Paper review: using ontologies for verification and validation of workflow-based experiments

|  |  |  |  |
| --- | --- | --- | --- |
| Journal | Autores | Link - scopus | Compilado por |
| Web Semantics: Science, Services and Agentson the World Wide Web | Tomasz Miksa, Andreas Rauber |  | Luiz Gustavo Dias - UFF |

Resumo

No domínio de e-science, experimentos científicos requerem ferramentas especiais, software, e *workflows* que permitam pesquisadores reproduzir, transformar, visualizar, e interpretar dados. Entretanto estudos recentes apontam que muitos estudos não podem ser replicados devidos a dificuldades na infraestrutura utilizada, deste modo surgem os mecanismos para proveniência, que são acoplados em mecanismos de *workflow* para aumentar a replicabilidade dos estudos. No entanto o problema de armazenamento de dados continua sendo assim, e proposto no trabalho um modelo que integra ontologias que descreve o *workflow* bem como o seu ambiente (de alto e baixo nível, como *software, hardware,* e arquivos). Foi utilizada uma ferramenta que monitora o *workflow* e cria automaticamente o modelo de contexto, também foi criada uma ontologia nomeada VPlan que permite a modelagem de requisitos e validação. As ontologias criadas foram testadas em cinco *workflows* no Taverna, que diferem em dependências de *software* e serviços adicionais.

Visão geral

Os autores introduzem o trabalho ressaltando um dos vários fatores influenciadores na não-reprodutibilidade de experimentos científicos, como a infraestrutura e dependências de terceiros (arquivos, *software*, e bibliotecas por exemplo). Além do mais, enfatizam que, *workflows* foram desenvolvidos para empregar padronização, bem como ocultar a complexidade da infraestrutura do experimento. Ferramentas como VisTrails, Taverna, e Kepler, são utilizadas em várias áreas ciências, e permitem por exemplo, que o pesquisador represente graficamente os experimentos em forma de fluxos, construídos com elementos pré-definidos, proporcionando a reprodutibilidade, e o reuso dos experimentos. Para fundamentar o problema, os autores citam que, um trabalho recente, relata que apenas 30% dos quase 1500 *workflows* desenvolvidos na ferramenta Taverna, publicados no myExperiment podem ser executados novamente (o que não significa que a parcela executável traz resultados corretos). Os autores afirmam que todas as dependências relacionadas a execução de *workflows*, compõem um contexto que precisa ser capturado e verificado para determinar se a reexecução do mesmo produz resultados verdadeiramente corretos. Embasado nisso, foi utilizada na pesquisa o VFramework, que pode ser utilizado para verificar e validar execuções de um *workflow*, a ferramenta também utiliza o modelo de contexto para documentar os ambientes em que o fluxo e executado, e permite comparar execuções do *workflow* sem a necessidade de acessar dois ambientes ao mesmo tempo. O modelo de contexto utiliza ontologias que descrevem o *workflow* e o seu ambiente, que inclui informações de alto e baixo nível. Os fluxos desenvolvidos na ferramenta Taverna, são relacionadas a três domínios específicos: análise de dados de sensores de engenharia civil, classificação de musica em recuperação de informação, e pesquisa clinica medica, que possuem várias dependências , e foram avaliados em diferentes sistemas operacionais.

VFramework

Os objetivos da ferramenta não são relacionadas apenas a re-executação de experimentos, mas incluem por exemplo, preservação digital de configurações que *worflows* necessitam para ser executados posteriormente (longo prazo), em ambientes computacionais diferentes. O que se torna uma característica essencial tendo em vista a velocidade do avanço tecnológico, tanto de *software* quanto de *hardware*. A maneira mais viável para comparar a execução de um mesmo *workflow* em diferente contextos, é coletar dados referentes a execução no ambiente original, e usa-los como referencia no ambiente no qual será re-executado. VFramework trabalha essa maneira em dois passos básicos: primeiro ele coleta as informações relacionadas a execução do ambiente original; após essa captura, a ferramenta utiliza as informações para verificar e validar a re-execução no ambiente de implantação. O modelo de contexto armazena essas informações no primeiro passo, e as fornece no segundo.

Os autores utilizaram dois modelos de contexto para validar a re-execução, o primeiro modelo, da execução em ambiente original, e o segundo modelo em ambiente de reimplantação. Se houver diferenças entre os modelos, é gerada uma lista de diferenças entre os ambientes em questão, permitindo verificar prováveis causas. Os dados verificados não são apenas os de entrada e saída, mas também os dados intermediários entre processos do fluxo.

O modelo de contexto pelo VFramework é composto por cinco partes principais:

* *Workflow model core*: elemento central que descreve o modelo do *workflow* e serve de base para outras etapas;
* *Workflow dependencies*:descrição de *hardware* e *software* utilizados para execução do *workflow*;
* *Workflow instance data*:dados utilizados, processados, produzidos;
* *File format specification*: formato de arquivos utilizados no *workflow*;
* *Validation requirements*: requisitos e métricas utilizados para validação;

workflow model core

*Workflows* podem ser executados dentro de um mecanismo especifico, o que não implica que os mesmos são independentes do ambiente ao qual são executados. Dentro dos *workflows* podem existir iterações com outro *software* que está sendo executado pelo sistema (como banco de dados por exemplo). Portanto antes de iniciar o fluxo, frequentemente outros serviços devem ser inicializados pelo ambiente. Todos estes serviços pertencem ao contexto do *workflow* e devem ser capturados em seu ambiente original*.*

Conhecendo o contexto, é possível identificar as dependências e definir seus limites para a execução do *workflow*. Tais dependências devem ser analisadas, verificadas e validadas, para confirmar a replicabilidade do *workflow*. A analise do *Workflow Model Core* permite identificar quais dependências são necessárias para a execução do *workflow* e qual delas devem ser coletados os dados para verificação e validação.

Os autores utilizaram Taverna2Archi e Archi2OWL, a primeira ferramenta lê o *workflow* e gera o modelo Archi2OWL, enquanto a segunda transforma o modelo em uma ontologia OWL. O processo de conversão do *workflow* em uma ontologia OWL é feita de maneira automática, e não requer que usuários manipulem ArchiMate.

Os benefícios listados, com o uso de ontologias para representar o modelo, é a possibilidade de aplicar queries, para por exemplo:

1. Identificar se o *workflow* tem dependências externas, como por exemplo *web services*;
2. Verificar se algum script utiliza bibliotecas externas;
3. Listar possíveis licenças utilizadas no *workflow*;

Listar as etapas do *workflow* bem como suas entradas e saídas.

Workflow dependencies

Dependências relacionadas a *workflows* são todos os componentes utilizados durante sua execução. Se tratando de sistemas Linux por exemplo, a versão exata de todos os pacotes deve ser documentadas. Dependências externas são todos os componentes utilizados para a execução do *workflow*, mas não estão hospedados na mesma plataforma que o executa, como por exemplo *web services*. Esse passo e importante, pois pode acontecer por exemplo, de um *web service* não estar mais disponível, ou ser atualizado para uma versão mais nova, o que pode resultar em resultados alterados quando comparados a execução original. Uma solução para este problema levantado pelos autores é sugerir um serviço diferente que e compatível com o original.

workflow instance data

*Workflows* diferem na maneira como produzem dados, o que depende da sua implementação. Alguns processam dados em memoria do sistema e no final gravam dados no disco rígido, outros salvam resultados intermediários no disco e portanto, habilitam a execução e validação de dados intermediários.

A forma de como os *workflows* fazem este procedimento interferem na captura de instancias de *workflows*. Taverna permite exportar a proveniência utilizando a ontologia Janus, que foi utilizada para modelar dados de instancias de *workflows*.

file format specification

A proveniência também foi utilizada para enriquecer o contexto do modelo com informações sobre o formato de dados de cada uma das etapas do *workflow*. Esta etapa e necessária, pois facilita o processo de validação permitindo a escolha de uma ferramenta que realize esse tipo de comparação. Na pesquisa, os autores utilizaram a ferramenta DROID que realiza identificações automáticas de formatos de arquivo, e pode identificar mais de 250 tipos de formatos. Tambem foi utilizada a ontologia PREMIS, que e um padrão para preservação de metadados, baseado em um modelo de dados que defin entidades descritas (objetos, eventos, e agentes), propriedades, e relações.

Estudo de caso

O primeiro *workflow* é relacionado ao domínio música com o objetivo de com categorizar músicas cuja sua categoria e desconhecida. O *workflow* trabalha com bibliotecas locais para a extração de fragmentos da musica e recursos web para aquisição de dados musicais. O segundo *workflow* avaliado lida com analise de dados de sensores em engenharia, para analisar a segurança de barragens, caracterizando seu comportamento estrutural, e utiliza serviços web e linguagem R para fins estatísticos.

Os outros três *workflows* são utilizados no domínio da biomedicina e são utilizados para investigações de aspectos da doença de Huntington, que utilizam recursos e R para realizar analises estatísticas para identificar genes, e *web services* para representar processos biológicos.

VFramework foi aplicado em todos os cinco *workflows*, com a finalidade de responder cinco perguntas:

* Q1: Como podemos verificar e validar *workflows* executados em ambientes diferentes?
* Q2: Qual o impacto das dependências de *software* na verificação automática?
* Q3: Como podemos identificar dependências do *workflow*?
* Q4: Quais dados devem ser publicados com o *workflow* para que seja possível verificar e validar?
* Q5: Como podemos validar *workflows* com um conjunto de métricas definidas?

Q1: Como podemos verificar e validar *workflows* executados em ambientes diferentes?

Os experimentos foram re-executados em sistemas operacionais que diferiam em suas versões, distribuições e arquiteturas. Quando o ambiente diferia na versão e distribuição, foi possível utilizar todas as partes do modelo de contexto para análise. Para execução em Windows e Linux a comparação foi limitada devido diferenças na arquitetura do sistema (foi necessário por exemplo procurar por programas equivalentes e avaliar a compatibilidade manualmente).

Q2: Qual o impacto das dependências de *software* na verificação automática?

Os *workflows* analisados diferem quanto ao número de dependências adicionais. Entretanto nos casos que era necessária uma quantidade menor de dependências, a verificação e validação pode ser automaticamente executada. Entretanto em casos que dependiam de um numero maior de dependência, a verificação e validação era mais custosa. Para todos os casos o modelo de contexto atuou de forma universal, detectando dependências faltantes.

Q3: Como podemos identificar dependências do *workflow*?

Dois *workflows* analisados faziam uso de R. Para um as dependências foram identificadas por meio de monitoração da execução do *workflow*. Para outro foram realizadas consultas SPARQL sob o modelo.

Q4: Quais dados devem ser publicados com o *workflow* para que seja possível verificar e validar?

Foram avaliados duas opções: A primeira é publicar apenas o modelo de contexto, e a segunda publicar arquivos e pacotes. Ambas mostraram ser bem-sucedidas, o modelo de contexto e uma fonte suficiente de informações para verificação do *workflow*.

Q5: Como podemos validar *workflows* com um conjunto de métricas definidas?

A estratégia aplicada para geração de validação métricas que geram requisitos para cada etapa do *workflow*

e verifica cada valor usando um comparador de formato correspondente, provou funcionar corretamente, mas tem suas limitações. A estratégia foi capaz de detectar alterações em diferentes estágios do fluxo de trabalho processamento e com base nestes, fomos capazes de identificar etapas em que a reexecução do fluxo de trabalho foi alterada. A aplicação de as métricas específicas de formato nos permitiu comparar corretamente os dados

coletados para validação. Entretanto dependendo dos arquivos comparados, o resultado pode não ser satisfatório, como se por exemplo, os arquivos de entrada/saída fizerem uso de dados como data/hora.

Análise segundo o leitor

1. Qual tipo de proveniência abordada no trabalho?

**Essa informação não é explicita no texto, entretanto os autores citam que a proveniência capturada é relacionada a dependências para re-execução de experimentos. Desta forma assume-se que a proveniência abordada é a proveniência retrospectiva.**

1. Qual tipo de ontologia utilizada no trabalho?

**São utilizadas varias ontologias (VPlan, Janus (Taverna), PREMIS) para criar um modelo de contexto. Entretanto não são utilizadas ontologias de domínios específicos, tendo em vista que vários *workflows* de diversos domínios são avaliados.**

1. Em qual etapa do ciclo de vida do *workflow* a ontologia foi aplicada?
2. Qual a principal vantagem em se utilizar ontologia no contexto da pesquisa?

**A principal vantagem percebida no trabalho, foi que através da formalização via ontologias. É possível identificar através da relação entre seus componentes, dependências, diferenças e similaridades entre diferentes execuções de um mesmo *workflow.***

1. Questões de glanularidade são abordadas no artigo?

**Sim. A granularidade e abordada quando levada em consideração a descrição manual das dependências.**