**UNIVERSIDADE FEDERAL DE JUIZ DE FORA**
**INSTITUTO DE CIÊNCIAS EXATAS**
**PÓS GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO**

**Maria Luiza Furtuozo Falci**

# Melhoria de processos de software através da combinação de Proveniência de dados, Ontologias, Redes complexas e Visualizações

**Juiz de Fora**
**2018**

**Maria Luiza Furtuozo Falci**

# Melhoria de processos de software através da combinação de Proveniência de dados, Ontologias, Redes complexas e Visualizações

Dissertação apresentada ao Programa de Pós-graduação em Ciência da Computação, da Universidade Federal de Juiz de Fora como requisito parcial a obtenção do grau de Mestre em Ciência da Computação.

Orientadora: Regina Maria Maciel Braga.

**Juiz de Fora**
**2018**

Ficha catalográfica elaborada através do programa de geração
automática da Biblioteca Universitária da UFJF,
com os dados fornecidos pelo(a) autor(a)

Furtuozo Falci, Maria Luiza.
Melhoria de processos de software através da combinação de Proveniência de dados, Ontologias, Redes complexas e Visualizações / Maria Luiza Furtuozo Falci. -- 2018.
124 f. : il.

Orientadora: Regina Maria Maciel Braga
Dissertação (mestrado acadêmico) - Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Programa de Pós Graduação em Ciência da Computação, 2018.

1. Processos de Software. 2. Tomada de decisão. 3. Ontologia. 4. Rede Complexa. 5. Proveniência de dados. I. Maciel Braga, Regina Maria , orient. II. Título.

**Maria Luiza Furtuozo Falci**

**Melhoria de processos de software através da combinação de Proveniência de dados, Ontologias, Redes complexas e Visualizações**

Dissertação apresentada ao Programa de Pós-graduação em Ciência da Computação, da Universidade Federal de Juiz de Fora como requisito parcial a obtenção do grau de Mestre em Ciência da Computação.

Aprovada em 20 de setembro de 2018

BANCA EXAMINADORA

_____

Regina Maria Maciel Braga – Dsc. - Orientador

Universidade Federal de Juiz de Fora

_____

Mário Antônio Ribeiro Dantas, Dsc.

Universidade Federal de Juiz de Fora

_____

Daniel Cardoso Moraes de Oliveira, Dsc.

Universidade Federal Fluminense

*Dedico este trabalho aos meus pais, por sempre acreditarem e investirem em mim e no meu futuro.*

## AGRADECIMENTOS

A Deus por ter me guiado por todo caminho,

Aos meus pais Rosa e Waldyr por todo apoio e amor incondicional,

Aos meus irmãos João Victor e Luís Gustavo por todo amor e paciência,

Ao Heleno por todo amor, paciência e incontável ajuda,

A todos os professores do PGCC pelos ensinamentos, apoio e encorajamento,

Em especial à Regina Braga, por ter me orientado, por todo apoio e ajuda,

Aos meus professores da graduação do IF Sudeste MG, principalmente Marco Antônio e Filippe Jabour que me incentivaram a seguir para o mestrado,

À Suzi por toda ajuda,

Às minhas amigas da computação Camila, Laura, Lili e Tamillys por todo apoio e amizade,

Às minhas amigas Bianca e Mariana pela amizade e paciência,

Aos meus amigos do NEnc Márcio, Lenita, Iuri, Pedro, Vitor e Weiner, por toda ajuda e suporte,

Aos outros amigos que estiveram ao meu lado, me motivando sempre.

“Aqueles que não conseguem lembrar o passado estão condenados a repeti-lo”

(George Santayana)

## RESUMO

O processo de desenvolvimento de *software* é uma atividade complexa, que é influenciada por diferentes fatores, e pode ser surpreendida por um comportamento inesperado do *software*. Devido a sua importância cada vez maior nos dias de hoje, a necessidade de melhoria na qualidade do *software* e seus processos é de extrema importância. Uma forma de melhorar processos de *software* é através da análise de dados de execuções anteriores, dados estes que para serem coletados necessitam do controle e monitoramento dos processos. O presente trabalho propõe uma arquitetura que engloba modelos de proveniência de dados, ontologia e rede complexa, para modelar a proveniência na área de processos de desenvolvimento *software*, além de permitir a extração de conhecimento implícito nos dados. A arquitetura conta também com uma camada de visualização para dar suporte à compreensão do comportamento dos dados a gerentes de projetos, e dessa forma os mesmos possam tomar decisões orientadas a dados e melhorar futuras execuções. A arquitetura proposta foi avaliada através da utilização de dados reais e estudo com participação de um gerente de projetos.



Palavras-chave: Processos de *software*, Tomada de decisão, Ontologia, Rede Complexa, Proveniência de dados.

## ABSTRACT

Software development process is a complex activity, which is influenced by many factors and can be surprised by an unexpected software behavior. Software's importance has grown exponentially in the past few years, which makes software improvement extremely necessary, as it is present in many different aspects of daily life. Analyze data from previous executions may be a good tactic to deal with software unpredictability, and to record processes' data is necessary to implement software monitoring and control. The present work proposes an architecture that encompasses provenance data, ontology and complex network models to structure data provenance in software process' domain and allow implicit knowledge extraction. The architecture proposed has a visualization layer to support project managers' data comprehension, allowing them to have data-oriented decision making and improve future process executions. The proposed architecture was evaluated with real companies' data and through a study with a specialist participation.



Keywords: Software Processes, Decision Making, Ontology, Complex Network, Data Provenance.

## LISTA DE ILUSTRAÇÕES

## LISTA DE TABELAS

**SUMÁRIO**

# 1. INTRODUÇÃO

## 1.1. MOTIVAÇÃO

Desenvolvimento de *software* é uma atividade complexa, que depende de vários fatores, e pode ser surpreendida por um comportamento inesperado advindo do *software* (FUGGETTA e DI NITTO, 2014), (JONES e BONSIGNOUR, 2011). Este comportamento inesperado pode estar relacionado a diferentes aspectos, como a influência de fatores humanos, organizacionais ou até mesmo ambientais. Um exemplo de fator que poderia gerar um comportamento inesperado é a mudança de fuso horário. Muitas vezes o funcionamento do software depende do horário de um local, e essa mudança poderia acarretar em problemas inesperados.

O monitoramento de diferentes fases dos processos de desenvolvimento pode ser uma boa tática para lidar com a imprevisibilidade do *software*, melhorando também sua qualidade. Uma vez que o monitoramento ocorre e dados são coletados sobre suas operações, os mesmos podem ser utilizados para controlar e alterar o comportamento dos processos quando necessário (BOSCH, 2016). Como no exemplo da mudança de fuso horário, caso fossem coletados dados sobre módulos do software que são influenciados pelo horário local, os mesmos poderiam ser utilizados para encontrar a fonte de possíveis erros. Além disso, dados sobre erros que foram corrigidos podem ser utilizados para melhorar futuras execuções.

Uma forma de utilizar estes dados é extraindo conhecimento sobre os mesmos, o que pode ser realizado, por exemplo, através de ontologias, que na área de ciência da computação são definições formais de conceptualizações (GUARINO, 1998), e, portanto, permitem a modelagem do conhecimento em um contexto, além da utilização de máquinas de inferências para extrair conhecimento implícito sobre os dados. Os dados coletados podem também ser representados em diferentes formatos, como no formato de rede, e análises sobre os mesmos podem ser realizadas para compreender seu comportamento.

Processos de *software* podem ser caracterizados como um conjunto de atividades necessárias para desenvolver e implementar *software*, executado por um grupo de pessoas organizadas de acordo com uma estrutura organizacional e apoiada por ferramentas conceituais e técnicas (ACUNA et al., 2001). A utilização de *software* está presente em várias áreas do cotidiano, inclusive em áreas críticas onde erros podem causar grandes prejuízos e colocar a segurança de pessoas em risco, o que aumenta a necessidade de melhoria contínua nos processos, que influencia diretamente na qualidade do *software*.

Uma forma de monitorar processos de *software* de maneira adequada é através do controle dos processos, que é um fator crítico para desenvolver *software* com qualidade. Como o desenvolvimento de *software* é uma atividade complexa e imprevisível, não é possível estabelecer um modelo restrito de passos a serem seguidos, mas é muito importante garantir que os processos caminhem para o resultado desejado, e isso pode ser feito através do controle dos processos, onde o comportamento dos processos pode ser acompanhado e guiado ao longo dos acontecimentos (FUGGETTA e DI NITTO, 2014).

Para dar suporte aos processos de controle de *software*, o uso de ferramentas e técnicas adequadas é essencial. Processos de monitoramento e melhoria contínua são mecanismos importantes para serem implementados, e podem ocorrer através do conhecimento de informações relativas aos processos, considerando que, baseado nas informações de modelos de processos, assim como baseado em execuções de processos similares ou anteriores, é possível refinar novas execuções e prover qualidade e melhorias aos processos.

Portanto, com o monitoramento e análise dos modelos de execução de dados, é possível trazer conhecimento sobre execuções anteriores para melhorar novos processos. Uma forma de obter esse conhecimento é através do uso de técnicas de proveniência de dados. Proveniência de dados descreve a origem e histórico de uso de um dado (BUNEMAN et al., 2001). Ou seja, são metadados relevantes que podem prover um melhor entendimento sobre os dados, e disponibilizar informações sobre sua origem, quais processos o geraram, o que os influenciou, seu contexto e como foram produzidos, além do caminho que percorram até serem salvos.

## 1.2. PROBLEMA

Os processos de desenvolvimento de *software* são assessorados por várias ferramentas, incluindo ferramentas de gerência de configuração, que conseguem capturar e salvar dados sobre os processos. Entretanto, dados produzidos por diferentes ferramentas e gerenciados por diferentes setores usualmente são representados em diferentes formatos. Mesmo tendo inúmeras ferramentas de diferentes tipos disponíveis para dar suporte aos processos de *software*, ainda há uma carência de ferramentas voltadas para uso intensivo de dados que permitam gerentes de projetos possam tomar decisões baseadas em dados, e também não há opções de frameworks para integração e aquisição de informações em diferentes formatos (NAEDELE et al., 2014).

Apesar de muitas empresas guardarem dados sobre seus processos, muitas vezes eles não são utilizados para extração de conhecimento e suporte à tomada de decisão. Um dos motivos é a necessidade de conhecimento especializado para a mineração e interpretação dos dados. Decisões muitas vezes são tomadas baseadas na experiência dos funcionários mais experientes (ou mais bem pagos) e não baseadas em dados (BOSCH, 2017).

## 1.3. QUESTÕES DE PESQUISA

Com base na motivação e problema apresentados, a questão de pesquisa do presente trabalho, que será nomeada QP0, é apresentada a seguir:

***QP0: Como utilizar dados em diferentes formatos sobre processos de software, e seu conhecimento implícito, para melhorar a compreensão de gerentes de projetos sobre os processos, e melhorar futuras execuções?***

No presente trabalho é proposta uma arquitetura denominada OntoComplex, a qual engloba modelos de proveniência, ontologias, redes complexas e visualizações para extrair conhecimento sobre dados de processos de software e dar suporte à compreensão dos mesmos. Os modelos de ontologia e rede complexa são avaliados através de questões de competência.

A questão de pesquisa QP0 é subdividida em três questões que serão respondidas através da avaliação proposta no presente trabalho:

- (QP1). É possível responder às questões de competência através de buscas na rede complexa e ontologia?
- (QP2). A OntoComplex é capaz de extrair conhecimento implícito dos dados?
- (QP3). O conhecimento extraído é capaz de dar suporte ao entendimento sobre os dados através de visualizações?

Essas questões ajudam a responder a questão geral, pois através delas é possível avaliar se a arquitetura proposta no presente trabalho propõe uma solução viável de ***como utilizar dados em diferentes formatos sobre processos de software, e seu conhecimento implícito*** através de suas camadas, ***para melhorar a compreensão de gerentes de projetos sobre os processos,*** através de suas visualizações, ***e melhorar futuras execuções*** a partir da compreensão dos dados por parte do gerente.

## 1.4. JUSTIFICATIVAS

Considerando o contexto de processos de *software*, proveniência de dados pode trazer melhorias através de seus modelos e da análise de execuções anteriores. Para poder facilitar a captura e integração de dados de proveniência, existem alguns modelos como OPM (MOREAU et al., 2011) e PROV (GROTH e MOREAU, 2013). O modelo PROV é o modelo padrão indicado pela W3C[1] (The World Wide Web Consortium)**,** e portanto o mais utilizado na comunidade acadêmica, o qual conta com um conjunto de documentos para sua definição, como o Prov-DM (BELHAJJAME et al., 2013b), um modelo de dados, e o Prov-O (BELHAJJAME et al., 2013a), um modelo de ontologia baseado no modelo de dados proposto. Essas especificações, além de permitir a modelagem da proveniência, dão suporte à extração de conhecimento implícito nos dados.

O modelo PROV tem algumas extensões específicas, como o ProvONE (CUEVAS-VICENTTÍN et al., 2016), criado especificamente para representar o contexto de *workflows* científicos, dessa forma permitindo que diferentes tipos de dados possam ser traduzidos para o formato do modelo. Dessa forma, dados sobre *workflows* científicos podem ser utilizados para adquirir conhecimento e melhorar *workflows* semelhantes ou futuras execuções do mesmo.

O modelo ProvONE pode ser estendido para ser usado na área de processos de *software*, modelando assim o conhecimento da área. Além do mais, acreditamos que nesse contexto, é possível extrair conhecimento implícito de dados de proveniência, o que pode melhorar futuras execuções, como, por exemplo, para encontrar tarefas obsoletas, desenvolvedores sobrecarregados, entre outras informações relacionadas a execuções de processos. O modelo estendido pode também ser utilizado como base para uso dos dados em diferentes formatos como, por exemplo, redes complexas, as quais podem ser analisadas para extrair conhecimento do comportamento dos dados como uma rede. Os dados em diferentes formatos podem ser integrados através do modelo de proveniência e o conhecimento implícito sobre os dados pode ser extraído e disponibilizado através de visualizações, para melhorar o entendimento de gerentes de projetos sobre os dados.

[1] W3C é uma comunidade internacional que desenvolve modelos padrão abertos para garantir o crescimento a longo prazo da Web (https://www.w3.org/)

## 1.5. OBJETIVO

O presente trabalho tem como objetivo propor técnicas para fomentar o uso de dados como apoio a tomada de decisões estratégicas para melhoria dos processos de desenvolvimento de *software*. Para isso, essa dissertação utiliza modelos de proveniência de dados, aliados ao uso de análises que permitam a extração de conhecimento implícito com base nesses modelos, através da utilização de ontologias e redes complexas, além de visualizações dessas análises para disponibilizar o conhecimento extraído. O trabalho visa também avaliar essa abordagem.

Esses objetivos visam melhorar a compreensão sobre os dados através de visualizações, e assim dar suporte a melhorias em futuras execuções. Para atingir esse objetivo, foram definidas três etapas:

- Definição de modelos;
- Proposta de uma arquitetura para implementação dos modelos;
- Avaliação dos modelos e da arquitetura com base em dados reais de processos de *software*.

## 1.6. METODOLOGIA

A metodologia utilizada neste estudo foi realizada em 5 etapas principais, sendo elas: (i) revisão *quasi*-sistemática da literatura, (ii) proposta de modelos de proveniência, dados, ontologia e rede complexa, (iii) proposta de uma arquitetura para implementação dos modelos propostos e suporte à visualização, (iv) avaliação da ontologia e rede complexa através de respostas à questões de competência, (v) dois estudos para avaliar a viabilidade da arquitetura, e avaliação com especialista em gerência de projetos.

A revisão *quasi*-sistemática da literatura teve como objetivo buscar como a tomada de decisão orientada a dados tem sido desenvolvida na área de processos de *software*, e quais ferramentas ou técnicas foram propostas para dar suporte a essa abordagem. Além disso, a revisão teve como objetivo encontrar oportunidades de pesquisa na área a partir da identificação de carências de propostas.

Foi definida uma extensão do modelo de proveniência ProvONE para adaptação do mesmo ao contexto de processos de *software*. Além disso, foram definidos modelos de ontologia e rede complexa com base no modelo de proveniência proposto, os quais têm como objetivo extrair conhecimento implícito sobre os dados e analisá-los sobre uma diferente perspectiva, para conhecimento sobre os mesmos possa ser extraído.

A arquitetura proposta teve como objetivo implementar os modelos propostos, bem como disponibilizar o conhecimento extraído através de visualizações para dar suporte ao entendimento do comportamento dos dados para gerente de projetos. Dessa forma, os mesmos podem compreender o comportamento dos dados, tomar melhores decisões e melhorar futuras execuções de processos.

A avaliação realizada teve como objetivo avaliar a ontologia, rede complexa e arquitetura propostas. A primeira etapa contou com a resposta a questões de competência através da utilização da ontologia e rede complexa, para verificar a capacidade dessas ferramentas de extrair conhecimento sobre os dados e responder às perguntas sobre os mesmos. A segunda etapa foi dividida em três partes, cuja primeira contou com a avaliação da arquitetura através da utilização de dados reais de uma empresa de desenvolvimento de *software*, para encontrar pontos de melhoria na arquitetura. A segunda parte contou com a avaliação da arquitetura, também com dados reais, após as melhorias implementadas; e a terceira parte contou com a participação de um gerente de projetos para avaliar se a arquitetura foi capaz de dar suporte à compreensão dos dados através de suas ferramentas de extração e visualizações.

## 1.7. ORGANIZAÇÃO DO TEXTO

O trabalho está dividido em 5 capítulos, além desta introdução. O capítulo 2 apresenta uma revisão da literatura e trabalhos relacionados, enquanto o capítulo 3 apresenta os modelos propostos, que incluem uma extensão de modelo de proveniência, denominado PROVONE-Ext, e uma ontologia (PROVONEExt-O) e rede complexa (PROVONE-Complex) para serem utilizados no contexto de processos de *software*.

O capítulo 4 apresenta a arquitetura ONTOCOMPLEX, que utiliza os modelos propostos para extração de conhecimento. O capítulo 5 apresenta a avaliação dos modelos e da arquitetura, além de ameaças a validade. E por fim, o capítulo 6 apresenta considerações finais e trabalhos futuros.

# 2. FUNDAMENTAÇÃO TEÓRICA

## 2.1. INTRODUÇÃO

Neste capítulo serão apresentados conceitos que servem como fundamentação teórica para o presente trabalho, a saber, proveniência de dados, modelos de proveniência, ontologia e processos de *software*. Além disso, os principais trabalhos relacionados a proposta são discutidos e analisados.

## 2.2. PROCESSOS DE SOFTWARE

Assim como mencionado anteriormente, *software* tem se tornado cada dia mais presente e importante na sociedade atual, e tem influenciado vários setores do cotidiano (FUGGETTA e DI NITTO, 2014). Um dos componentes mais importantes para o desenvolvimento de *software* com qualidade, custos controlados e prazos respeitados é a modelagem e execução de processos de *software*, que são definidos por Fuggetta (2000) como: "um conjunto de políticas, estrutura organizacional, tecnologias, procedimentos e artefatos necessários para conceber, desenvolver, implantar e manter produtos de *software*".

De maneira mais informal, um processo de *software* pode ser definido também como um conjunto de atividades que podem produzir artefatos (produtos), e para isso consome recursos, adota procedimentos e utiliza artefatos (insumos) produzidos por outras atividades (GUIZZARDI et al., 2008). E ainda, uma atividade pode ser dividida em subatividades, e pode depender da realização de atividades anteriores para que possam ser iniciadas.

Recursos são utilizados para apoiar a execução de atividades, ou usados durante a execução. Os recursos podem ser recursos humanos (agentes humanos que desempenham as atividades), recursos de hardware (equipamentos utilizados nas atividades) e recursos de *software* (qualquer produto de *software* usado na atividade). Além disso, procedimentos podem ser adotados durante as atividades como, por exemplo, modelos (GUIZZARDI et al., 2008).

Neste contexto, Processos de *Software* se assemelham a *Workflows* Científicos. A Tabela 1 compara alguns conceitos semelhantes entre processos de *software* e *workflows* científicos, considerando a nomenclatura utilizada pelo modelo PROVONE, detalhado na seção 2.5.

Tabela 1 - Comparação de conceitos de *workflows* científicos e processos de *software*, considerando a notação utilizada no modelo PROVONE

| ***Workflows* científicos** | **Definição** | **Processos de *Software*** | **Definição** |
|---|---|---|---|
| ***Workflow*** | É composto por um conjunto de atividades, denominadas no modelo ProvONE[2] como *Programs* | **Processos de *software*** | Conjunto de atividades |
| **Program / Atividade** | Pode gerar artefatos | **Atividades** | Podem produzir artefatos |
| | Pode consumir artefatos | | Pode consumir artefatos |
| | Pode consumir artefatos produzidos por outros Programs (atividades) | | Utiliza artefatos produzidos por outras atividades |
| | Pode possuir subprograms (sub-atividades) | | Pode ser dividida em sub-atividades |
| | Pode depender de uma atividade anterior para ser iniciada, principalmente se depender de seus outputs | | Pode depender da finalização de atividades anteriores para serem iniciadas |
| | Podem ter parâmetros padrões para serem executados | | Podem consumir recursos, artefatos e procedimentos (como por exemplo modelos) |
| **Entity** | São utilizados para apoiar execuções de *Programs* (atividades) | **Recursos** | São utilizados para apoiar atividades |
| | Podem ser de diferentes categorias, como visualizações, documentos | | Podem ser de diferentes categorias, como *software* ou hardware |
| **User** | Execuções podem estar ligadas a usuários responsáveis por elas | | Podem precisar de apoio de recursos humanos (como gerentes de projetos, desenvolvedores) |

[2] O modelo ProvOne será detalhado na seção 2.5.

Devido às semelhanças entre os dois conceitos, o modelo ProvONE foi utilizado para ser estendido para se adaptar ao contexto de processos de *software*. A extensão do modelo é apresentada no capítulo 3.

## 2.3. ONTOLOGIA

Ontologia na área de ciência da computação é a definição formal de uma conceptualização (GUARINO, 1998). A definição de Guarino é uma definição clássica e ressalta os benefícios de se ter um conceito compartilhado, pois permite que diferentes indivíduos compartilhem o entendimento de um mesmo conceito. Portanto, ontologias facilitam a interoperabilidade de diferentes indivíduos, mesmo que estes tratem definições similares, mas com nomenclaturas distintas.

Outro ponto importante na definição de Guarino (1998) é a questão do formalismo da definição. Uma ontologia formal consiste em um conjunto de axiomas que descrevem conhecimento explícito na ontologia. A partir deste conhecimento explícito, podem-se criar novas regras que permitem a descoberta de novos relacionamentos entre os conceitos, permitindo a inferência de conhecimento implícito (REN et al., 2014). Essas inferências enriquecem o contexto porque possibilitam um maior entendimento sobre os conceitos, já que extraem conhecimento que não estava disponível no conjunto original dos dados.

Neste contexto de formalização dos modelos ontológicos, a linguagem OWL (Web Ontology Language)[3] foi desenvolvida (MCGUINNES e VAN HARMELEN, 2004). Através da utilização dessa linguagem, é possível escrever classes que representam os conceitos, as quais podem possuir indivíduos, relações explícitas entre essas classes e axiomas, que restringem o significado dos conceitos e relações. Esses axiomas permitem a extração de conhecimento e inferência de conhecimento implícito.

Para ilustrar a utilização de ontologias para inferência de conhecimento, supondo que uma ontologia sobre família fosse criada, onde dois indivíduos A e B de uma classe se relacionam através de uma relação que indica que os mesmos são irmãos, e que o indivíduo A tem uma relação com um indivíduo C que indica ser seu filho. A ontologia seria capaz, desde que tenha uma regra que a permita, inferir que o indivíduo B é tio do indivíduo C. Essa inferência é um exemplo de conhecimento implícito que a ontologia pode extrair.

[3] https://www.w3.org/OWL/

## 2.4. REDES COMPLEXAS

Uma rede pode ser definida, de forma geral, como uma codificação de relacionamento entre objetos (FIGUEIREDO, 2011). Ou seja, em uma rede diferentes indivíduos (representados por nós) se relacionam através de arestas. Essa definição se aplica em diversas áreas: a rede mundial de computadores, a Internet, redes neurais e sistemas biológicos são exemplos de sistemas que contém unidades dinâmicas interconectadas (BOCCALETTI et al., 2006). Além desses exemplos, redes sociais são conhecidas por conectarem pessoas através de relacionamentos virtuais, e as mesmas podem ter outros tipos de relacionamentos, como amorosos, de amizade ou profissionais.

As redes podem ser caracterizadas e modeladas para compreender sua estrutura e comportamento. Podem existir em uma rede diferentes tipos de nós. Um exemplo que ilustra essa situação seria uma rede definida com o objetivo de estudar a saúde de indivíduos. Cada indivíduo representa um nó da rede, esses indivíduos podem se conectar por relacionamentos de diferentes tipos, como relacionamentos familiares, amorosos ou de trabalho. Esses relacionamentos podem ter diferentes pesos de acordo com a proximidade e convivência dos indivíduos, que podem criar pesos para a as relações de acordo com uma fórmula, a qual depende de diferentes fatores, como horas de convivência por ano.

Neste exemplo, quanto mais relações um indivíduo possui na rede, maior é sua centralidade, ou seja, mais centralizado esse nó é representado na rede. Essa rede pode possuir outros tipos de nós, como, por exemplo, a cidade onde vivem, a empresa onde trabalham. Fenômenos podem ser estudados nessa rede, como as chances de uma doença se espalhar de acordo com as relações dos indivíduos, ou o impacto da remoção de um nó da rede. Dependendo da quantidade e dos tipos de relacionamento que um indivíduo possui, o impacto de sua remoção da rede, que poderia acontecer, por exemplo, por sua morte, esse impacto pode ser maior ou menor.

Outros exemplos de fenômenos que podem ser estudados são discutidos no exemplo de FIGUEIREDO (2011), onde um sociólogo interessado em estudar como boatos se espalham em uma rede de pessoas pode realizar análises como quais indivíduos mais favorecem o espalhamento, o tempo para o boato chegar a todos os indivíduos da rede, entre outras características. As redes também têm sido propostas em outras áreas, como na medicina, onde são propostas para análise de disseminação de doenças.

Na computação a definição de redes é equivalente à definição de grafos (FIGUEIREDO, 2011). Os nós de uma rede são equivalentes aos vértices de um grafo, e as relações às arestas. A utilização de grafos transcendeu dos problemas rotas, fluxo e coloração

para a aplicação em bancos de dados. Identificando a necessidade de representação de dados conectados, e devido à limitação de representação dos mesmos em bancos de dados relacionais, principalmente em relação à quão devagar pode ser a busca por grande quantidade de dados conectados, foi identificada a oportunidade de representar esses dados em grafos (Robinson et al., 2013). Com essa nova abordagem, não só os dados são valorizados, mas também a relação entre eles. Desta maneira, os dados podem ser representados como nós de um grafo, ou de uma rede, e as relações entre os mesmos podem ser analisadas, bem como a relevância da participação dos nós na rede. Além disso, outras características podem ser avaliadas, como a centralidade de um nó na rede, os graus de entrada e saída, entre outras características. O *Facebook* é um exemplo de onde não só as pessoas (e seus dados) são importantes, mas também o relacionamento entre elas (Robinson et al., 2013). A utilização desse formato de dados tem crescido exponencialmente, e vem sendo cada vez mais abordada para diferentes análises.

## 2.5. PROVENIÊNCIA DE DADOS

Proveniência de dados é a descrição do histórico de um dado desde sua criação até sua gravação em um banco, bem como os processos que o influenciaram (BUNEMAN et al., 2001). A proveniência de dados é muito importante na verificação da integridade de um dado, permitindo um melhor entendimento de como esse dado foi transformado. É importante principalmente quando há a necessidade de entender o que o influenciou, como, e qual sua origem. Essa necessidade pode vir para entender erros ou comportamentos estranhos do dado, mas também pode vir para extrair conhecimento e melhorar processos futuros.

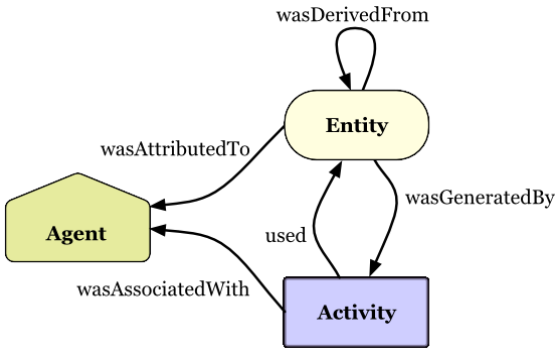
Para que a proveniência seja capturada, os processos que utilizam os dados precisam ser monitorados. Alguns modelos de proveniência foram desenvolvidos com o objetivo de estruturar a proveniência em diferentes contextos. Esses modelos apresentam classes e relações causais entre as classes, pelas quais os dados podem ser influenciados. Podemos citar como exemplo de modelos de proveniência o modelo OPM (MOREAU et al., 2011) e o modelo PROV (BELHAJJAME et al., 2012), detalhado mais a frente.

Podemos dividir os tipos de proveniência em dois tipos: retrospectiva e prospectiva. A proveniência retrospectiva trata dos processos que influenciaram um dado até o presente momento, e a prospectiva trata dos modelos utilizados para se gerar a proveniência. Por exemplo, em um contexto de processos de *software*, a proveniência retrospectiva está relacionada aos dados e processos da execução do processo de *software*. Já a proveniência prospectiva está relacionada ao modelo utilizado para se gerar a execução.

### 2.5.1. PROV

O PROV (BELHAJJAME et al., 2012) é um modelo de proveniência padrão, utilizando para capturar proveniência de forma integrada, recomendado pelo W3C[4]. O PROV expressa registros de proveniência que contém descrições de agentes e atividades que influenciam um objeto (entidade) ou estão envolvidas em sua produção e entrega. Uma característica a ser destacada no modelo são suas relações causais, que tentam abranger diversas perspectivas considerando proveniência de objetos digitais. A Figura 1 apresenta as principais construções do modelo e as Tabelas 2 e 3 apresentam suas principais classes e relações causais.

Figura 1 - Modelo de proveniência PROV




Fonte: (BELHAJJAME et al., 2012)

Tabela 2 - Classes do modelo PROV

| | |
|---|---|
| **Entity** | A classe "Entity" é utilizada para representar qualquer tipo de entidade, como por exemplo, física, conceitual ou digital. |
| **Agent** | A classe "Agent" se refere aos agentes que podem influenciar na proveniência representada, como por exemplo, um *software* que realiza uma ação, ou um usuário. |
| **Activity** | A classe "Activity" se relaciona às atividades, como por exemplo, processos ou ações. |

Tabela 3 - Relações do modelo PROV

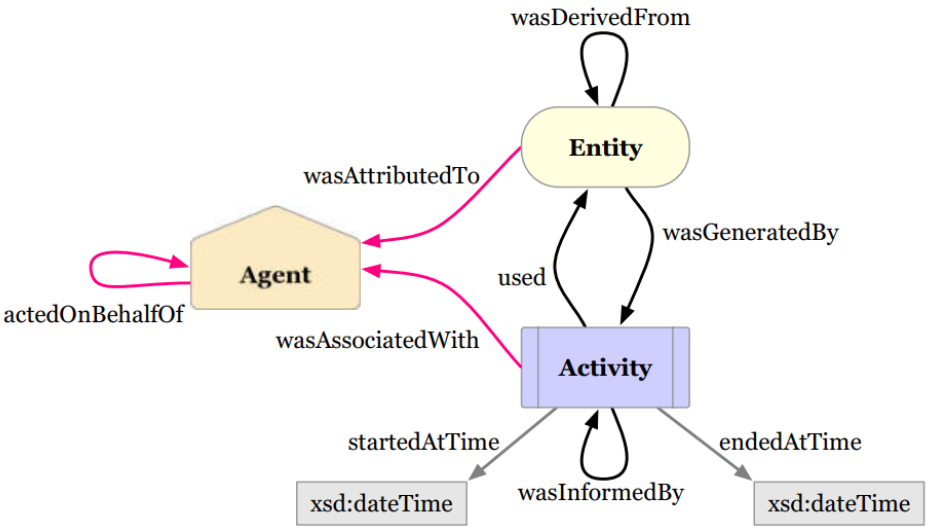
| | |
|---|---|
| **wasAttributedTo** | Relaciona um agente a uma entidade a qual ele foi atribuído |
| **wasAssociatedWith** | Relaciona um agente a uma atividade da qual ele tem responsabilidade sobre |
| **used** | Relaciona uma entidade a uma atividade que a usou |
| **wasGeneratedBy** | Relaciona uma atividade a uma entidade pela qual foi gerada |
| **wasDerivedFrom** | Relaciona uma entidade a outra pela qual a mesma foi gerada |

[4] https://www.w3.org/Consortium/

O modelo PROV conta com um conjunto de modelos para sua definição. Dentre estes, destaca-se o modelo PROV-O, que é uma ontologia associada às construções básicas do modelo. A partir da PROV-O, regras de completude e inferência foram especificadas, provendo construções lógicas passíveis de serem usadas e estendidas para possivelmente gerar conhecimento implícito.

A PROV-O expressa o modelo de dados do PROV (PROV-DM) utilizando a linguagem OWL2. Essa ontologia propõe um conjunto de classes, propriedades e restrições que podem ser usadas para representar o intercâmbio de informações de proveniência gerado em diferentes sistemas e sob diferentes contextos. Assim como o modelo PROV, a ontologia PROV-O pode ser estendida para ser especializada. Essa extensão é feita através da criação de novas classes e propriedades de diferentes aplicações e domínios. A Figura 2 apresenta as classes e propriedades da ontologia PROV-O.

Figura 2 - Ontologia PROV-O




Fonte: (BELHAJJAME et al., 2013a)

Conforme já dito, é importante destacar que o modelo PROV é genérico e pode ser estendido para diferentes contextos. Uma das suas extensões é o ProvONE (CUEVAS-VICENTTÍN et al., 2016), que visa representar a proveniência de dados no contexto de *workflows* científicos e será detalhado na seção seguinte.

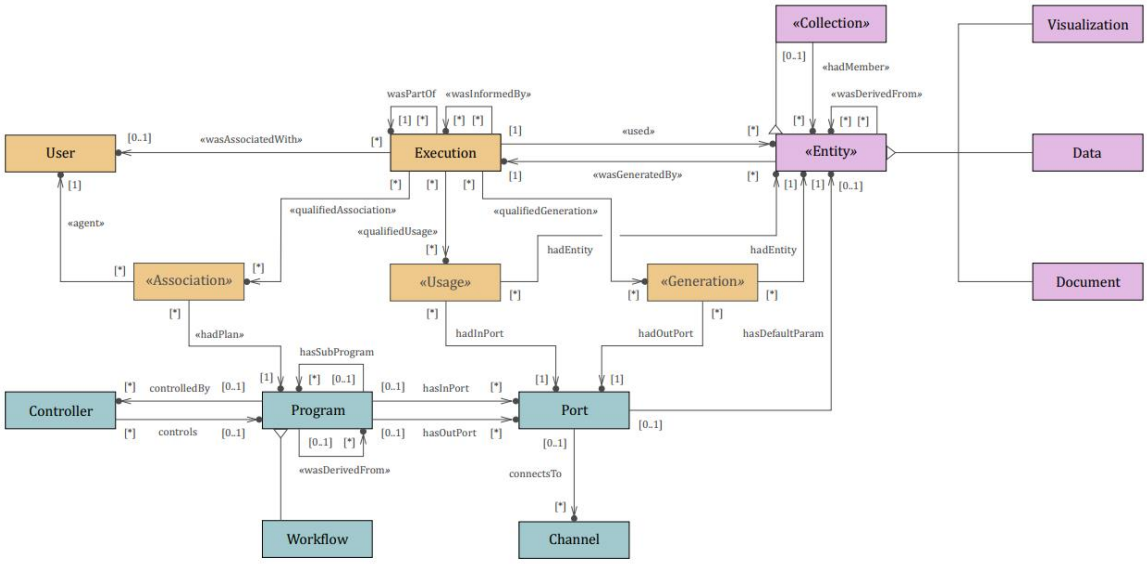
## 2.6. PROVONE

ProvONE (CUEVAS-VICENTTÍN et al., 2016) é um modelo de proveniência de dados, uma extensão do modelo PROV para ser aplicada ao contexto de *Workflows* científicos. Nesse contexto muitas vezes traços de execuções são coletados e armazenados

através de sistemas de gerência de *workflows*. Devido à existência de diversos sistemas que utilizam diferentes formatos para armazenar os dados, existe uma dificuldade de interoperabilidade e compartilhamento de dados vindos de diferentes sistemas de gerenciamentos de *workflows*.

Um mesmo *workflow*, ou *workflows* semelhantes, caso gerenciados por ferramentas distintas, são representados em formatos diferentes. Isso impossibilita que cientistas possam analisar e comparar os dados de formatos diferentes, mesmo que sejam referentes ao mesmo *workflow*. O modelo ProvONE foi proposto com o intuito de criar um modelo canônico para o contexto de *workflows* científicos, para que a proveniência dos dados possa ser representada em formato comum. A Figura 3 ilustra apresenta o modelo ProvONE.

Figura 3 - Modelo de proveniência ProvONE




Fonte: (CUEVAS-VICENTTÍN et al., 2016)

O modelo divide suas classes e relações em três aspectos: *Workflow*, traços de execução (Trace) e estrutura de dados (Data Structure). As informações sobre cada uma dessas categorias sãos apresentada a seguir de acordo com o modelo ProvONE apresenta em (CUEVAS-VICENTTÍN et al., 2016).

- *Workflow*: A classe *Program* representa as várias atividades que formam um *Workflow*. *Programs* podem ser compostos por subprograms, e um *Program* específico pode ser definido como um *workflow*. Um *Program* pode consumir artefatos através de *Input Ports*, ou gerar artefatos através de *Output Ports*. Essas *Ports* são conectadas com *Programs* através de *Channels*. O output de um *Program* pode servir como input de um outro *Program*. *Workflows* podem ter parâmetros padrões para serem executados, que são associados aos

*Programs* que o constituem. Esses parâmetros são descritos através da classe *Entity*.

- Trace: Os traços de execução associados a um *Workflow* são representados no ProvONE através da classe *Execution*. Cada instância de *Execution* representa um *Program* (e seu plano de execução, que pode ser um *workflow*, ou o passo de atividades para concluir um *Program*). Essas instâncias podem ser associadas a um usuário responsável pela execução. Para a execução de um *Program* uma série de inputs da classe *Entity* é lida através das portas de Input e são usados para gerar outputs também da classe *Entity* enviados pelas portas de output. Esses outputs podem ser dados, visualizações, documentos e itens, dependendo do objetivo do *workflow*. Quando um output é gerado ou um input consumido, são gravados através de propriedades do modelo.
- Data structure: As várias entidades associadas às instâncias do *workflow* e traços são representadas pela classe de *Data*, *Visualization* e *Document*. As classes de dados são definidas para serem genéricas e representam itens de dados de vários tipos (por ex CSV, JSON, XML, etc). A classe *Visualization* é definida para representar os outputs de *workflows*, que podem ser por ex (JPG, PNG, MP4, etc). A classe *Document* representa documentos publicados e não publicados que foram criados como resultado de uma *Execution* de *Program* ou *Workflow*. Essa classe não representa todos os dados, mas outros tipos podem ser representados através da classe *Collection*.
- Representação da evolução de um *Workflow*: o modelo não cobre representações específicas de diferentes ferramentas de gerenciamento de *workflows*, mas as características específicas podem ser representadas através da relação *wasDerivedFrom* do PROV.

A especificação de cada classe e relação é detalhada na Tabela 4, adaptada do documento original do ProvONE (CUEVAS-VICENTTÍN et al., 2016).

Tabela 4 - Classes e Relações do modelo ProvONE

| **Construções do ProvONE** | **Tipo** | **Nome** | **Especificação** |
|---|---|---|---|
| ***Workflow*** | **Class** | Program | Representa uma tarefa computacional |
| | | Port | Permite que um Program consiga enviar ou receber |

| | | | |
|---|---|---|---|
| | | | itens da classe Entity |
| | | Channel | Permite a conexão de um “Port” definido para um “Program” |
| | | Controller | Especifica um Program que controla outros Programs em um modelo computacional |
| | | *Workflow* | Representa um experimento computacional |
| | **Association** | *hasSubProgram* | Especifica uma composição recursiva de Programs |
| | | *controlledBy* | Relaciona um Controller a seu Program de origem |
| | | *controls* | Relaciona um Controller a seu Program destino |
| | | *hasInPort* | Especifica uma Port de um Program, a qual é utilizada para receber Inputs |
| | | *hasOutPort* | Especifica uma Port de um Program, a qual é utilizada para enviar Outputs |
| | | *hasDefaultParam* | Especifica que uma determinada input Port tem um indivíduo específico de Entity como parâmeto padrão |
| | | *connectsTo* | Especifica um Channel que uma Port se conecta |
| | | *wasDerivedFrom* | Descreve a evolução de Programs e *Workflows* |
| **Trace** | **Class** | Execution | Representa a execução de um Program. Caso o Program seja um *Workflow*, a execução representa um traço dessa execução |
| | | Association | Assinala a responsabilidade de um Agent sobre uma atividade |
| | | Usage | Representa a utilização de uma Entity por uma Activity |
| | | Genertion | Representa a geração de uma Entity por uma atividade |
| | | User | Representa o usuário responsável pela execução de uma Execution |
| | **Association** | *used* | Assinala que uma Execution usou uma entidade como input para sua execução |
| | | *wasGeneratedBy* | Indica que uma Execution produziu uma Entity como produto de sua execução |

| | | | |
|---|---|---|---|
| | | *wasAssociatedWith* | Indica que um usuário foi associado a uma execução específica |
| | | *wasInformedBy* | Indica que uma Execution se comunica com outra Execution através das portas input-output e desencadeia sua execução |
| | | *wasPartOf* | Indica que uma Execution possui uma determinada sub execução |
| | | *qualifiedAssociation* | Assinala que um Agent tem responsabilidade sobre uma Activity |
| | | *agent* | Indica um Agent que influencia um recurso |
| | | *hadPlan* | Indica a adoção de um Program por um Agent |
| | | *qualifiedUsage* | Indica quando uma Execution usa uma Entity |
| | | *hadInPort* | Indica a Port de uma Execution específica foi utilizada como Input Port |
| | | *hadEntity* | Indica uma Entity que foi utilizada por uma Execution em particular |
| | | *qualifiedGeneration* | Inica que uma Execution gerou uma Entity como output de seu processo |
| | | hadOutPort | Especifica que uma Port de uma Execution específica foi utilizada como Output Port |
| **Data Structure** | **Class** | Entity | Representa instâncias de dados, visualizações, documentos, ou de outras classes do tipo Entity. |
| | | Collection | Indica coleções de entidades. |
| | | Data | Representa a unidade básica de informação consumida ou produzida por um Program |
| | | Visualization | Representa a unidade de informação consumida ou produzida por um program em forma de representação visual |
| | | Document | Representa o corpo de informação produzido como resultado de uma Execution |
| | **Association** | *wasDerivedFrom* | Indica a dependência entre itens de Data produzidos |

| | | | |
|---|---|---|---|
| | | | durante a execução de um *workflow* |
| | | *hadMember* | Especifica indivíduos de Data que formam parte de uma Collection |

O modelo de proveniência ProvONE foi utilizado como base para o desenvolvimento de um modelo de proveniência para o contexto de processos de desenvolvimento de *software*. Este modelo será apresentado no capítulo 3.

## 2.7. REVISÃO QUASI-SISTEMÁTICA DA LITERATURA

Levando em consideração o contexto da pesquisa atual, foi realizada uma revisão *quasi*-sistemática da literatura para buscar artigos sobre a utilização de dados de processos de *software* para tomada de decisão estratégica, ou seja, sobre a tomada de decisão orientada a dados no contexto de processos de *software*. Uma Revisão *quasi*-Sistemática da literatura é definida por (ABRANTES e TRAVASSOS, 2007) como um estudo secundário, mas ainda assim sistemático, onde o objetivo é a caracterização da área, porém não há comparação e não é possível aplicação de meta-análise.

### 2.7.1. Objetivo

Bosch (2017) apresenta, entre outras coisas, as dificuldades de se utilizar dados para a tomada de decisão. O autor fala que apesar de muitas empresas ainda não tomarem decisões baseadas em dados sobre seus processos, essa abordagem já vem sendo implementada em algumas companhias. O objetivo inicial é investigar, através da revisão, informações sobre como tomadas de decisão orientada a dados tem sido utilizadas para melhorar processos de *software*, e quais abordagens e ferramentas foram propostas para dar suporte às mesmas.

### 2.7.2. Formulação da Pergunta

As perguntas formuladas têm como objetivo guiar a identificação de estudos relevantes, e extração de dados sobre os mesmos. Duas perguntas foram formuladas no presente estudo para investigar a utilização de tomada de decisão orientada a dados na área de processos de *software*.

**Q1: Como a tomada de decisão orientada a dados tem sido utilizada para melhorar processos de *software*?**

A pergunta Q1 é mais geral, e tem como objetivo investigar o cenário sobre tomadas de decisão orientada a dados, sobre como tem sido utilizadas para melhorar processos de *software*. Ou seja, se tem sido utilizada amplamente, se ainda tem encontrado resistência de empresas, se tem encontrado dificuldades para ser implantada, entre outras características que ilustrem como esse tipo de tomada de decisão tem sido aplicado.

**Q2: Quais abordagens, técnicas ou ferramentas foram propostas para dar suporte à tomada de decisão orientada a dados no contexto de processos de *software*?**

Essa pergunta é mais específica e tem como objetivo buscar quais ferramentas, técnicas ou abordagens foram propostas para dar suporte à tomada de decisão orientada a dados.

**Problema**: Encontrar como a abordagem de tomada de decisão baseada em dados vem sendo utilizada no contexto de processos de *software* para gerar melhorias, e quais ferramentas ou técnicas apoiam essa abordagem.

**Aplicação**: Investigar o cenário de tomadas de decisão baseada em dados, e até que ponto essa abordagem vem sendo aplicada, e como tem sido apoiada.

### 2.7.3. **Seleção de Fontes**

Foram selecionadas algumas fontes de busca dos trabalhos, as quais são amplamente utilizadas na área de ciência da computação, sendo elas:

Science Direct, Scopus, ACM Digital Library, IEEE Digital Library, ISI Web of Science.

### 2.7.4. **Palavras-chave**

Segundo (WOHLIN et al., 2012), as questões de pesquisa devem ser bem pensadas e devem levar em conta aspectos como: 1) População (population): para qual grupo de pessoas, programas ou negócios a revisão é de interesse; 2) Intervenção (intervention): qual tecnologia, ferramenta ou procedimento está sendo estudado; 3) Resultado (outcome): o que é melhorado através da intervenção. Além desses aspectos, são propostos a comparação, que assim como mencionado anteriormente não é o intuito da presente revisão, e o contexto, o qual define o contexto em que o estudo é definido, mas que no caso da presente revisão é restringido através da população.

Foram definidas palavras-chave para a pesquisa com base nesses aspectos. As palavras foram definidas em inglês, pois se deseja buscar por artigos neste idioma. As palavras-chave da pesquisa são apresentadas a seguir:

População (population): “Software project manager”, “Project owner”, “Software Engineer”, “Software Developer”.

Essas palavras chave foram selecionadas porque no contexto de processos de *software* esse é o grupo de pessoas as quais é de interesse a intervenção estudada.

Intervenção (intervention): “Data oriented decision making”, “Data driven decision making”, “Informed decision”, “Data oriented management”.

A intervenção estudada é a tomada de decisão orientada a dados, ou seja, a utilização de dados para apoiar a tomada de decisões.

Resultado (outcome): “Process Improvement”, “Software improvement”.

O resultado desejado é a melhoria de processos, e consequentemente a melhoria do *software*.

### 2.7.5. **Critérios de inclusão e Exclusão**

Alguns critérios de inclusão e exclusão foram definidos para aceitação ou rejeição dos artigos de acordo com suas características, e são apresentados a seguir:

Os critérios de inclusão têm como objetivo aceitar artigos que possam contribuir para a caracterização de tomadas de decisão orientada a dados, e a extração de informações sobre como essa abordagem vem sendo aplicada. São estes os critérios de Inclusão: Os documentos devem estar disponíveis na internet; Os documentos devem estar escritos em inglês; Publicações devem abordar o tema de tomada de decisão orientada a dados no contexto de processos de *software*; Falar sobre melhorias obtidas através de tomadas de decisão orientada a dados.

Os critérios de exclusão têm como objetivo excluir os artigos que não tragam conhecimento sobre a tomada de decisão orientada a dados no contexto de processos de *software*. Foram os seguintes os critérios de exclusão nesta pesquisa: comunicação de anais de conferências; Trabalhos em outro escopo que não seja na área de ciência da computação e engenharia de *software*; Não falar sobre tomada de decisão orientada a dados.

### 2.7.6. **Processo de Seleção dos Estudos**

O processo de seleção dos estudos passa por algumas etapas, são elas:

- Importação da bibtex[5] dos artigos, que são arquivos usados para descrever e processar listas de referências, e inclui títulos e resumos;
- Leitura de títulos e abstracts de cada uma das publicações para pré-seleção baseada nos mesmos;
- Leitura completa dos artigos aceitos na etapa anterior e aplicação dos critérios de inclusão e exclusão;
- Extração de informações referentes às questões formuladas sobre tomada de decisão orientada a dados.

### 2.7.7. **Estratégia de Extração de Informações e Sumarização dos Resultados**

Foram extraídas sobre cada um dos artigos selecionados as seguintes informações:

- Título
- Autores
- Ano de publicação
- Como tomada de decisão orientada a dados tem sido aplicada no contexto de processos de *software*
- Quais ferramentas, técnicas e abordagens foram criadas para apoiar tomadas de decisão orientadas a dados no contexto de processos de *software*

As informações extraídas de cada um dos artigos aceitos serão apresentadas em tabelas. Essas informações foram extraídas a partir de análises para identificar a ideia principal e as demais informações listadas acima.

### 2.7.8. **String de Busca**

Uma String de busca foi definida utilizando as palavras-chave do estudo para ser aplicada às bases selecionadas. A string é apresentada a seguir:

("Software project manager" OR "Project owner" OR "Software Engineer" OR "Software Developer") AND ("Data oriented decision making" OR "Data driven decision making" OR "Informed decision" OR "Data oriented management") AND ("Process Improvement" OR "Software improvement")

[5] http://www.bibtex.org/

### 2.7.9. **Execução de Buscas**

Buscas foram realizadas no dia 18 de agosto de 2017 nas bases selecionadas utilizando a string de busca apresentada. Foram necessárias algumas modificações para aplicá-la ao padrão especificado por cada base.

A biblioteca digital Science Direct retornou 80 artigos, e as demais não retornaram trabalhos. Desta forma, buscando investigar a validade dos resultados obtidos, foram realizadas modificações para ampliar o escopo da string. Observou-se que os novos resultados não incluíam trabalhos relevantes de acordo com o protocolo de pesquisa. Portanto, optou-se por manter a string original, mantendo as palavras-chave definidas.

As referências dos resultados foram manipuladas através da plataforma Parsifal[6], onde títulos e abstracts de todos os artigos foram lidos, e as publicações que respeitavam ao contexto selecionado foram aceitas para leitura completa.

Foram aceitos na primeira seleção 15 artigos para leitura completa, aos quais foram aplicados os critérios de inclusão e exclusão, e dos quais 4 foram aceitos para extração de informações.

### 2.7.10. **Extração de Informações**

Serão apresentadas informações extraídas de cada um dos artigos, nas tabelas 5, 6, 7, 8. A tabela 9 apresenta uma comparação do conteúdo extraído dos artigos de acordo com suas características.

Tabela 5 - Informações extraídas sobre artigo 1

| **Informação extraída** | **Informações sobre artigo 1** |
|---|---|
| Título | A decision support framework for metrics selection in goal-based measurement programs: GQM-DSFMS |
| Autores | Cigdem Gencel, Kai Petersen, Aftab Ahmad Mughal, Muhammad Imran Iqbal |
| Ano de publicação | 2013 |
| Como tomada de decisão orientada a dados tem sido aplicadas no contexto de processos de *software* | O artigo fala sobre a utilização de dados para embasar tomada de decisão. Segundo os autores existem inúmeras abordagens para apoiar empresas no planejamento de seus programas de medições (para selecionar |

[6] https://parsif.al/

| | |
|---|---|
| | dados que serão usados para tomada de decisão). Essas ferramentas apoiam a ligação entre a necessidade de medições e seus objetivos. Entretanto, sustentar e gerenciar as métricas selecionadas a longo prazo é um desafio por várias razões, que são apontadas no artigo. |
| Quais ferramentas, técnicas e abordagens foram criadas para apoiar tomadas de decisão orientadas a dados no contexto de processos de *software* | O trabalho propõe um framework para dar suporte à decisão de seleção de métricas com base na abordagem Goal Question Metric (GQM). Segundo os autores, o framework proposto dá suporte à tomada de decisão baseada em dados sobre quais métricas coletar considerando o orçamento disponível, focando os esforços nos objetivos e métricas mais importantes, principalmente em ambientes muito dinâmicos onde as necessidades mudam com frequência. |

Tabela 6 - Informações extraídas sobre artigo 2

| **Informação extraída** | **Informações sobre artigo 2** |
|---|---|
| Título | Manufacturing execution systems: A vision for managing *software* development |
| Autores | Martin Naedele, Hong-Mei Chen, Rick Kazman, Yuanfang Cai, Lu Xiao, Carlos V.A. Silva |
| Ano de publicação | 2014 |
| Como tomada de decisão orientada a dados tem sido aplicadas no contexto de processos de *software* | O trabalho aborda a coleta de métricas, que segundo os autores é o fator mais importante para o controle de processos. Entretanto, na prática dificilmente ferramentas para coleta automatizada de métricas é implantada. São apresentadas dez áreas funcionais onde dados podem ser coletados e utilizados para tomar decisões, e são apresentadas lacunas que as ferramentas de engenharia de *software* não dão suporte, as quais estão presentes nos seguintes aspectos: integração |

| | |
|---|---|
| | de uso de dados sobre pessoas; infraestrutura de integração e coleta de dados unificados; frameworks conceituais direcionados a ciclos de melhoria a partir de dados de desenvolvimento; suporte de projeção e simulação integrados. |
| Quais ferramentas, técnicas e abordagens foram criadas para apoiar tomadas de decisão orientadas a dados no contexto de processos de *software* | O trabalho propõe a adaptação de conhecimento da área de sistemas de execução de manufatura para a área de engenharia de *software*, o que aumentaria a previsibilidade do desenvolvimento de *software* através da integração de informações, que proveriam base para um suporte automatizado de ferramentas que melhorariam tomadas de decisão. O trabalho propõe um protótipo de sistema de suporte de decisão, que é utilizado para ilustrar a viabilidade de utilizar princípios da área de sistemas de execução de manufatura para a gerência de processos de desenvolvimento de *software*. |

Tabela 7 - Informações extraídas sobre artigo 3

| **Informação extraída** | **Informações sobre artigo 3** |
|---|---|
| Título | An ontology-based approach for integrating tools supporting the *software* measurement process |
| Autores | Vinícius Soares Fonseca, Monalessa Perini Barcellos, Ricardo de Almeida Falbo |
| Ano de publicação | 2016 |
| Como tomada de decisão orientada a dados tem sido aplicadas no contexto de processos de *software* | O trabalho aponta que diferentes ferramentas de coleta de métricas são propostas para diferentes processos de *software*, porém há uma grande dificuldade na integração entre as diferentes ferramentas. |
| Quais ferramentas, técnicas e abordagens foram criadas para apoiar tomadas de | O trabalho propõe uma abordagem baseada em ontologia para integração de sistemas de |

| | |
|---|---|
| decisão orientadas a dados no contexto de processos de *software* | medição, para integrar diferentes ferramentas com o objetivo de dar suporte ao processo de medição de *software*. |

Tabela 8 - Informações extraídas sobre artigo 4

| **Informação extraída** | **Informações sobre artigo 4** |
|---|---|
| Título | A systematic literature review on *software* measurement programs |
| Autores | Touseef Tahir, Ghulam Rasool, Cigdem Gencel |
| Ano de publicação | 2016 |
| Como tomada de decisão orientada a dados tem sido aplicadas no contexto de processos de *software* | Os autores falam da importância do *software* hoje em dia, e que medições são vitais para que a engenharia de *software* possa caracterizar, avaliar, prever e melhorar entidades de *software* como por exemplo seus processos. O artigo é uma revisão sistemática que encontrou 35 modelos de planejamento de medições e 11 ferramentas associadas, e a maioria deles foi feita com base em abordagens baseadas em objetivos. Os resultados do estudo apresentam que ainda é um desafio a implantação de programas de medições, e que muitas vezes falta sustentabilidade, objetivos claros e instrumentos de medição corretos, recursos, tempo e orçamento. |
| Quais ferramentas, técnicas e abordagens foram criadas para apoiar tomadas de decisão orientadas a dados no contexto de processos de *software* | A revisão que apresenta ferramentas utilizadas para implementação de programas de medições de *software*, assim como fatores de sucesso e fracasso dos mesmos. Entre eles estão presentes modelos de planejamento de medições propostos para serem utilizados como parte da iniciativa de melhorar processos de *software*. Esses modelos visam escolher as melhores métricas para serem analisadas com esse objetivo. |

Tabela 9 - Tabela comparativa entre os artigos aceitos na revisão

| Características | Artigo 1 | Artigo 2 | Artigo 3 | Artigo 4 |
|---|---|---|---|---|
| **Utilização de dados para tomar decisões** | Sim | Sim | Sim | Sim |
| **Extração de conhecimento implícito** | Não | Não, se fala sobre análises, mas não sobre extração de conhecimento implícito dos dados | Não, o foco da proposta é a integração de diferentes ferramentas, e não a extração de conhecimento | Não |
| **Ferramentas para apoiar a tomada de decisão orientada a dados** | Sim, ferramenta para apoiar tomada de decisão sobre seleção de métricas | Sim, um protótipo é proposto | Não, mas indiretamente a integração de diferentes ferramentas de coleta de dados pode dar suporte à tomada de decisão orientada a dados | Sim, apresenta alguns trabalhos que falam sobre o tema. Porém os mesmos apenas mencionam tomada de decisão sobre seleção de métricas. |
| **Apoio à visualização para facilitar a compreensão dos dados** | Não | Não, no trabalho não há uma preocupação em gerar visualizações para facilitar a compreensão dos dados | Sim, o trabalho apresenta um estudo de caso onde um recurso utilizado foi a geração de gráficos com as métricas coletadas | Sim, menciona algumas ferramentas disponíveis no mercado que apoiam visualizações. |
| **Utilização de modelos de proveniência dos dados** | Não | Não | Não | Não |
| **Melhoria de processos de software** | Não | Sim | Sim, a proposta pode dar suporte a melhoria de processos de software | Sim, cita alguns trabalhos que falam sobre modelos de planejamento de métricas com foco em melhorias de processos de software. |
| **Suporte à integração de dados em diferentes formatos para apoiar tomada de decisão** | Não | Não, fala sobre a utilização de dados de diferentes projetos, mas não em diferentes formatos | Sim | Não |

### 2.7.11. **Análise dos Resultados Obtidos**

Naedele et al. (2015) apresenta os desafios de processos de desenvolvimento de *software*, como a falta de previsibilidade a respeito de custos, tempo e qualidade, e sobre como esses desafios se assemelham aos enfrentados na área de sistemas de execução de manufatura (MESs). Os MESs são sistemas de suporte a decisão para gerentes da área de manufatura, e dão suporte à melhoria de processos através de métricas. Segundo os autores, esses sistemas são capazes de coletar, analisar, integrar e apresentar os dados gerados em produções industriais para que funcionários consigam entender melhor o comportamento dos processos a partir dos dados, e consigam reagir de forma mais rápida. O trabalho propõe um protótipo de sistema de suporte de decisão, que é utilizado para ilustrar a viabilidade de utilizar princípios da área de MESs para a gerência de processos de desenvolvimento de *software*. O trabalho não menciona modelos de proveniência, extração de conhecimento implícito, apenas propõe um protótipo genérico para utilização do conhecimento com base nos conhecimentos de MESs.

Tahir et al. (2016) fala sobre *Software* Measurement Programs (MPs) ou programas de medição de *software*, que são importantes ferramentas para entendimento, avaliação, gerência e melhoria de processos, produtos e recursos. O trabalho apresenta uma revisão sistemática da literatura que busca modelos de planejamento de MPs, além de ferramentas utilizadas para implementação e fatores de sucesso e fracasso. Segundo as descobertas da revisão, alguns modelos de planejamento de medições foram propostos com o intuito de melhorar processos de *software*. Os trabalhos mencionados na revisão de Tahir não foram encontrados pela revisão *quasi*-sistemática apresentada no presente trabalho pois possuem escopos diferentes, já que os trabalhos citados pelos autores têm o foco em modelos de MPs. Ainda assim, a revisão como um todo traz conhecimento sobre a área de tomada de decisão orientada a dados, e por isso foi aceita na revisão.

Gencel et al. (2013) propõe um framework para suporte a decisão de seleção de métricas com base na abordagem Goal Question Metric (GQM). O trabalho fala sobre os desafios da seleção de métricas e da necessidade da escolha de métricas apropriadas. Além disso, é salientada a importância da seleção de métricas e de sua utilização para tomar decisões baseadas em dados para melhor gerenciar processos de *software*. O framework proposto inclui a seleção de métricas baseada no GQM, e incorpora mecanismos de tomada de decisão. No trabalho é apresentado um estudo de caso que compara as escolhas de métricas a partir do framework proposto com as escolhas feitas por especialista. O foco do trabalho é na

tomada de decisão sobre as métricas a serem selecionadas. Não se fala sobre suporte à tomada de decisão sobre os processos de *software* através da utilização de métricas.

Fonseca et al. (2017) fala sobre a utilização de diversas ferramentas para dar suporte a diferentes processos. Apesar de muitas dessas ferramentas não terem como objetivo dar suporte a medições, elas geralmente capturam dados sobre os processos que dão suporte, dados esses que podem ser importantes para tomadas de decisão. Como muitas ferramentas diferentes costumam ser utilizadas para dar suporte a diferentes processos, elas coletam dados em formatos diferentes. Para que essas ferramentas possam dar suporte às medições de *software*, elas precisam ser integradas, e como utilizam modelos diferentes, isso pode ser um desafio. O trabalho propõe uma abordagem baseada em ontologia para integração de sistemas de medição. A integração de ferramentas tem como objetivo dar suporte ao processo de medição de *software*.

Os trabalhos analisados na revisão ajudaram a compreender melhor a área de tomada de decisão orientada a dados no contexto de processos de *software*. De acordo com as informações obtidas, ainda é um desafio a coleta de dados para serem utilizados como fonte de conhecimento para apoiar a tomada de decisão. Alguns autores salientaram que muitas vezes decisões ainda são tomadas sem o auxílio dos dados sobre processos devido à essas dificuldades, mas também devido à falta de ferramentas de suporte à extração de conhecimento, e à necessidade de conhecimento especializado para análise dos dados.

Todos os trabalhos analisados focam na coleta de métricas. Alguns afirmam que métricas podem ser coletadas através de ferramentas de gerência de configuração, mas não falam muito sobre como utilizar esses dados para dar suporte à tomada de decisão.

O presente trabalho propõe um modelo de proveniência na área de processos de *software*, onde as métricas a serem coletadas são representadas pelas classes do modelo. Além do modelo, são propostas ontologia e rede complexa, para extrair conhecimento implícito através de suas regras e inferências, e análise dos dados como uma rede complexa, ambas abordagens favorecem a extração de conhecimento dos dados. Além disso, propomos uma camada de visualização para apresentar o conhecimento extraído a gerentes de projetos, que podem assim analisar os dados, compreendê-los, e assim tomarem melhores decisões e melhorar execuções futuras.

A proposta visa preencher a lacuna sobre a integração dos dados, através do modelo de proveniência que modela a proveniência da área. Além disso, o modelo permite que as métricas a serem coletadas sejam pré-definidas de acordo com o contexto. O trabalho também aborda o desafio da utilização dos dados, mencionada por alguns autores devido à

dificuldade de compreensão e extração de informações relevantes sobre os dados, já que as tecnologias propostas visam extrair conhecimento e apresenta-lo aos gerentes, fazendo com que os mesmos não precisem ser especializados em buscas para adquirir conhecimento sobre os dados. Outras contribuições do trabalho serão mencionadas nos próximos capítulos.

## 2.8. OUTROS TRABALHOS RELACIONADOS

Além da revisão quasi-sistemática descrita na seção anterior, foi também realizada uma busca ad-hoc por trabalhos que utilizam o modelo PROVONE. Apresentamos a seguir os principais resultados.

Foi realizada uma pesquisa por trabalhos relacionados ao modelo de proveniência ProvONE com o intuito de verificar se existiam propostas que ligavam o uso do modelo a processos de *software*. Um dos trabalhos relacionados ao ProvONE introduz o DataONE, uma organização multi-institucional, multinacional e interdisciplinar, que conta com a colaboração de diversos pesquisadores que visam contribuir para a comunidade científica na área de ciência de dados (ALLARD, 2012). O grupo desenvolve uma cyber-infraestrutura e estrutura organizacional para dar suporte ao ciclo de vida de dados ambientais, biológicos e ecológicos. Além disso, o grupo propõe ferramentas para serem utilizadas por pesquisadores, educadores e os pesquisadores em geral, que contribuem com o suporte de descobrimento de dados, integração e análise (incluindo visualização).

Allard (2012) menciona também as dificuldades de lidar com um grande volume de dados disponíveis, e os desafios da utilização dos mesmos para pesquisas de diferentes contextos. O trabalho fala ainda sobre a importância da interoperabilidade e cooperação entre diferentes pesquisadores e áreas de pesquisa. Além disso, no texto são citados problemas relacionados a modelos de proveniência de dados, dificuldade do acesso aos repositórios de dados, a complexidade dos diferentes ambientes relacionados, além de outras dificuldades como confiabilidade nos dados, entre outros. O Modelo ProvONE é uma das contribuições para o grupo DataONE, e foi proposto para modelar a proveniência de *workflows* científicos e dar suporte à pesquisa (CUEVAS-VICENTTÍN et al., 2016).

Segundo Prabhune et al. (2016), modelar a proveniência de dados no ProvONE é uma tarefa árdua, já que não é realizada de forma automatizada. Para tanto, os autores propõem uma ferramenta chamada Prov2ONE que gera a proveniência prospectiva para *workflows* científicos definidos no BPEL4WS[7]. A partir de então o grafo de proveniência é

[7] http://www.daml.org/services/swsl/materials/bpel11.pdf

atualizado com a proveniência retrospectiva relevante, favorecendo a interoperabilidade e análise dos *workflows* e suas proveniências associadas.

Cuevas-Vicenttín (2014) propõe um repositório visando sua integração com o DataONE. A experiência dos autores demonstra o potencial adquirido por suportar vários tipos de consultas pela proveniência de dados. Ainda houve a preocupação de desenvolver um repositório equipado com uma visualização voltada aos usuários. São apresentado ainda desafios enfrentados pelo PBase, como a definição de um modelo de proveniência que seja compatível com os principais sistemas de *workflows* científico, caracterização de funcionalidades requeridas para consultas específicas e desenvolvimento de uma estrutura que permita usuários avaliarem as consultas eficientemente.

Oliveira et al. (2016) fala sobre os modelos e ferramentas que ajudam os pesquisadores a estudarem a proveniência de dados. Os autores citam diferentes modelos existentes e falam sobre a colaboração que ocorre na comunidade científica. Além disso, falam também sobre as dificuldades enfrentadas devido a vários modelos existentes de proveniência de dados, e a dificuldade de interoperabilidade que ocorre devido à utilização de diferentes modelos. Os autores então propõem uma padronização de dois modelos (SciCumulus e e-Science Central) a partir de uma integração utilizando o ProvONE. A intenção é fazer com que os diferentes *workflows*, que a princípio utilizavam modelos diferentes de proveniência de dados, possam ser padronizados para posterior comparação entre eles. É proposta uma forma de converter proveniências de dados de diferentes modelos para que possam ficar no padrão do modelo ProvONE.

## 2.9. RESUMO DO CAPÍTULO

No presente capítulo foram apresentados referenciais teóricos que servem como base para o presente trabalho, referentes a ontologias, proveniência de dados, modelos de proveniência e processos de *software*. Foram também apresentadas semelhanças entre processos de *software* e *workflows* científicos. Além disso, foi apresentada uma revisão *quasi*-sistemática, desenvolvida na área de tomada de decisão orientada a dados, no contexto de processos de *software*. A revisão teve com o objetivo responder questões de pesquisa sobre a área. Foram também apresentados outros trabalhos relacionados aos tópicos do presente trabalho.

O próximo capítulo apresenta os modelos propostos no presente trabalho, que posteriormente servirão como base para a proposta de uma arquitetura.

# 3. MODELO PROVONEEXT

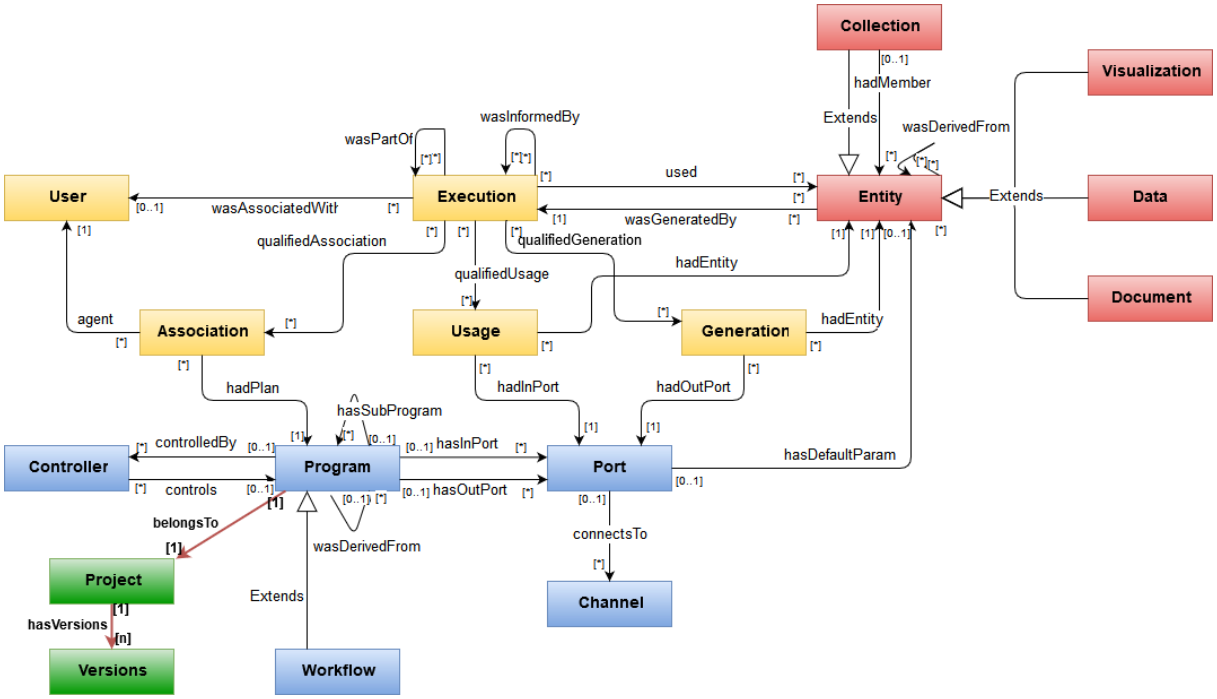
## 3.1. INTRODUÇÃO

Como mencionado no capítulo de fundamentação teórica, proveniência de dados descreve o histórico de um fragmento de dado, e os processos pelo qual ele passou (BUNEMAN et al., 2001). O ProvONE (CUEVAS-VICENTTÍN et al., 2016) é uma das extensões do modelo PROV, desenvolvida para o contexto de *workflows* científicos, como detalhado no capítulo 2, que permite a captura e processamento de dados de proveniência.

Para ser utilizando no contexto de *workflows* científicos, o modelo possui classes que são similares a entidades do domínio de Processo de Desenvolvimento de *Software* (PDS), que de certa forma se assemelha a *workflows* científicos, já que ambos possuem atividades a serem executadas, suas execuções, o produto (outputs) e requisitos (inputs) dessas execuções, além de usuários responsáveis por elas. Devido a essa semelhança, o modelo ProvONE foi selecionado para ser utilizado e adaptado para o contexto de PDS, domínio estudado no contexto desta dissertação.

Para que pudesse ser adaptado, foram adicionadas ao modelo novas classes, propriedades e regras específicas, detalhadas nas próximas subseções. A Figura 4 apresenta algumas das principais classes e propriedades dessa extensão, denominada ProvONEExt.

Figura 4 - ProvONEExt, extensão do modelo ProvONE



Algumas classes do modelo original não foram utilizadas no contexto de processos de software, como por exemplo *Channel, Workflow* e *Collection.* Essas classes foram mantidas na representação do modelo estendido para não descaracterizar o modelo original, e caso futuramente haja necessidade de utilizar as mesmas.

Além das classes originais do ProvONE, que foram adaptadas para o contexto de PDS, foram adicionadas duas novas classes ao modelo, as classes *Project* e *Versions*. A classe *Project* representa os diferentes projetos sendo desenvolvidos em uma empresa, onde cada projeto pode ter diferentes *Programs*, que representam processos de desenvolvimento de *software*. E cada *Program* pode possuir diferentes *Executions*, que representam execuções de *Programs*, ou seja, o trabalho executado para desenvolver o que um *Program*. Além disso, foi adicionada ao modelo a classe *Versions*, que representa as diversas versões de um projeto. O relacionamento entre *Project* e *Versions* é denominado *hasVersion*, e entre *Program* e *Project* existe a relação *belongsTo*. O modelo proposto foi validado como compatível com o modelo Prov através da ferramenta ProvValidator[8] (MOREAU et al., 2014).

Uma das características importantes em PDS é a tomada de decisão por parte dos gerentes de projeto em relação à execução do processo de desenvolvimento, com o objetivo de otimizar custos, prazos e melhoria da qualidade. Neste contexto, o uso de informações estratégicas para a tomada de decisão é importante. Assim, a descoberta de informações implícitas nos dados que possam levar a melhorias no processo torna-se fundamental. O uso de ontologias e o processamento de dados através de inferências, além da análise de informações estruturais ligadas a conectividade do modelo, são alguns dos caminhos para a descoberta de informações implícitas.

A extensão do modelo ProvONE pode ser associada a uma ontologia, onde classes e relações são representadas por entidades e propriedades da ontologia, respectivamente. A ontologia permite uma melhor representação da proveniência de dados no contexto de PDS, auxiliando principalmente na extração de conhecimento implícito a partir de suas regras e inferências. Outra vantagem da utilização de ontologias está na criação de axiomas (property chains), que potencializam a gerência de conhecimento através da criação de novas relações, que podem ser inferidas através da utilização de máquinas de inferência.

Por outro lado, esse modelo também pode ser interpretado como uma rede, onde entidades são associadas aos nós e propriedades às arestas, que podem ser direcionadas e

[8] https://openprovenance.org/services/view/validator
Versão utilizada para validação: Validator 0.7.3-SNAPSHOT, ProvToolbox 0.7.4-SNAPSHOT, prov-service-0.7.3-SNAPSHOT-docker

possuir pesos. A partir dessa associação, a rede pode ser instanciada com dados reais sobre processos de desenvolvimento de *software*, e ser analisada para compreender o comportamento dos dados na rede. Alguns exemplos de análises que podem ser desempenhadas são relativas a centralidade dos nós, se ele possui muitas conexões comparado a outros nós, qual o impacto da remoção desse nó, entre outras.

Desta forma, o modelo proposto na Figura 4 foi estendido para a especificação de uma ontologia conjugada a uma rede complexa, o que permite uma representação semântica dos processos, melhorando a compreensão de seu comportamento, o que pode prover informações estratégicas e suporte à tomada de decisão. Além disso, uma arquitetura foi proposta como parte do framework de gerenciamento de projetos, proposto em (COSTA et al., 2016b) e (DALPRA et al., 2015), com a intenção de extrair conhecimento estratégico e melhorar futuras execuções de processos e ajudar na tomada de decisão. Apresentamos a seguir o detalhamento da ontologia ProvOneExt e da rede complexa gerada.

## 3.2. ONTOLOGIA PROVONEEXT-O

Como mencionado anteriormente, uma ontologia foi criada baseada no modelo ProvONEExt, com o objetivo de se extrair conhecimento implícito, para auxiliar na tomada de decisão. O principal objetivo da ontologia é modelar semanticamente uma área de conhecimento, e criar regras que possam extrair conhecimento novo. Além das classes e propriedades do modelo, foram adicionadas cadeias de relações (*property chains*), que permitem a extração de conhecimento na ontologia uma vez que este conhecimento não estava disponível diretamente no modelo inicial.

Parte importante na modelagem de uma ontologia em uma área de conhecimento é a definição de questões a serem respondidas pela ontologia. Estas questões são chamadas questões de competência (REN et al., 2014) e devem ser explicitadas na fase de análise de viabilidade da ontologia.

A ontologia desenvolvida no contexto deste trabalho tem como principal objetivo a extração de conhecimento implícito para melhoria do processo de decisão de gerentes de projetos de desenvolvimento de *software*. Desta forma, a questões de competência explicitadas devem ser diretamente relacionadas a este objetivo. A subseção seguinte detalha estas questões no contexto da ontologia ProvOneExt-O.

### 3.2.1. Questões de Competência

Entrevistas com gerentes de projetos foram realizadas, além de pesquisas sobre o contexto de gerência de projetos, para detalhar que tipos de questões seriam relevantes para dar base ao conhecimento de processos de software. As Questões de Competência sobre PDS foram criadas com o objetivo de melhorar o conhecimento a respeito do processo de desenvolvimento e com isso auxiliar na tomada de decisão para melhoria do processo por parte dos gerentes de projeto. Neste sentido, tanto a ontologia ProvOneExt-O quanto a rede complexa relacionada, foram desenvolvidas para explicitar essas questões de competência. As questões variam de questões mais gerais, onde o conhecimento extraído a partir delas é mais amplo, até questões mais específicas sobre indivíduos dos processos. A Tabela 10 apresenta as questões de competência e a importância da resposta de cada uma das perguntas. Para facilitar a compreensão, as questões serão respondidas no capítulo 5, a partir do experimento conduzido com dados reais de empresas de desenvolvimento de *software*.

Tabela 10 - Questões de Competência relacionadas a ontologia ProvOneExt-O

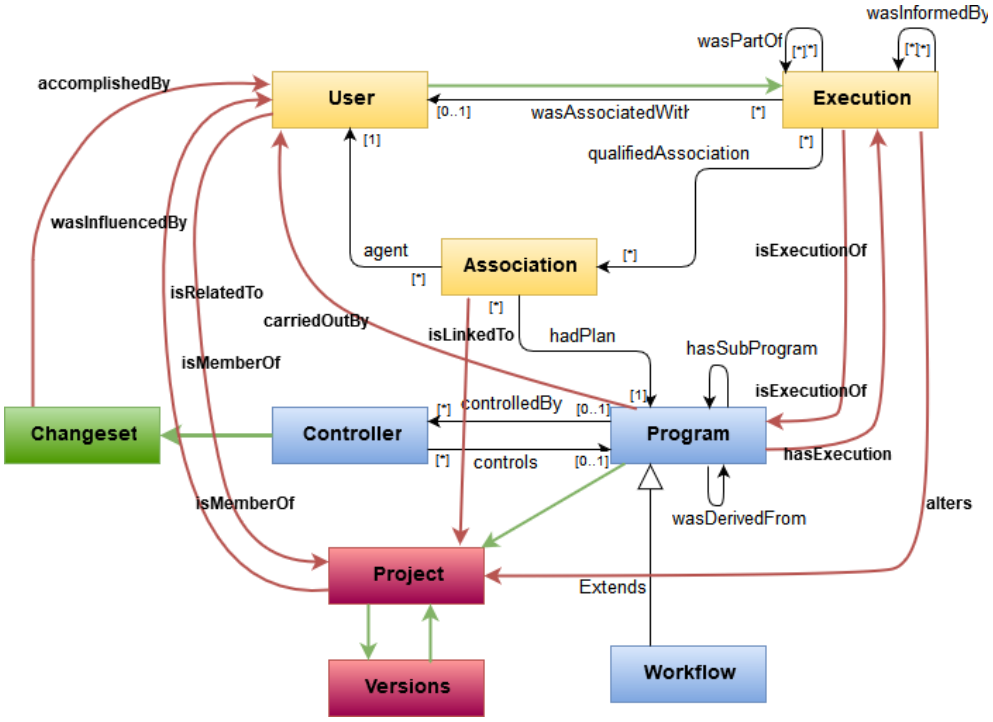
| Questão | Importância |
|---|---|
| Quais execuções foram associadas a um usuário específico? | Compreender melhor a proveniência prospectiva relacionada a um usuário e tentar entender como ele influencia essas execuções, ou talvez encontrar um padrão nas execuções associadas a um usuário, criando um perfil para associar futuras execuções. |
| A qual usuário uma execução específica foi associada? | Buscar a proveniência relacionada a uma execução, e investigar como um usuário a influenciou, seja positiva ou negativamente. |
| A qual associação um usuário específico está ligado? | Investigar se um usuário está ligado a uma associação, que se liga a um projeto. Associações permitem investigar se o usuário está formalmente ligado a um projeto. |
| A qual usuário uma associação específica está ligada? | Buscar a qual usuário uma associação está ligada. Uma associação sempre se liga a um usuário e um projeto, ela faz a intermediação formal da relação entre um usuário e um projeto. |
| A qual projeto uma associação específica está ligada? | Descobrir a qual projeto uma associação está ligada, essa pergunta é complementar a anterior. |
| Um usuário específico está ligado a execuções de qual projeto? | Investigar a proveniência retrospectiva de projetos, e descobrir quais usuários trabalharam ativamente para ele, e se influenciaram o projeto positiva ou negativamente. Um usuário pode estar ligado a |

| | |
|---|---|
| | execuções de um projeto mesmo sem estar relacionado a ele formalmente. |
| A qual projeto um usuário específico foi associado? | Investiga a qual projeto um usuário foi formalmente designado. |
| Por quais usuários um projeto é influenciado? | Buscar a proveniência relacionada a um projeto, e quais usuários o influenciaram. Essa investigação pode ser feita para descobrir, por exemplo, quais usuários influenciaram positivamente o projeto e poderiam ser indicados para outros projetos. Caso o projeto não esteja sendo desenvolvido como esperado, é possível investigar qual é a influência dos usuários nesse fator. |
| Quais versões um projeto possui? | Essa pergunta busca investigar quais são as versões de um projeto. Essa busca pode ser feita para compreender a proveniência de um projeto, e se suas versões o influenciam positiva ou negativamente, como por exemplo, através de atrasos. Além disso, investigações mais profundas podem ser feitas para saber, por exemplo, sobre a quantidade, tempo, ou outras características das versões. |
| Uma versão pertence a qual projeto? | Essa pergunta busca a proveniência relacionada a uma versão, para saber qual projeto ela influencia. |
| A qual projeto um *program* está relacionado? | Busca a proveniência de um *Program*, qual projeto ele influencia. |
| Um *program* está relacionado a qual/quais usuários? | Busca a proveniência de um *Program* em relação a quais usuários o influenciam. Pode criar um perfil de execuções ligadas a um usuário, ou buscar se o usuário foi relevante para o sucesso ou algo não esperado no desenvolvimento do *Program*. |
| Qual projeto uma execução altera/influencia? | Busca a proveniência de uma execução, e qual projeto ela influencia. Essa investigação pode ser feita após identificar algum comportamento estranho em um *Program*, para identificar o impacto que ele gerou em um projeto. |
| Uma execução está relacionada a qual *Program*? | Busca a proveniência de um *Program*, e quais execuções o influenciam, principalmente se uma execução tiver um comportamento estranho, qual o impacto que ela gerou no desenvolvimento de um *Program*. |
| Quais execuções um *Program* possui? | Buscar a proveniência de um *Program*, e investigar o comportamento das execuções, seja para identificar possíveis problemas ou fatores de sucesso para melhorar futuras execuções. |

### 3.2.2. Detalhamento da ontologia

Com base nas questões de competência a serem respondidas, a ontologia ProvOneExt-O foi desenvolvida, a partir do modelo estendido do ProvOne, apresentado na Figura 4. A partir do modelo da Figura 4, novas relações e axiomas (representados pelas property chains) foram especificados. A Figura 5 apresenta a ontologia ProvOneExt-O, com suas classes, relações e axiomas (representados em vermelho).

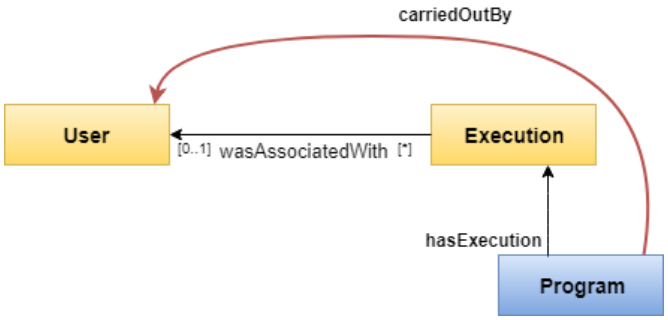
Figura 5 - Ontologia baseada na extensão do modelo ProvONE




Os axiomas especificados na ontologia ProvOneExt-O foram implementados a partir do uso de property chains. Property chains são relações que agregam outros relacionamentos, ou seja, são a união de dois ou mais relacionamentos para a derivação de um novo. A Figura 6 ilustra um exemplo de property chain, denominada *carriedOutBy*. O axioma é especificado através da property chain que estabelece que quando um indivíduo da classe *Program* possui a relação "*hasExecution*" com um indivíduo da classe *Execution*, e esse indivíduo de *Execution* possui a relação "*wasAssociatedWith*" com um indivíduo da classe *User*, é possível inferir que o indivíduo de *Program* possui um relacionamento "*carriedOutBy*" com o indivíduo da classe *User*, como ilustrado na Figura 6 e representado pela seguinte notação:"*wasAssociatedWith o hasExecution: carriedOutBy*". Na Figura 6, a property chain "*carriedOutBy*" associa duas relações: "*hasExecution*" e "*wasAssociatedWith*". Isso significa que é possível extrair o conhecimento implícito nos

dados referente aos usuários que influenciam um *Program* diretamente, relação que não existia antes no modelo. Um usuário influencia um *Program* caso ele esteja ligado a uma execução deste Program. Dessa forma, através da relação "carriedOutBy" é possível inferir todos os usuários que influenciaram um Program.

Figura 6 - Property chain carriedOutBy




Além do axioma apresentado na Figura 6, outros axiomas foram adicionados à ontologia, a partir de property chains, para enriquecer as possibilidades de extração de conhecimento, que são apresentados na Tabela 11:

Tabela 11 - Property Chains adicionadas na ontologia ProvOneExt-O

| **Identificação** | **Expressão Formal** | **Descrição** |
|---|---|---|
| ***isExecutionOf*** | qualifiedAssociation o hadPlan | Relação entre *Execution* e *Program*, que permite que se saiba qual *Program* gerou determinada execução. Pode ser importante no contexto de PDS pois permite a busca pela proveniência de uma execução. Essa relação permite a investigação sobre o *Program* de origem a partir de uma execução diretamente, e pode ser muito útil quando há necessidade de compreender melhor sua origem. |
| ***hasExecution*** | Inversa de isExecutionOf | Inversa à property chain *isExecutionOf*, a propriedade *hasExecution* liga *Programs* às suas execuções, e permite que se saiba quais execuções estão ligadas a um *Program*. Antes da criação dessas propriedades não havia uma ligação direta entre um *Program* e suas execuções. Essa relação pode ser muito útil quando se identifica algo não esperado em uma *Execution*, e há necessidade de buscar por sua proveniência. |
| ***alters*** | isExecutionOf o belongsTo | A property chain *alters* liga *Execution* a *Project*, e permite que seja buscado o projeto ao qual uma execução está ligada. Caso seja identificado algum comportamento em uma |

| | | |
|---|---|---|
| | | *Execution* que seja crítico, é possível investigar qual projeto ela influencia, para averiguar possíveis impactos. |
| ***isLinkedTo*** | hadPlan o belongsTo | A property chain *isLinkedTo* permite a ligação entre *Association* e *Project*, e pode ser inferida caso *hadPlan* e *belongsTo* sejam satisfeitos. Essa propriedade permite a busca de qual projeto uma *Association* se liga, e consequentemente a qual projeto ela associa formalmente um usuário. |
| ***isRelatedTo*** | associates o alters | A property chain *isRelatedTo* liga um *User* a um *Project*, e pode ser inferida quando o usuário possui a relação *associates* com uma execução, e essa por sua vez está ligada a um *Project* pela *alters*. Esse axioma averigua se um usuário está ligado a execuções de um projeto. No contexto de PDS, é muito importante para investigar se um usuário está trabalhando para um projeto sem ter sido oficialmente associado a ele. Essa propriedade deve ser comparada à *isMemberOf*, que verifica se um usuário está formalmente associado a um projeto. |
| ***isMemberOf*** | isAgentOf o isLinkedTo | Semelhante a propriedade "*isRelatedTo*", a relação *isMemberOf* liga um *User* a um *Project*, porém só pode ser inferida quando um *User* possui a ligação *isAgentOf* (inversa à ligação "*agent*") com uma *Association*, e essa possui a ligação *isLinkedTo* com um *Project*. Quando um *User* está ligado a um *Project* pela propriedade *isRelatedTo*, porém não possui a ligação com o mesmo *Project* através de *isMemberOf*, significa que ele não está associado a um projeto formalmente, mas está ligado a *Executions* do mesmo. |
| ***wasInfluencedBy*** | inversa da property chain is RelatedTo | A propriedade *wasInfuencedBy* liga *Project* a *User*, e é inversa à *isRelatedTo* (ocorre quando *isRelatedTo* é satisfeita). Essa ligação permite a busca de *Users* que influenciaram um determinado *Project*, ou seja, executaram *Executions* de *Programs* relacionadas a um *Project*. Caso um projeto não esteja funcionando como esperado, essa propriedade permite a busca por quais usuários o influenciam, e que possivelmente pode ajudar a resolver o problema. |
| ***carriedOutBy*** | hasExecution o wasAssociatedWith | Essa é uma propriedade entre *Program* e *User*, e pode ser inferida quando um *Program* está ligado a uma *Execution*, a qual está relacionada a um *User*. Dessa forma, é |

| | | |
|---|---|---|
| | | possível buscar quais usuários influenciam um *Program*. É possível investigar quais perfis de *Program* um *User* costuma executar, e futuramente associá-lo a *Programs* similares. Além disso, é possível identificar caso um *Program* se destaque, quais *Users* o influenciaram. |

Com o uso de axiomas, a ontologia tem um papel importante na extração de conhecimento implícito, o que auxilia na compreensão e na tomada de decisão baseada nos dados. As property chains combinam relações diferentes, criando uma nova relação que antes era inexistente. Relações e property chains podem ser inferidas pela máquina de inferência, de acordo com as regras criadas e indivíduos disponíveis.

A property chain "isRelatedTo", permite a extração do conhecimento implícito nos dados referente aos usuários que influenciam um *Project* diretamente por estar relacionado a execuções do mesmo. Através dessa relação é possível inferir todos os usuários que influenciaram um *Project*. Essa informação pode ser muito relevante caso se esteja investigando usuários relacionados a atividades do *Project* em questão, seja por algum fator discrepante, como por exemplo muitas horas de trabalho para pouco progresso, ou caso tenham identificado que usuários não associados formalmente ao projeto (que possuem a relação isMemberOf com o projeto) estejam mesmo assim trabalhando para o mesmo. A partir da descoberta de quais usuários estavam envolvidos com execuções de um *Project*, pode-se investigar qual decisão tomar, como por exemplo, alocar esse usuário para o projeto, ou designar a ele atividades de um projeto para o qual ele está relacionado formalmente.

## 3.3. REDE COMPLEXA PROVONEEXT-COMPLEX

Foi identificado que algumas questões relacionadas a estrutura do modelo poderiam auxiliar na tomada de decisão estratégica por parte dos gerentes de projeto. Assim, foi proposta a criação de uma rede complexa a partir do modelo ProvOneExt para também auxiliar na derivação de conhecimento estratégico. Foram então identificadas questões de competência específicas relacionadas à parte estrutural do modelo. Dessa forma, para responder a essas questões de competência, o modelo da rede complexa utilizou classes do modelo proposto ProvOneExt, e foram acrescentadas classes e relações específicas relacionadas a modelagem estrutural. O modelo gerado, denominado ProvOneExt-Complex tem como objetivo ser um modelo complementar a ontologia, para extrair conhecimento

estrutural, derivando conhecimento estratégico complementar ao derivado pelo modelo ontológico.

Entre as principais vantagens de se utilizar a rede complexa está a possibilidade de analisar os dados e relações como um todo. É possível analisar, por exemplo, o impacto da retirada de um nó na rede. Quanto mais ligações ele possui, principalmente se essas ligações tiverem maiores pesos, maior será esse impacto. No contexto de PDS pode se analisar, por exemplo, o impacto da retirada de um usuário da rede. No capítulo de avaliação da abordagem algumas análises são apresentadas dentro desse contexto.

### 3.3.1. Questões de Competência

Considerando a parte estrutural do modelo, algumas questões de competência relacionadas a PDS foram especificadas para auxiliar na derivação de conhecimento estratégico. A Tabela 12 apresenta as questões de competência, e as respostas dessas questões são apresentadas no capítulo de avaliação.

Tabela 12 - Questões de Competência a serem respondidas com a rede complexa

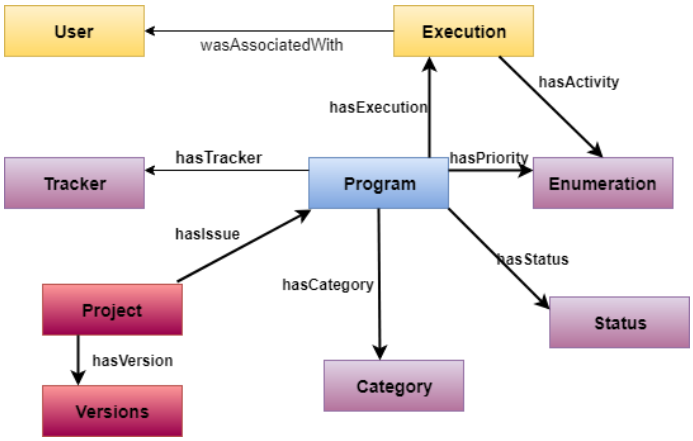
| **Questão** | **Importância** |
|---|---|
| Quantas execuções estão ligadas a cada usuário? | Investigar a importância de um usuário na rede. Quanto mais execuções ele está ligado, maior é sua participação, e maior pode ser o impacto caso seja removido da rede. |
| Quais tipos de atividades são desenvolvidos? | Diferentes tipos de atividades podem ser desenvolvidos entre as execuções de *Programs*. Essa pergunta procura responder a qual tipo de atividade uma execução se refere. |
| Quais *Programs* possuem maior quantidade de execuções? | Essa pergunta busca analisar qual a importância de um *Program* em uma rede. Quanto maior a quantidade de execuções que um *Program* possui, maior é o impacto de sua possível remoção da rede, principalmente se forem *Programs* críticos. Além disso, maior pode ser a influência desse *Program* em seus projetos relacionados. |
| Quais tipos de *Programs* possuem maior quantidade de relações? | Investigar o comportamento dos dados quanto à distribuição de *Programs* por tipo de atividade. Dessa forma é possível descobrir quais tipos de *Programs* são mais executados nos projetos, e caso algum comportamento estranho seja identificado, é possível pesquisar mais profundamente para entender o porquê dessa distribuição. |
| Quais prioridades de *Programs* possuem a | Essa questão permite buscar quais |

| | |
|---|---|
| maior quantidade relações? | prioridades de *Programs* mais ocorrem. Dessa forma é possível investigar se há algum padrão ou alguma prioridade que ocorra mais, e caso haja algo não esperado, investigar o porquê dessa distribuição. Por exemplo, se as prioridades sempre são muito altas, porque isso ocorre? Haveria possibilidade de fazer algo para que nem sempre fossem altas, como manutenções preventivas ao invés de corretivas? |
| Quais categorias de *Programs* possuem maior quantidade de ligações? | Essa pergunta permite o entendimento da distribuição de *Programs* por categorias, como por exemplo, quantos *Programs* estão relacionados a "*bugs*". |
| Qual é o status da maioria dos *Programs*? | A busca pelos status de *Programs* permite descobrir a distribuição de *Programs* de acordo com seus status, por exemplo, se um *Program* já foi concluído. |
| A quantos *Programs* cada projeto está relacionado? | Essa pergunta busca a quantidade de *Programs* que cada projeto possui. A partir dessa resposta é possível identificar o perfil dos projetos, se possui muitas atividades ou se está pausado. De acordo com essa distribuição, caso algo não esperado seja identificado, investigações mais profundas podem ser feitas para buscar razões pela qual a distribuição se encontra desta forma. |
| Quais projetos possuem mais versões? | Um projeto pode possuir diversas versões. Essa pergunta busca responder quantas e quais versões cada projeto possui. Essa quantidade pode variar de acordo com o nível de atividade ou o tempo que o projeto tem sido desenvolvido. Caso sejam identificadas menos versões que o esperado, ou mais, é possível realizar investigações mais profundas para entender o porquê dessa característica. |

### 3.3.2. Detalhamento

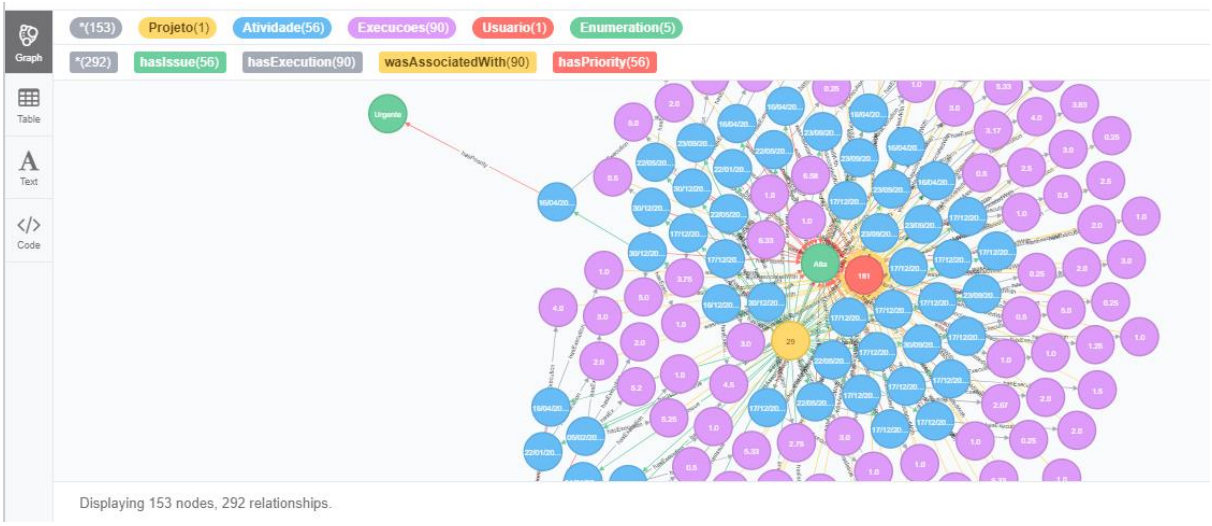
A Figura 7 apresenta a rede complexa ProvOneExt-Complex, especificada a partir do modelo ProvOneExt, e estendida com o intuito de responder questões específicas relacionadas a parte estrutural do modelo, que não poderiam ser respondidas a partir do modelo básico e nem a partir da ontologia.

Figura 7 - Rede complexa baseada na extensão do modelo ProvONE




As classes adicionadas na rede complexa representam características das classes *Execution* e *Program*, como status e categorias. O objetivo da rede complexa é permitir a extração de conhecimento adicional sobre os dados, além do conhecimento extraído através da ontologia. Dessa forma, essas classes foram criadas para permitir análises diferenciadas e uma análise topológica sobre características na rede. Com essas análises é possível compreender a importância de cada uma das características, principalmente qual seria o impacto de sua remoção da rede, ou grau de entrada e saída, análises que não poderiam ser feitas caso essas características não fossem representadas como classes. Além disso, é possível analisar visualmente a importância da característica na rede como resultado de buscas. A Figura 8 mostra um exemplo dessa visualização, onde em verde são apresentadas prioridades de atividades. A centralidade do nó “Alta” permite observar que nessa busca especificamente, a grande maioria das atividades possui prioridade alta. Essa análise pode ser feita de forma intuitiva devido a construção da rede. Caso as características fossem detalhes dos nós, para obter essa informação, cada nó de atividade teria que ser analisado individualmente para descobrir qual prioridade dele.

Figura 8 - Exemplo de busca com visualização das características na rede complexa




As classes adicionadas à rede são apresentadas na Tabela 13:

Tabela 13 - Classes adicionadas à rede complexa

| Classe | Importância |
|---|---|
| *Tracker* | Essa classe foi criada para ter dados sobre o tipo de *Program*, como "Bug" ou "Melhoria". Através dessa classe é possível analisar quais tipos de *Programs* tem ocorrido mais na rede, e o impacto que ele gera no comportamento dos dados. |
| *Enumeration* | Essa classe possui dados sobre *Execution* e *Program* que seriam enumeradas em um banco de dados. Como atividades desenvolvidas em uma execução, ou prioridades de um *Program*. |
| *Status* | Essa classe possui dados sobre o status de um *Program*, como, por exemplo, se é um *Program* novo, ou se já foi fechado. |
| *Category* | A classe *Category* possui dados sobre as categorias de um *Program*. |

## 3.4. IMPORTÂNCIA DOS MODELOS GERADOS

Para auxiliar na tomada de decisão por parte dos gerentes de projetos, foram criados diferentes modelos. O modelo de proveniência ProvONEExt, uma extensão do ProvONE, foi criado para adaptar o modelo ao contexto de Processos de Desenvolvimento de *Software*. Esse modelo possui classes e propriedades que representam entidades e relacionamentos do contexto, e é um modelo amplo que pode ser utilizado para obter a proveniência de diferentes partes do contexto de PDS, inclusive a geração e utilização de documentos, visualizações e dados.

A ontologia ProvOneExt-O foi desenvolvida utilizando como base o modelo do ProvONEExt. Ela permite a modelagem do conhecimento no contexto de PDS, e a extração de conhecimento implícito através de inferências, que podem ser feitas sobre as regras definidas. A rede complexa foi desenvolvida também com base no modelo ProvONEExt. Essa abordagem permite que sejam feitas análises sobre o comportamento dos dados na rede, como grau de entrada e saída dos nós, ou o impacto da remoção de um nó da rede.

As análises da ontologia e rede complexa se complementam. Na rede complexa é possível realizar análises sobre a importância de um nó na rede, e essa importância está relacionada à quantidade de relações que um nó possui, como grau de entrada, saída ou impacto da remoção de um nó. Portanto, a maioria das perguntas sobre o contexto que a rede complexa ajuda a responder é quantitativa. Além disso, como a rede possui classes diferentes que estão relacionadas às características de *Programs* e *Executions*, por exemplo, ela colabora para uma compreensão do comportamento dos dados, e a distribuição dos mesmos de acordo com suas características. Por outro lado, a ontologia permite uma melhor compreensão da proveniência sobre o contexto. As perguntas que a ontologia ajuda a responder são relativas a proveniência dos dados, e como indivíduos da ontologia foram influenciados ou influenciam. Essas análises auxiliam na compreensão sobre o comportamento e importância dos dados, e auxiliam na tomada de decisão.

Para facilitar a utilização dos modelos criados, uma arquitetura foi proposta para dar suporte à extração de conhecimento e à tomada de decisão. Essa arquitetura engloba os modelos apresentados no presente capítulo, e é descrita no capítulo 4. A ontologia e rede complexa podem ser criadas e instanciadas com dados de projetos específicos, e análises podem ser feitas para extrair conhecimento implícito.

Um exemplo de análise que pode ser feita através da arquitetura é a busca pela compreensão do comportamento de atividades. Na rede complexa é possível investigar a quantidade de execuções ligadas a cada atividade. Essa análise pode ser feita por um gerente de projetos através das visualizações disponíveis da arquitetura. Caso detectado algo discrepante, como uma quantidade muito grande de execuções ligadas a uma atividade, é possível investigar na rede características ligadas *à* mesma e suas execuções. Na ontologia é possível buscar a proveniência ligada à atividade, como a qual usuário ela está ligado e qual projeto ela influencia. Essa proveniência pode ser visualizada a partir da filtragem do indivíduo na ferramenta de visualização Visionary (COSTA et al. 2016a), a qual faz parte da arquitetura proposta.

## 3.5. RESUMO DO CAPÍTULO

O presente capítulo apresentou o ProvONEExt, uma extensão do modelo de proveniência ProvONE, estendido para se adaptar ao contexto de processos de desenvolvimento de *software*. Baseadas nesse modelo foram apresentadas também uma ontologia e uma rede complexa, propostas para extrair conhecimento sobre dados de PDS, melhorando o entendimento de PDS, além de auxiliar na tomada de decisão. A ontologia modela a proveniência da área, e através das relações e property chains adicionadas, juntamente com uma máquina de inferência, permite inferir conhecimento e extrair conhecimento implícito nos dados. A rede complexa apresentada permite análises topológicas, como o impacto da remoção de um nó.

Os modelos apresentados neste capítulo são parte de uma arquitetura proposta no presente trabalho, e que será detalhada no capítulo 4. As questões de competência relacionadas à ontologia e rede complexa, apresentadas nas seções 3.2.1 e 3.3.1 respectivamente, são respondidas no capítulo 5 através de análises da ontologia e rede complexa.

# 4. ARQUITETURA ONTOCOMPLEX

## 4.1. INTRODUÇÃO

Apresentamos neste capítulo a arquitetura OntoComplex, especificada com o objetivo de prover suporte aos gerentes de projeto na tomada de decisão sobre os Processos de Desenvolvimento de *Software*. A OntoComplex é alinhada com as abordagens que apoiam a tomada de decisão em projetos de *software* baseada em dados (BOSCH, 2017). Para isso, a arquitetura possui uma camada de visualização, que ajuda na compreensão dos dados, uma camada de análise, que investiga novas conexões entre os dados, e um tradutor, que traduz os dados de processos em formatos variados, para o modelo de proveniência proposto. Assim, o principal objetivo é extrair conhecimento implícito em dados de processos de desenvolvimento de *software*, possibilitar diferentes análises, além de visualizações que possam auxiliar gerentes de projetos a compreender melhor os dados e ter suporte à tomada de decisão. A arquitetura proposta será detalhada nas seções seguintes.

## 4.2. DETALHAMENTO DA ARQUITETURA ONTOCOMPLEX

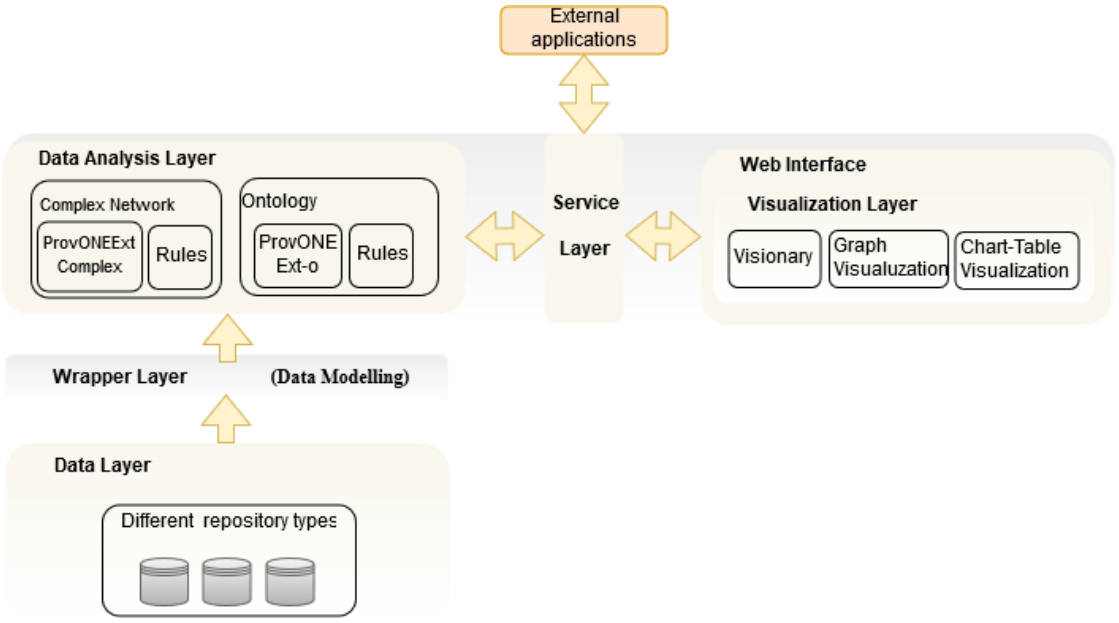
A arquitetura OntoComplex usa ontologias e redes complexas, em conjunto com o modelo ProvONEExt, como modelos para apoio a tomada de decisão. O principal objetivo da arquitetura é extrair conhecimento estratégico de dados sobre o processo de desenvolvimento de *software*. Acreditamos que com as informações providas pela arquitetura, gerentes de projetos possam entender melhor o comportamento dos processos, tomar decisões baseadas em dados e como isso pode ajudar a melhorar futuras execuções.

Para isso, a OntoComplex foi especificada com base no modelo arquitetural em camadas, conjugado com o modelo orientado a serviços (BUSCHMANN et al., 1996). A Figura 9 apresenta uma visão alto nível da arquitetura. A primeira camada é a **Camada de Dados**, que engloba dados de processos de *software* de diferentes aplicações. Esses dados podem ser heterogêneos e provenientes de diferentes repositórios. Para isso, os mesmos devem ser traduzidos para o modelo básico utilizado pela arquitetura (ProvOneExt). A **Camada Wrapper** é responsável por fazer a tradução destes dados para o modelo ProvONEExt.

A ontologia e a rede complexa são os componentes da arquitetura que permitem a extração de conhecimento através de suas estruturas e regras, caracterizando **a Camada de Análise de Dados**. A **Camada de Visualização** tem por objetivo facilitar o entendimento dos

dados, a partir do uso de *dashboards* conjugados a técnicas de visualização adequadas para visualização de dados de proveniência. Estas camadas da arquitetura serão detalhadas nas próximas subseções.

Figura 9 - Arquitetura OntoComplex




### 4.2.1. Camada de Dados

A **Camada de Dados** da arquitetura tem como objetivo englobar dados de diferentes tipos de processos de desenvolvimento de *software*, desde processos com ênfase na fase de captura de requisitos e modelagem até processos de suporte a manutenção de *software*, como por exemplo, dados de requisições de manutenção e melhoria, que serão utilizados para ilustrar a abordagem dessa dissertação.

Para isso, os dados são capturados a partir dos repositórios de ferramentas de controle de processos, entre elas, ferramentas de controle de mudanças, como o Mantis[9]. Os dados são capturados durante os processos, e disponibilizados para processamento e uso na **Camada de Análise de Dados** da OntoComplex, a partir da **Camada Wrapper**.

Estes dados coletados podem ser disponibilizados em formatos diferentes, como por exemplo no formato “sql” ou em “csv”. Os dados são heterogêneos, mas podem ser integrados através da utilização do *wrapper*, que adapta os dados para o formato do modelo ProvONEExt. Um detalhamento do tipo de dado tratado nessa camada será apresentado no capítulo 5 desta dissertação.

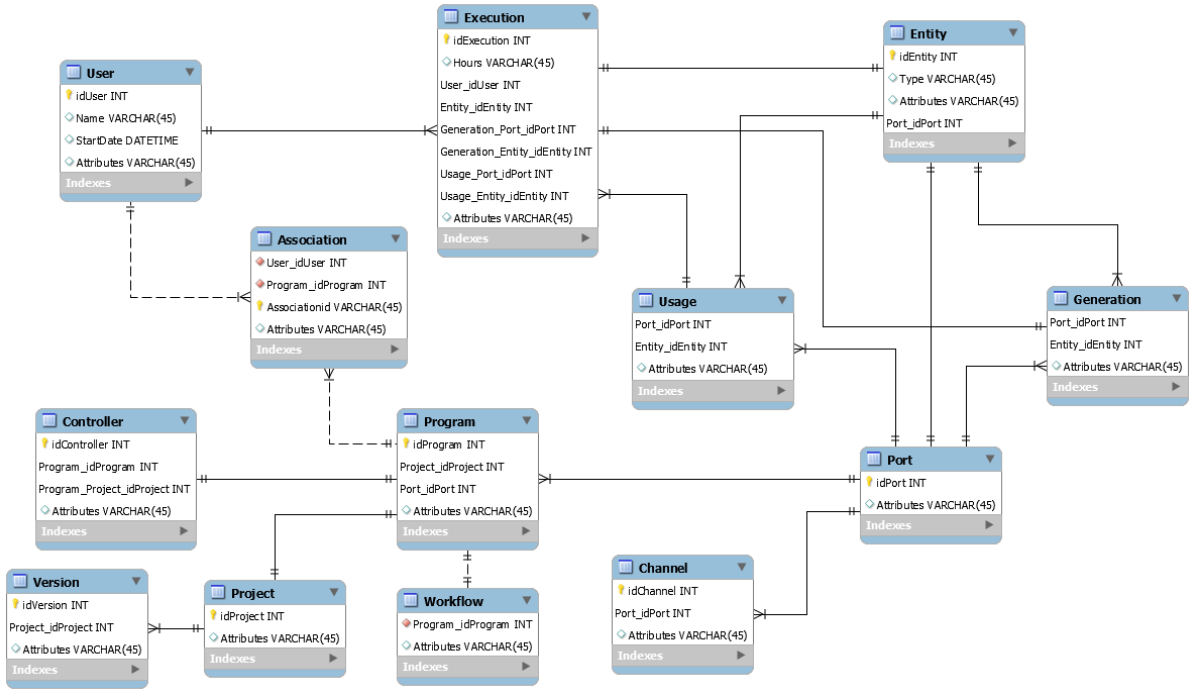
[9] https://www.mantisbt.org/

Assim, é possível, a partir dos dados disponibilizados nessa camada, realizar análises para a derivação de novos relacionamentos, que podem auxiliar gerentes de projetos a entender melhor o comportamento dos dados e a tomar melhores decisões.

### 4.2.2. Camada Wrapper

Assim como mencionado anteriormente, os dados utilizados na arquitetura podem estar disponíveis em formatos diversos. A **Camada Wrapper** tem como objetivo traduzir o modelo de dados original para o modelo utilizado pela arquitetura OntoComplex, ou seja, traduz os dados de origem para o modelo ProvOneExt. Para isso, a camada utiliza um banco de dados relacional para recepcionar os dados vindos da Camada de Dados, que devem possuir descritores que permitam a compreensão dos dados originais. O banco de dados é modelado de acordo com o modelo ProvOneExt (Figura 10), onde as tabelas estão relacionadas às classes do modelo, e as relações baseadas nas propriedades do modelo e para o qual os dados devem ser traduzidos. O modelo é genérico, mas pode ser especializado de acordo com os dados disponíveis, e a necessidade de extração de conhecimento. Um exemplo de uso desta camada será também apresentado no capítulo 5.

Figura 10 - Diagrama entidade relacionamento ProvONEExt




### 4.2.3. Camada de Análise de Dados

Nesta camada os dados traduzidos para o modelo ProvOneExt são processados pela ontologia e pela rede complexa, e analisados com objetivo de estabelecer novas

conexões, e, a partir destas extrair conhecimento estratégico para facilitar a tomada de decisão relacionada as execuções dos processos.

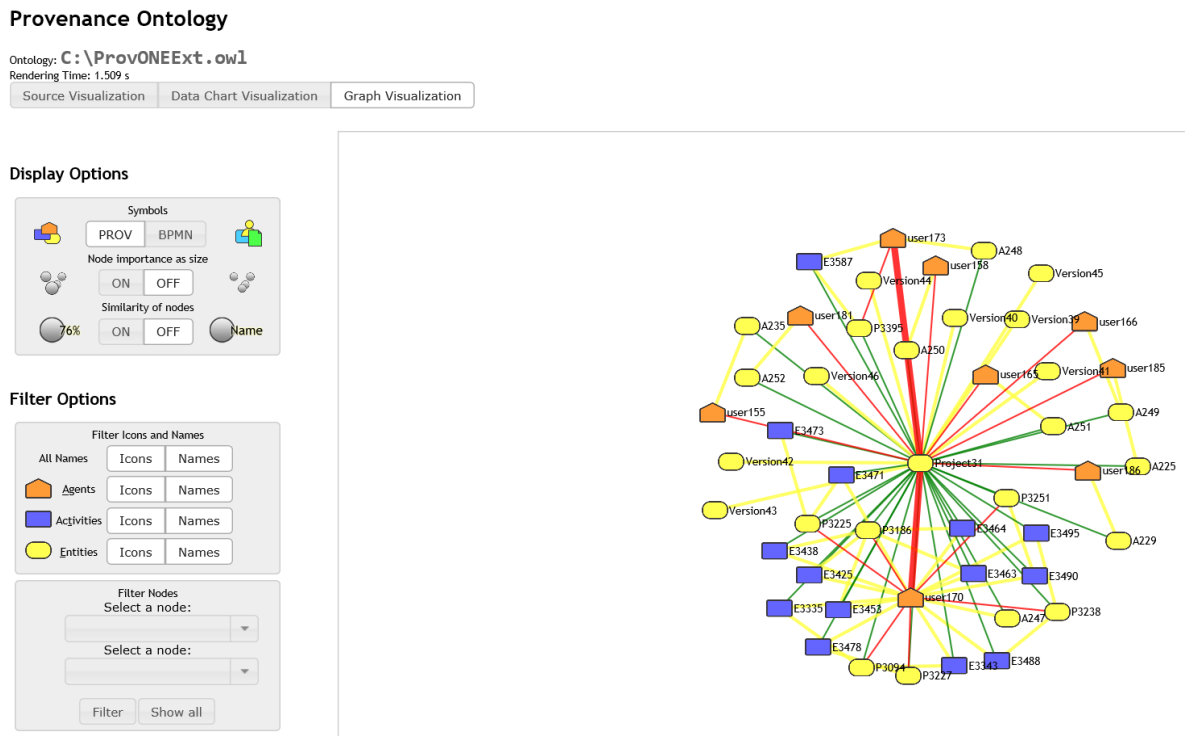
Detalhamos a seguir como se dá o uso da ontologia e da rede complexa no contexto da arquitetura.

#### *4.2.3.1. ProvONEExt-O*

As análises semânticas realizadas pela arquitetura são obtidas a partir do uso da ontologia ProvONEExt-O. O objetivo da ontologia, conforme apresentado no capítulo 3, é prover regras ontológicas que permitam a derivação de conhecimento estratégico sobre os dados através de inferências sobre projetos, suas versões, atividades e suas execuções, além de usuários conectados a elas. Através das relações e inferências, é possível derivar conhecimento novo sobre os dados que não se encontram no banco original.

Para ilustrar o uso da ontologia, um exemplo é apresentado a seguir. A Figura 11 apresenta a visualização da proveniência dos processos de *software* relacionadas ao "Projeto 31". Essa visualização é feita através da ferramenta Visionary da **Camada de Visualização**, que é utilizada para visualizar a ontologia, suas relações, indivíduos e inferências.

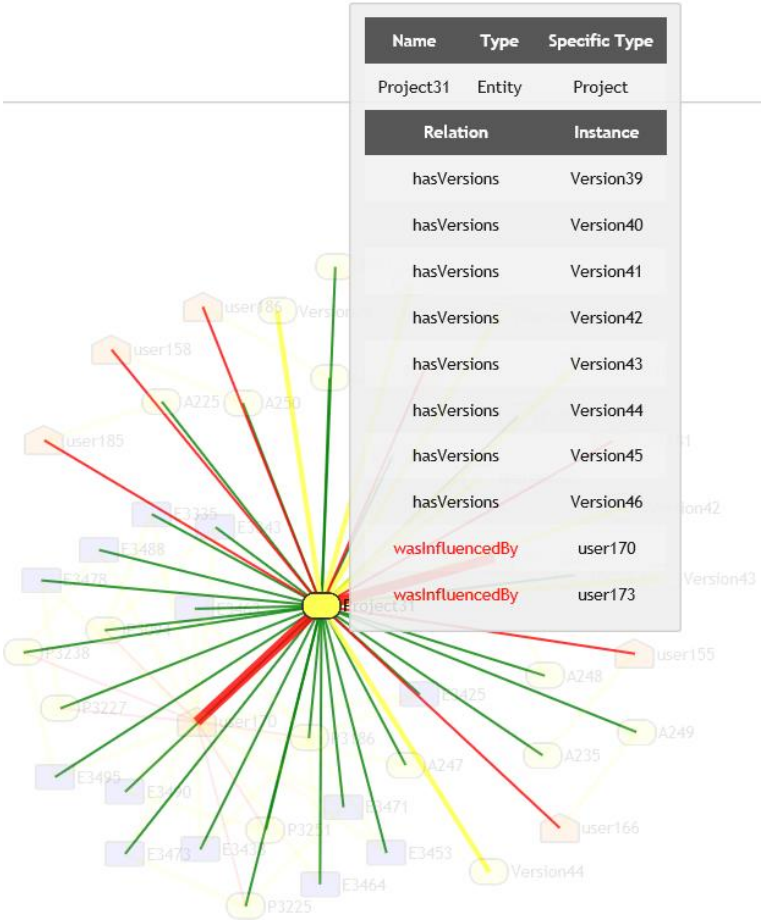
Figura 11 - Visualização da proveniência do Projeto 31 no Visionary




Supondo que um gerente de projetos estivesse insatisfeito com o rendimento das atividades do "Projeto 31", e a partir de então fosse investigar a participação de usuários associados ao projeto, assim como atividades e execuções relacionadas a cada um deles. O

modelo ProvONEExt não provê todas essas relações diretamente, porém através da ontologia elas podem ser inferidas a partir das regras definidas e da utilização da máquina de inferências. Um exemplo é ilustrado na Figura 12, onde o "projeto 31" é filtrado na ferramenta de visualização da ontologia, onde podemos visualizar as relações das quais o projeto é o domínio. É possível observar nessa imagem duas relações do tipo *"wasInfluencedBy"* inferidas pela ontologia ProvONEExt-o. Essa relação afirma que o projeto foi influenciado pelos usuários "user170" e "user173", ou seja, os mesmos estão relacionados a execuções do projeto.

Figura 12 - Visualização de uma filtragem da proveniência relacionada Projeto 31 no Visionary




Ao clicar em um indivíduo da visualização, é possível observar as relações das quais um indivíduo é domínio[10], assim como os indivíduos imagens das relações, como pode ser observado na Figura 13. Em vermelho aparecem as relações inferidas pela ontologia. A relação *"associates"* relaciona o usuário "user170" a todas as execuções que ele é responsável. A relação *"isRelatedTo"* indica que o usuário está relacionado a execuções do

[10] Domínio neste contexto se refere ao sujeito da relação ontológica, ou seja, *domain* em inglês e imagem, se refere ao objeto da relação ontológica, ou *range* em inglês

"Projeto 31", e *"isMemberOf"* indica que o usuário foi formalmente designado para o projeto. A relação *"verifiedAssociation"* é inferida quando o usuário está relacionado formalmente ao projeto e está relacionado a execuções do mesmo. A Figura 14 mostra as relações do usuário "user 173".

Figura 13 - Visualização de proveniência relacionada user170 no Visionary ao clicar no indivíduo

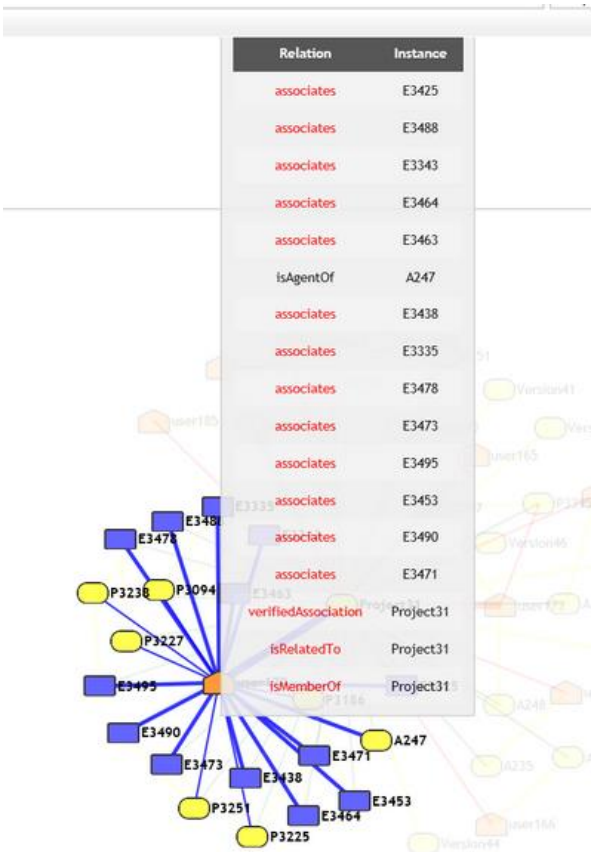
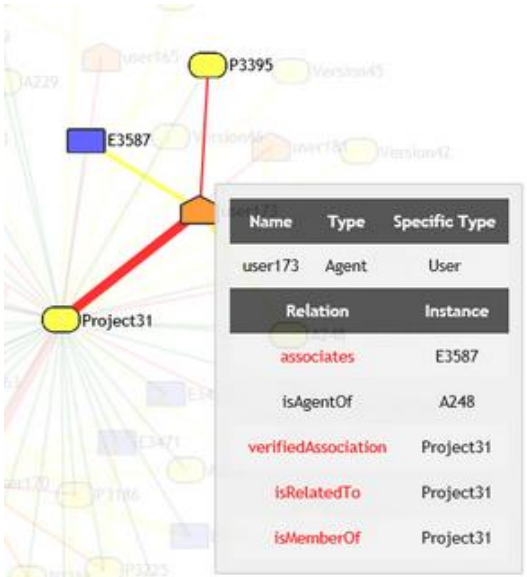


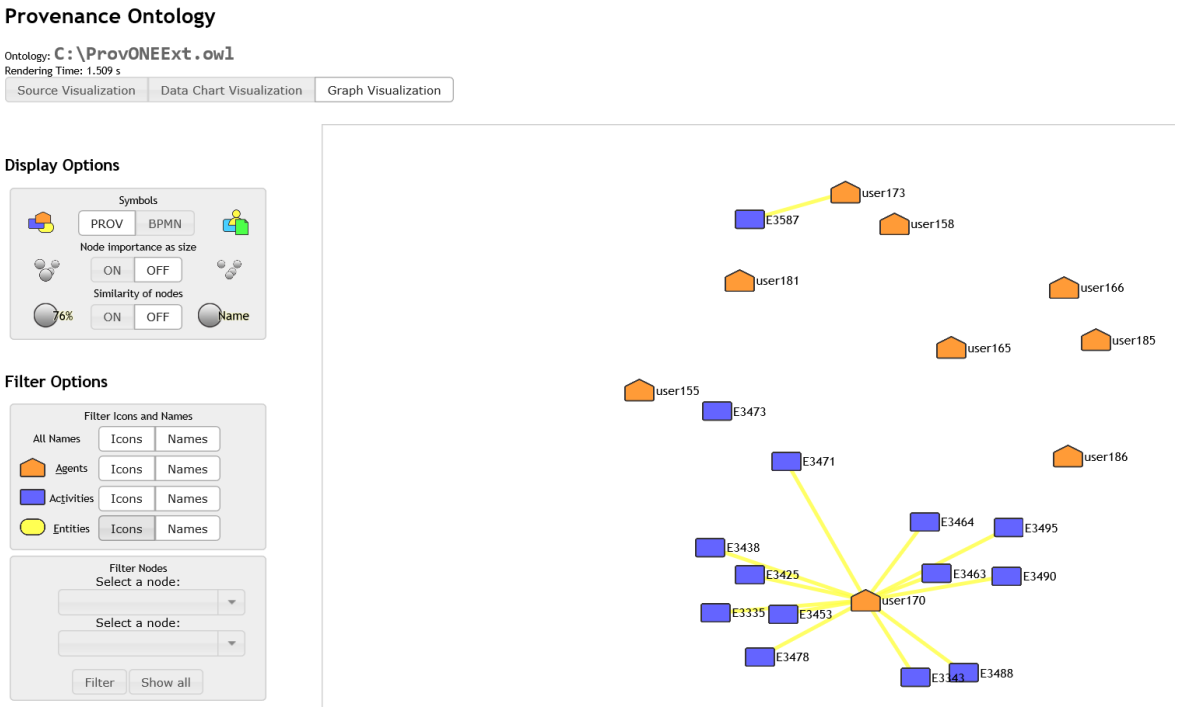

Figura 14 - Visualização de proveniência relacionada user173 no Visionary ao clicar no indivíduo



Na ferramenta Visionary, integrante da arquitetura OntoComplex, o gerente poderia clicar no filtro “Icons” da opção *Entities*, para omitir as entidades e visualizar apenas Atividades (execuções de tarefas) e Agentes (Usuários). Essa filtragem permite visualizar todos os usuários e execuções de tarefas relacionadas à proveniência do “projeto 31”. Essa investigação poderia estar sendo realizada para identificar usuários alocados para o projeto que não participam ativamente do mesmo. Dessa forma, o gerente poderia buscar identificar razões pelas quais os usuários não estão se dedicando, e tentar alocar atividades para os usuários inativos.

Após a filtragem, a ferramenta apresenta apenas nós de atividades e agentes, como pode ser observado na Figura 15. Nessa imagem é possível observar que apesar do projeto estar associado a 9 usuários (nós da cor laranja), apenas 2 estão ligados a atividades (execuções de atividades, representados na cor azul). A partir dessa constatação, investigações mais profundas podem ser realizadas para levantar hipóteses de o porquê esses usuários estarem associados a esse projeto mas não possuírem execuções relacionadas a ele.

Figura 15 - Visualização da proveniência do Projeto 31 no Visionary, com filtragem de nós do tipo entidade




### *4.2.3.2. ProvONEExt-Complex*

Outro tipo de análise realizada pela OntoComplex são as análises estruturais processadas a partir de redes complexas. Esta rede complexa foi criada a partir do modelo ProvONEExt, apresentado no capítulo 3. Os nós foram criados de acordo com as classes do

modelo, e arestas foram adicionadas baseadas nas relações das classes do modelo da rede. Após a definição inicial, dados da camada de dados podem ser inseridos para que as análises possam ser realizadas.

A rede complexa pode ser analisada para verificar, por exemplo, a relevância dos nós na rede complexa, o impacto em caso de ataque (se um nó fosse removido da rede repentinamente), além do grau de entrada e de saída, que analisam quantas relações chegam ou saem dos nós.

Desta forma, a rede é utilizada para realizar análises complementares à ontologia. Seguindo a análise iniciada na subseção 4.2.3.1, a partir da identificação de usuários que não possuem execuções do "Projeto 31" apesar de estarem associados a ele, podem ser realizadas buscas na rede complexa para investigar informações adicionais sobre eles e seus comportamentos na rede complexa.

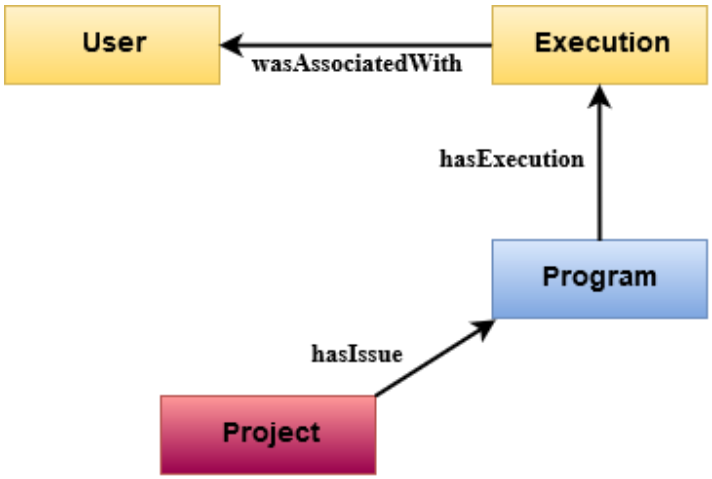
Enquanto a visualização apresentada na Figura 11 apresenta a proveniência relacionada apenas ao "Projeto 31", na rede é possível buscar relações de um usuário na rede como um todo. Além disso, como apresentado no Capítulo 3, o modelo da rede complexa disponibiliza outros tipos de informações.

Seguido o exemplo do "Projeto 31", podem ser realizadas buscas para verificar a importância na rede de cada um dos usuários que não possuem execuções do "Projeto 31". Essas buscas são realizadas aqui utilizando a plataforma Neo4j[11] para facilitar a visualização das respostas das buscas.

Como nesse cenário o interesse é buscar pela participação de usuários específicos em relação à projetos, atividades e execuções, buscas em *cypher*[12] foram realizadas na rede complexa para obter essas informações. Para tanto, foram buscados Projetos (nós do tipo **Project**) relacionados a atividades (nós **Program**), que possuem execuções (nós **Execution**) pelas quais um usuário (nós **User**) é responsável. O retorno das buscas mostra apenas os projetos dos quais os usuários participam ativamente, estando ligados a execuções de atividades do mesmo. As buscas retornam relações que representam a fração da rede ilustrada na Figura 16. Quanto maior a quantidade de nós um usuário estiver relacionado, maior é o impacto de uma possível remoção desse nó da rede, já que ele foi responsável por uma quantidade maior de execuções. Esse impacto varia de acordo com a quantidade de horas gasta em cada uma das execuções, e de acordo com a prioridade, entre outros fatores.

[11] https://neo4j.com/
[12] *Cypher* é a linguagem utilizada para escrever queries no Neo4j https://neo4j.com/developer/cypher-query-language/

Figura 16 - Seleção da rede complexa

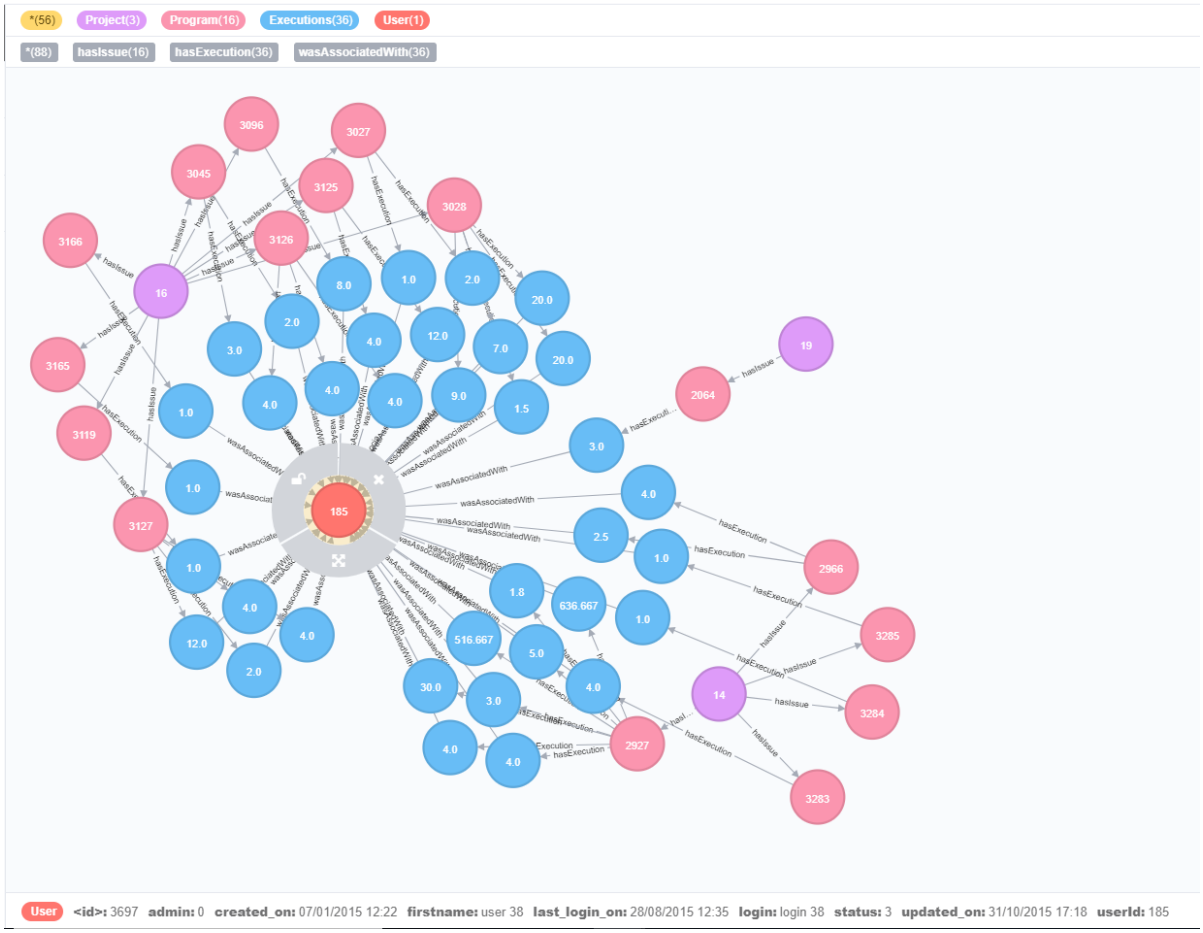



Na Figura 17 é possível observar o resultado da busca dos projetos, atividades (**programs** no modelo) e execuções relacionados ao "user 185", um dos usuários que está associado ao "Projeto 31", mas não executa atividades do mesmo. O retorno da busca mostra apenas os projetos dos quais o usuário participou ativamente, por isso não mostra o nó do "Projeto 31", o qual ele é associado mas não participou ativamente.

Na imagem 17, o nó do usuário é representada em vermelho, e ao clicar no nó é possível observar informações relacionadas a ele no rodapé, como a data de criação do registro do usuário (created_on), como é ilustrado na figura. É possível trocar as informações mostradas no centro do nó, nos nós azuis que representam execução de atividades a informação disponível no centro do nó é relativa a quantidade de horas que cada execução durou. Na imagem pode se observar que o "user 185" participou ativamente de 3 projetos (representados em lilás), e está ligado a atividades (**programs**, representados em rosa) e execuções de atividades (em azul) dos mesmos.

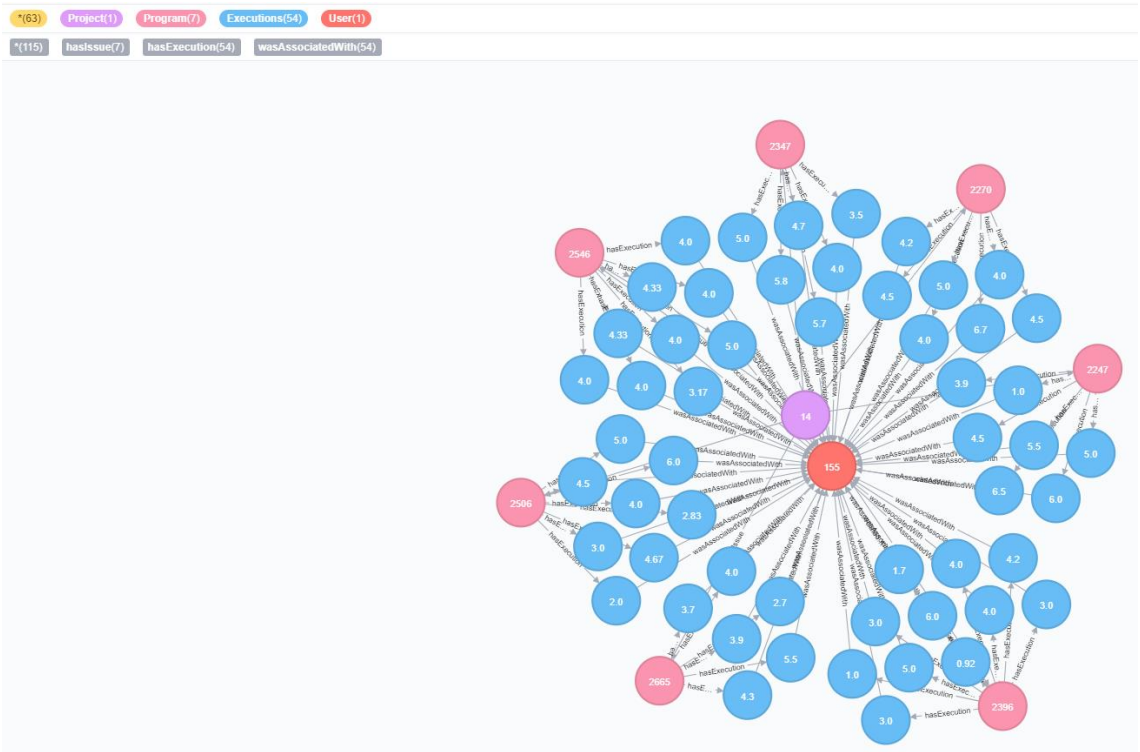
A seleção dos dados apresentada na Imagem 13 é uma "sub-rede". Quanto mais centralizado está um nó, maior é a quantidade de conexões que ele possui, e maior o impacto de uma possível remoção desse nó na rede. O "user 185" possivelmente não está relacionado a execuções do "Projeto 31" por estar ativo em oturos projetos. Pode ser também investigado sobre as datas das execuções dos projetos e a data da associação do usuário ao "Projeto 31", para mais respostas.

Figura 17 - Participação do usuário 185 na rede




A Figura 18 apresenta a participação do “user 155” na rede, onde se pode observar que o usuário está relacionado a atividades e execuções do “projeto 14”. Nessa imagem a quantidade de execuções e de horas por execução é relativamente homogênea, diferente do que é apresentado nas relações da Figura 17 relativas ao “user 185”. É possível identificar padrões de execuções relacionadas ao usuário. Caso seja de interesse da investigação, também se pode buscar informações sobre as atividades relacionadas a cada execução, categorias relacionadas às atividades, entre outras disponíveis na rede. O “user 155” pode não estar executando atividades do “projeto 31” por estar ativo em outro projeto.

Figura 18 - Participação do usuário 155 na rede




A Figura 19 apresenta a investigação sobre a participação do “user 158” na rede. É possível observar que o usuário não está ligado a atividades e execuções de nenhum projeto. Pode-se buscar por mais informações para saber quando esse usuário foi cadastrado o sistema e quando foi seu último *login*. Possivelmente o usuário foi registrado recentemente e por isso não possui execuções ligadas a ele.

Figura 19 - Participação do usuário 158 na rede

```
$ MATCH p=(:Project)-->(:Program)-->(:Executions)-->(:User {userId: 158}) RETURN p LIMIT 1000000
```

Table

(no changes, no records)

### 4.2.4. Camada de Visualização

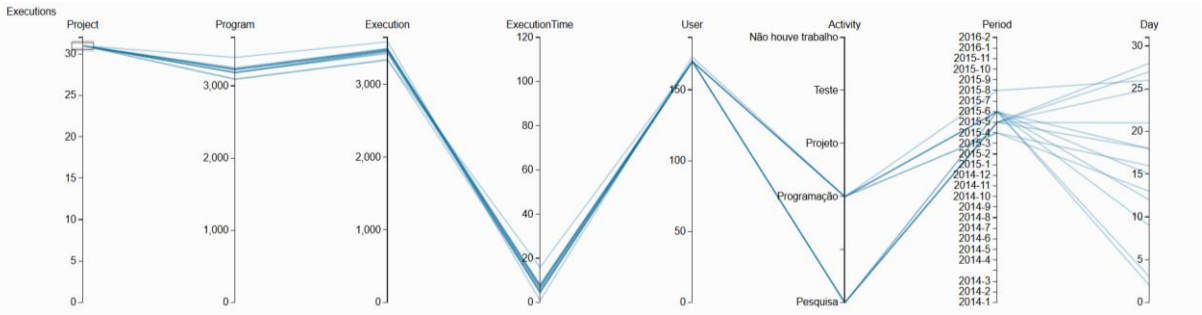
A **Camada de Visualização** ajuda na compreensão dos dados, e tem como um de seus componentes um *frontEnd* web. A **Camada de Análise de Dados** extrai conhecimento implícito dos dados, e a **Camada de Visualização** aprimora o entendimento do conhecimento extraído.

Diferentes gráficos para visualização foram utilizados para verificar o mais adequado para visualizar dados de processos de desenvolvimento de *software*. Uma visualização ampla, dinâmica e de fácil entendimento era necessária para dar suporte ao entendimento dos dados.

Desta forma, a **Camada de Visualização** conta com diferentes tipos de visualização. Uma das visualizações associa gráficos a tabelas, o que permite que os dados do gráfico sejam filtrados a partir de seleções nas tabelas, como ilustrado na Figura 20, onde o "Projeto 31" é selecionado para que se mostre apenas registros relacionados a ele. Também pode ser mudada a ordem das colunas ao clicar na mesma e posicionar no lugar desejado, além de mostrar o caminho de um registro específico da tabela no gráfico ao colocar o cursor em cima de um dado específico, como é ilustrado na Figura 21. Dependendo da coluna selecionada, outras visualizações são sugeridas na parte de baixo, no canto direito da visualização, assim como é representado nas Figuras 20 e 21. Nessas figuras cada coluna representa um tipo de dado, e as arestas representam a relação entre eles.

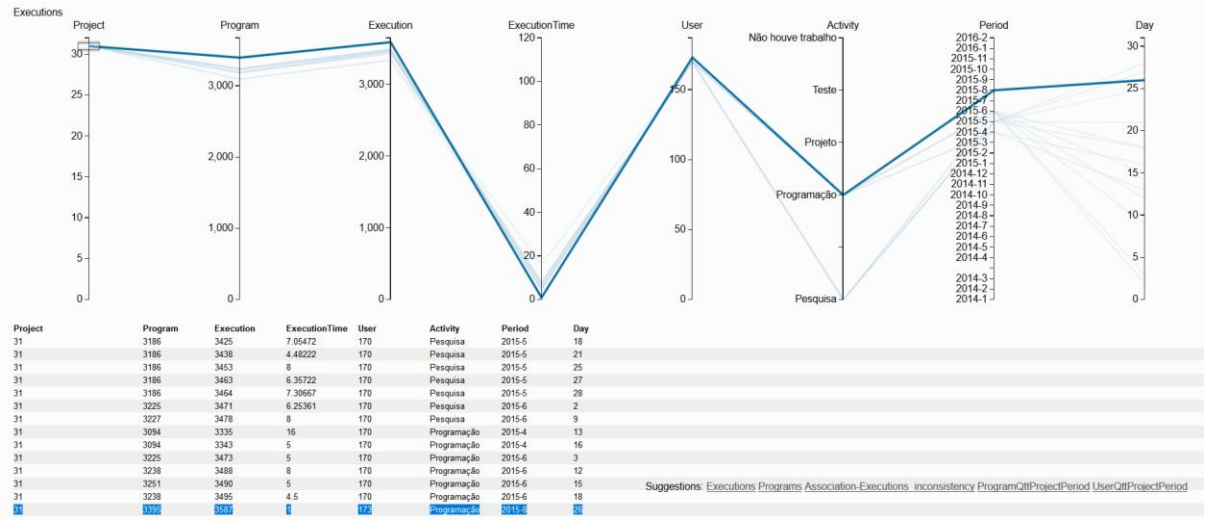
Ainda nessas Figuras é possível observar mais um tipo de visualização para complementar as análises anteriores sobre o "Projeto 31". Nessa visualização as atividades desenvolvidas no projeto são relacionadas a programação e pesquisa. Além disso, o período em que as atividades foram executadas está dentro do espectro em que o "user 158" está cadastrado na rede. Com isso, pode ser observado que há algo de errado no comportamento do usuário, e o gerente deve tomar alguma decisão, como contactar o chefe direto do usuário, ou contactar o usuário para que ele seja questionado sobre sua inatividade na rede, apesar de ter logado recentemente no sistema, e investigar as causas da sua inatividade, como, por exemplo, ele pode ter executado atividades, mas as mesmas não foram registradas no sistema. Ou talvez o usuário não trabalhe mais para a empresa, mas teve seu login utilizado. A partir da constatação do problema baseada nos dados, o gerente do projeto tem evidências de comportamentos que podem estar prejudicando o projeto, e pode tomar decisões para melhorar execuções futuras.

Figura 20 - Visualização com filtragem sobre atividades e execuções do Projeto 31




| Project | Program | Execution | ExecutionTime | User | Activity | Period | Day |
|---|---|---|---|---|---|---|---|
| 31 | 3186 | 3425 | 7.05472 | 170 | Pesquisa | 2015-5 | 18 |
| 31 | 3186 | 3438 | 4.48222 | 170 | Pesquisa | 2015-5 | 21 |
| 31 | 3186 | 3453 | 8 | 170 | Pesquisa | 2015-5 | 25 |
| 31 | 3186 | 3463 | 6.35722 | 170 | Pesquisa | 2015-5 | 27 |
| 31 | 3186 | 3464 | 7.30667 | 170 | Pesquisa | 2015-5 | 28 |
| 31 | 3225 | 3471 | 6.25361 | 170 | Pesquisa | 2015-6 | 2 |
| 31 | 3227 | 3478 | 8 | 170 | Pesquisa | 2015-6 | 9 |
| 31 | 3094 | 3335 | 16 | 170 | Programação | 2015-4 | 13 |
| 31 | 3094 | 3343 | 5 | 170 | Programação | 2015-4 | 16 |
| 31 | 3225 | 3473 | 5 | 170 | Programação | 2015-6 | 3 |
| 31 | 3238 | 3488 | 8 | 170 | Programação | 2015-6 | 12 |
| 31 | 3251 | 3490 | 5 | 170 | Programação | 2015-6 | 15 |
| 31 | 3238 | 3495 | 4.5 | 170 | Programação | 2015-6 | 18 |
| 31 | 3395 | 3687 | 1 | 173 | Programação | 2015-8 | 26 |

Figura 21 - Visualização com filtragem sobre atividades e execuções do Projeto 31, e filtragem sobre uma linha da tabela




### 4.2.5. Camada de Serviços

A **Camada de Serviços** tem o objetivo de permitir a comunicação da arquitetura com outras aplicações relacionadas, e fornecer análises adequadas a partir dos dados processados. Neste sentido, a Camada de Serviço é responsável por gerar as visualizações e torná-las disponíveis em arquivos *json(JavaScript Object Notation)*[13] que podem ser consumidos por outras aplicações ou ferramentas. A Camada de Serviços também provê métodos que permitem a realização de buscas em *cypher* na rede complexa, e seus resultados são traduzidos para *json*. Além disso, essa camada consegue realizar buscas em *sparql*[14] na ontologia para extrair conhecimento da mesma.

## 4.3. RESUMO DO CAPÍTULO

A arquitetura apresentada no presente capítulo apresenta uma abordagem para extrair conhecimento implícito em dados de processos de desenvolvimento de *software* através da utilização de modelo de proveniência, ontologia e rede complexa. A arquitetura também conta com o suporte de visualizações para uma melhor compreensão dos dados.

A arquitetura visa dar suporte à extração de conhecimento implícito nos dados, além de auxiliar na tomada de decisão. A extração de conhecimento é feita através de análises ontológicas, que permitem a inferência de conhecimento implícito a partir de regras da ontologia, e também através de análises estruturais a partir de uma rede complexa.

[13] https://www.json.org/
[14] Sparql é a linguagem de query para RDF https://www.w3.org/TR/rdf-sparql-query/

O presente capítulo apresentou a arquitetura, suas camadas e exemplos de utilização de cada uma das camadas. O capítulo 5 apresenta a avaliação da arquitetura através de dois estudos de caso, com duas etapas, a primeira é realizada através de um experimento automatizado, onde são realizadas análises da ontologia e rede complexa, e a segunda parte do experimento conta com julgamento humano para avaliar as análises e a abordagem.

# 5. AVALIAÇÃO DA ARQUITETURA ONTOCOMPLEX

## 5.1. INTRODUÇÃO

O presente capítulo apresenta a avaliação da arquitetura OntoComplex, realizada em duas etapas. Na primeira etapa foi realizada uma avaliação com parte de dados reais de uma empresa de desenvolvimento de *software* de grande porte (> de 100 empregados). O objetivo foi verificar se os modelos conseguiam responder de maneira adequada às questões de competência definidas na especificação dos mesmos e como isso responder também as duas primeiras questões de pesquisa definida para esta dissertação (QP1) "É possível responder às questões de competência através de buscas na rede complexa e ontologia?" e (QP2) "A OntoComplex é capaz de extrair conhecimento implícito nos dados?". Na segunda etapa, dois estudos de caso com dados reais de empresas de desenvolvimento de *software* foram executados, com o objetivo de verificar a viabilidade da arquitetura OntoComplex e sua capacidade de extração de conhecimento implícito e responder as questões de pesquisa (QP2) "A OntoComplex é capaz de extrair conhecimento implícito nos dados?", reforçando os resultados obtidos na etapa 1 da avaliação e (QP3) "O conhecimento extraído é capaz de dar suporte ao entendimento sobre os dados através de visualizações?". O primeiro estudo se trata de um estudo piloto, descrito em (FALCI et al., 2018), e o segundo um estudo regular. Ao final do estudo regular, uma entrevista com um gerente de projetos relacionado a empresa que disponibilizou os dados foi realizada, com o intuito de verificar a aplicabilidade da arquitetura. Na descrição dos estudos, são detalhados a metodologia usada para avaliação, ameaças a validade e resultados.

## 5.2. DEFINIÇÃO DO ESCOPO DA AVALIAÇÃO

O escopo dessa avaliação foi definido baseado no método GQM (VAN SOLINGEN et al., 2002) de acordo com o seguinte:

**Analisar:** A abordagem OntoComplex

**Com o objetivo de:** avaliar sua viabilidade

**Em relação à:** extração de conhecimento implícito sobre os dados de processos de *software* e fornecimento de informações relevantes para compreensão do comportamento dos dados através de visualizações do ponto de vista de gerentes de projetos.

De acordo com o escopo definido foram derivadas as questões de pesquisa:

- (QP1). É possível responder às questões de competência através de buscas na rede complexa e ontologia?
- (QP2). A OntoComplex é capaz de extrair conhecimento implícito dos dados?
- (QP3). O conhecimento extraído é capaz de dar suporte ao entendimento sobre os dados através de visualizações?

A avaliação da abordagem OntoComplex foi desenvolvida em 2 etapas, assim como descrito na Tabela 14:

Tabela 14 - Etapas da avaliação da Ontocomplex

| Etapa | Parte | Descrição | Objetivo |
|---|---|---|---|
| **Etapa 1** | | Avaliação com dados parciais do estudo regular. | Responder às questões de competência e avaliar a capacidade da OntoComplex de extrair conhecimento implícito, respondendo às questões de pesquisa QP1 e QP2 e observar pontos de melhoria nos modelos. |
| **Etapa 2** | Parte 1 | Estudo piloto com dados de processos reais de uma empresa de desenvolvimento de *software* de médio porte (cerca de 25 empregados) | Avaliar a OntoComplex , suas análises e visualizações, com o objetivo de encontrar possíveis ajustes e pontos de melhoria. |
| | Parte 2 | Estudo regular utilizando dados de processos de desenvolvimento de uma empresa de grande porte (> 100 empregados) | Avaliar a segunda versão da OntoComplex, aprimorada após o estudo piloto. |
| | Parte 3 | Avaliação com gerente de projetos | Avaliação do suporte da abordagem para o entendimento dos dados. Com as três partes dessa etapa de análises, é possível responder as questões de pesquisa QP2 e QP3 |

## 5.3. ETAPA 1

Nesta primeira etapa, o objetivo foi verificar se com o uso dos modelos detalhados no capítulo 3, é possível responder as Questões de Competência e consequentemente avaliar as QP 1 e 2 desta dissertação.

### 5.3.1. Definição do Estudo

Avaliar se os modelos especificados no contexto da arquitetura OntoComplex, instanciados com dados reais de processos de desenvolvimento de *software*, conseguem responder de maneira adequada as questões de competência definidas na sua criação.

### 5.3.2. Planejamento do Estudo

Conforme já dito, dados parciais de uma empresa de desenvolvimento de *software* de grande porte (> 100 empregados) foram instanciados nos modelos e através do uso das ferramentas de visualização da arquitetura OntoComplex, as questões de competência foram utilizadas para verificar se os modelos com os dados instanciados conseguem responder as questões de competência especificadas. Assim, as análises foram realizadas através da visualização promovida pela ferramenta Visionary, que apoia a visualização da proveniência dos dados através da ontologia, e as análises da rede complexa foram realizadas através de *queries* em cypher[15].

### 5.3.3. Cenário e Resultados

A avaliação foi realizada em um computador com as seguintes configurações: Processador Intel Core i7-7700HQ CPU 2.80GHz 2.81GHz, Memória RAM 8GB, Windows 10. Como mencionado anteriormente, questões foram formuladas sobre o contexto de processos de desenvolvimento de *software*, com o objetivo de compreender melhor o cenário e extrair conhecimento, a partir das respostas as Questões de Competência (QC) definidas no capítulo 3. Estas questões foram formuladas para verificar se a partir dos modelos instanciados é possível extrair conhecimento implícito através de análises da ontologia e rede complexa baseadas no ProvONEExt.

Para tanto são feitas análises complementares entre a ontologia e a rede complexa. Na rede complexa são feitas consultas relativas a relevância de um nó na rede (através das arestas da rede), enquanto na ontologia são feitas consultas para avaliar a proveniência dos dados (através das propriedades) e entender suas relações.

### 5.3.4. Questões de Competência

As questões de competência foram formuladas visando avaliar cada uma das ligações da ontologia e da rede complexa.

[15] https://neo4j.com/developer/cypher-query-language/

#### *5.3.4.1. Questões de Competência relacionadas a Análises Estruturais - Redes Complexas*

A. **Quantas execuções estão ligadas a cada usuário?**

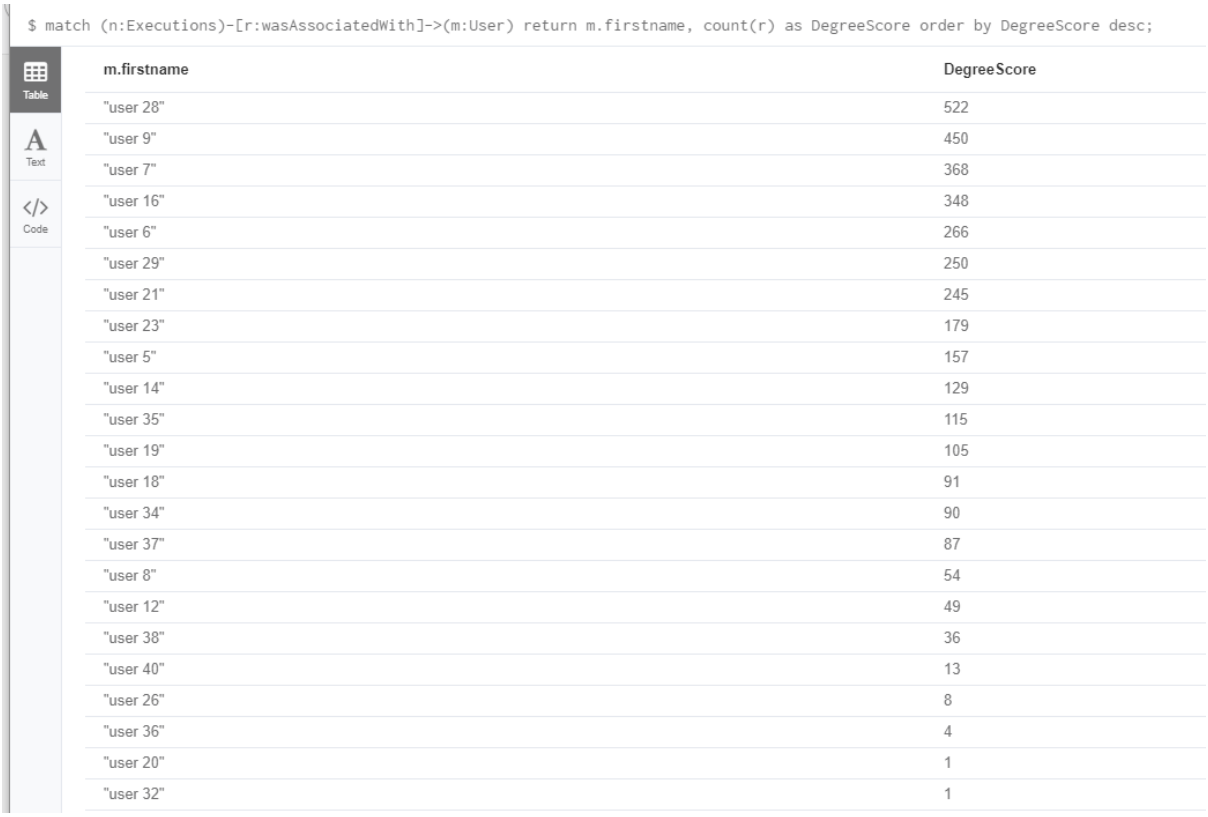
Esta questão de competência é possível ser respondida utilizando a relação causal *WasAssociatedWith*. Essa relação é uma relação do modelo ProvONEExt que também é utilizada na rede complexa. Ela associa execuções aos usuários responsáveis pelas mesmas. No caso da rede complexa, essa relação é muito importante para avaliar o grau de entrada de um nó de usuário. Quanto maior a quantidade de relações desse tipo recebidas por um usuário, maior seu grau de entrada, e consequentemente maior seria o impacto de uma possível remoção desse nó da rede, principalmente se as execuções ligadas a ele possuírem muitas horas de execução, ou dependendo do tipo de atividade desenvolvida nessa execução (e a importância desse tipo de atividade em um projeto).

Com essa consulta, que é uma consulta geral que busca a distribuição de execuções por todos os usuários da rede, é possível observar a distribuição de execuções por usuário, e se há algum comportamento estranho nessa distribuição. Por exemplo, na Figura 22 é possível observar que enquanto o “user 28” é responsável por 522 execuções, e o “user 9” é responsável por 450, a maioria dos usuários possui menos de 200 execuções, e alguns possuem menos de 20, o que torna a distribuição desbalanceada.

A partir dessa constatação outras consultas podem ser realizadas para verificar o porquê dessa distribuição heterogênea, como “Quando ocorreu a primeira e última execução do ‘user 28’?” “Quando ocorreram as execuções do ‘user40’?”. Os nós de execuções possuem as datas de execução, como pode ser observado na Figura 23. Dessa forma é possível investigar a quanto tempo um usuário atua na rede e se ainda está ativo. Caso suas execuções sejam antigas, possivelmente o usuário não está mais ativo na rede, e caso seja um novo usuário ele pode não possuir muitas execuções por seu tempo de atividade.

Usuários que possuem muitas execuções podem causar um maior impacto caso sejam removidos da rede, pois eram responsáveis por grande parte das execuções.

Figura 22 - Distribuição de execuções por usuários da rede

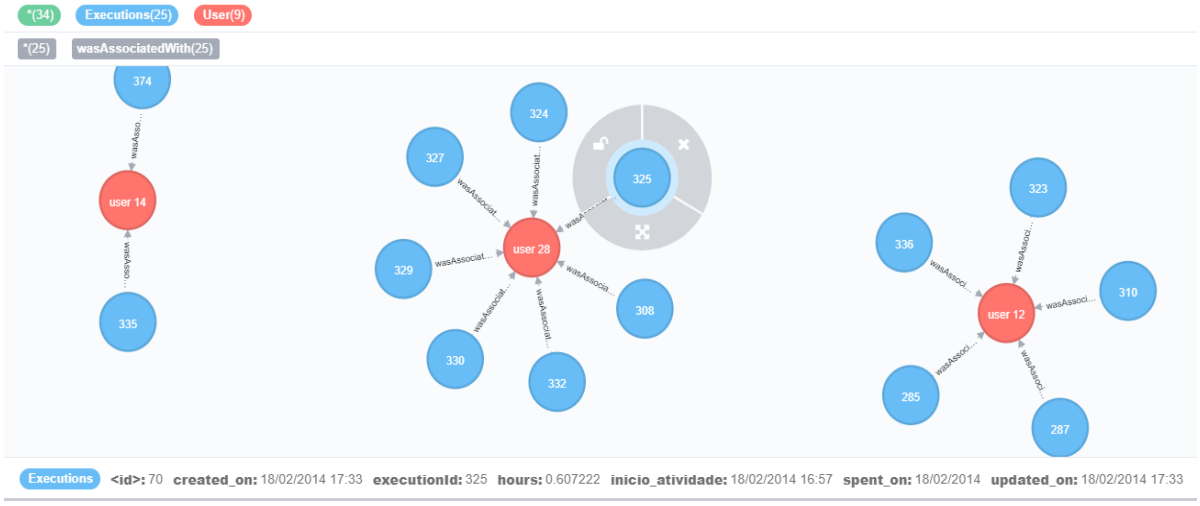


```
$ match (n:Executions)-[r:wasAssociatedWith]->(m:User) return m.firstname, count(r) as DegreeScore order by DegreeScore desc;
```

Table | Text | Code

| m.firstname | DegreeScore |
|---|---|
| "user 28" | 522 |
| "user 9" | 450 |
| "user 7" | 368 |
| "user 16" | 348 |
| "user 6" | 266 |
| "user 29" | 250 |
| "user 21" | 245 |
| "user 23" | 179 |
| "user 5" | 157 |
| "user 14" | 129 |
| "user 35" | 115 |
| "user 19" | 105 |
| "user 18" | 91 |
| "user 34" | 90 |
| "user 37" | 87 |
| "user 8" | 54 |
| "user 12" | 49 |
| "user 38" | 36 |
| "user 40" | 13 |
| "user 26" | 8 |
| "user 36" | 4 |
| "user 20" | 1 |
| "user 32" | 1 |

Figura 23 - Detalhes dos nós da rede




**B. Quais tipos de atividades são desenvolvidas?**

Esta questão de competência pode ser respondida utilizando a relação causal *HasActivity*. Essa relação ocorre entre um nó de execução e um nó do tipo *enumeration*, onde são enumerados todos os tipos de atividades que podem ser executados em uma execução. Através dessa relação é possível obter o grau de entrada de cada um dos tipos de atividade desenvolvidas, pertencentes ao tipo *enumeration*, e consequentemente a relevância desses tipos na rede complexa.

Com a resposta dessa questão de competência é possível saber quais tipos de atividades são desenvolvidos na rede, e então é possível investigar melhor cada uma delas. A Figura 24 representa quais tipos de atividades eram desenvolvidos na empresa cujos dados foram utilizados nessa avaliação.

Uma possível pergunta complementar seria: “Quantas execuções estão ligadas a cada um dos tipos de atividades?”**.** Dessa forma é possível compreender a distribuição de execuções por cada tipo de atividade. Na Figura 25 é possível observar a resposta dessa pergunta de acordo com a rede complexa em questão. A maioria das execuções se trata de programação, enquanto as execuções equivalentes a testes são quase ¼ desse valor. Esse seria um fator importante para ser analisado para averiguar se testes suficientes estão sendo realizados. Além disso, na distribuição dos dados pode-se observar que 96 execuções estão relacionadas com a atividade “Não houve trabalho”. É possível investigar quanto tempo foi gasto nessas atividades, se não houve trabalho, e caso tenha sido um tempo considerável, avaliar quais usuários estavam relacionados a essas execuções, para saber o que houve de errado, e tentar não repetir os mesmos padrões em futuras execuções.

Figura 24- Tipos de atividades da rede

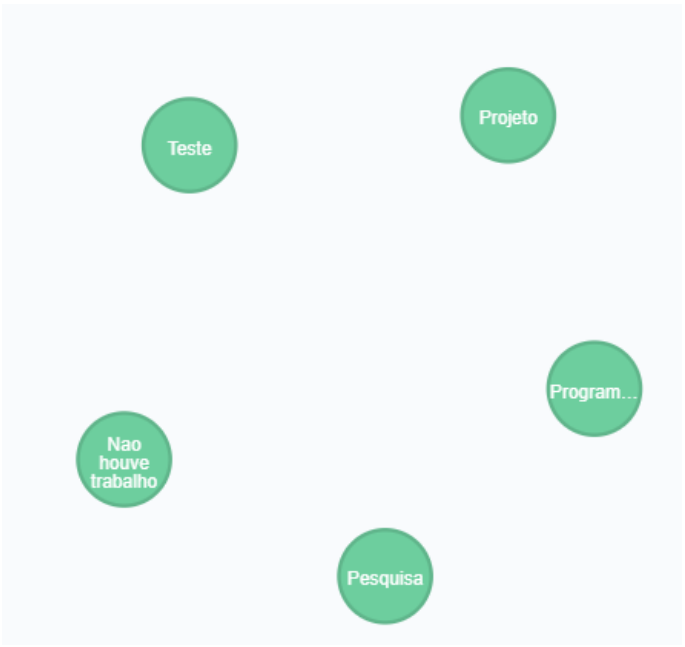



Figura 25 - Distribuição de nós por tipo de atividade

```
$ MATCH p=()-[r:hasActivity]->(a) RETURN a.name, count(r) as ExecQtt
```

Table | Text | Code

| a.name | ExecQtt |
|---|---|
| "Pesquisa" | 614 |
| "Nao houve trabalho" | 96 |
| "Teste" | 543 |
| "Projeto" | 33 |
| "Programacao" | 2282 |

**C. Quais atividades (programs) possuem maior quantidade de execuções?**

Essa questão de competência pode ser respondida através da relação *HasExecution*, que ocorre entre atividades (nós do tipo *program*) e suas execuções (nós tipo *execution*). Essa relação permite a investigação do grau de saída das atividades, e consequentemente que se saiba qual possui maior quantidade de execuções, e a qual causaria maior o impacto em caso de uma possível remoção da rede.

Quando uma atividade possui uma quantidade muito grande de execuções, pode ser investigado o motivo para tal situação. No caso oposto, quando uma atividade possui uma quantidade muito pequena de execuções, principalmente se não for o esperado de acordo com sua prioridade, e a atividade já tiver sido encerrada, motivos para que ela tenha sido bem-sucedida podem ser estudados para que os fatores de sucesso possam ser implementados em futuras execuções.

Ou seja, a resposta dessa questão permite que se analise a produtividade de atividades, se seu desempenho está ocorrendo como esperado, além de avaliar quais atividades demandam maior esforço para serem desenvolvidas, e se houve alguma discrepante, investigar o porquê houveram tantas execuções ligadas a uma mesma atividade.

A Figura 26 ilustra a resposta dessa pergunta através da rede. Análises relacionadas podem ser realizadas, como "**Ao todo, quanto tempo foi gasto para desenvolver essa atividade?"**, **"Quais usuários estavam relacionados às execuções de maior prioridade?"**, **"Qual era a prioridade dessa atividade?"**, entre outras análises. Dessa forma, é possível descobrir o cenário e entender o porquê de tantas execuções, e evitar que futuras atividades demandem tanto tempo para serem desenvolvidas.

Figura 26 - Distribuição de execuções por atividade

| a.programId | OutcomeDegreeScore |
|---|---|
| 488 | 52 |
| 2064 | 41 |
| 1663 | 40 |
| 870 | 37 |
| 850 | 34 |
| 2050 | 31 |
| 2067 | 31 |
| 2102 | 29 |
| 2769 | 24 |
| 1656 | 24 |
| 1875 | 24 |
| 2068 | 23 |
| 2272 | 22 |
| 2104 | 21 |
| 865 | 20 |
| 2940 | 20 |

**D. Quais trackers possuem maior quantidade de atividades (programs) relacionadas?**

Essa pergunta pode ser respondida através da relação *HasTracker*, que ocorre entre uma atividade (nós **program**) e nós do tipo *tracker* (que possuem informações adicionais sobre atividades, e atuam como rastreadores). Essa relação permite que se investigue o grau de entrada de *trackers*, e consequentemente que se saiba qual possui maior importância na rede e qual seria o impacto se esse tipo de *tracker* fosse removido da rede. Além disso, essa informação permite que se saiba qual é a distribuição de atividades por *trackers*.

Respondendo essa pergunta é possível investigar quais tipos de *trackers* possuem mais atividades relacionadas a ele. De acordo com a resposta obtida na rede, ilustrada na Figura 27, a maioria das atividades são relacionados a *bugs*, quase 5 vezes mais do que os relacionados a melhorias. Essa distribuição mostra que a maioria das correções é feita devido a erros, e não a manutenções preventivas.

Uma investigação pode ser realizada para averiguar qual impacto seria gerado caso mais correções preventivas fossem feitas, ao invés de manutenções corretivas. Caso houvesse o desejo de mudar o comportamento em relação às manutenções, esse seria um ponto de partida para análises mais profundas, como por exemplo, investigar quais usuários estão relacionados a cada tipo de atividade, quanto tempo é gasto nas execuções de cada tipo, etc. Dessa forma é possível descobrir o que valeria mais a pena para a empresa.

Figura 27 - Distribuição de nós por *Tracker*

```
$ match (a:Program)-[b:hasTracker]->(c:Tracker) return c.name,count(a) as DegreeScore order by DegreeScore;
```

Table | Text | Code

| c.name | DegreeScore |
|---|---|
| "Continua" | 7 |
| "FalhaNoTeste" | 14 |
| "Requisito" | 26 |
| "Subtarefa" | 37 |
| "Gerencial" | 55 |
| "Carga" | 98 |
| "Contratual" | 162 |
| "Bug DEV" | 210 |
| "Refactoring" | 224 |
| "Melhoria" | 441 |
| "Bug" | 2169 |

**E. Quais prioridades possuem a maior quantidade de atividades?**

Essa questão de competência pode ser respondida através da busca pelos nós que possuem a relação *HasPriority*, que é uma relação entre nós de atividades e nós do tipo *enumeration*, que como mencionado anteriormente possuem enumerações, e nesse caso enumerações de tipos de prioridades. Através dessa investigação é possível descobrir o grau de entrada de cada tipo de prioridade, compreender como é a distribuição de atividades por prioridade, consequentemente saber qual é mais relevante na rede, e se seria maior o impacto caso ocorresse a remoção desse nó.

Com essa resposta é possível investigar a distribuição de prioridades de atividades na rede. A Figura 28 representa um cenário real da importância desta questão, onde muitas requisições têm prioridade alta e pouquíssimas têm prioridade baixa. Uma possível análise seria investigar se foi por falta de manutenção preventiva, o que acabou gerando muito mais manutenções corretivas que têm alta prioridade, ou o que causa tantas entradas com prioridade alta.

Figura 28 - Distribuição de nós por prioridade

| ProgramQuantity | Priority |
|---|---|
| 128 | "Imediata" |
| 441 | "Urgente" |
| 381 | "Altissima" |
| 1944 | "Alta" |
| 515 | "Normal" |
| 34 | "Baixa" |

**F. Quais categorias possuem relação com a maior quantidade de atividades?**

Essa questão de competência pode ser respondida através da busca por relações do tipo *HasCategory*, que é uma relação entre atividades (nós do tipo *Program*) e categorias (nós do tipo *ProgramCategories*). A resposta dessa pergunta apresenta o grau de entrada de categorias, e permite a compreensão da distribuição de atividades por categorias. Quanto maior o grau de entrada, maior a importância de uma categoria na rede, e maior seria o impacto da remoção desse nó da rede.

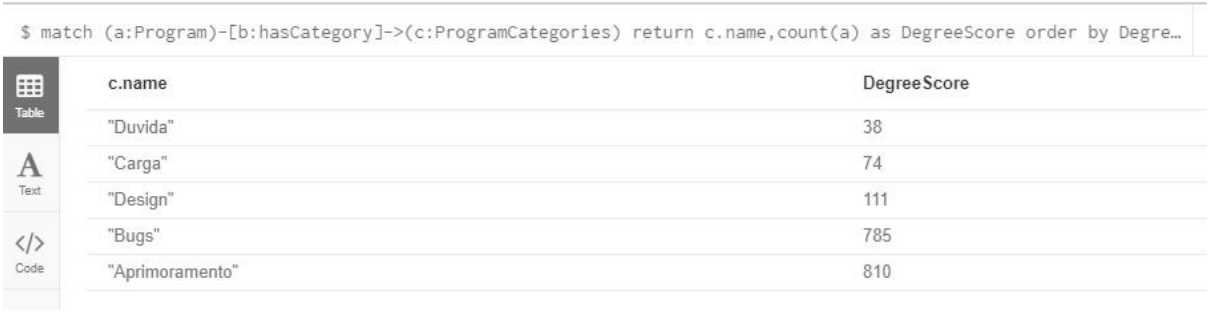
De acordo com a resposta obtida nos dados utilizados, apresentada na Figura 29, há uma grande quantidade de requisitos para aprimoramento, e quase a mesma quantidade relacionada a “bugs”, enquanto quase 7 vezes menos registros se relacionam com a fase de projeto. O gerente pode investigar se essa distribuição está dentro do esperado para os projetos.

Dentro do resultado obtido com essa busca na rede, representado na Figura 29, é possível observar uma semelhança dos nomes das categorias se comparado aos *trackers* da Figura 27. Seria interessante investigar de onde provêm as respostas, se há alguma diferença entre os conceitos de *Tracker* e *Categories*, e porque são homônimos.

Além disso, é possível observar que o comportamento da rede se mostra diferente nos dois resultados, já que na Figura 27 aparentemente há mais atividades ligadas a *Bugs* e na Figura 29 há mais relacionamentos com Aprimoramento.

Análises complementares podem ser feitas para verificar as datas das atividades de cada um dos retornos, e checar se houve uma mudança no comportamento dos dados, se o uso de categorias foi descontinuado, se há uma interseção entre os dados, etc.

Figura 29 - Distribuição de nós por categoria

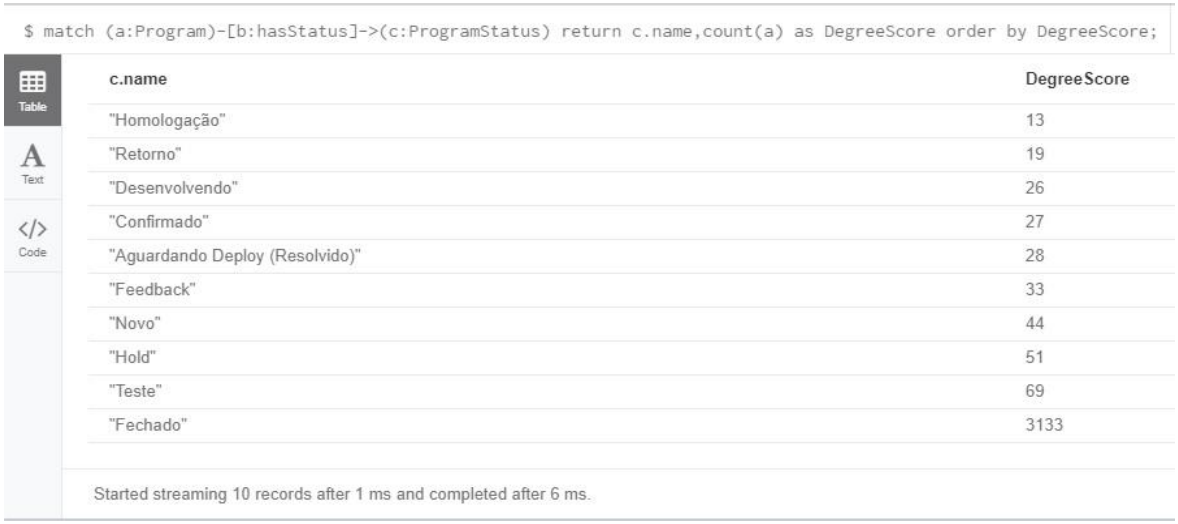



**G. Quais status possuem a maior quantidade de atividades?**

Essa questão de competência pode ser respondida através da relação *HasStatus*, que ocorre entre atividades (nós do tipo *Program*) e status (nós do tipo *ProgramStatus*). Quanto maior a quantidade de relações de um status, maior seu grau de entrada, e maior sua importância na rede. Essa resposta permite a compreensão do comportamento de atividades segundo os status que elas se relacionam.

A Figura 30 apresenta os resultados da rede, quantos processos estão abertos, e de quais tipos eles são. Dessa forma é possível fazer análises encadeadas, para investigar, por exemplo, quais usuários estão ligados a testes, ou desenvolvimento. Além disso, é apresentada a quantidade de atividades que já estão fechados (concluídas), que são a grande maioria.

Figura 30 - Distribuição de nós por Status




H. **A quantas atividades cada projeto está relacionado?**

Essa questão de competência pode ser respondida através da relação *HasIssue*, que ocorre entre um projeto e suas atividades. Através dessa relação é possível buscar quais projetos possuem maior grau de saída. Quanto maior o grau de saída de um projeto, maior é a produtividade do projeto, maior a probabilidade deste projeto estar mais desenvolvido, e maior o impacto da remoção desse projeto da rede. Projetos com maior quantidade de atividades possuem maior possibilidade de ter casos de onde se pode extrair conhecimento para aplicar a execuções futuras.

Sabendo a quantidade de atividades que foram realizadas para cada Projeto, é possível investigar quais projetos requerem mais atividades para chegarem a seu ponto atual, o que pode ser uma informação interessante para gerentes saberem, por exemplo, quanto tempo e esforço foram demandados para se chegar a um determinado ponto nos projetos.

Além disso, análises integradas podem ser desenvolvidas. É possível investigar, como representado na Figura 31, a quantidade de atividades relacionadas a cada projeto, a quantidade total de execuções para cada uma delas, e quantas horas no total foram gastas para executar todas as atividades abertas até o momento atual da busca. Dessa forma fica mais claro o esforço gasto (em horas) para que cada um dos projetos chegasse ao ponto em que está no momento da busca. Podem ser feitas análises comparando se o esforço gasto em projetos diferentes gerou um avanço proporcional, e se algum projeto gastou em geral mais tempo que outros e está mais imaturo, o que pode ser feito para melhorar, e porque não houve evolução como nos outros. Ou quais foram os fatores que influenciaram o projeto positivamente, para que futuros projetos possam se beneficiar desse conhecimento.

Figura 31 - Relação de projetos, atividades, execuções e horas

| Project | Issues | Executions | Hours |
|---|---|---|---|
| 14 | 1133 | 1133 | 38789.635059400025 |
| 16 | 1246 | 1246 | 10273.9110553 |
| 17 | 221 | 221 | 6166.5373887000005 |
| 31 | 14 | 14 | 3205.9439999999995 |
| 32 | 97 | 97 | 2016.181 |
| 29 | 441 | 441 | 1656.9886669999996 |
| 30 | 233 | 233 | 582.7499999999999 |
| 19 | 101 | 101 | 552.365389 |
| 20 | 58 | 58 | 516.267 |
| 26 | 3 | 3 | 115.951556 |
| 24 | 16 | 16 | 20.861667 |
| 15 | 1 | 1 | 3.35 |
| 27 | 2 | 2 | 2.5 |

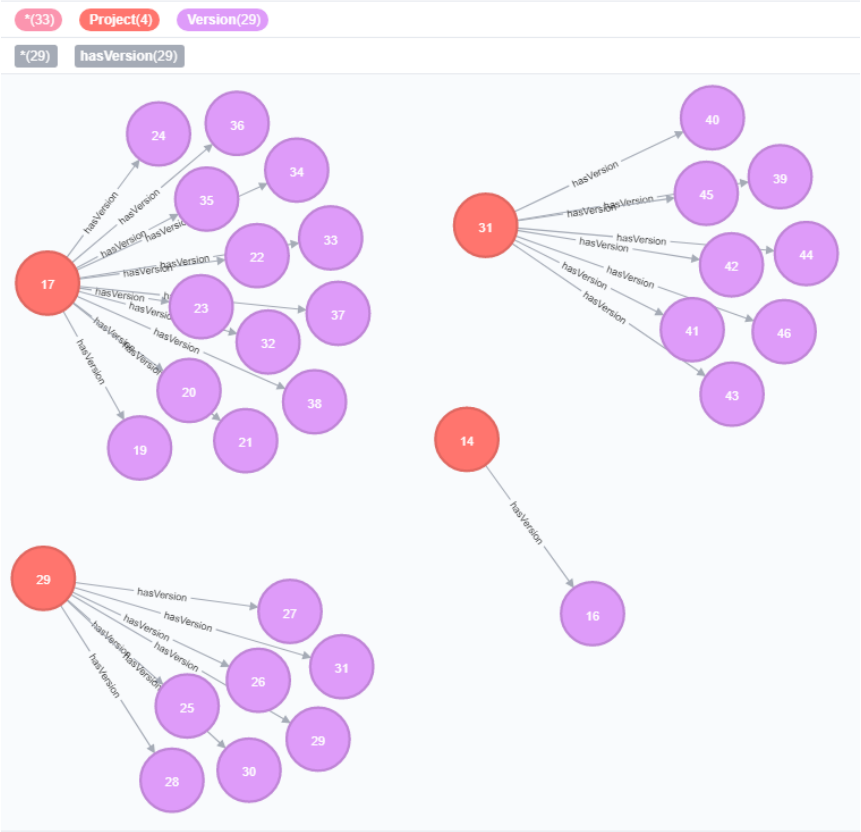
**I. Quais projetos possuem mais versões?**

Essa questão de competência pode ser respondida através da relação *HasVersion*, que ocorre entre projetos e versões. Através dessa investigação é possível obter o grau de saída de projetos relativo a versões. Quanto maior o grau de saída de um projeto em relação às versões, provavelmente maior é sua maturidade, e maior sua importância na rede.

Com essa resposta é possível visualizar quais projetos possuem mais versões ligadas a eles. A Figura 32 apresenta os projetos, representados em vermelho, e as versões de cada um deles. Os nós de versões possuem dados sobre datas de criação, atualizações e versão de lançamento. Dessa forma, caso necessário, é possível investigar quais versões foram entregues com atrasos e o porquê.

Além disso, é possível realizar uma busca pela proveniência de projetos ou versões na ontologia ProvOenExt-O. O conhecimento extraído através dessas análises pode identificar problemas que ocorreram durante o desenvolvimento de determinadas versões para aprimorar versões futuras.

Figura 32 - Distribuição de versões por projetos




### *5.3.4.2. Questões de Competência Relacionadas a Análises Semânticas - Ontologia*

Diferente das respostas obtidas através da rede complexa, que são estruturais e feitas de acordo com a relevância dos nós na rede, as análises ontológicas são centradas em um indivíduo específico e relacionadas ao contexto semântico dos mesmos. Através da utilização da ferramenta Visionary, é possível filtrar um indivíduo e visualizar as relações ligadas a ele. Além disso, é possível omitir tipos de nós da visualização, para ficar mais evidente a proveniência de outros tipos. As investigações sobre os dados devem ser feitas de forma conjunta, usando a rede complexa, ontologia e visualizações. Através da rede complexa e visualizações, é possível identificar padrões de comportamento anômalos nos dados, e a ontologia serve como um segundo passo para investigar a proveniência relacionada aos indivíduos que apresentam esse comportamento. Os dados utilizados para análise nesta seção são referentes apenas a 1 projeto, para facilitar a compreensão das análises.

**A. Quais execuções foram associadas a um usuário específico (“user173”, “user170”)?**

Essa questão de competência pode ser respondida através da relação *Associates*. Essa relação é uma propriedade da ontologia entre usuário (classe *User*) e execução (classe

*Execution*). Essa relação é inferida pela ontologia quando a relação inversa (*wasAssociatedWith)* é satisfeita.

Na Figura 33 é possível observar que na amostra de dados utilizada o “user173” possui apenas 1 ligação “*Associates*”, ou seja, está ligado apenas a uma execução. Em contrapartida, na Figura 34 é possível observar que o “user170” possui 12 ligações “*Associates*”.

Essa resposta auxilia no entendimento da distribuição de execuções por usuários. Além disso, como apresentado na Figura 35, é possível filtrar elementos de algum tipo, como, por exemplo, entidades, visualizar a proveniência de atividades e agentes (usuários) como um todo. Essas visualizações permitem uma compreensão melhor do comportamento dos dados, que não seria possível na base original.

Levando em conta a distribuição das execuções por usuários nesse projeto, análises complementares podem ser executadas para compreender o porquê um usuário possui tantas execuções ligadas a ele, enquanto outro possui apenas uma, e vários outros usuários não possuem ligações com execuções, como pode ser observado na Figura 35. Cenários possíveis que explicariam essa distribuição desbalanceada seriam que talvez os usuários com poucas ou nenhuma execução são novos no projeto, ou talvez foram realocados para outro projeto.

Figura 33 - Proveniência relacionada ao usuário 173

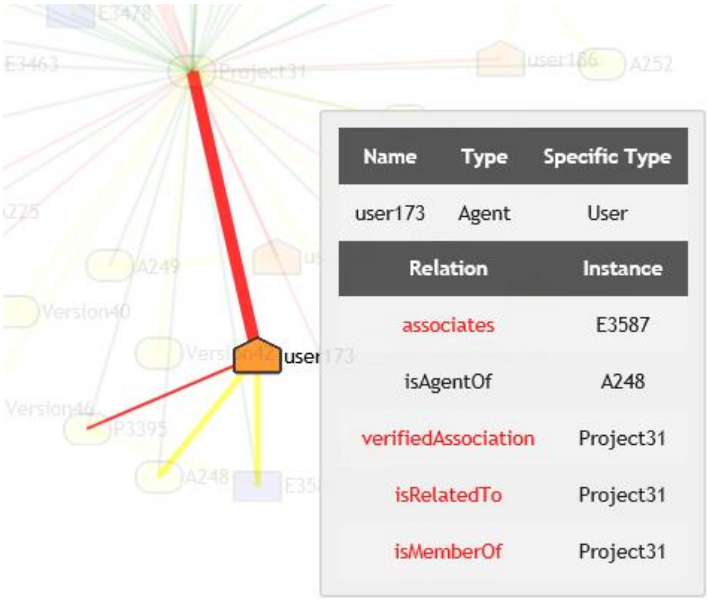
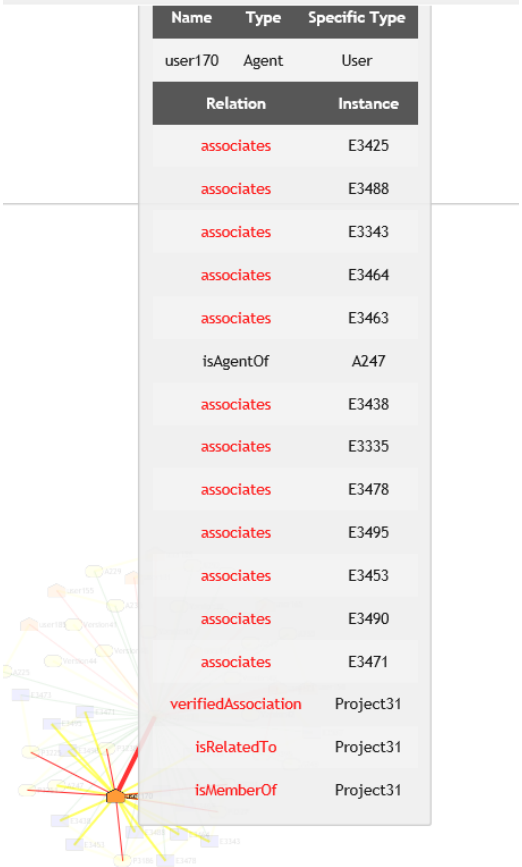


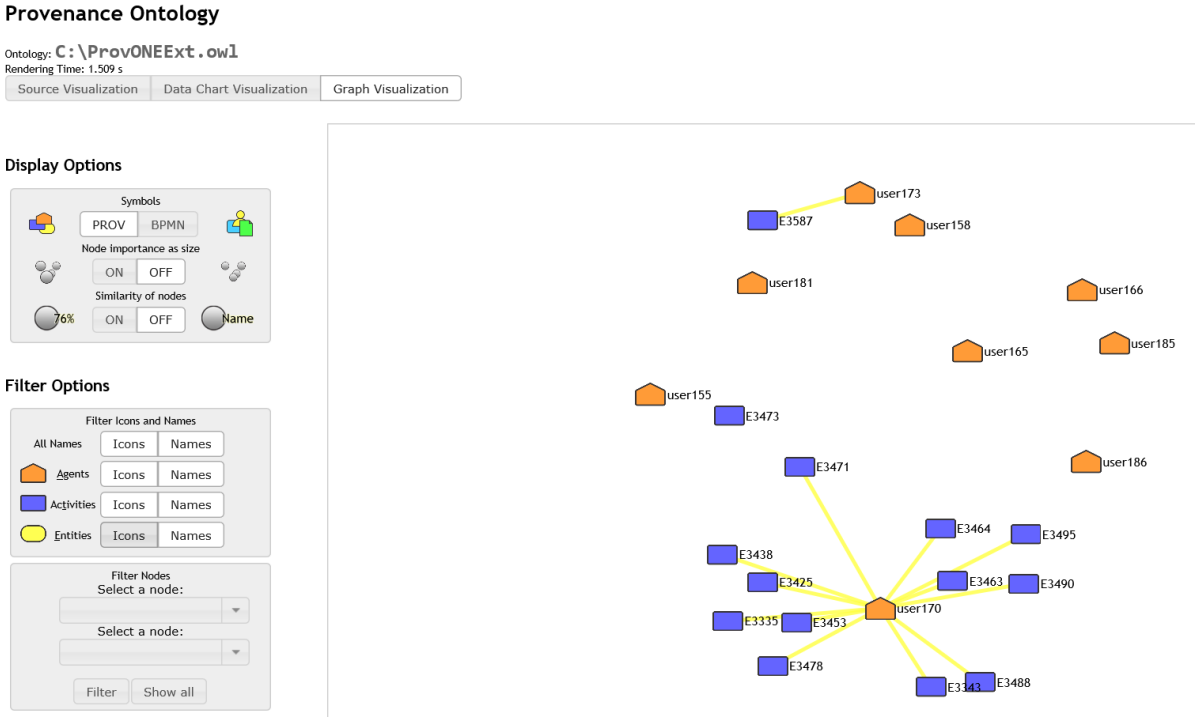
Figura 34 - Proveniência relacionada ao usuário 170



| Name | Type | Specific Type |
|---|---|---|
| user170 | Agent | User |

| Relation | Instance |
|---|---|
| associates | E3425 |
| associates | E3488 |
| associates | E3343 |
| associates | E3464 |
| associates | E3463 |
| isAgentOf | A247 |
| associates | E3438 |
| associates | E3335 |
| associates | E3478 |
| associates | E3495 |
| associates | E3453 |
| associates | E3490 |
| associates | E3471 |
| verifiedAssociation | Project31 |
| isRelatedTo | Project31 |
| isMemberOf | Project31 |

Figura 35 - Visualização do projeto 31 com filtragem de entidades na ferramenta Visionary



**B. A qual usuário a execução “E3478” foi associada?**

Essa questão de competência pode ser respondida através da relação *WasAssociatedWith.* Essa relação é originária do modelo ProvONE. Essa relação busca a proveniência de uma execução, e qual usuário a influenciou.

Na ferramenta Visionary essa resposta pode ser obtida através da filtragem do nó de execução, nesse caso específico, a execução “E3478”, como ilustrado na Figura 36. Dessa forma pode se visualizar todas as relações desse nó, inclusive a relação *wasAssociatedWith*.

Essa consulta pode ser o aprofundamento da investigação sobre uma atividade (*Program*), quando se identifica algo não esperado, como, por exemplo, quantidades excessivas de execuções, ou muito tempo para uma execução específica ocorrer. E então a investigação de qual usuário está ligado à essa execução (ou todas as execuções de uma atividade) pode ser interessante para uma investigação mais ampla. A Figura 37 mostra uma filtragem de uma atividade (*Program*) e uma execução dessa atividade, como exemplo dessa possível investigação. Além disso, futuras atividades similares a essa podem ser associados ao mesmo usuário, caso os usuários ligados a ele tenham um bom desempenho.

Figura 36 - Proveniência relacionada à execução E3478

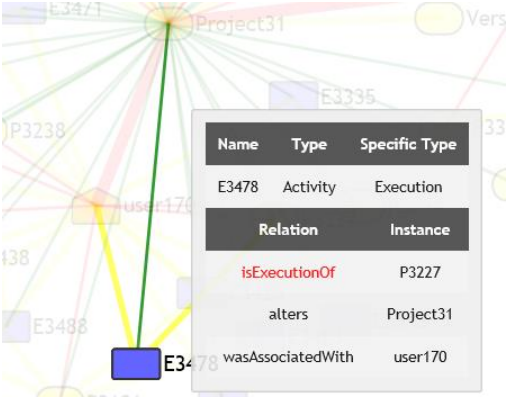
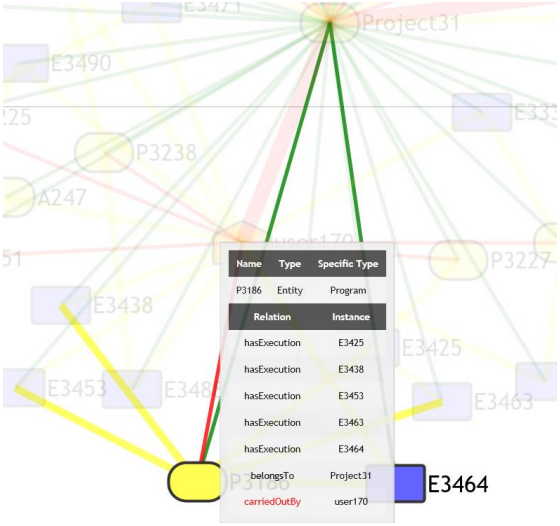



Figura 37 - Filtragem de uma atividade e uma de suas execuções na ferramenta Visionary

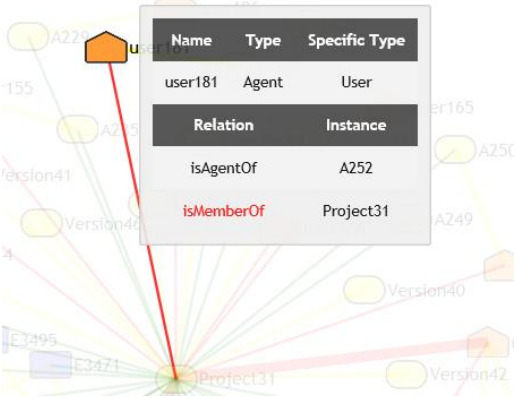


**C. A qual associação o usuário “user181” está ligado?**

Essa questão de competência pode ser respondida através da propriedade da ontologia *isAgentOf*, que pode ser inferida pela ontologia, e é inversa à propriedade *agent*. Essa relação (*isAgentOf*) indica a relação de um usuário (classe *User*) a uma associação (classe *Association*).

No modelo ProvONEExt os usuários (*Users*) são ligados oficialmente a projetos (*Projects*) através de associações (indivíduos da classe *Association*). Ou seja, para que um usuário esteja formalmente registrado em um projeto, ele deve estar ligado a uma associação através da ligação “*isAgentOf*”, e essa se liga ao projeto em questão. Portanto, com essa resposta é possível descobrir a quais associações um usuário está conectado. É possível buscar por essa relação através da filtragem de um usuário específico, como ilustrado na Figura 38.

Figura 38 - Proveniência relacionada ao usuário 181

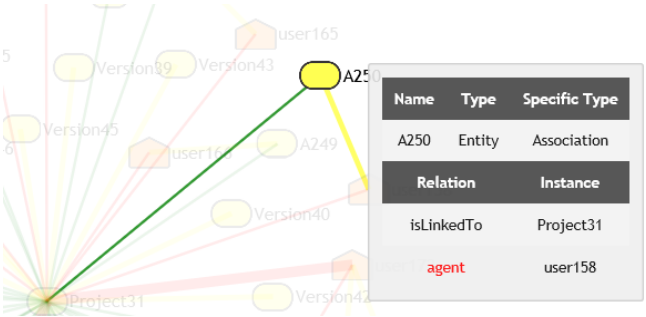



**D. A qual usuário a associação “A250” está ligada?**

Essa questão de competência pode ser respondida através da relação *Agent*, herdada do modelo ProvONE, que relaciona uma associação (*Association*) a um usuário (*User*).

A relação Agent é inversa à *isAgentOf* (mencionada na consulta anterior), e tem o objetivo de responder a qual usuário uma associação está ligada. É possível obter essa resposta através da filtragem de uma associação específica, como ilustrado na Figura 39.

Figura 39 - Proveniência relacionada à associação A250

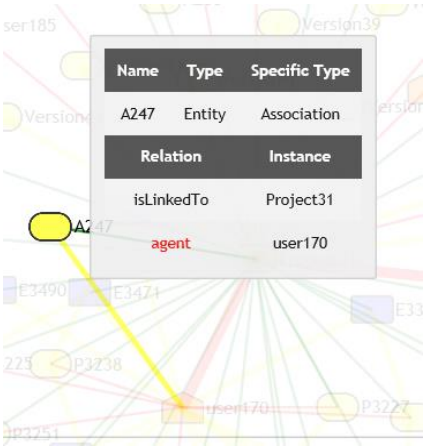


**E. A qual projeto a associação “A247” está ligada?**

Essa questão de competência pode ser respondida através da relação *isLinkedTo*, que relaciona uma associação a um projeto.

Essa resposta pode ser obtida através da filtragem de uma associação específica, como ilustrado na Figura 40, e tem como objetivo responder a qual projeto um indivíduo da classe *Association* está ligado. Essa busca pode ser importante caso haja necessidade de investigar a quais projetos um usuário está ligado formalmente, já que *isLinkedTo* é a relação entre uma *Association* e um projeto, e é utilizada na property chain *isMemberOf* (que verifica se um usuário está alocado para trabalhar em um projeto). Mais detalhes sobre essa relação são apresentados na QC G.

Figura 40 - Proveniência relacionada à associação A247

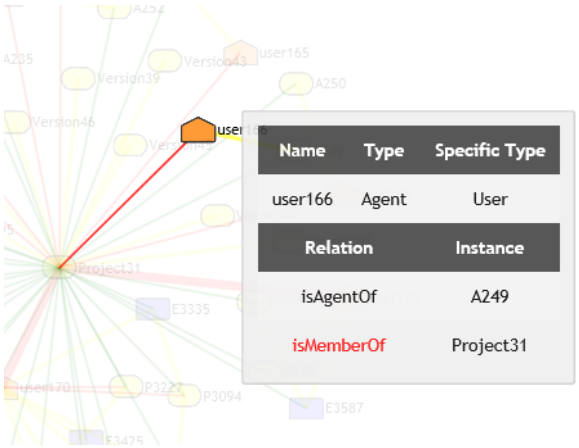



**F. A qual projeto o usuário “user166” foi associado?**

Essa questão de competência pode ser respondida através da relação *isMemberOf*, que é uma property chain do modelo ProvONEExt-O, e pode ser inferida quando um usuário possui relação com uma associação, e a mesma possui relação com um projeto, e logo o usuário é membro do projeto (possui a relação *isMemberOf*) com o projeto.

Na ferramenta Visionary essa resposta pode ser obtida através da filtragem de um usuário específico, no caso o “user166”, como ilustrado na Figura 41. Usuários são considerados membros de um projeto quando possuem associação com o mesmo, e, portanto, estão formalmente alocados ao projeto.

Figura 41 - Proveniência relacionada ao usuário 166




**G. O usuário "user173" está ligado a execuções de qual projeto?**

Essa questão de competência pode ser respondida através da relação *isRelatedTo*, que é uma property chain do modelo ProvONEExt-O, e pode ser inferida pela ontologia quando um usuário está relacionado a uma execução, e a mesma está relacionada a um projeto. Logo, quando um usuário está relacionado a execuções de um projeto, ele possui a relação *isRelatedTo* com esse projeto.

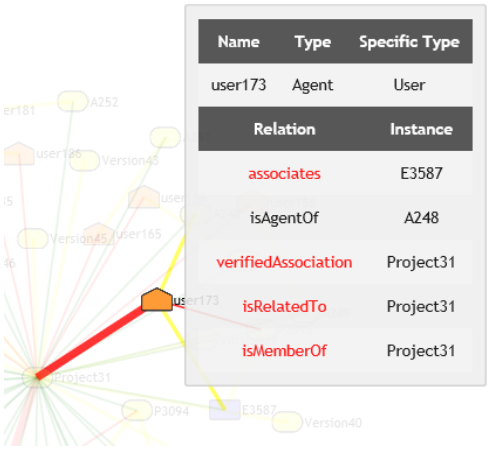
Essa resposta pode ser obtida na ferramenta Visionary através da filtragem de um usuário específico, como ilustrado na Figura 42. Essa relação é uma property chain, e busca quais usuários estão ligados a execuções de um *Project*, e, portanto, trabalham para ele. Porém um usuário pode estar ligado a execuções (e diretamente trabalhar para um projeto) mas não estar relacionado a ele formalmente (que é o que checa a property chain *isMemberOf*, mencionada nas perguntas E e F). Ou seja, existem cenários sobre a atuação de um usuário:

- usuário estar está ligado a execuções de um projeto => possui relação *isRelatedTo* com o projeto
- usuário está ligado formalmente ao projeto => possui relação *isMemberOf* com o projeto

As relações *isMemberOf* e *isRelatedTo* devem ser confrontadas para verificar possíveis inconsistências em três cenários:

1. Se o usuário possui as duas relações => ele possui execuções do projeto e está ligado a ele formalmente (como ilustrado na Figura 42)
2. Se o usuário possui apenas relação *isMemberOf* => está ligado ao projeto mas não está ligado a execuções dele (como ilustrado na Figura 41)
3. Se o usuário possui apenas *isRelatedTo* => possui execuções mas não está alocado formalmente ao projeto

Figura 42 - Proveniência relacionada ao usuário 173




**H. Por quais usuários o projeto "Project31" é influenciado?**

Essa questão de competência pode ser respondida através da relação *wasInfluencedBy* que é uma relação inversa à relação *isRelatedTo*, e pode ser inferida pela ontologia sempre que uma relação desse tipo ocorre. Ou seja, quando um usuário está relacionado a execuções de atividades de um projeto, o projeto possui a relação *wasInfluencedBy* com o usuário, e isso indica a proveniência retrospectiva do projeto, e quais usuários o influenciaram até o presente momento.

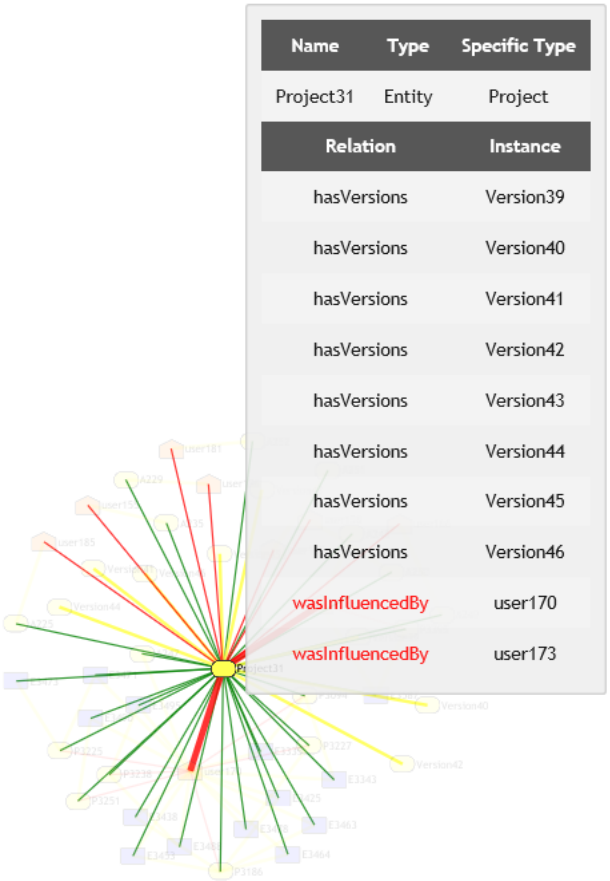
Essa resposta pode ser obtida na ferramenta Visionary através da filtragem de um *Project* específico, como ilustrado na Figura 43. Através dessa property chain é possível visualizar todos os usuários que influenciam um projeto (estão ligados a execuções de atividades do projeto). Essa investigação pode ser necessária principalmente em casos que o projeto não está se desenvolvendo como esperado. Caso o projeto esteja atrasado, podem ser tomadas decisões quanto aos usuários alocados para eles, como, por exemplo, treinamentos, ou adição de novos usuários caso haja poucos. Caso o projeto esteja tendo um desempenho melhor que o esperado, pode ser estudado o comportamento dos usuários para extrair conhecimento e tentar motivar futuros usuários a terem um desempenho melhor.

**I. Quais versões o projeto "Project31" possui?**

Essa questão de competência pode ser respondida através da relação *hasVersions*, que é uma relação entre projeto (*Project*) e suas versões (*Version*). Essa relação apresenta a proveniência do projeto, e todas as versões que ele possui.

Essa resposta pode ser obtida no Visionary através da filtragem de um projeto específico, como ilustrado na Figura 43, e através dela é possível observar todas as versões que pertencem a um projeto.

Figura 43 - Proveniência relacionada ao projeto 31

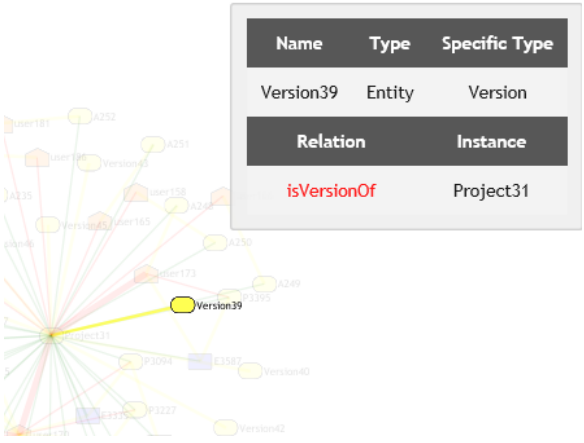



**J. O indivíduo "Version39" é uma versão de qual projeto?**

Essa questão de competência pode ser respondida através da relação *isVersionOf*, que é inversa à relação *hasVersion*, e pode ser inferida sempre que sua inversa ocorre. Essa relação apresenta a proveniência retrospectiva de uma versão, e o projeto que a gerou.

Através da filtragem do indivíduo "Version39" é possível visualizar de qual projeto ele é uma versão, como pode ser observado na Figura 44.

Figura 44 - Proveniência relacionada à versão 39



**K. A qual projeto a atividade (*program*) "P3228" está relacionada?**

Essa questão de competência pode ser respondida através da relação *belongsTo* que é uma relação entre uma atividade e o projeto a qual ela se relaciona. Essa relação indica qual projeto uma atividade influenciará quando for executada.

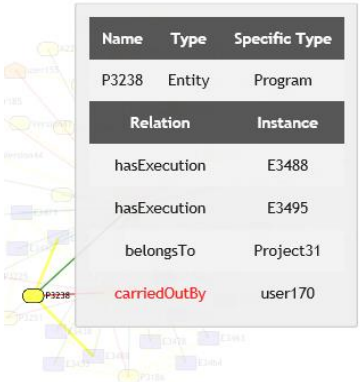
Através da filtragem da atividade "P3228", é possível visualizar que ela está relacionada ao *Project31*, como pode ser observado na Figura 45.

**L. A atividade "P3238" está ligada a qual/quais usuários?**

Essa questão de competência pode ser respondida através da relação *carriedOutBy*, uma property chain que pode ser inferida quando uma atividade possui uma execução, que por sua vez é associada a um usuário, e portanto a atividade possui a relação *carriedOutBy* com o usuário em questão. Essa relação permite a busca pela proveniência relacionada a uma atividade, e que se saiba quais usuários influenciaram na resolução dessa atividade.

Através da filtragem da atividade "*P3238*" é possível observar as suas relações com usuários, nesse caso sua resolução foi encarregada pelo "*user170"*, como pode ser observado na Figura 45.

Figura 45 - Proveniência relacionada à atividade P3238




**M. Qual projeto a execução "E3587" altera/influencia?**

Essa questão de competência pode ser respondida através da relação *alters*, que é uma property chain que pode ser inferida pela ontologia, e é satisfeita quando uma execução está relacionada a uma atividade, que por sua vez está relacionada a um projeto, e, portanto essa execução possui a relação *alters* com o projeto.

Essa consulta pode ser respondida através da filtragem do nó "*E3587*", onde é possível ver que esse indivíduo possui a relação *alters* com o "*Project31*", como ilustrado na Figura 46. Essa busca pode ser relevante caso seja identificado algum comportamento

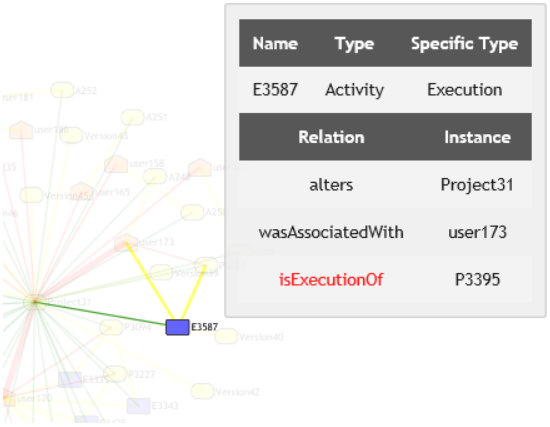
estranho na execução, como por exemplo, grande quantidade de tempo gasto para realizá-la, e estejam buscando quais projetos ela pode ter influenciado.

**N. A execução "E3587" está relacionada a qual atividade?**

Essa questão de competência pode ser respondida através da relação *isExecutionOf*, que liga uma execução à sua atividade de origem. Ou seja, essa relação permite a busca pela proveniência de uma execução, para descobrir a qual atividade ela se relaciona e pretende contribuir para a resolução.

Essa resposta pode ser obtida através da filtragem do nó "*E3587*" na ferramenta Visionary, onde é possível notar suas relações, inclusive a relação *isExecutionOf* que se refere a qual atividade uma execução está relacionada (de qual *program* ela é uma execução), como ilustrado na Figura 46, onde é possível observar que essa execução está relacionada à atividade "*P3395*".

Figura 46 - Proveniência relacionada à execução E3587

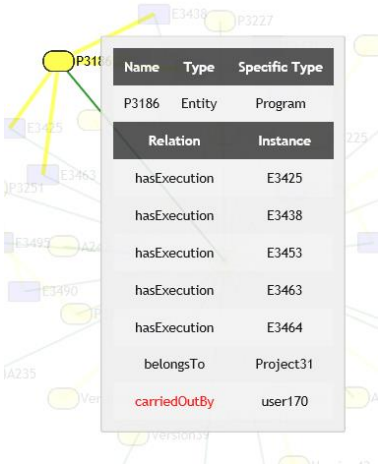



**O. Quais execuções a atividade "P3186" possui? (está ligado a)**

Essa questão de competência pode ser respondida através da relação *hasExecution*, e é oposta à relação *isExecutionOf*. Essa relação permite a busca por todas as execuções que uma atividade possui. Caso haja necessidade de investigar fatores de sucesso ou fracasso de uma atividade, investigar as execuções ligadas a ela pode ser muito relevante.

Essa questão de competência pode ser respondida através da filtragem do nó "*P3186*", dessa forma é possível visualizar todas as execuções que estão ligadas a essa atividade. Como pode ser observado na Figura 47, estão relacionadas à atividade em questão, 5 diferentes execuções através da relação *hasExecution*.

Figura 47 - Proveniência relacionada à atividade P3186




### 5.3.5. Análise dos Resultados

O modelo ProvONEExt possui algumas relações que auxiliam na busca por respostas às Questões de Competência (QC). Entretanto, considerando QC mais específicas, houve a necessidade do uso das propriedades específicas da ontologia (semântica) e rede complexa (estrutura), e a criação de novas relações nestes modelos. Na ontologia foram criadas regras ontológicas específicas, através do uso de *property chains*, e na rede complexa foram criadas novas classes e relações para representar características do contexto de processos de *software*, como *Trackers*, *ProgramStatus* (status de atividades), *ProgramCategories* (categorias de atividades), para que pudessem ter sua importância avaliada como nós da rede.

Na ontologia foram adicionadas duas novas relações, *hasMember*, que é oposta à relação *isMemberOf,* e indica quais membros um projeto possui, e a relação *verifiedAssociation*, uma property chain que é satisfeita quando um usuário possui execuções de uma atividade relacionada a um projeto (consequentemente possui a relação *isRelatedTo* com o projeto), e também possui a relação *isMemberOf,* e está associado formalmente ao projeto. No modelo da rede complexa foi identificado que o nome da relação *hasActivity* pode ser melhorado, assim como o nome *enumeration* da classe de enumeração, que pode ser mudado e os dados do nó divididos em mais categorias. Essas alterações poderão ser implementadas em trabalhos futuros para melhorar futuras análises.

Assim, considerando os novos modelos utilizados, ou seja, modelo ProvOneExt, ontologia ProvOneExt-O e rede complexa ProvOneComplex, todas as questões de competência foram respondidas, e muitas delas só puderam ser respondidas devido ao novo conhecimento extraído devido às novas relações dos modelos. Essa extração de conhecimento

implícito ocorreu através de inferências da ontologia, ou aos novos tipos de análise realizados estruturando os dados como uma rede complexa.

Como todas as questões de competência foram respondidas, a Questão de Pesquisa 1 (É possível responder às questões de competência através de buscas na rede complexa e ontologia?), pode ser respondida positivamente.

A Questão de Pesquisa 2 (A OntoComplex é capaz de extrair conhecimento implícito nos dados?), também pode ser respondida através das análises apresentadas nesta etapa da avaliação, já que grande parte do conhecimento utilizado para responder às questões de competência foi obtido através do conhecimento implícito extraído dos dados através da utilização da ontologia e suas inferências, e da rede complexa e suas análises.

Portanto, esta primeira etapa da avaliação teve como objetivo avaliar a ontologia e rede complexa, e verificar se as mesmas seriam capazes de extrair conhecimento implícito dos dados e responder às questões de competência. A próxima etapa do estudo, a etapa 2, que tem como objetivo avaliar a arquitetura OntoComplex como um todo, será apresentada na próxima subseção.

### 5.3.6. Ameaças à Validade

Algumas ameaças à validade foram detectadas. O estudo foi conduzido pela própria pesquisadora, o que pode trazer algum viés, uma vez que a pesquisadora tem pleno conhecimento das QC e dos dados utilizados no estudo. Outra ameaça a validade é em relação ao estudo ter sido realizado apenas sobre um conjunto específico de dados. A utilização de outros conjuntos de dados pode trazer novas questões não ressaltadas neste estudo. No entanto, na etapa 2 foi conduzido um estudo piloto com outro conjunto de dados, o que pode minorar essa segunda ameaça.

Outra questão a ser destacada é que algumas relações originais do modelo não puderam ser utilizadas com os dados disponibilizados. Por exemplo, as relações *hadPlan* e *qualifiedAssociation* do modelo ProvONEExt-o não puderam ser utilizadas. Não foi utilizada a classe "*Changesets*" do modelo ProvONEExt-O, pois essa classe, na base de dados original disponibilizada pela empresa, possuía cerca de 30.000 registros, o que aumentaria muito o escopo das análises, e, portanto, não foi utilizado para as análises iniciais.

Nas respostas às questões de competência utilizando a rede complexa, alguns tipos de nós apresentaram dados similares, apesar de se referirem a dados diferentes do banco, como apresentado na Questão F da subseção 5.3.4.1., onde foi identificado que os nós do tipo

*Tracker* possuem dados homônimos aos do tipo *ProgramCategories* (que se referem às categorias de atividades).

## 5.4. ETAPA 2

A partir dos resultados obtidos na Etapa 1 da avaliação e com a realização dos ajustes necessários nos modelos, foram conduzidas novas avaliações, utilizando como instrumento de pesquisa o Estudo de Caso (YIN, 2014). De acordo com (YIN, 2014), estudo de caso é a abordagem mais adequada para se avaliar a viabilidade de uma abordagem, utilizando contextos reais.

Assim, foram definidos dois estudos de caso utilizando dados reais de duas empresas de desenvolvimento de *software*. O primeiro estudo de caso, chamado de estudo piloto foi definido e implementado em uma empresa brasileira de desenvolvimento de *software* de porte médio, com cerca de 25 empregados. Este estudo foi apresentado em (FALCI et al., 2018) e serviu como estudo piloto para se verificar a necessidades de ajustes na arquitetura e seus componentes de análise e visualização, uma vez que os modelos foram avaliados e ajustados na Etapa 1.

Após este estudo inicial, alguns ajustes na arquitetura foram realizados e um segundo estudo de caso foi executado para verificar a viabilidade de uso da arquitetura após os ajustes necessários. Para este segundo estudo de caso, dados de uma empresa de desenvolvimento de *software* de grande porte, com cerca de 100 empregados foram utilizados. Como parte da avaliação do segundo estudo de caso, análises realizadas sobre os dados foram apresentadas a um dos gerentes de projeto da empresa, especialista na área de processos de *software*, para que ele avaliasse a extração de conhecimento implícito e suporte ao entendimento dos dados através das visualizações da abordagem.

### 5.4.1. Definição do Estudo

Avaliar se a arquitetura OntoComplex, utilizando dados reais de processos de desenvolvimento de *software*, consegue responder de maneira adequada as questões de pesquisa QP2 “A OntoComplex é capaz de extrair conhecimento implícito nos dados?”, já avaliada na Etapa 1 mas com os ajustes realizados, considerando adequado uma nova avaliação dessa QP, e QP3 “O conhecimento extraído é capaz de dar suporte ao entendimento sobre os dados através de visualizações?”.

### 5.4.2. Planejamento do Estudo

Conforme já dito, dados reais de duas empresas de desenvolvimento de *software* brasileira, denominadas Empresa1 (cerca de 25 empregados) e Empresa2 (> 100 empregados) foram carregados na OntoComplex. A partir destes dados, análises foram realizadas e as ferramentas de visualização da arquitetura foram utilizadas para verificar a viabilidade de se responder as QP 2 e 3, a partir da extração de conhecimento implícito sobre os dados de processos de *software* e fornecimento de informações relevantes para compreensão do comportamento dos dados através de visualizações do ponto de vista de gerentes de projetos.

### 5.4.3. Cenário e Resultados

A avaliação foi realizada em um computador com as seguintes configurações: Processador Intel Core i7-7700HQ CPU 2.80GHz 2.81GHz, Memória RAM 8GB, Windows 10.

O objetivo foi avaliar as funcionalidades da abordagem OntoComplex de extração de dados implícitos e suporte à compreensão dos dados através das visualizações. Foram realizadas análises das redes complexas e ontologias para extrair conhecimento sobre os dados e responder às questões de competência, e observações sobre as respostas foram realizadas para coletar evidências sobre o uso da OntoComplex e sua capacidade de derivar informações estratégicas. As visualizações das análises foram apresentadas a um especialista para que ele avaliasse a extração de conhecimento implícito e o suporte à compreensão dos dados através das visualizações. As respostas do especialista foram analisadas para se verificar a capacidade de suporte à tomada de decisão baseada no fornecimento de informação estratégica, provida através da extração de conhecimento implícito.

## 5.5. ESTUDO PILOTO

### 5.5.1. Objetivos

O estudo piloto teve como objetivo verificar a viabilidade de uso da OntoComplex e verificar possíveis ajustes nas funcionalidades e tecnologias utilizadas. O estudo completo foi detalhado em (FALCI et al., 2018). Apresentamos aqui os principais resultados.

### 5.5.2. Caracterização do Objeto de Estudo

Os dados utilizados no estudo piloto foram obtidos através de uma parceria com uma empresa brasileira de desenvolvimento de *software* de porte médio, com cerca de 25 empregados. Os dados foram disponibilizados em formato de planilhas, e foram modelados de

acordo o modelo inicial da OntoComplex. Após a tradução dos dados, a base de dados da OntoComplex persistiu 5624 registros com 29 atributos. Estes dados foram utilizados para avaliar as questões de pesquisa QP 2 e QP3, além de servir de base de testes para verificar falhas e pontos de melhoria da arquitetura.

### 5.5.3. Cenário e Resultados

O primeiro passo do estudo piloto foi a compreensão dos dados disponibilizados pela empresa parceira. Após a compreensão e conversão dos dados para o modelo disponibilizado pela arquitetura, os mesmos foram instanciados na ontologia de acordo com as classes e relações estabelecidas. Após a instanciação dos dados, inferências foram processadas para realizar as asserções e processar a extração de conhecimento implícito nos dados. As inferências obtidas foram analisadas para verificar possíveis pontos que poderiam ser aprimorados, que serão detalhados a seguir.

O próximo passo foi a inserção dos dados na rede complexa e processamento de análises topológicas da rede. Foram analisados aspectos da rede como centralidade dos nós, grau de entrada e saída, e vulnerabilidade em caso de ataque (caso um nó seja removido da rede de forma inesperada). Possíveis melhorias foram levantadas para aprimorar a rede complexa da OntoComplex.

Com esse estudo foi observado que é possível inferir conhecimento através da ontologia, e que através das análises da rede complexa é possível extrair conhecimento novo sobre os dados e sua relevância na rede. No entanto, alguns aspectos necessitaram de melhoria:

- As análises realizadas, principalmente na rede complexa, eram focadas na utilização e geração de dados pelas execuções. Foi identificada a necessidade de mais análises sobre as atividades e suas execuções, além de usuários responsáveis pelas mesmas, e projetos aos quais se relacionavam. Por conta disso, foi identificada a necessidade de extensão do modelo para possibilitar essas análises;
- O primeiro modelo extendido do ProvONE possuía apenas as classes *User* (usuário), *Process* (atividade), *ProcessExec* (execução de atividade), *e Data* (dado). Foi identificada a necessidade de mais classes para representar, por exemplo, projetos e versões de projetos, associação de usuários a atividades, classe de controle das atividades, além das relações entre essas classes. Optou-se por criar uma nova versão do modelo, baseado na nova versão do

ProvONE, o qual atende alguma dessas necessidades, e foi ser estendido para se adaptar completamente;

- Foi identificada a necessidade de se criar *property chains* para gerar novo conhecimento sobre a ontologia;
- Foi observado que o modelo da rede complexa agregaria mais conhecimento se o modelo apresentasse classes distintas da ontologia, e dessa forma as análises pudessem ser feitas de forma conjunta, porém com detalhes distintos, mantendo a mesma base. O modelo da rede complexa poderia ter classes para representar, por exemplo, os status e categorias de atividade para que tais características pudessem ser analisadas levando em consideração sua participação e importância na rede;
- Para verificar a adequabilidade das análises realizadas, a camada de visualização foi utilizada para visualizar a proveniência dos dados, e foi identificada a necessidade da implementação de mais um tipo de visualização, onde os dados pudessem ser visualizados de forma geral, porém ao mesmo tempo fosse possível observar detalhes dos dados. A visualização criada para atender essa necessidade mostra de forma conjunta os dados em gráficos e tabelas.

Desta forma, ajustes foram realizados na arquitetura e para verificar a viabilidade desta nova versão, um estudo regular foi realizado, com o objetivo de verificar a viabilidade da arquitetura e a possibilidade de ser verificar as questões de pesquisa propostas. A seção a seguir descreve em detalhes este estudo de caso.

## 5.6. ESTUDO REGULAR

### 5.6.1. Objetivos

Avaliar a viabilidade da segunda versão da arquitetura OntoComplex, aprimorada após o estudo piloto, e apresentada no capítulo 4. Essa avaliação ocorre através da verificação do comportamento da ontologia e rede complexa, bem como de suas inferências e análises estruturais, respectivamente, ao serem instanciadas com uma grande massa de dados reais de uma empresa de desenvolvimento de *software*, e sua capacidade de prover informações estratégicas sobre os dados. Além disso, foram avaliadas as visualizações disponibilizadas pela arquitetura quanto ao suporte da compreensão do conhecimento implícito extraído pelos componentes da arquitetura.

### 5.6.2. Caracterização do Objeto de Estudo

Os dados utilizados no estudo regular são provenientes de uma empresa de desenvolvimento de *software* com aproximadamente 100 empregados. A base disponibilizada possui dados de gerência de configuração, os quais foram instanciados no modelo ProvOneExt, após a tradução do modelo utilizado pela ferramenta Mantis. Os dados foram utilizados para instanciar a ontologia e rede complexa, os quais foram analisados através de suas visualizações para verificar o conhecimento estratégico provido pelas ferramentas.

### 5.6.3. Cenário e Resultados

No segundo estudo, a base com dados de gerência de configuração foi disponibilizada pela empresa de desenvolvimento de *software*. Estes dados incluíam dados sobre projetos, usuários, requisições de alteração, execuções das requisições, entre outros dados relacionados a esse contexto.

A base de dados foi disponibilizada como um banco de dados relacional, e possuía 222.323 registros distribuídos em 79 tabelas. O primeiro passo da modelagem dos dados foi o entendimento dos dados e a função de cada uma das tabelas e seus campos.

O banco de dados foi modelado para se adaptar ao modelo ProvOneExt, considerando a nova versão após as alterações realizadas no estudo piloto. Os dados foram associados às entidades do modelo, de acordo com o tipo de dados aos quais se referiam. Após a conversão dos dados para o ProvOneExt, os dados foram instanciados na ontologia, considerando a correspondência com as classes do modelo. A máquina de inferência Pellet foi então utilizada, para inferir asserções, de acordo com as regras definidas na ontologia. O Pellet foi o algoritmo de inferência selecionado por sua capacidade de suportar property chain assertions (asserções sobre cadeias de relações), o que não é possível com outras máquinas de inferência.

Os resultados obtidos após o processamento da inferência foram conferidos para verificar se as regras funcionam como se esperava. Alguns ajustes foram feitos para melhorar as asserções iniciais.

Como resultado, pôde-se verificar que as inferências da ontologia extraíram de maneira adequada conhecimento implícito nos dados. As regras construídas permitiram inclusive inferências de relações inversas e de property chains.

Como segunda etapa no processamento das análises da OntoComplex, o processamento das informações na rede complexa também foi verificado. Na rede complexa, após definições iniciais, dados foram instanciados na rede complexa. Arestas foram

adicionadas baseadas nas relações das classes do modelo estendido. Após a instanciação, a rede complexa foi analisada para verificar, por exemplo, a relevância dos nós na rede complexa, o impacto em caso de ataque (se um nó fosse removido da rede repentinamente), grau de entrada e de saída, que analisam quantas relações chegam ou saem dos nós. Essas análises foram feitas através de buscas em *cypher*.

Para obter as visualizações sobre a ontologia, foram geradas diferentes versões da mesma. Cada versão possui dados sobre um dos projetos da empresa. Essa divisão foi feita para que a visualização dos dados na ferramenta Visionary fosse mais clara, considerando o volume elevado de dados.

A partir dos dados analisados e com as visualizações utilizando as ferramentas disponíveis na OntoComplex, uma avaliação com um especialista no domínio foi conduzida. Na seção seguinte apresentamos os resultados dessa avaliação.

#### *5.6.3.1. Estudo com especialista*

Para responder à questão de pesquisa QP3 (O conhecimento extraído é capaz de melhorar o entendimento sobre os dados?) foi realizada uma avaliação com um especialista, gerente de projetos da empresa que disponibilizou os dados analisados.

A avaliação foi realizada a partir da disponibilização dos dados processados pela OntoComplex, através das análises realizada pela ProvOneExt-O e ProvOneExt-Complex e visualizados utilizando a ferramenta Visionary e o retorno de consulta em realizadas em cypher. O especialista pôde avaliar a visualização da proveniência a partir dos dados da ontologia e suas inferências, e da rede complexa e suas análises, com o objetivo de verificar se elas contribuem com a extração de conhecimento implícito nos dados, e se essas extrações disponibilizadas através de visualizações dão suporte à compreensão dos dados.

##### 5.6.3.1.1. Caracterização do indivíduo

Foi definido que o experimento seria desenvolvido com especialista da área de processos de desenvolvimento de *software*, um profissional que possuísse experiência com gerência de projetos, e conhecesse o contexto dos dados utilizados no estudo regular.

O indivíduo selecionado para participar do estudo tem conhecimento da base de dados disponibilizada para o segundo estudo, e possui experiência na área de gerência de projetos. É do sexo masculino, tem 32 anos e trabalhou por cerca de 5 anos com projetos na empresa alvo do estudo.

#### 5.6.3.1.2. Condução da Avaliação com o Especialista

Não foi informado ao participante qualquer informação sobre ontologia, proveniência de dados, inferências ou rede complexa. Apenas foram apresentadas as ferramentas com as visualizações dos dados (e inferências, no caso da ontologia), e foi pedido que ele respondesse às perguntas do questionário baseado nas visualizações, assim como aconteceria em um cenário real.

O problema apresentado era real, por ser relacionado a partir de dados de uma empresa de desenvolvimento de *software*, e conta com as seguintes atividades:

(1) leitura da seção 1 de perguntas do formulário (com questões relacionadas à proveniência dos dados) – anotação do horário de início;

(2) busca pelas respostas na visualização provida pela ferramenta Visionary;

(3) respostas às perguntas da seção 1 – anotação do horário de fim de resposta às perguntas;

(4) leitura da seção 2 de perguntas do formulário (com questões sobre a participação dos dados na rede complexa) – anotação do horário de início da busca por respostas;

(5) busca pelas respostas nas visualizações disponíveis da rede complexa;

(6) respostas às perguntas da seção 2 – anotação do horário de fim de respostas às perguntas;

(8) leitura da seção 3 de perguntas do formulário (com questões sobre o comportamento dos dados de execução como um todo) – anotação do horário de início de busca pelas respostas do questionário na visualização;

(9) busca pelas respostas nas visualizações que intercalam tabela e dados relacionados;

(10) respostas às perguntas da seção 3 – anotação do horário de fim das respostas ao questionário

Após o participante responder os questionários, foi perguntado a ele se tinha alguma consideração a ser feita sobre a etapa, as quais são detalhadas na subseção de limitações e ameaças à validade. Os questionários das três etapas estão nos anexos do presente trabalho.

Foi criado um cenário com base no problema ilustrado no capítulo 4, onde um gerente de projetos, insatisfeito com o rendimento das atividades do projeto, pretende investigar sobre a produtividade de seus usuários. Nessa análise diferentes pontos da arquitetura são avaliados.

A avaliação foi dividida em 3. Antes de cada etapa, apenas uma breve explicação sobre as funcionalidades das ferramentas de visualização foi feita. Sobre os dados, apenas foi informado que se tratava de dados de um projeto, e foi pedido para que ele respondesse questões sobre os dados através das visualizações. A intenção era avaliar se o usuário conseguiria responder perguntas sobre os dados, o que comprovaria que ele entendeu o comportamento deles, e consequentemente que a arquitetura foi capaz de dar suporte à compreensão dos dados.

A primeira etapa da avaliação apresenta a proveniência do projeto investigada através da ferramenta Visionary, onde são exibidas as inferências realizadas pela ontologia. Nessa etapa são feitas perguntas para avaliar se o gerente de projetos consegue compreender o comportamento dos dados da visualização. A Figura 48 ilustra a visualização apresentada ao usuário. Os nomes dos tipos dos indivíduos foram apresentados de acordo com o contexto, e não com o modelo. Como ilustrado na Figura 48, os nós do tipo *Program* da ontologia foram apresentados como Atividade, para facilitar o entendimento. A Tabela 15 relaciona os nomes do modelo ProvONEExt, os nomes utilizados na visualização Visionary na avaliação com especialista, e o padrão de nomes criados para os indivíduos de cada um dos tipos. Essas alterações foram feitas para que o modelo ficasse transparente ao especialista.

O questionário sobre a primeira etapa continha, por exemplo, perguntas sobre usuários membros do projeto apresentado, sobre as execuções relacionadas a cada usuário, sobre quais usuários não possuíam atividades. Essas perguntas seriam essenciais na investigação sobre a produtividade dos usuários do projeto. Além dessas perguntas, foram feitas outras perguntas cujas respostas podiam ser obtidas a partir de inferências da ontologia, como por exemplo, “Qual usuário está relacionado à atividade ‘P3225’?”. A relação entre *Usuário* e *Atividade* pode ser obtida através da inferência de uma das property chains do modelo. As inferências criam novas relações entre os dados, que podem ser visualizadas na ferramenta apresentada.

Figura 48 - Visualização de toda proveniência do Projeto 31 pela ferramenta Visionary

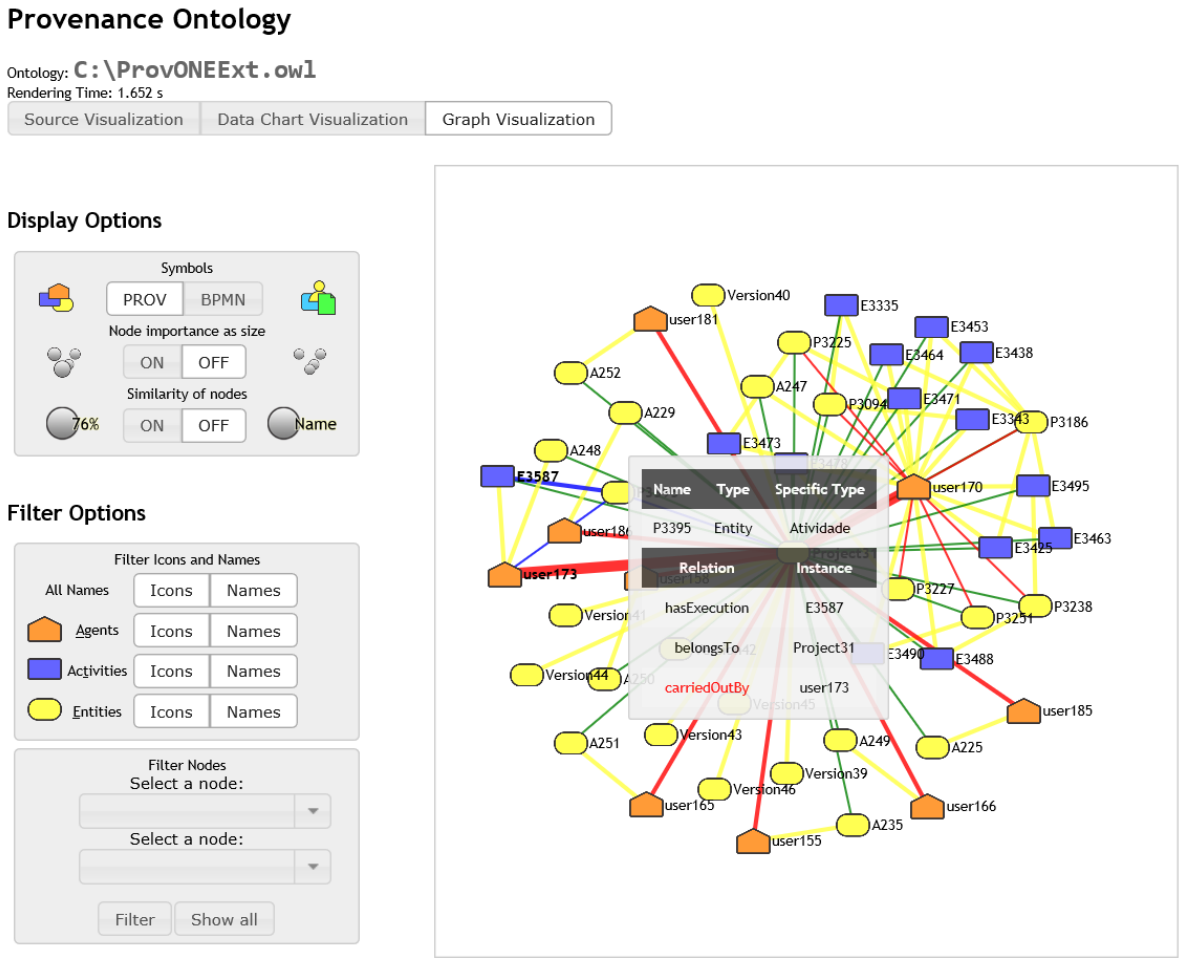



Tabela 15 - Nomes das classes ontologia, seus equivalentes na visualização e padrão de nomes dos indivíduos utilizados

| **Tipo de nó - modelo ProvONEExt-o** | **Nome do tipo de nó – Visualização Visionary** | **Padrão dos nomes criados** |
|---|---|---|
| Program | Atividade | P + id do registro original da base de dados |
| Execution | Execução | E + id do registro original da base de dados |
| Association | Association | A + id do registro original da base de dados |
| User | Usuário | user + id do registro original da base de dados |
| Version | Versão | V + id do registro original da base de dados |
| Project | Projeto | Project + id do registro original da base de dados |

O participante foi capaz de responder às 7 questões do questionário e acertar todas, e demorou 7 minutos para responder o questionário. As dificuldades e lições aprendidas

nesta etapa, assim como das outras etapas dessa parte da avaliação, serão apresentadas na próxima subseção.

Na segunda etapa foi apresentado ao usuário o retorno de duas consultas em *cypher* no Neo4j[16], que foi utilizado para dar suporte à visualização. Apenas os retornos, apresentados na Figura 49, foram apresentados, pois não fazia parte da avaliação testar se o usuário possuía a capacidade de realizar buscas sobre a rede complexa, e sim se arquitetura conseguia extrair conhecimento implícito nos dados e dar suporte à compreensão dos dados através das visualizações.

Figura 49 - Retorno de buscas em *cypher* com o suporte da plataforma Neo4j

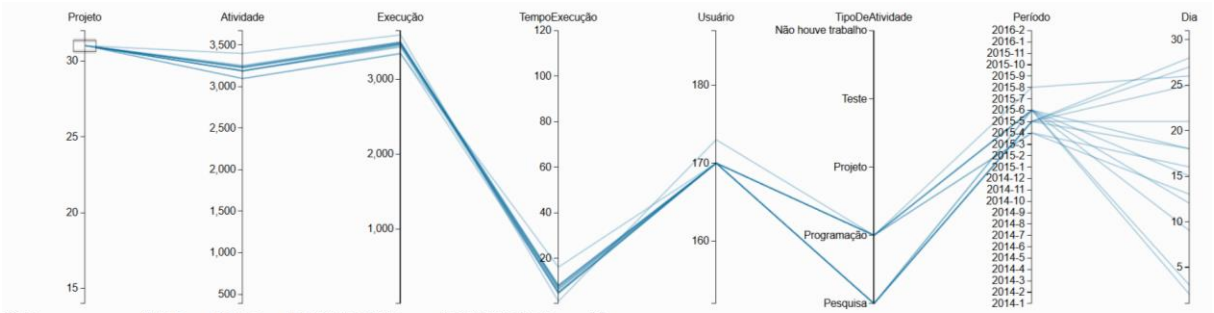

Portanto, as perguntas da segunda etapa tinham como objetivo avaliar se as visualizações da rede complexa, bem como a construção da mesma, as relações e as informações sobre os nós conseguiam dar suporte à compreensão dos dados. Algumas características da rede, como por exemplo, grau de saída foi indiretamente questionada ao participante. Por exemplo, foi feita a pergunta “A quantas execuções o “usuário 181” está relacionado?”, e a resposta dessa pergunta é o grau de entrada do usuário. A pergunta “Quem possui mais relações com execuções, o usuário 181 ou 185?” compara os graus de entrada dos dois usuários, e consequentemente a importância dos mesmos na rede em questão de

[16] https://neo4j.com/

quantidade de ligações. Além disso, a resposta da pergunta “Qual é a prioridade da maioria das atividades relacionadas ao usuário 181?” além de indicar o perfil de execuções do usuário, ainda pode indicar se o impacto da remoção de um nó seria maior ou menor de acordo com a prioridade das execuções que o mesmo é responsável. O participante respondeu corretamente às 6 questões da etapa 2 em 5 minutos.

Na etapa 3 foi apresentada a visualização que engloba gráfico e tabela. Foi apresentada uma visualização sobre dados de execuções. Foram feitas perguntas complementares às análises anteriores, sobre os dados do “Projeto 31”, como por exemplo, para investigar quais tipos de atividades foram desenvolvidas no projeto e qual o período que ocorreram as atividades. Essa etapa da avaliação tinha como intuito verificar se a ferramenta era intuitiva e conseguia disponibilizar uma visualização que desse suporte à compreensão dos dados. O participante conseguiu responder às 3 questões dessa etapa corretamente em 3 minutos.

Figura 50 - Visualização sobre dados de execuções




| Projeto | Atividade | Execução | TempoExecução | Usuário | TipoDeAtividade | Período | Dia |
|---|---|---|---|---|---|---|---|
| 31 | 3186 | 3425 | 7.05472 | 170 | Pesquisa | 2015-5 | 18 |
| 31 | 3186 | 3438 | 4.48222 | 170 | Pesquisa | 2015-5 | 21 |
| 31 | 3186 | 3453 | 8 | 170 | Pesquisa | 2015-5 | 25 |
| 31 | 3186 | 3463 | 6.35722 | 170 | Pesquisa | 2015-5 | 27 |
| 31 | 3186 | 3464 | 7.30667 | 170 | Pesquisa | 2015-5 | 28 |
| 31 | 3225 | 3471 | 6.25361 | 170 | Pesquisa | 2015-6 | 2 |
| 31 | 3227 | 3478 | 8 | 170 | Pesquisa | 2015-6 | 9 |
| 31 | 3094 | 3335 | 16 | 170 | Programação | 2015-4 | 13 |
| 31 | 3094 | 3343 | 5 | 170 | Programação | 2015-4 | 16 |
| 31 | 3225 | 3473 | 5 | 170 | Programação | 2015-6 | 3 |
| 31 | 3238 | 3488 | 8 | 170 | Programação | 2015-6 | 12 |
| 31 | 3251 | 3490 | 5 | 170 | Programação | 2015-6 | 15 |
| 31 | 3238 | 3495 | 4.5 | 170 | Programação | 2015-6 | 18 |
| 31 | 3395 | 3587 | 1 | 173 | Programação | 2015-8 | 26 |

### 5.6.4. Avaliação do Estudo

A segunda etapa da avaliação foi desenvolvida em três partes. A primeira foi realizada através de um estudo piloto utilizando dados reais de uma empresa desenvolvimento de *software*. Esse estudo teve como objetivo foi avaliar a arquitetura OntoComplex e encontrar possíveis pontos de melhoria, os quais foram identificados e aplicados para melhorar o modelo de proveniência proposto, além da ontologia, rede complexa e visualizações.

A segunda parte teve como objetivo avaliar a viabilidade da arquitetura OntoComplex após a implementação das melhorias identificadas no estudo piloto. Essa parte

foi realizada através de um estudo regular onde dados sobre processos de *software* de uma empresa de grande porte foram utilizados. Os dados foram modelados de acordo com o modelo ProvONEExt, proposto como extensão da nova versão do ProvONE. Após a modelagem dos dados, os mesmos foram utilizados como instâncias da ontologia e rede complexa, também aprimoradas após a proposta de novos modelos.

As inferências da ontologia foram analisadas através da ferramenta Visionary, que dá suporte à camada de visualização, onde é possível visualizar a proveniência dos dados e criar filtragens para visualizar seções dos dados. Análises da rede complexa foram realizadas com apoio da plataforma Neo4j, onde buscas puderam ser visualizadas. Além disso, dados sobre os processos foram visualizados com suporte da nova visualização proposta que integra visualização de gráfico e tabela.

Com base nesses resultados, com estes estudos foi possível confirmar QP2 (“A OntoComplex é capaz de extrair conhecimento implícito nos dados?”) positivamente. Essa resposta pôde ser obtida através do conhecimento extraído pelas inferências e análises da rede complexa, e disponibilizadas pelas visualizações. Dessa forma foi possível compreender que a OntoComplex é capaz de extrair conhecimento implícito nos dados, já que o conhecimento disponibilizado pela arquitetura não estava disponível nos dados originalmente.

A terceira parte do estudo contou com a participação de um especialista, o qual respondeu perguntas sobre os dados através de visualizações, as quais foram disponibilizadas pela arquitetura proposta sobre a ontologia e suas inferências, além de análises da rede complexa e visualização de gráfico e tabela sobre os dados.

O especialista foi capaz de responder a todas as perguntas sobre os dados com ajuda das visualizações sobre o conhecimento extraído pela arquitetura. Dessa forma, essa parte do estudo ajudou a responder a questão QP3 (“O conhecimento extraído é capaz de dar suporte ao entendimento sobre os dados através de visualizações?”) positivamente, já que o especialista foi capaz de responder a todas as perguntas que avaliavam sua compreensão sobre os dados através das visualizações.

#### *5.6.4.1. Limitações e Ameaças à validade*

Foram detectadas algumas ameaças à validade do estudo. Uma delas é o fato de que o especialista que participou dessa avaliação conhecer os dados utilizados. Entretanto, a base contava com um grande volume de dados e muitas relações apresentadas foram derivadas a partir de inferências, ou apresentadas em um formato completamente diferente do original, em forma de rede complexa ou como a visualização que integra gráfico e tabela.

Acreditamos que essa ameaça não invalide os resultados do experimento. O especialista não foi informado antes do estudo quais dados estavam sendo utilizados.

Nessa avaliação foi apresentada ao usuário uma seleção dos dados originais, referentes a apenas um projeto. A visualização na ferramenta Visionary atualmente é comprometida, quando um grande volume de dados é apresentado. Desta forma, não foi possível a visualização completa do modelo (que possui mais de 36000 indivíduos). Para lidar com essa limitação, a solução utilizada foi analisar cada projeto individualmente, e não todos os projetos de uma vez.

A visualização utilizada na etapa 2 foi feita com apoio da plataforma Neo4j. Não foi desenvolvido no presente trabalho visualizações sobre os dados de queries em *cypher*. Entretanto, é possível utilizar uma extensão do Neo4j como servidor REST, onde podem ser feitas buscas em *cypher* e obter retornos no formato JSON. A partir desse retorno é possível criar visualizações com o apoio da biblioteca D3.js[17], utilizada para apoiar as outras visualizações apresentadas no presente trabalho.

Sobre as lições aprendidas, algumas observações foram feitas durante o estudo. Na primeira etapa, o gerente declarou que teve uma dificuldade inicial de entender o padrão dos nomes dos indivíduos. Portanto, o nome dos indivíduos deve ser escolhido cuidadosamente para não dificultar o entendimento dos usuários. Na visualização utilizada na rede complexa, os nomes dos tipos dos nós são cruciais para o entendimento dos dados. Quando há tipos que englobam categorias diferentes, o ideal é separar em mais tipos. Por exemplo, quando houver um nó que possui diferentes tipos de enumerações, o ideal é separar essas enumerações em diferentes classes de enumerações. E por fim, na visualização que engloba gráfico e tabela, foi relatada uma dificuldade para selecionar dados das arestas quando o alcance é muito curto. Talvez a seleção de “brushing”, feita ao fazer uma seleção na aresta com o cursor, seja difícil em algumas situações, e seja necessário incluir outros tipos de filtragem. Esses detalhes podem ser abordados em trabalhos futuros.

Consideramos os resultados da avaliação satisfatórios. No entanto, estes resultados não podem ser generalizados, e novos estudos devem ser conduzidos, com outras bases de dados, uma vez que a avaliação utilizada só é válida para o cenário e dados utilizados.

[17] https://d3js.org/

## 5.7. RESUMO DO CAPÍTULO

O presente capítulo apresentou a avaliação da arquitetura OntoComplex. A primeira avaliação apresentada teve como objetivo utilizar a ontologia e rede complexa propostas para ajudar a responder questões de competência e, consequentemente, responder à questão de pesquisa QP1 (“É possível responder às questões de competência através de buscas na rede complexa e ontologia?”).

A segunda parte da avaliação foi dividida em três partes. A primeira parte contou com um estudo piloto, o qual teve como objetivo encontrar possíveis melhorias a serem feitas na arquitetura e seus elementos. O segundo estudo, um estudo regular, teve como objetivo avaliar a nova versão da arquitetura. Esses estudos ajudaram a responder à questão de pesquisa QP2 (“A OntoComplex é capaz de extrair conhecimento implícito nos dados?”), enquanto a terceira parte do estudo contou com a participação de um especialista da área de processos de *software* e ajudou a responder à questão de pesquisa QP3 (“O conhecimento extraído é capaz de dar suporte ao entendimento sobre os dados através de visualizações?”). Foram também apresentadas limitações e ameaças à validade dos estudos.

O próximo capítulo apresenta as conclusões do presente trabalho, assim como trabalhos futuros.

# 6. CONSIDERAÇÕES FINAIS

## 6.1. CONTRIBUIÇÕES

O presente trabalho teve como objetivo propor a arquitetura OntoComplex para dar suporte à integração de dados de processos de *software* em diferentes formatos. Essa integração pode ser feita com a ajuda de uma das camadas da arquitetura, onde é apresentado um modelo genérico de banco de dados relacional, que recebe os dados e seus descritores através de outra camada da mesma. Cabe ressaltar ainda que a parte de tradução dos dados pode ser um grande desafio dependendo dos dados disponíveis.

Após a modelagem os dados são utilizados por ontologia e rede complexa para extrair conhecimento implícito e prover análises sob uma diferente perspectiva, e disponibilizar o conhecimento extraído através de visualizações para dar suporte à compreensão dos dados para gerentes de projetos. As visualizações são disponibilizadas em três formatos diferentes, dependendo das necessidades de visualização da proveniência dos dados, dos dados como uma rede, ou de forma mais geral e detalhada ao mesmo tempo.

Desta forma, o trabalho proposto oferece uma abordagem que provê uma solução para necessidades da área, reportadas em estudos relacionados, encontrados através da revisão *quasi*-sistemática da literatura. Essas dificuldades incluem principalmente a falta de abordagens que dão suporte à integração de dados em diferentes formatos, e ferramentas de suporte à utilização de dados para tomada de decisão.

A ontologia e rede complexa propostas foram analisadas para responder questões de competência, que tiveram como objetivo avaliar os modelos e extrair conhecimento implícito sobre os dados de processos de software. A arquitetura foi avaliada através da utilização de bases de dados reais e com a participação de um especialista da área. Os resultados foram apresentados através desta dissertação e de artigo científico.

O trabalho atual apresentou questões de pesquisa, que agora serão revisitadas. No capítulo de introdução foi apresentada uma questão de pesquisa (QP0), e no capítulo 5 foram apresentadas três questões (QP1, QP2, QP3), propostas a partir do escopo da avaliação. As questões de pesquisa apresentadas foram as seguintes:

**QP0: Como utilizar dados em diferentes formatos sobre processos de *software*, e seu conhecimento implícito, para melhorar a compreensão de gerentes de projetos sobre os processos, e melhorar futuras execuções?**

**QP1:** É possível responder às questões de competência através de buscas na rede complexa e ontologia?

**QP2:** A OntoComplex é capaz de extrair conhecimento implícito nos dados?

**QP3:** O conhecimento extraído é capaz de dar suporte ao entendimento sobre os dados através de visualizações?

A Questão de Pesquisa 1 foi ser respondida positivamente através das respostas às questões de competência, realizadas no capítulo 5, seção 5.3.4. As questões de competência ajudam a extrair conhecimento implícito sobre a área de processos de *software* com o auxílio da ontologia e rede complexa, o que também ajudou a responder positivamente a segunda questão de pesquisa.

A primeira e segunda parte do estudo que avaliaram a arquitetura através da utilização de duas bases de dados reais. Essas partes reforçam a resposta positiva à questão de pesquisa 2, pois através da utilização da arquitetura foi possível extrair conhecimento implícito sobre as bases de dados. A última parte deste estudo contou com a avaliação de um especialista da empresa que disponibilizou a segunda base de dados real. Essa avaliação permitiu a resposta à questão de pesquisa 3, pois com a mesma foi possível observar que o conhecimento extraído através da arquitetura é capaz de dar suporte ao entendimento sobre os dados através de visualizações. Esse suporte permitiu que o especialista respondesse a todas as perguntas dos questionários sobre os dados.

Sobre a questão de pesquisa QP0, há indícios de que a arquitetura proposta possibilitou a integração de diferentes tipos de dados de acordo com o modelo de dados proposto, e a extração de conhecimento implícito através da utilização das ferramentas propostas, o qual foi disponibilizado através de visualizações para gerente de projetos. Desta forma, foi possível dar suporte à compreensão dos dados sobre os processos, e a partir dessa compreensão gerentes podem tomar decisões baseadas em dados, e consequentemente melhorar futuras execuções com o conhecimento adquirido.

Há indícios de que o objetivo principal foi alcançado através da proposta da arquitetura OntoComplex. Apesar de existirem outras abordagens, encontradas através da revisão apresentada, que dão suporte à seleção de métricas de processos para dar suporte à tomada de decisão orientada a dados, não foram encontradas propostas que englobem as tecnologias utilizadas no presente trabalho, como modelos de proveniência de dados, modelos de dados, ontologia, rede complexa e visualizações para extrair conhecimento e dar suporte à compreensão dos dados, e consequentemente dar suporte à tomada de decisão e melhorar futuras execuções.

## 6.2. LIMITAÇÕES

O presente trabalho possui algumas limitações, as quais serão apresentadas a seguir:

- O modelo de proveniência proposto no presente trabalho, o ProvONEExt, foi avaliado após seu aprimoramento de acordo com as análises pertinentes. O modelo é genérico no contexto de processos de *software*, e possui classes e relações que permitem diversos tipos de análises, com focos diferenciados. O presente trabalho apresentou a avaliação com base na proveniência relacionada a projetos, versões de projetos, atividades, suas execuções e usuários. Análises sobre a geração e utilização de documentos, e sobre a classe de controle não foram executadas utilizando o modelo apresentado no presente trabalho, mas foram realizadas utilizando o modelo inicial, reportado em (FALCI et al., 2018).
- As questões de competência utilizadas para avaliar a ontologia e rede complexa foram desenvolvidas após entrevistas com gerentes, de onde foi extraído conhecimento que serviu como base para entendimento de quais questões seriam interessantes ser respondidas no contexto.
- A avaliação final contou com um gerente de projetos para responder perguntas sobre os dados com base nas visualizações, e avaliações com mais gerentes devem ser conduzidas para dar um melhor embasamento na avaliação.
- As visualizações utilizadas na avaliação com o gerente de projetos podem ser aprimoradas. No caso da visualização da proveniência dos dados, deve ser buscada uma solução para visualizar dados de diversos projetos, e no caso da visualização da rede complexa, deve ser desenvolvida uma visualização independente da plataforma Neo4j e integrada ao projeto.

## 6.3. TRABALHOS FUTUROS

As limitações anteriormente apresentadas apresentam novas demandas para aprimorar a proposta do presente trabalho, como por exemplo:

- Avaliação do modelo ProvONEExt por completo, utilizando todas as classes propostas no documento após seu aprimoramento;
- Aprimoramento das visualizações, para melhorar a interação do usuário com a arquitetura, além do desenvolvimento de uma visualização da rede complexa independente da plataforma utilizada para visualização;

- Nova avaliação com outros gerentes de projetos para avaliar a arquitetura e suas visualizações;
- Trabalhos futuros podem abordar a coleta automática de dados para alimentação da arquitetura proposta no presente trabalho. Além disso, podem ser estudadas dinâmicas para avaliar automaticamente quais análises são mais importantes em um determinado momento;
- Podem ser estudados meios de realizar as análises da ontologia e rede complexa automaticamente, e disponibilizá-las em um *dashboard*, sem a necessidade de análises manuais;
- A arquitetura proposta pode ser adaptada para outros contextos através da utilização de diferentes modelos de proveniência de dados, ontologias e redes complexas.

## REFERÊNCIAS


ABRANTES, J. F., & TRAVASSOS, G. H. (2007). Revisão quasi-Sistemática da Literatura: Caracterização de Métodos Ágeis de Desenvolvimento de Software. **Relatório Técnico RT-ES 714/07, Programa de Engenharia de Sistemas de Computação**, *COPPE/UFRJ*.

ACUNA, S. T., DE ANTONIO, A., FERRE, X., MATÉ, L., & LÓPEZ, M. (2001). The software process: Modeling, evaluation and improvement. In **Handbook of Software Engineering and Knowledge Engineering***:* Volume I: Fundamentals (pp. 193-237).

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

BELHAJJAME, K., B'Far, R., Cheney, J., Coppens, S., Cresswell, S., Gil, Y., ... & Miles, S. (2013b). Prov-dm: The prov data model. **W3C Recommendation**.

BELHAJJAME, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., Zhao, J., (2013a), PROV-O: The PROV Ontology. **W3C Recommendation.** Disponível em: http://www.w3.org/TR/prov-o

BELHAJJAME, K., Deus, H., Garijo, D., Klyne, G., Missier, P., Soiland-Reyes, S., & Zednik, S. (2012). Prov model primer. **W3C Recommendation** URL: http://www. w3. org/TR/provprimer.

BOCCALETTI, S., LATORA, V., MORENO, Y., CHAVEZ, M., & HWANG, D. U. (2006). Complex networks: Structure and dynamics. **Physics reports**, *424*(4-5), 175-308.

BOSCH, J. (2016). Speed, data, and ecosystems: the future of software engineering. **IEEE Software**, 33(1), 82-88.

BOSCH, J. (2017). **Speed, data, and ecosystems:** Excelling in a software-driven world. CRC Press.

BUNEMAN, P., KHANNA, S., & WANG-CHIEW, T. (2001, January). Why and where: A characterization of data provenance. In *International conference on database theory* (pp. 316-330). **Springer**, Berlin, Heidelberg.

BUSCHMANN, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). **A system of patterns:** Pattern-oriented software architecture. Wiley Publishing.

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**.

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.

CUEVAS-VICENTTÍN, V., KIANMAJD, P., LUDASCHER, B., MISSIER, P.,

CHIRIGATI, F., WEI, Y., ... & DEY, S. (2014). The PBase scientific workflow provenance repository. **International Journal of Digital Curation**, *9*(2), 28-38.

CUEVAS-VICENTTÍN, V., LUDASCHER, B., MISSIER, P., BELHAJJAME, K., CHIRIGATI, F., WEI, Y., and ALTINTAS, I. 2016. ProvONE: **A PROV Extension Data Model for Scientific Workflow Provenance**. Disponível em: http://jenkins-1.dataone.org/jenkins/view/Documentation%20Projects/job/ProvONE-Documentation-trunk/ws/provenance/ProvONE/v1/provone.html

DALPRA, H. L., COSTA, G. C. B., SIQUEIRA, 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**.

FALCI M. L. F., BRAGA R., SROËLE V. and DAVID J. (2018). Software Process Improvement through the Combination of Data Provenance, Ontologies and Complex Networks.In **Proceedings of the 20th International Conference on Enterprise Information Systems** - *Volume 2: ICEIS*, ISBN 978-989-758-298-1, pages 61-70. DOI: 10.5220/0006700400610070

FIGUEIREDO, Daniel Ratton. Introdução a redes complexas. **Atualizaçoes em Informática**, p. 303-358, 2011. Disponível em: http://www.land.ufrj.br/~daniel/JAI-RC/JAI-RC.pdf

FONSECA, V. S., BARCELLOS, M. P., & de ALMEIDA FALBO, R. (2017). An ontology-based approach for integrating tools supporting the software measurement process. **Science of Computer Programming**, 135, 20-44.

FUGGETTA, A. (2000). Software process: a roadmap. In **Proceedings of the Conference on The Future of Software Engineering** (ICSE '00).

FUGGETTA, A., & DI NITTO, E. (2014, May). Software process. In **Proceedings of the on Future of Software Engineering** (pp. 1-12). ACM.

GENCEL, C., PETERSEN, K., MUGHAL, A. A., & IQBAL, M. I. (2013). A decision support framework for metrics selection in goal-based measurement programs: GQM-DSFMS. **Journal of Systems and Software**, 86(12), 3091-3108.

GROTH, P., & MOREAU, L. (2013). PROV-overview. An overview of the PROV family of documents. **W3C Recommendation** Disponível em: http://www.w3.org/TR/2013/NOTE-prov-overview-20130430

GUARINO, N. (Ed.). (1998). Formal ontology in information systems: **Proceedings of the first international conference** *(FOIS'98), June 6-8, Trento, Italy* (Vol. 46). IOS press.

GUIZZARDI, G., Falbo, R. A., & GUIZZARDI, R. S. S. (2008). A importância de Ontologias de Fundamentação para a Engenharia de Ontologias de Domínio: o caso do domínio de Processos de Software. **Revista IEEE América Latina**, 6(3), 244-251.

JONES, C., & BONSIGNOUR, O. (2011). **The economics of software quality**. Addison-Wesley Professional.

MCGUINNESS, D. L., & VAN HARMELEN, F. (2004). OWL web ontology language overview. **W3C recommendation**, *10*(10). Disponível em: https://www.w3.org/TR/owl-features/

MOREAU, L., CLIFFORD, B., FREIRE, J., FUTRELLE, J., GIL, Y., GROTH, P., ... & PLALE, B. (2011). The open provenance model core specification (v1. 1). **Future generation computer systems**, *27*(6), 743-756.

MOREAU, L., HUYNH, T. D., MICHAELIDES, D. (2014) An online validator for provenance: Algorithmic design, testing, and API. In: **International Conference on Fundamental Approaches to Software Engineering**. Springer, Berlin, Heidelberg, p. 291-305.

NAEDELE, M., CHEN, H. M., KAZMAN, R., CAI, Y., XIAO, L., & SILVA, C. V. (2015). Manufacturing execution systems: A vision for managing software development. **Journal of systems and Software**, 101, 59-68.

NAEDELE, M., KAZMAN, R., & CAI, Y. (2014). Making the case for a manufacturing execution system for software development. **Communications of the ACM**, *57*(12), 33-36

OLIVEIRA, W., MISSIER, P., OCANÃ, K., de OLIVEIRA, D., & BRAGANHOLO, V. (2016, June). Analyzing provenance across heterogeneous provenance graphs. In **International Provenance and Annotation Workshop** (pp. 57-70). Springer, Cham.

OWL. Web Ontology Language. **W3C**. 2012. Disponível em: https://www.w3.org/OWL/

PRABHUNE, A., ZWEIG, A., STOTZKA, R., GERTZ, M., & HESSER, J. (2016, June). Prov2ONE: an algorithm for automatically constructing ProvONE provenance graphs. In **International Provenance and Annotation Workshop** (pp. 204-208). Springer, Cham.

REN, Y., PARVIZI, A., MELLISH, C., PAN, J. Z., VAN DEEMTER, K., & STEVENS, R. (2014, May). Towards competency question-driven ontology authoring. **In European Semantic Web Conference** (pp. 752-767). Springer, Cham.

ROBINSON, I., WEBBER, J., & EIFREM, E. (2013). **Graph databases**. O'Reilly Media, Inc.

TAHIR, T., RASOOL, G., & GENCEL, C. (2016). A systematic literature review on software measurement programs**. Information and Software Technology**, 73, 101-121.

VAN SOLINGEN, R., BASILI, V., CALDIER, G., & ROMBACH, H. D. (2002). Goal question metric (gqm) approach. **Encyclopedia of software engineering**.

WOHLIN, C., RUNESON, P., HÖDT, M., OHLSSON, M. C., REGNELL, B., & WESSLÉN, A. (2012). **Experimentation in software engineering**. Springer Science & Business Media.

YIN, R. K. (2014). **Case study research**: Design and methods (Fifth). Sage Publications.

# ANEXOS

## Termo de Consentimento Livre Esclarecido

OBJETIVO DO ESTUDO
Este estudo visa realizar uma investigação sobre uma abordagem de apoio à extração de conhecimento de dados de processos de software e suporte à compreensão dos dados através de visualizações

PROCEDIMENTO
Neste estudo, você deverá responder algumas perguntas sobre o comportamento dos dados. Todos os documentos utilizados neste estudo serão apresentados ao participante e deverão ser preenchidos pelo próprio. Você receberá orientações sobre como realizar as atividades. Para participar deste estudo solicitamos a sua especial colaboração em:

(1) Responder aos questionários sobre os dados;
(2) Permitir que os dados resultantes da sua participação sejam estudados;
(3) Informar o tempo gasto nas atividades;
(4) Quando os dados forem coletados, seu nome será removido destes e não será utilizado em nenhum momento durante a apresentação dos resultados.

CONFIDENCIALIDADE
Eu estou ciente de que meu nome não será divulgado em hipótese alguma. Também estou ciente de que os dados obtidos por meio deste estudo serão mantidos sob confidencialidade.

BENEFÍCIOS E LIBERDADE DE DESISTÊNCIA
Eu entendo que, uma vez o experimento tenha terminado, os trabalhos que desenvolvi serão estudados. Os benefícios que receberei deste estudo são limitados ao aprendizado do material que é distribuído e ensinado. Também entendo que sou livre para realizar perguntas a qualquer momento, solicitar que qualquer informação relacionada a minha pessoa não seja incluída no estudo ou comunicar minha desistência de participação, sem qualquer penalidade. Por fim, declaro que participo de livre e espontânea vontade com o único intuito de contribuir para o avanço e desenvolvimento de técnicas e processos para a Engenharia de Software.

PESQUISADORA RESPONSÁVEL
Maria Luiza Furtuozo Falci
Programa de Pós Graduação em Ciência da computação
PGCC-UFJF

PROFESSORA RESPONSÁVEL
Profa. Regina Braga
Programa de Pós Graduação em Ciência da computação
PGCC-UFJF

Nome (em letra de forma): _____
Assinatura: _____

Data:_____

**Formulário Avaliação**

**Questões Parte 1**

Hora de início das respostas __:__

1) Quantos usuários são membros do projeto “Project31”?
   _____
   _____
2) Quais usuários estão relacionados a execuções?
   _____
   _____
3) Qual usuário está relacionado a maior quantidade de execuções?
   _____
   _____
4) Qual usuário está relacionado à atividade “P3225” ?
   _____
   _____
5) Quais execuções a atividade “P3225” possui?
   _____
   _____
6) O usuário “user173” está relacionado a qual/quais execução/execuções?
   _____
   _____
7) Quantos usuários não possuem execuções relacionadas a eles?
   _____
   _____

Hora do fim das respostas __:__

**Questões Parte 2**

Hora de início das respostas __:__

1) A quantas execuções o usuário 181 está relacionado?
   _____
   _____
2) Qual é a data do último login do usuário 181?
   _____
   _____
3) Qual é a data de criação do projeto ligado ao usuário 181?
   _____
   _____
4) Qual é a prioridade da maioria das atividades relacionadas ao usuário 181?
   _____
   _____
5) Quem possui mais relações, o usuário 181 ou 185?
   _____
   _____
6) O usuário 185 está relacionado a execuções de atividades de quantos projetos?
   _____
   _____

Hora do fim das respostas __:__

**Questões Parte 3**

Hora de início das respostas __:__

1) As atividades do projeto 31 são de qual tipo?
   _____
   _____

2) Qual foi o período em que ocorreram atividades do projeto?
   _____
   _____

3) Quais usuários estão relacionados às atividades de programação?
   _____
   _____

Hora do fim das respostas __:__