overwhelmed agents, among other information related to processes executions.

This paper presents OntoComplex, an architecture that uses ontology, complex networks and inferences

Furtuozo Falci, M., Braga, R., Strõele, V. and David, J.
Software Process Improvement through the Combination of Data Provenance, Ontologies and Complex Networks.
DOI: 10.5220/0006700400610070
In *Proceedings of the 20th International Conference on Enterprise Information Systems (ICEIS 2018)*, pages 61-70
ISBN: 978-989-758-298-1

to derive implicit knowledge from provenance data related to software process. Its main goal is to assist software managers to improve future executions and reuse processes.

As a first step to OntoComplex specification, in order to identify the deficiencies in the domain, a systematic review was conducted to find relevant works in the area, specially works which propose techniques, frameworks, tools or procedures that could support project managers to make decisions based on data.

(Gencel et al, 2013) and (Naedele et al, 2015) propose models to deals with metrics and measurement programs. (Fonseca et al, 2017) proposed an ontology-based approach for measurement systems integration.

A systematic literature review was performed by (Tahir et al, 2016) to answer questions about software measurement programs. Challenges faced by companies which adopt data-driven decision making are presented by (Bosch, 2017).

This paper is organized as follows: Section 1 presents the introduction. Section 2 discusses the research context. In Section 3 OntoComplex architecture is presented and in Section 4 some related works are detailed. Finally, in Section 5 conclusions are discussed.

2 RESEARCH CONTEXT

Software development is a complex and unpredictable activity (Acuna et al, 2000). Through software process monitoring and analysis, it is possible to bring knowledge from previous process to new processes, finding and understanding vulnerabilities and deficiencies. One approach that can help in that tasks is data provenance, a “description of the origins of a piece of data and the process by which it arrived in a database” (Buneman et al., 2001).

Data provenance is a metadata that helps a better understanding about data origin, which process have generated or influenced it, its context and how it was produced used until the moment it was stored. Provenance describes which processes and steps are related to a data lifecycle. Considering software process context, a process uses a data and can generate another data, which can be input in another process or be relevant for a software manager decision making, which, based on this information can find the origin of errors, data accuracy, related users, execution time, execution date, used configuration parameters, among others. Benefits from using data provenance can be even more relevant when associated to technologies that help finding implicit knowledge, improving decision making strategies. Among these technologies, we can cite ontologies (Prabhune et al., 2016) and complex networks (Harary, 1969).

Ontologies are models that represent knowledge about a domain, which are able to express entities, relationships and rules related to that domain (Prabhune et al., 2016). The Prov-O ontology, part of PROV model from W3C, is specified using OWL2 language (Web Ontology Language). The Prov-O has classes, properties and restrictions that can be used to represent and acquire provenance information, generated by different systems and in different contexts. This model can also be specialized, adding new classes and properties to express knowledge from different applications and domains (Belhajjame et al., 2012). In this context, ProvONE (Cuevas-Vicenttín et al., 2015) is a Prov-O ontology specialization, to be used in scientific workflows domain.

ProvONE ontology can also be interpreted as a graph, with nodes and edges. In this paper is proposed a representation of this model as a complex network, where nodes and edges are related to the model entities and properties. With this new representation it is possible to better understand data influence and behaviour, which may generate strategic information to help decision making. In this vein, OntoComplex architecture was proposed considering different technologies, as part of the process management framework proposed in (Costa et al., 2016b) and (Dalpra et al., 2015) with the main objective to derive strategic knowledge to improve future software process executions.

3 OntOComplex

Considering the ProvONE model, and the use of ontologies, complex networks and visualization techniques, this section details OntoComplex architecture (Figure 1), which aims to provide software process improvement. The main goal of this architecture is to use software process and its execution data analysis, to help managers to make decisions based on acquired knowledge to improve future executions.

A partnership with a medium size development company was proposed with the goal of evaluating OntoComplex functionalities and its architecture use. A real dataset from error solution and new functionalities requirements from an ERP (Enterprise

Resource Planning) was used. The dataset was collected from the company database using the Mantis[1] software. After collecting data, they were processed based on interviews made with company´s managers and developers. These data have information about activities flow and information generated by each process.

The layers are detailed in Figure 1.

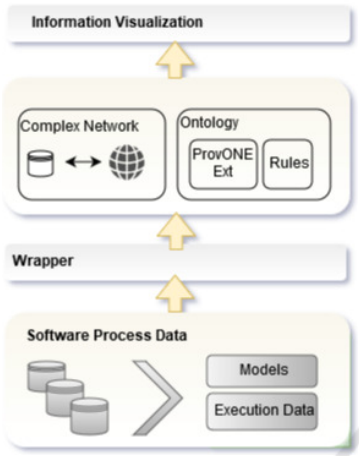

Figure 1: OntoComplex architecture.

3.1 Software Process Data

As stated before, the main goal of the architecture is to improve future software process executions. Thus, it was decided to support multiple software process data repositories, in multiple formats, in order to make the process management easier. Therefore, it is possible to use the approach in shared environments, allowing the data reuse in similar processes, which were generated by different process execution tools, in different organizations.

In order to make it possible, it was specified an intermediate layer (Wrapper on Figure 1), to make the necessary conversions between different data formats. To each data format it is necessary a new specific wrapper. However, OntoComplex provides a generic wrapper that needs to be specialized to specific formats, as necessary.

The dataset from the ERP Company was available in Mantis tool data format. Thus, the generic wrapper provided by OntoComplex was specialized to convert this specific format to PROVONE model. After the conversion, the dataset had 5624 registers with 29 attributes. The data were loaded into the ontology and ontological analysis were processed, together with complex network topological analysis.

[1] https://www.mantisbt.org/

3.2 Data Analysis

The use of ontological classes, their relationships and rules allow the derivation of new knowledge extracted from data. OntoComplex architecture uses these functionalities to derive strategic knowledge considering software process data. However, to make it possible, it was necessary to create specific extensions to software process context. As stated before, ProvONE model is a PROV extension for scientific workflows context. Considering the similarities between scientific workflows and software processes, it was decided to use ProvONE as a base model to create the specific extensions. Starting from the ProvONE model, it was created an extended ontology (ProvONEExt) presented on Figure 2. Figure 2 presents the extensions made on the model, which are mainly related to new relationships between classes and new ontological rules. As PROV is a generic model and ProvONE has more specific classes for scientific workflows, the model was adapted.

Based on this model (Figure 2), software process data extracted from the data layer were converted, loaded into the ProvONEExt ontology, and the ontological analysis were processed (starting from the specified ontological rules) as well as the complex network topological analysis. Details of how ontological analysis were made using the dataset provided by the ERP Company are presented in the following subsections.

3.2.1 Ontological Analysis

As it was mentioned before, ProvONEExt was created as a ProvONE extension, and it has ProvONE original classes and properties, besides original and added properties (created in this work context) that complement the model. Figure 3 presents object properties from ProvONEExt.

To provide a better understanding of ProvONEExt ontology use, the dataset from ERP Company was associated to the ontology's classes and subclasses, and all the relations were defined. The dataset is related to software process maintenance and include information such as: who requested the maintenance, its priority, to which user the maintenance was attributed to, the frequency which the errors happen, the submission date, the product version, and other information.

After initial definitions, it was chosen a reasoner (Pellet) for inferring knowledge. Figure 4 presents

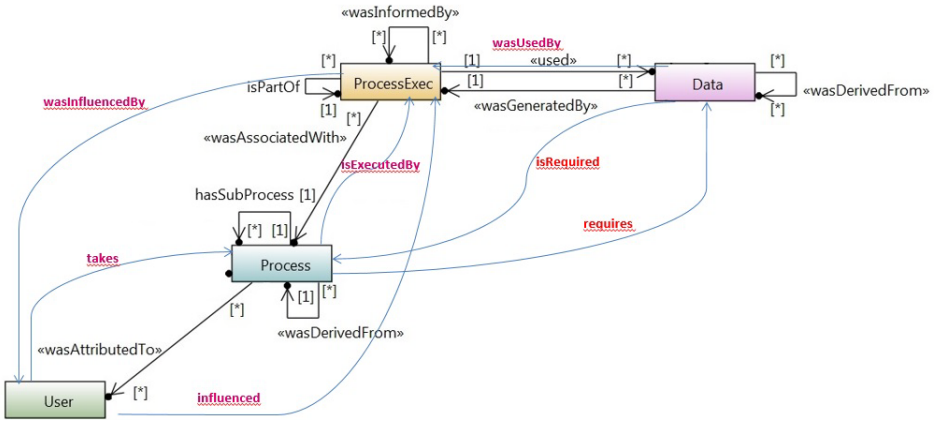

Figure 2: Data Provenance Model ProvONE [15], with new extensions.

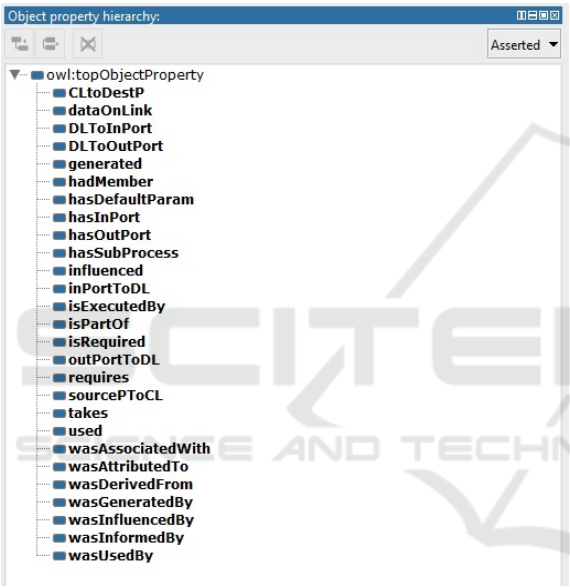

Figure 3: Some properties added to ProvONEExt ontology, using the software Protégé tool, version 5.1.0.

some information inferred by reasoned following the ontological rules. Figure 4 shows specifically inferences about software versions that were influenced by other versions.

For example, software version 2.5.05 was generated by other versions' maintenance indicated by the relationship 'wasGeneratedBy'.

Based on the knowledge extracted from ontology through inferences, it was possible to analyse and use information to make strategic decisions using the visualization layer. For example, it is possible to know which software versions were used for which other software executions, and which version was generated by that use, among other information. It is strategic decision because it makes easier to understand the impact of one version into other, and it is possible to make deep analysis if the manager considers that the impact was big, for example related budget cost and if these costs can decrease.

3.2.2 Complex Network Topological Analysis

The modelling of the complex network, in the OntoComplex context, was made with the data instantiated into ProvONEExt ontology, generating a graph through existing ontological relationships.

Considering ERP company dataset, it was created nodes and edges based on the model. It was also created weights for the edges[2] considering the dataset. Based on these data, information could be derived using complex network data mining techniques, analysing the network and its nodes.

OntoComplex analysis intend to define the complex network characteristics, and as consequence, extract knowledge from its data, for example centrality analysis, income degree, outcome degree, vulnerability in case of attacks, among others. These analyses allow a better understanding of the network, knowing the impacts that the network would have with the removal of a node. For example, considering the ERP Company, if a developer makes the majorities of the maintenance activities, how the network would be impacted if this developer was removed from it, how is the impact into the company if this person quits the job. Is there someone who would be able to substitute this person?

[2] The weights were defined based on the dataset. Thus, the severity or execution time of the process, was used to define the edge weights in this specific dataset.

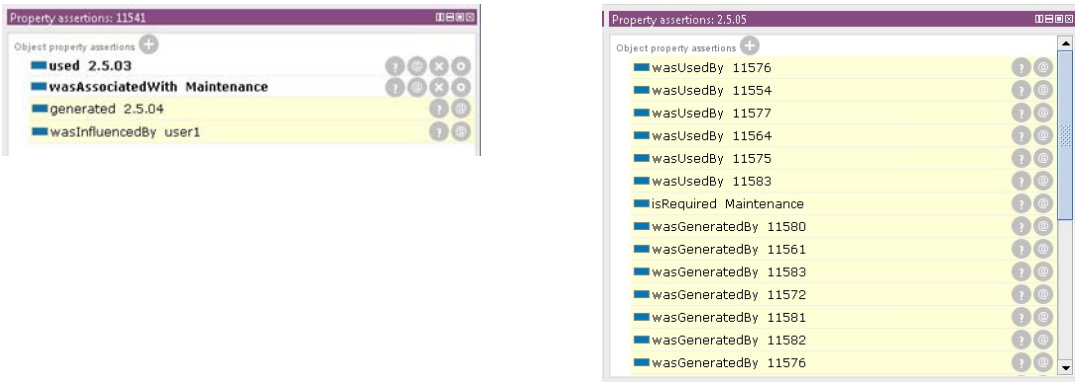

Figure 4: ProvONEExt ontology Inferences and Assertions a) and b), using the Protégé 5.1.0.

Moreover, with the complex network, it is possible to have an overview of companies' software processes, and it is also possible to analyse selected nodes and specific edges to understand them better considering a particular behaviour.

Following the initial complex network modelling, the network was populated with 5000 real process execution data. Some of this data, such as a process execution 'severity' and 'execution time', were used to define the edge weights of the graph, to relate a node to other, for example, an activity and the software version to which it is related.

After the complex network structure definition and storage using the Neo4j[3] database management system, which allows graph queries, the network analysis was processed in order to obtain relevant information about data and execution processes. One analysis example is related to the most critical activities and which users are related to them. Two different analysis can be made, the first one is numerical, where can be showed the most complex activities, the execution time, etc. The other possible analysis is visual.

In Figure 5 it is illustrated the relation between a developer (red nodes), the process executions (in blue), and the software version which was influenced by these executions (in green). It is possible to observe in Figure 5a) that the developer is connected to a big amount of execution processes, much more than the developer on Figure 5b). It is possible to make some analysis based on that observations, for example investigate why a developer has much more execution processes assumed, if these executions are harder than others, what would be the impact if this user quit the company, etc. The answers obtained from these questions help to better understand the importance of each network element, for example developers/users, as it is illustrated in Figure 5.

To have a general overview of the complex network using the ERP execution data, Figure 6 represents all execution data from the related process model.

Complex network analysis together with ontological analysis can derive useful strategic knowledge to software managers. As a practical example of how this work, if a manager clicks on a developer node, it can be analysed which processes were influenced by him and what is the intensity of this process, as well as information inferred using ontological rules. Thus, in the same view, it is possible to fire the two types of analysis, ontological and topological.

However, it is important to highlight that ontological and complex networks analysis provide the data analysis from different perspectives and granularities, and one complements the other. Ontological analysis is applicable when the analysis must be done in a more detailed point of view, in specific data, showing its details (Figure 7). Analysis related to complex networks are less specific and based on the nodes neighbours and graphic data visualization in general (Figure 8).

3.2.3 Information Visualization

Information visualization layer was specified to facilitate software process visualization and analysis. This layer includes Visionary framework (Costa et al., 2016a) to make better understanding of data processed by OntoComplex analysis layer. The visualization layer loads provenance data and uses the D3.js[4] framework to generate the visualizations.

As it was highlighted in the previous section, ontologies and complex networks allow a better visualization of data, its relations, and the main implicit knowledge present on data. Considering the

[3] https://neo4j.com/

[4] https://d3js.org/

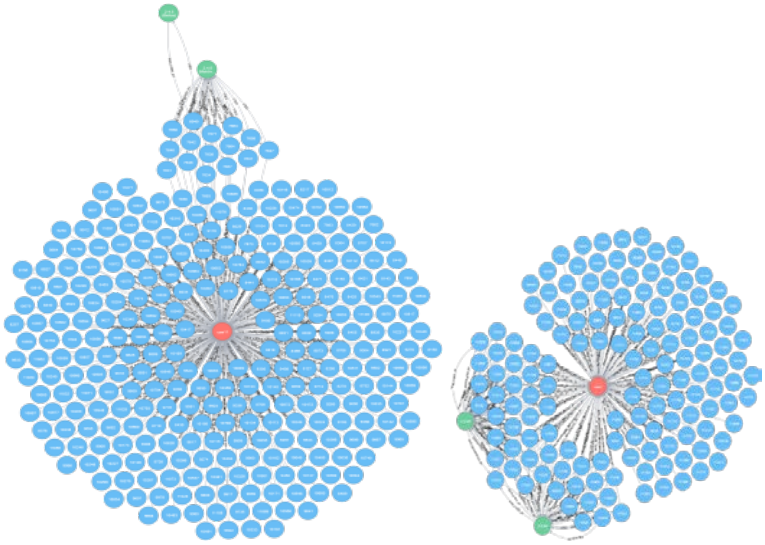

Figure 5: Complex network selections a) and b) show different relations between developers and maintenance activities, and the software versions related to them.

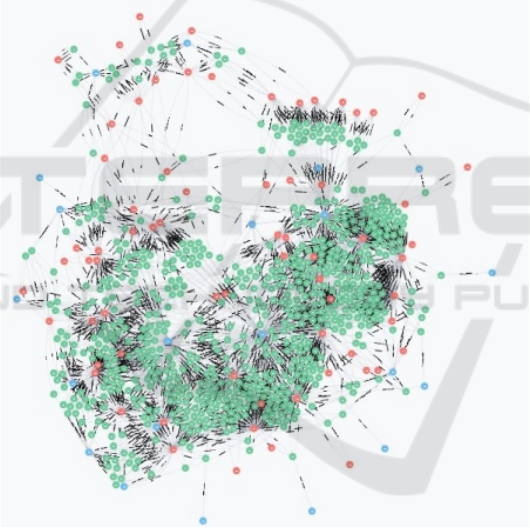

Figure 6: Complete view from the complex network data.

ERP company dataset, with the use of ontological and inferences rules, for example, it is possible to click on execution process data, on developers' data, on software versions, and see with which other data these ones are related, and what are the relation types. With the complex network based analysis, it can be observed a relation between data, and observe their behaviour on the network.

Some examples of visualizations generated by the Visualization layer are presented on Figure 7. The main nodes' names are available by default (with an option of showing all the names), and with one click on a specific node it is possible to show details and its more specific connections to other nodes. Nodes use a structured layout to automatically adjust its positions, to avoid any of them to be hided. It is also possible to show more information from a node if it is selected, including its related elements and relationships, helping to make established tasks.

The visualization presented in Figure 8 represents some immediate information to the system user. For example, the developer 'Support_4' has few connections with other elements, which allows a deduction that this user can be allocated to simpler tasks. On the other hand, 'VB6_2' has a lot of connections, which can indicate that this developer has been involved with more activities. The final

decision to be made is up to the software manager based on these data analysis.

4 RELATED WORKS

Related work will be analysed in the following paragraphs through different perspectives, a) the provenance data used, considering provenance data models and b) the approaches proposed to improve software processes.

DataONE is a multi-institutional, multinational and interdisciplinary organization that has collaboration from different researchers who intend to contribute to the scientific community with data science, using data provenance (Allard, 2012).

Cuevas-Vicenttín (Cuevas-Vicenttín et al., 2014a) proposes a repository named Pbase intending the integration with DataONE. The authors experience shows the potential acquired for supporting different search types through data provenance. Challenges faced by PBase are presented, as the definition of a provenance model that is compatible with the main scientific workflows, as well as functionalities characterization required to specific searches and also the development of a structure that allows users to evaluate queries effectively.

One of drawbacks of ProvONE is to model data provenance without a tool support. In this vein, a tool called Prov2ONE was proposed by (Prabhune et al., 2016) to generate the prospective provenance to scientific workflows. The provenance graph is them updated with the relevant retrospective provenance, improving the academic community interoperability, facilitating the share and workflows analysis.

Oliveira (Oliveira et al., 2016) discusses about the challenges faced by the academic community considering the different provenance models available, and the difficulty to interoperate them. The authors proposed a common model for two different models (SciCumulus e e-Science Central), using the ProvONE. The goal is to allow different workflows, which use different workflow models, to have a common model, allowing its reuse.

Considering approaches that deal with process improvement, as state before, a systematic review was conducted to find relevant works in the area, especially works which propose techniques, frameworks, tools or procedures that could support project managers to make decisions based on data.

In (Gencel et al, 2013) it is proposed a framework to assist software organizations to choose the right metrics when planning measurement programs. (Naedele et al, 2015) presents how knowledge from manufacturing execution systems can be used in software engineering. The authors provide a framework for increasing software development predictivity and quality, and developed a prototype that presents analysis that can help in software development decision making.

(Fonseca et al, 2017) proposed an ontology-based approach for measurement systems integration. The ontology helps organizations to integrate different systems and helps them to establish measurement process and tool support.

Another recent systematic literature review was performed by (Tahir et al, 2016) to answer questions about software measurement programs, considering measurement models, tools and practices, whether they are discussed in the literature and success and failure models.

(Bosch, 2017) presents challenges faced by companies which adopt data-driven decision making. The author explains the transition from a traditional decision-making company to a data oriented decision-making company.

Although the works above improves the state of art, none of them have proposed tools which use data from development processes, to extract implicit knowledge from these data, and at the end help managers to make decisions based on the processes, in direction to a data-oriented decision-making approach (Bosh, 2017).

5 FINAL REMARKS

Knowledge extraction from datasets is an activity that can be very complex because of the huge amount of data. Software managers frequently don't know models, tools or techniques that can support them to extract knowledge and consequently make better strategic decisions.

This paper proposes an architecture that has data provenance models, ontologies and complex networks, used together to help project managers to make better decisions.

To make an initial evaluation of the approach presented, real data from a medium size development company were used and from the analysis made, it was possible to extract strategic knowledge from data, presenting initial evidences of the architecture utility. However, a validation through a formal experiment is necessary, as well as architecture improvements, mainly considering its performance. Therefore, formal evaluations are being planned and this pilot evaluation presented in this paper serves as a first step to the architecture improvement.

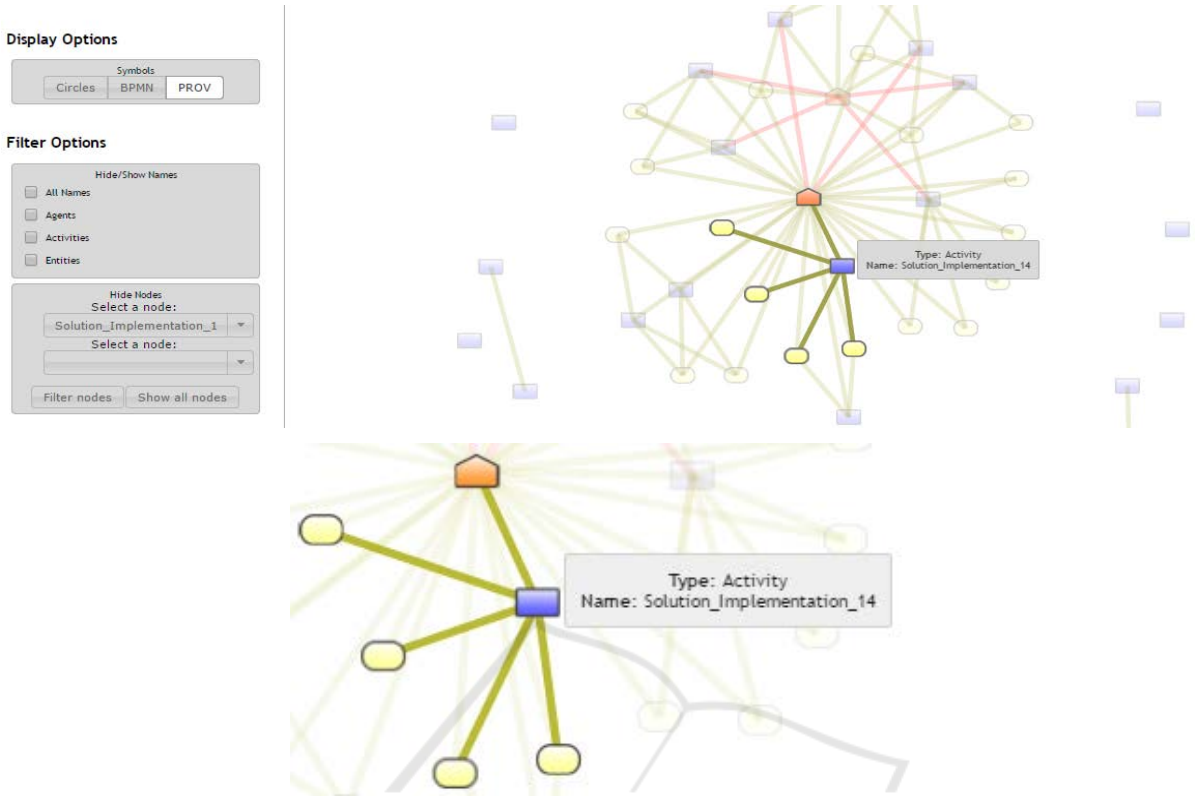

Figure 7: ERP process data visualization selection.

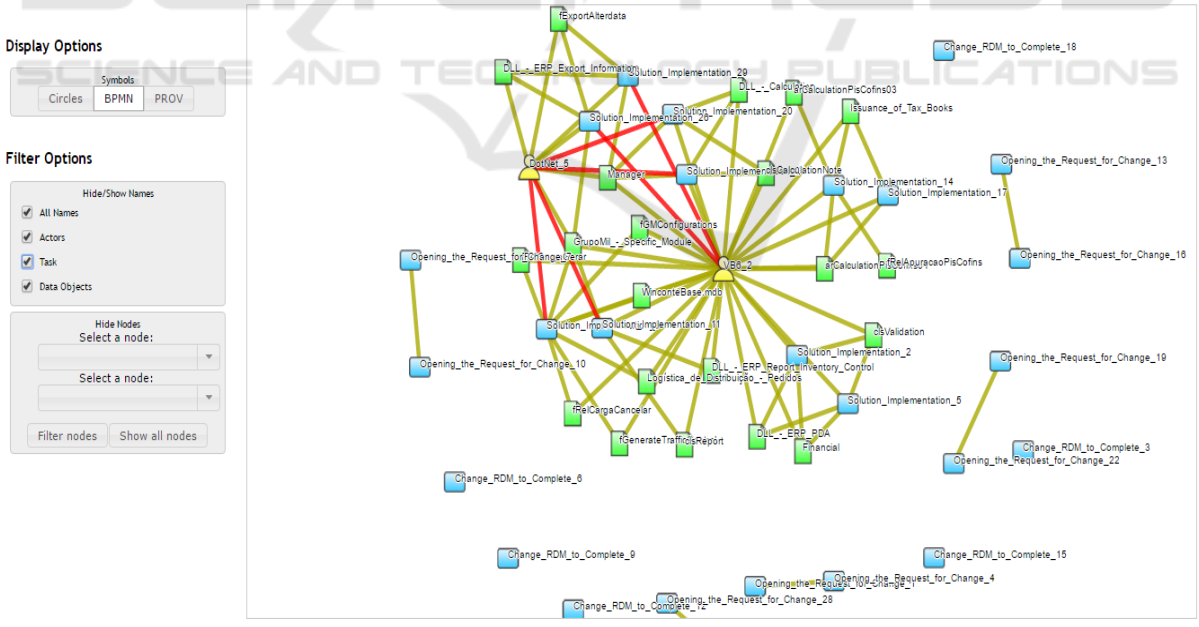

Figure 8: PROV Visualization.

REFERENCES

Acuna, S. T., De Antonio, A., Ferre, X., Lopez, M., Mate, L., and Estero, S. 2000. The Software Process: Modelling, Evaluation and Improvement. *Handbook of Software Engineering and Knowledge Engineering*, pp. 1–35.

Allard, S. 2012. DataONE: Facilitating eScience through collaboration. *Journal of eScience Librarianship*, *1*(1), 3.

Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., and Zhao, J. 2012. Prov-o: The prov ontology. *W3C Working Draft*.

Bosch, J. (2017). *Speed, Data, and Ecosystems: Excelling in a Software-Driven World*, CRC Press.

Buneman, P., Khanna, S., and Wang-Chiew, T. 2001. Why and where: A characterization of data provenance. In *International conference on database theory* (pp. 316-330). Springer Berlin Heidelberg.

Costa, G. C., Schots, M., Oliveira, W. E., LO, H., Dalpra, C. M., Braga, R., and Campos, F. (2016a). SPPV: Visualizing Software Process Provenance Data. *Sociedade Brasileira de Computação–SBC*, 49.

Costa, G. C. B., Werner, C. M., & Braga, R. (2016b). Software Process Performance Improvement Using Data Provenance and Ontology. In *International Conference on Business Process Management* (pp. 55-71). Springer International Publishing.

Cuevas-Vicenttín, V., Kianmajd, P., Ludäscher, B., Missier, P., Chirigati, F., Wei, Y., and Dey, S. 2014. The PBase scientific workflow provenance repository. *International Journal of Digital Curation*, *9*(2), 28-38.

Cuevas-Vicenttín, V., Ludäscher, B., Missier, P., Belhajjame, K., Chirigati, F., Wei, Y., and Altintas, I. 2015. *ProvONE: A PROV Extension Data Model for Scientific Workflow Provenance*. DOI = http://vcvcomputing.com/provone/provone.html

Dalpra, H. L., Costa, G. C. B., Sirqueira, T. F., Braga, R. M., Campos, F., Werner, C. M. L., & David, J. M. N. (2015). Using Ontology and Data Provenance to Improve Software Processes. In *ONTOBRAS*.

Fuggetta, A. 2000. Software process: a roadmap. *Proceedings of the Conference on the Future of Software Engineering. ACM.*

Groth, P., and Moreau, L. 2013. *PROV-Overview*. An overview of the PROV Family of Documents.

Harary, F. *Graph theory*. Addison. Reading, MA, 1969.

Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., and Plale, B. 2011. The open provenance model core specification (v1. 1). *Future generation computer systems*, *27*(6), 743-756.

Oliveira, W., Missier, P., Ocaña, K., de Oliveira, D., and Braganholo, V. 2016. Analyzing Provenance Across Heterogeneous Provenance Graphs. In *International Provenance and Annotation Workshop* (pp. 57-70). Springer International Publishing.

Prabhune, A., Zweig, A., Stotzka, R., Gertz, M., and Hesser, J. 2016. Prov2ONE: An Algorithm for Automatically Constructing ProvONE Provenance Graphs. In *International Provenance and Annotation Workshop* (pp. 204-208). Springer International Publishing.

APPENDIX

In order to give more information about ProvONEExt ontology, in appendix it is presented its Description Logics (DL) Syntax.

Classes	Object Properties	
Activity	CLtoDestP	outPortToDL
Agent	IrreflexiveObjectPropertyCLtoDestP	IrreflexiveObjectPropertyoutPortToDL
Collection	$\exists$ CLtoDestP Thing $\sqsubseteq$ SeqCtrlLink	$\exists$ outPortToDL Thing $\sqsubseteq$ OutputPort
Data	$\top \sqsubseteq \forall$ CLtoDestP Process	$\top \sqsubseteq \forall$ outPortToDL DataLink
Data $\sqsubseteq$ Entity	DLToInPort	requires
DataLink	IrreflexiveObjectPropertyDLToInPort	isRequired $\equiv$ requires$^-$
DataLink $\sqsubseteq$ Entity	$\exists$ DLToInPort Thing $\sqsubseteq$ DataLink	$\exists$ requires Thing $\sqsubseteq$ Process
Entity	$\top \sqsubseteq \forall$ DLToInPort InputPort	$\top \sqsubseteq \forall$ requires Data
InputPort	DLToOutPort	sourcePToCL
InputPort $\sqsubseteq$ Entity	IrreflexiveObjectPropertyDLToOutPort	IrreflexiveObjectPropertysourcePToCL
OutputPort	$\exists$ DLToOutPort Thing $\sqsubseteq$ DataLink	$\exists$ sourcePToCL Thing $\sqsubseteq$ Process
OutputPort $\sqsubseteq$ Entity	$\top \sqsubseteq \forall$ DLToOutPort OutputPort	$\top \sqsubseteq \forall$ sourcePToCL SeqCtrlLink
Plan	dataOnLink	takes
Process	IrreflexiveObjectPropertydataOnLink	wasAttributedTo $\equiv$ takes$^-$
Process $\sqsubseteq$ Plan	$\exists$ dataOnLink Thing $\sqsubseteq$ Data	used
Process $\sqsubseteq$ Entity	$\top \sqsubseteq \forall$ dataOnLink DataLink	used $\equiv$ wasUsedBy$^-$
ProcessExec	generated	IrreflexiveObjectPropertyused
ProcessExec $\sqsubseteq$ Activity	generated $\equiv$ wasGeneratedBy$^-$	$\exists$ used Thing $\sqsubseteq$ ProcessExec
SeqCtrlLink	hadMember	$\top \sqsubseteq \forall$ used Data
SeqCtrlLink $\sqsubseteq$ Entity	IrreflexiveObjectPropertyhadMember	wasAssociatedWith
User	$\exists$ hadMember Thing $\sqsubseteq$ Collection	isExecutedBy $\equiv$ wasAssociatedWith$^-$
User $\sqsubseteq$ Agent	$\top \sqsubseteq \forall$ hadMember Data	IrreflexiveObjectPropertywasAssociatedWith
Workflow	hasDefaultParam	$\exists$ wasAssociatedWith Thing $\sqsubseteq$ Activity
Workflow $\sqsubseteq$ Process	IrreflexiveObjectPropertyhasDefaultParam	$\top \sqsubseteq \forall$ wasAssociatedWith Process
	$\exists$ hasDefaultParam Thing $\sqsubseteq$ InputPort	wasAttributedTo
	$\top \sqsubseteq \forall$ hasDefaultParam Data	wasAttributedTo $\equiv$ takes$^-$
	hasInPort	IrreflexiveObjectPropertywasAttributedTo
	IrreflexiveObjectPropertyhasInPort	$\exists$ wasAttributedTo Thing $\sqsubseteq$ Process
	$\exists$ hasInPort Thing $\sqsubseteq$ Process	$\top \sqsubseteq \forall$ wasAttributedTo User
	$\top \sqsubseteq \forall$ hasInPort InputPort	wasDerivedFrom
	hasOutPort	$\exists$ wasDerivedFrom Thing $\sqsubseteq$ Data
	IrreflexiveObjectPropertyhasOutPort	$\exists$ wasDerivedFrom Thing $\sqsubseteq$ Workflow
	$\exists$ hasOutPort Thing $\sqsubseteq$ Process	$\exists$ wasDerivedFrom Thing $\sqsubseteq$ Process
	$\top \sqsubseteq \forall$ hasOutPort OutputPort	$\top \sqsubseteq \forall$ wasDerivedFrom Process
	hasSubProcess	$\top \sqsubseteq \forall$ wasDerivedFrom Workflow
	$\exists$ hasSubProcess Thing $\sqsubseteq$ Process	$\top \sqsubseteq \forall$ wasDerivedFrom Data
	$\top \sqsubseteq \forall$ hasSubProcess Process	wasGeneratedBy
	inPortToDL	generated $\equiv$ wasGeneratedBy$^-$
	IrreflexiveObjectPropertyinPortToDL	IrreflexiveObjectPropertywasGeneratedBy
	$\exists$ inPortToDL Thing $\sqsubseteq$ InputPort	$\exists$ wasGeneratedBy Thing $\sqsubseteq$ Data
	$\top \sqsubseteq \forall$ inPortToDL DataLink	$\top \sqsubseteq \forall$ wasGeneratedBy ProcessExec
	influenced	wasInfluencedBy
	influenced $\equiv$ wasInfluencedBy$^-$	influenced $\equiv$ wasInfluencedBy$^-$
	$\exists$ influenced Thing $\sqsubseteq$ User	$\exists$ wasInfluencedBy Thing $\sqsubseteq$ ProcessExec
	$\top \sqsubseteq \forall$ influenced ProcessExec	$\top \sqsubseteq \forall$ wasInfluencedBy User
	isExecutedBy	wasInformedBy
	isExecutedBy $\equiv$ wasAssociatedWith$^-$	$\exists$ wasInformedBy Thing $\sqsubseteq$ ProcessExec
	isPartOf	$\top \sqsubseteq \forall$ wasInformedBy ProcessExec
	$\exists$ isPartOf Thing $\sqsubseteq$ ProcessExec	wasUsedBy
	$\top \sqsubseteq \forall$ isPartOf ProcessExec	used $\equiv$ wasUsedBy$^-$
	isRequired	
	isRequired $\equiv$ requires$^-$	