CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA CELSO SUCKOW DA FONSECA

DEPIN

Sistemas ara Internet

Letícia Silva de Moura

INTEGRAÇÃO E MIGRAÇÃO DE INTERFACE DE LINHA DE COMANDO PARA WORKFLOW: UM RELATO DE EXPERIÊNCIA DO SISTEMA DE IMPUTAÇÃO APPRAISAL

RIO DE JANEIRO

2019

Letícia Silva de Moura

INTEGRAÇÃO E MIGRAÇÃO DE INTERFACE DE LINHA DE COMANDO PARA WORKFLOW: UM RELATO DE EXPERIÊNCIA DO SISTEMA DE IMPUTAÇÃO APPRAISAL

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Tecnologia em Sistemas para Internet, do Depin / EIC, do Centro Federal de Educação Tecnológica Celso Suckow da Fonseca.

Orientadores: Jorge de Abreu Soares e Diogo Silveira Mendonça

RIO DE JANEIRO
2019

Dedico este trabalho à minha família,
principalmente à minha mãe e minha avó.
À minha irmã, Fernanda, agradeço e peço
desculpas por todos os dias em que não
pude ir ao seu encontro.

AGRADECIMENTOS

As próximas palavras não farão jus a todo esforço e compreensão que meus familiares e amigos tiveram comigo durante a pesquisa e desenvolvimento desse trabalho. A todos vocês agradeço por estarem presentes em minha vida e nesse momento importante. Agradeço ao meu orientador Prof. Jorge Soares, pela paciência que teve comigo, por acreditar que eu seria capaz e por toda sabedoria com a qual me guiou. Agradeço também ao meu co-orientador que em todas as minhas dúvidas me norteou da melhor forma possível.

Aos meus colegas de faculdade e aos que a deixaram.

Aos meus professores que foram meus orientadores durante todo o curso.

À minha família, incluindo os que não são de sangue, por todo apoio que sempre me deram.

Ao meu namorado que se fez muito importante nessa reta final.

Aos meus amigos, que mesmo distantes, não deixaram de se fazer presentes, principalmente ao Alexandre, que sempre me apoiou.

Enfim, a todos os que por algum motivo contribuíram para a realização desta pesquisa.

RESUMO

MOURA, Letícia Silva de. **Integração e migração de interface de linha de comando para Workflow: Um relato de experiência do sistema de imputação Appraisal**. 2019. 47. Trabalho de Conclusão de Curso – Centro Federal de Educação Tecnológica Celso Suckow da Fonseca. Rio de Janeiro. Cidade, 2019.

O sistema de imputação de dados Appraisal foi projetado para funcionar com interface de linha de comando (CLI) e configuração por arquivos de propriedades. Tais configurações e interfaces, além de serem de difícil utilização, exigem esforço adicional para realizar a técnica de imputação composta, onde mais de um algoritmo é executado em sequência para que a imputação ocorra. No uso desta técnica, o Appraisal precisa ser executado mais de uma vez integrando a saída de uma execução como entrada da próxima, exigindo assim uma configuração flexível da sequência de execuções a serem realizadas. Com objetivo de permitir e facilitar tal integração, foi realizada a adaptação do Appraisal para ser executado utilizando o sistema de gerenciamento de workflow – ou somente sistema de workflow -Taverna.

Este trabalho consiste em um relato de experiência do processo de migração do Appraisal de CLI para uma interface de sistema de workflow, possibilitando assim sua integração de forma fácil e flexível. Contém relatado o passo a passo realizado, as dificuldades encontradas e os pontos fortes e fracos da ferramenta Taverna para realizar este processo. Nossa contribuição consiste em conhecimento e lições aprendidas para praticantes que porventura precisarem realizar um processo de migração e integração de interface semelhante (CLI para Workflow).

Palavras-chave: Imputação, Sistemas de Workflow, Interface de linha de comando, Integração de Software, Migração de Software.

ABSTRACT

MOURA, Letícia Silva de. **Integration and migration from command line interface to Workflow: An experience report of the Appraisal imputation system**. 2019. 47. Final work for the undergratuation - Federal Center of Technological Education – Rio de Janeiro. city, 2018.

The Appraisal is a data imputation system and it is designed to work from command line interface (CLI) and its configured by property files. These features made the Appraisal be difficult to use and require additional effort to run the composite imputation technique, where is executed more than one algorithm in sequence for the imputation to occur. In the use of composite imputation technique, the Appraisal needs to be executed more than once and integrate the output of an execution as input of the next execution and these run sequences require a flexible configuration. In order to allow and facilitate such integration, the Appraisal was adapted to be executed using the workflow management system - or just workflow system - Taverna.

This work is an experience report of the process of Appraisal migration from CLI for a workflow system interface. This migration makes possible an easy and flexible integration. This work reports the step by step accomplished, the difficulties encountered and the strengths and weaknesses of the Taverna tool to carry out this process. The contribution consists of knowledge and lessons learned for practitioners who may need to perform a similar interface migration and integration process (CLI for Workflow).

Keywords: Imputation, Workflow System, Command line Interface, Software Integration, Software Migration.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Lidar com dados ausentes – ou faltantes - é um problema recorrente e está presente em muitas bases de dados. Para lidar com essa questão da melhor maneira, os cientistas de dados muitas vezes precisam preencher esses valores de forma manual. A ação de preencher dados faltantes é chamada de imputação. Bases com dados ausentes são também chamadas de base suja.

Para auxiliar os cientistas e fazer com que os valores imputados sejam os mais próximos dos reais, foram desenvolvidas técnicas para realizar a imputação de forma computacional e lógica. Um dessas técnicas de imputação é a imputação composta.

Imputação composta é uma técnica baseada na combinação de estratégias, onde a escolha da estratégia influencia diretamente nos dados que serão imputados. A imputação composta foi proposta na tese de (SOARES, 2007), onde também foi apresentado o sistema Appraisal, sendo esse um software que possibilita a execução da técnica descrita acima. No Appraisal, as estratégias para a imputação dos dados são aplicadas a partir de comitês de complementação de dados ausentes. Para o trabalho atual, não é importante saber o que e quais são esses comitês.

Após a imputação composta, cientistas desenvolveram projetos e softwares baseados na respectiva técnica. Esses projetos são implementações de novas funcionalidades, incrementos, melhorias nas imputações do Appraisal, entre outros desenvolvimentos.

O objetivo desse trabalho começa a fazer sentido a partir do problema de integração do Appraisal. A cada novo projeto desenvolvido a partir dele, é preciso replicar seu código para fazer a modificação. O problema consiste nesses projetos nunca se integrarem, nem se comunicarem. É como uma esteira do git em que se criam branchs, mas esses nunca voltam para o master. Ou seja, os projetos avançam, mas sem comunicação entre eles e o Appraisal fica cada vez mais desatualizado.

Esse tipo de situação origina problemas como reescrita de código, mudanças sem controle, falta de padrão entre os sistemas, tempo gasto com desenvolvimento maior do que necessário, tendência a se ter sistemas cada vez mais isolados, entre outros.

O foco desse trabalho é sanar os problemas citados acima, que são derivados da falta de comunicação e integração do software Appraisal. Será proposta uma solução arquitetural para resolver esses problemas.

Foi realizado um estudo sobre os conceitos de arquitetura de software para achar a arquitetura mais apropriada para atender ao objetivo proposto. O estudo para escolha das arquiteturas candidatas se desenvolveu com foco em sistemas de workflow. Essa escolha se deu pois quando sistemas se tornam complexos ou possuem uma meta em comum, a solução é frequentemente alcançada através de workflows (BARKER; VAN HEMERT, 2008). Sistemas de workflow evitam que se reinvente a roda e se tenha retrabalho. Por essas características serem parecidas com as necessidades a serem atendidas do sistema Appraisal, workflow foi a arquitetura candidata escolhida para ser estudada.

A partir desse ponto, foram escolhidos 5 sistemas de workflow para serem analisados. A ordem dos sistemas de workflow a serem testados foi estabelecida a partir de critérios como trabalhos relacionados à imputação. Conforme fossem sendo realizados os testes, caso o primeiro sistema de workflow escolhido não atendesse à necessidade do trabalho, seria analisado um próximo sistema e assim por diante. O primeiro sistema de workflow escolhido para ser testado foi o Taverna. Como após alguns testes o Taverna mostrou-se capaz de atender ao proposto - e posteriormente isso foi comprovado -, os outros sistemas serão mencionados somente da seção de revisão de literatura.

2 REFERENCIAL TEÓRICO

Nesse capítulo, serão apresentados os conceitos que embasaram o trabalho proposto - como as técnicas de imputação e imputação composta -, o Appraisal e os conceitos de engenharia de software aplicados no desenvolvimento do trabalho - arquitetura de software e workflow.

2.1 IMPUTAÇÃO

Imputação é a técnica usada para substituição artificial estatística de dados faltantes (*missing data)*. Dados faltantes é um problema recorrente em pesquisas. Podem se dar por desistência de pessoas na resposta de questionários, por preguiça ou falta de capacidade de responder a perguntas, por exemplo (SCHAFER; OLSEN, 1998).

A técnica de imputação de dados surgiu com a ideia de encontrar uma maneira de preencher de forma plausível os dados faltantes (MARCIO JOSÉ DE ALMEIDA, 2016). Os primeiros métodos de imputação eram relativamente simples, como a média e a mediana do atributo. Atualmente, os métodos de imputação se dividem em imputação simples ou única e imputação múltipla (ENGELS; DIEHR, 2003). A imputação deu origem a n técnicas mais especificas de imputação, como a imputação composta

2.1.1 Imputação Composta

Imputação composta é uma técnica desenvolvida por (SOARES, 2007) onde a imputação atua em conjunto com a aplicação de estratégias. Nessa técnica, a imputação de um atributo é precedida pela aplicação de uma ou mais estratégias no processo de complementação de dados ausentes. O exemplo de uso de uma estratégia é utilizar o agrupamento de dados e a seleção de colunas, respectivamente, precedendo a imputação. A técnica é aplicada em cada dado v de uma base de dados com dados ausentes.

Os processos de imputação composta abrangem principalmente os conceitos descoberta de conhecimento em Bases de Dados, ordem de tarefas,

estratégia para a imputação de um atributo, algoritmos usados no processo de imputação e planos de imputação.

2.1.2 Appraisal

Appraisal é um sistema projetado para experimentar e analisar imputações compostas. Imputação composta é uma classe de técnicas de imputação que combinam uma ou mais tarefas usadas na etapa de mineração de dados para gerarem um novo valor a ser imputado. O Appraisal é ajustável desde a etapa de implementação de planos de imputação até a medição da qualidade do dado imputado (SOARES, 2007).Tem por objetivo avaliar os efeitos da aplicação das tarefas de seleção de atributos e agrupamento de dados precedendo à complementação de dados ausentes em bases de dados. Possui um conjunto com aplicação de comitês de complementação de dados para o processo de imputação.

O sistema é composto basicamente por quatro módulos:

Crowner, que é a execução dos planos de imputação.

Committee, que é o comitê de complementação de dados ausentes, onde gera-se valores de imputação baseados nos demais valores sugeridos pelas estratégias e os outros atributos da tabela.

Reviewer, que verifica a qualidade das sugestões de imputação produzidas pelos módulos *Crowner* e *Committee*.

Eraser, que simula valores ausentes em uma base de dados segundo um mecanismo e um percentual de ausência definidos pelo usuário.

O Appraisal possui uma arquitetura voltada à construção de classes abstratas que não contêm implementações específicas de nenhum algoritmo. Essa característica possibilita ao sistema ser adaptado, já que as funcionalidades oferecidas podem ser executadas de diferentes formas, com a variação dos algoritmos que já estejam implementados ou que porventura venham a ser integrados no sistema.

Após a primeira versão do software Appraisal, foram feitas novas pesquisas baseadas na imputação composta. Um exemplo é o sistema desenvolvido em (MEDINA, 2012), onde foram apresentadas técnicas de seleção com Algoritmos

Genéticos e agrupamento com Redes de Kohonen na aplicação da imputação composta.

Outro sistema relacionado a imputação composta, é o apresentado em (MONTEIRO, 2008). Uma variação do processo de imputação, proposto por SOARES (2007), precedido por tarefas usualmente aplicadas no processo de mineração de dados, onde avalia os resultados da aplicação de diversas estratégias de complementação de dados de natureza categórica em bases de dados que possuem valores ausentes em suas tuplas.

Há, também, o apresentado em (RIBEIRO, 2010) que utiliza proveniência para complementação de dados no contexto do processo de ETL (Extraction, Transformation, and Load). Onde aborda uma forma de alcançar melhores resultados a partir de técnicas de complementação que consideram o enriquecimento das tabelas com dados de proveniência.

2.2 WORKFLOW

Workflow (fluxo de trabalho) pode ser definido como um conjunto de etapas encadeadas que objetivam automatizar processos, a partir de um conjunto de regras definidas. Caracteriza-se também por ser reproduzível, ou seja, ser manipulado de forma mais simples e compreensível por diferentes pessoas (MARCIO JOSÉ DE ALMEIDA, 2016), o que possibilita um melhor gerenciamento do projeto.

Existem dois tipos de workflows: corporativo e científico. Foi inicialmente desenvolvido para o ambiente corporativo, mas workflows científicos se tornaram importantes e necessários para muitas aplicações.

Workflow corporativo é composto por uma coleção de processos, tarefas ou atividades automatizadas que representam uma atividade presente no negócio (LIU et al., 2010). Tem como característica a prioridade ao cliente. Auxilia na desburocratização de processos e no gerenciamento de tarefas. É tipicamente menos dinâmico que os científicos que tendem a mudar com mais frequência (TAYLOR et al., 2014).

Pode-se definir workflow científico como a especificação formal de um processo científico que representa os passos a serem executados em um

determinado experimento. É centrado na condução desses experimentos, visando automatizar tarefas e evitar retrabalho, o que possibilita aos pesquisadores perderem menos tempo com manuseio de recursos computacionais e focar mais em suas pesquisas (DEELMAN et al., 2009). Workflow Management (ou gerenciamento de fluxo de trabalho) diz respeito ao que está correlacionado ao workflow, como métodos, ideias, perspectivas, softwares e técnicas usados na modelagem do sistema. É possível modelar um workflow a partir de diferentes perspectivas, sendo as mais relevantes: fluxo de controle, fluxo de dados, organizacional e Tratamento de exceções (BRAGHETTO; RESUMO, 2014).

Workflows são atividades que envolvem a execução de tarefas e podem ser controlados por um coordenador humano ou por um sistema de gerenciamento de workflow - um software (RUSINKIEWICZ; SHETH, 1995).

2.3 SISTEMAS DE GERENCIAMENTO DE WORKFLOW

Sistemas de gerenciamento de workflow - ou somente sistemas de workflow - são softwares que ajudam na modelagem e análise dos processos de workflow. Proporcionam maior eficiência aos fluxos e melhor monitoramento dos processos. Sua arquitetura é aberta para oferecer suporte a várias plataformas e é projetada para facilitar mudanças de forma dinâmica (WEIN et al., 2008)

Um sistema de workflow não executa efetivamente tarefas, ele garante que as tarefas sejam executadas da forma correta. São responsáveis por gerenciar as informações relacionadas ao workflow, assegurando assim, que no momento correto as informações corretas cheguem à pessoa correta, ou seja, enviadas para o programa destinado certo no momento certo. Sua principal vantagem é ser um software genérico, o que significa poder ser aplicado em diversas situações. (OUYANG et al., 2015).

A partir do sistema de workflow é possível montar os passos do workflow, escolher e modificar o valor dos parâmetros a serem passados, configurar os serviços utilizados, entre outros. Nesse âmbito, existem diversas estruturas (sistemas) que permitem aos cientistas automatizar tarefas por esses sistemas de gerenciamento. Na seção abaixo serão apresentados alguns sistemas de workflow.

3 REVISÃO BIBLIOGRÁFICA

Nesse capítulo será apresentada uma abordagem mais técnica do sistema Appraisal, bem como os sistemas que derivaram dele e seus respectivos trabalhos. Serão também apresentados os sistemas de workflow escolhidos como candidatos.

3.1 SISTEMAS DE WORKFLOW EXISTENTES

Para a escolha dos sistemas de workflow candidatos que seriam estudados nesse trabalho foram levadas em consideração suas características e as principais áreas de pesquisa onde os mesmos foram mais aplicados. As características levadas em consideração foram: as linguagens suportadas - o Appraisal é desenvolvido em Java -, a aparente simplicidade no desenvolvimento do workflow, a quantidade de informações disponíveis sobre eles e a documentação.

Em (BARKER; VAN HEMERT, 2008) foi apresentada uma pesquisa dos sistemas de workflow mais populares. Entre eles os selecionados foram o Pegasus, GridNexus, Taverna e Triana. Dentro do domínio do Appraisal, esses foram os workflows que mais pareceram interessantes inicialmente, incluindo junto o Kepler, citado em (DEELMAN et al., 2009).

Nos próximos parágrafos será feita uma breve apresentação dos sistemas mencionados acima. Como o Taverna foi o sistema escolhido para ser utilizado nesse trabalho, ele será apresentado de forma mais profunda na próxima seção.

3.1.1 Pegasus

O Pegasus foi projetado para ser um compilador (plano/mapa), onde seria possível oferecer acesso direto ou indireto aos dados, caso esses já estivessem computados e disponíveis, executando-os em dados potencialmente distribuídos e recursos computacionais. Em algumas ocorrências compreenderia um acesso simplificado e em outras contém várias etapas inter-relacionadas (DEELMAN et al., 2015).

Possibilita a portabilidade, gerenciamento de dados, desempenho, proveniência, escalabilidade e ferramentas de confiabilidade e depuração - como o

pegasus-analyzer, sua ferramenta de depuração. No quesito desempenho, o Pegasus possui um “mapeador” que reordena, agrupa e prioriza tarefas com objetivo de otimizar. No quesito escalabilidade, ele possibilita manipular cálculos e dados grandes.

Em uma abstração, na entrada de dados no Pegasus há uma linguagem de alto nível que identifica apenas o cálculo, sem descrições de recursos, e de locais de dados (File Aware) (DEELMAN et al., 2015). Em sua execução foca no desempenho e confiabilidade do workflow. Ele localiza instintivamente locais físicos para os componentes e os dados. Além de coletar a procedência das informações em tempo de execução. Após esse mapeamento, o workflow executável é gerado de uma maneira específica para o workflow de destino e então é enviado para o mecanismo e seu agendador de tarefas no host de envio.

3.1.2 GridNexus

O GridNexus é um sistema gráfico para criar e executar workflows científico sem um ambiente Grid (BARKER; VAN HEMERT, 2008). A tecnologia de grid permite contínuas autenticação e autorização e fornece interfaces padrão para acesso a serviços remotos. Isso torna muito mais fácil automatizar acessos e o uso de recursos de computação distribuída (BROWN et al., 2005). Permite ao usuário montar processos complexos envolvendo recuperação de dados, análise e visualização através da construção de um gráfico acíclico direcionado (DAG) em um ambiente visual.

O objetivo do GridNexus é ser capaz de integrar de forma transparente recursos de computação de alto desempenho em um ambiente de computação em grid (grade) não exigindo que os cientistas conheçam a infraestrutura de rede e a computacional. A ideia é ser um construtor de aplicativos grid para incorporar uma interface gráfica e uma linguagem de script extensível e de alto nível. Foi projetado com a execução da interface em camadas separadas, assim os processos de alto nível (para cientistas) podem incluir processos de nível inferior (para programadores) (BROWN et al., 2005). Utiliza a linguagem JXPL. Os aplicativos que mais utilizam o GridNexus lidam com aprendizado de máquina e atividades complexas de

manipulação de dados. Como o LISP, o JXPL é adequado para esses tipos de aplicativos.

Uma vez projetado um workflow utilizando a interface gráfica do GridNexus, a GUI, ele é traduzido para a linguagem JXPL. O GridNexus separa a GUI da execução do workflow, portanto, uma vez construído um workflow (usando o JXPL) é possível executá-lo local ou remotamente.

3.1.3 Triana

O Triana é um ambiente gráfico de resolução de problemas (PSE - Problem Solving Environment) que opera a partir de um portal de usuário que permite a composição de aplicações científicas. Essa composição é feita arrastando unidades ou ferramentas de caixas de ferramentas (componentes de programação) e soltando-os em um bloco de rascunho ou espaço de trabalho. A conectividade entre as unidades é obtida desenhando cabos sujeitos à verificação de tipos

Possui uma arquitetura modular desacoplada e cada componente - unidade de execução - pode ser usado de forma individua ou coletiva pelos programadores ou pelos usuários finais. Um componente é a menor granularidade de trabalho que pode ser executada. Geralmente são classes Java (se for em outra linguagem precisa ter um código de quebra apropriado) com nome, portas de entrada e saída, parâmetros de nome/valor opcionais e um único algoritmo ou método de processo. Cada componente tem uma definição codificada em XML com seu nome especificado (SHIELDS; TAYLOR, 2004).

Triana é baseada nos fluxos de dados e de controle. A execução dos componentes dentro de Triana é acionada por esses fluxos. No caso do de dados, quando eles chegam à porta de entrada do componente acionam a execução. No caso do de controle, a execução do componente é acionada por um comando de controle.

Uma das principais características da linguagem Triana workflow é que ela não tem suporte explícito para construções de controle. Existe um componente de loop que controla a execução repetida em um sub-workflow e um componente lógico que controla a ramificação do workflow.

3.1.4 Kepler

Kepler é um gerenciador de workflow de código aberto (SOMMERVILLE, 1995) que objetiva desenvolver um framework para projetar, executar e implantar workflows científicos. Oferece uma grande variedade de modelos computacionais herdados do sistema Ptolomeu II, sob o qual foi construído. Utiliza a linguagem marcação de modelagem (Modelling Markup Language). O Kepler possui design orientado ao ator, onde os atores são blocos reutilizáveis independentes de computação, por exemplo, serviços da Web, chamadas a banco de dados, entre outros. Eles consomem dados de entradas e gravam em um conjunto de saídas (outports).

Uma das características do Kepler é a preocupações com a comunicação separada do ator (fluxo de dados) com a coordenação global do fluxo de trabalho, que é definida em um componente separado. Essa separação permite modelar o fluxo de dados de forma síncrona. É um sistema extensível, onde os desenvolvedores podem acrescentar funcionalidades ao sistema com facilidade e de acordo com as necessidades dos cientistas. Os usuários finais também podem estender a biblioteca de agentes usando os recursos de coleta de serviços da web para importar e executar novos componentes do sistema (GUIDE, 2004).

Sua interface gráfica foi herdada do Vergil de Ptolomeu II. A interface Vergil é intuitiva para manipular workflows científicos, ela serve como um editor para a linguagem de modelagem do Kepler, a Modeling Markup Language (MoML). MoML é baseada em XML e sua principal vantagem é o fato de workflows em formato XML poderem ser compartilhados entre cientistas com simplicidade.

3.1 TAVERNA

Taverna é um sistema de gerenciamento de workflow de código aberto e com reconhecimento de rede. Fornece um conjunto de middleware para dar suporte a cientistas que realizam experimentos através de simulação computacional com uso intensivo de dados em recursos distribuídos. Taverna é implementada como uma arquitetura orientada a serviços, baseada em padrões de serviço da Web (BARKER; VAN HEMERT, 2008).

Os workflows gerenciados podem ser projetados e executados em máquinas locais através do ambiente Taverna workbench ou em outros clientes/interfaces Web através do Taverna Server (ou Taverna commandline application - aplicativo de linha de comando). Esses modos de execução englobam diferentes estilos de workflows. O Taverna Server é um ambiente focado para o trabalho finalizado a uma comunidade maior de cientistas, fornecendo acesso a uma coleção workflows (geralmente através da interface web, a Taverna Player). A desvantagem é que não é possível alterar ou adicionar novos workflows à coleção. O Taverna Workbench foi o ambiente escolhido para o desenvolvimento do trabalho.

3.1.1 Taverna workbench

Taverna Workbench é um software da Apache para projetar e executar workflows. Conta com um ambiente de interface para manipular esses workflows. Esse ambiente permite que os cientistas testem métodos de análise, desenvolvam workflows a partir do zero ou componha-os a partir de workflows já existentes (WOLSTENCROFT et al., 2013). Os workflows são projetados em forma de diagrama com sequencias de passos que são definidas pelo usuário. Os diagramas são montados a partir de **serviços**, **portas de entradas** e **portas saídas** e ficam dispostos no painel *Workflow diagram*, localizados na perspectiva *Design*. O Taverna Workbench possui 3 perspectivas: *Design*, *Result* e *myExperiment*. Neste trabalho, foram utilizadas as perspectivas Design e Result. As áreas que serão apresentadas foram utilizadas nesse trabalho, as outras não serão detalhadas.

A perspectiva *Result* é onde se visualiza as saídas do workflow e é exibida quando o workflow é executado. A *Result* possui as áreas: *Workflow runs*, *Workflow results* e *Diagram graphs*. A área Diagram Graphs possui 2 abas e na aba *Graph* é onde é exibido um resumo do workflow executado. Abaixo da área *Diagram Graphs*, está a área *Workflow results*, onde são exibidas as portas de saída de workflow e seus respectivos valores. As portas saídas de workflow são listadas na parte direita (área *output ports*) e são separadas por abas. Os valores de cada porta são exibidos na parte esquerda (value output port, Figura 2). Os valores são exibidos após clicar no ‘Value 1’, localizado na área direito, abaixo de cada porta. As áreas não apresentadas não foram utilizadas no trabalho.

A perspectiva *Design* é onde o workflow é criado e manipulado. Nela existem as áreas: *Workflow Diagram*, *Workflow Explorer e Service Panel*. Na área *Workflow diagram* é onde o workflow fica disponível para ser criado e editado, através da interação e combinação de serviços. Na área *workflow explorer* são exibidos os serviços utilizados no diagrama – ou workflow. Na área *Service Panel* é onde os serviços ficam disponíveis para utilização.

Os serviços possibilitam ao desenvolvedor criar os passos do workflow. O painel de serviços (*service panel*) vem por padrão com alguns serviços disponíveis, como o Text constant, SpreeadsheetImport, Beanshell, entre outros. Cada serviço possui uma finalidade, no que diz respeito à programação, o serviço padrão é o Beanshell. Beanshell é um serviço de script/códigos Java simples. É possível importar serviços externos, mas alguns necessitam da instalação de plug-ins para serem importados. Todos os serviços intrínsecos possuem entradas e saídas.

Existem dois tipos de entrada e saída: as entradas e saídas dos serviços e as portas de entrada e saída do workflow. As entradas e saídas dos serviços são definidas em cada utilização de cada serviço e precisam ter seu tipo definido previamente. Esses tipos são definidos a partir de códigos (Depths) correspondentes. Por exemplo, uma saída do tipo string precisa ser definida como depth 0. As portas de entrada e saída de workflow são semelhantes aos serviços. A utilização das mesmas não é obrigatória, mas quando utilizadas precisam estar ligadas a alguma saída de algum serviço. É através das saídas de workflow que o Taverna Workbench exibe o conteúdo na perspectiva *Result.*

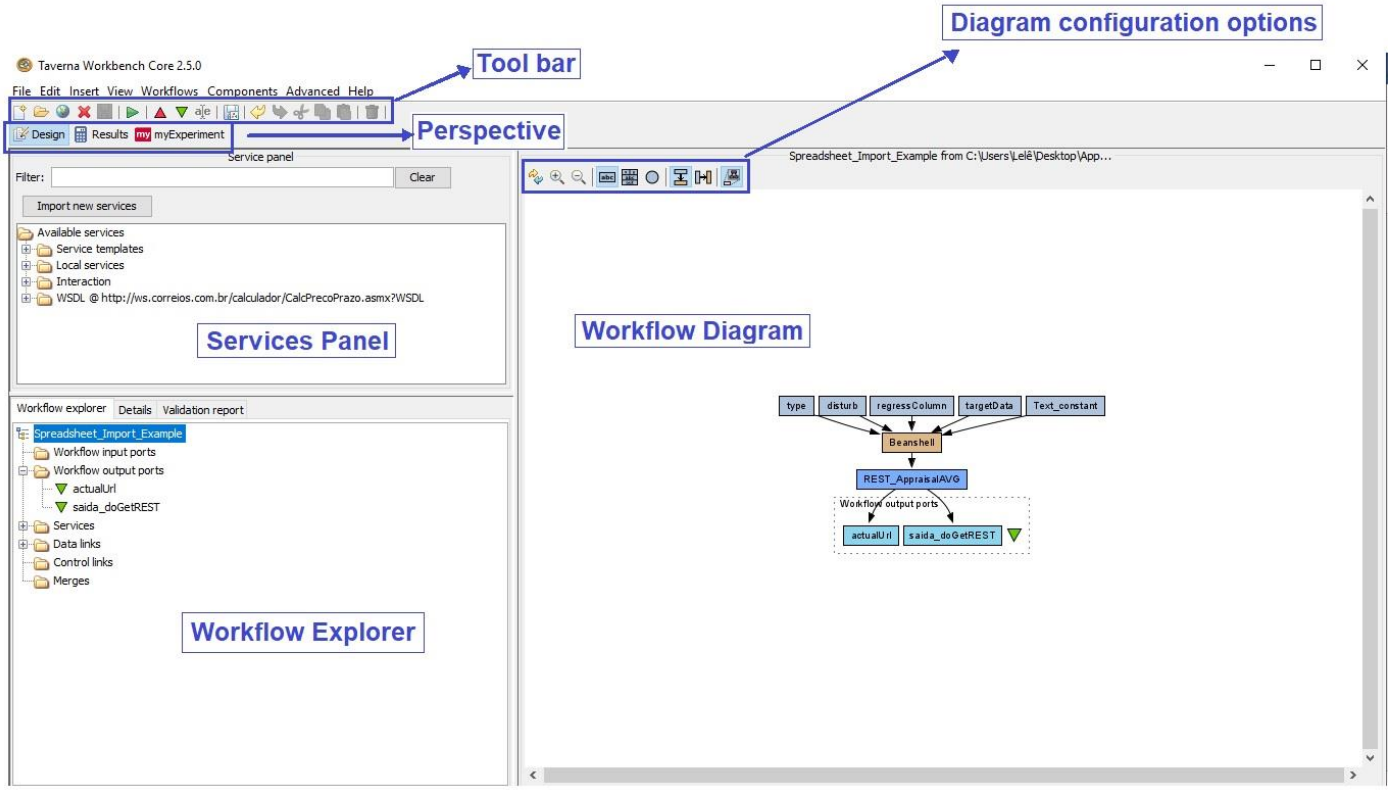


Figura 2: Perspectiva Design

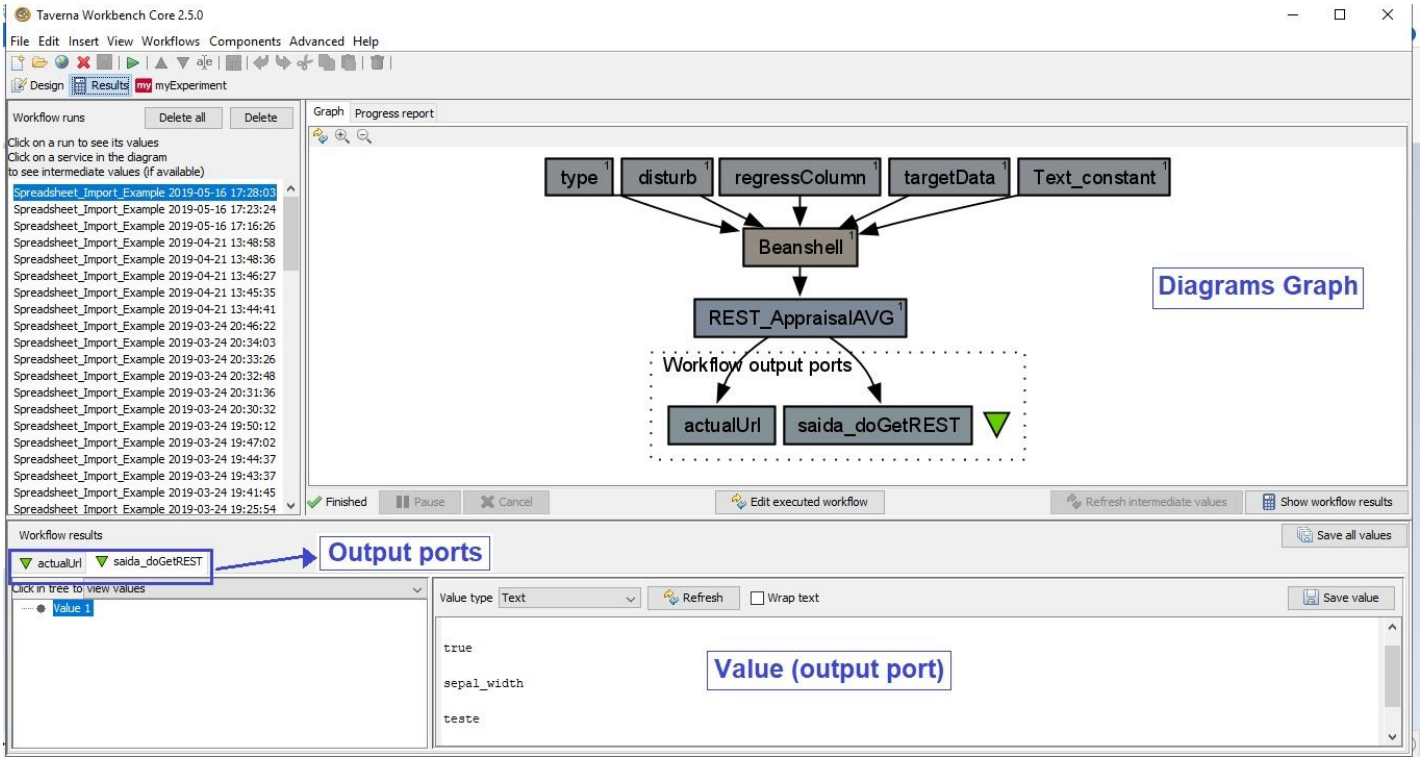


Figura 1: Perspectiva Results

3.2 ESTUDO DE CASOS SEMELHANTES

Dentre os artigos analisados relacionados a workflow e imputação de dados, foi analisado principalmente o artigo “Aprimorando Processos de Imputação Multivariada de Dados com workflows” (CASTANEDA; FERLIN; GOLDSCHMIDT, 2007), onde é apresentada uma plataforma baseada em workflow para tratar técnicas de imputação de dados. O artigo também apresenta uma forma de testar a plataforma a partir da prova de conceito.

O problema de valores ausentes está presente em diversas áreas. (CASTANEDA; FERLIN; GOLDSCHMIDT, 2007) propõe uma plataforma baseada em workflows capaz de automatizar a configuração, execução e avaliação de técnicas de imputação de dados. O trabalho cita dois principais métodos para solucionar problemas de ausência multivariada de dados: Joint-Modelling e Imputação Iterativa. Assim como o workflow proposto para o Appraisal, a plataforma utiliza também a prova de conceito que nesse caso foi feita através do estudo de caso de um mecanismo de imputação iterativa com e sem realimentação. “A imputação iterativa consiste na divisão de um problema multivariado em n problemas univariados, onde cada atributo que apresenta valores ausentes é resolvido de maneira independente, por técnicas tradicionais da solução de problemas univariados.”. “A imputação iterativa com realimentação é exemplo de uma implementação variante que re-introduz os valores imputados na base de dados ao final de cada iteração”.

A plataforma apresentada em (CASTANEDA; FERLIN; GOLDSCHMIDT, 2007) foi desenvolvida em Java e os resultados são escritos em disco, no formato de XML (para os valores imputados) e Excel (para o resultado das análises). Os autores optaram por possibilitar ao analista experimentar n implementações para que seja possível decidir-se pelo melhor resultado gerado. Para os autores, experimentar envolve quatro passos principais:

- Combinar diferentes implementações de imputação iterativa;
- Utilizar, onde possível, diferentes algoritmos para a resolução de cada iteração (onde o problema se torna univariado);
- Variar os diversos parâmetros de cada algoritmo empregado;

• Avaliar de maneira uniforme os resultados obtidos por cada combinação única dos fatores acima;

A plataforma trabalha a partir de componentes e isso proporcionou à tarefa de modificar experimentos ser simples e automatizada. Os componentes são passos, que foram desenvolvidos em forma de componentes, que precedem o processo de imputação. O sistema foi elaborado a partir do encadeamento das entradas e saídas desses componentes. São acionados através da passagem de dados e parâmetros de configuração. Para ilustrar o processo de experimento de imputação iterativa foi utilizado um diagrama baseado em UML que apresenta um plano de trabalho com esses passos. Nesse diagrama, são passos a escolha do padrão de ausência, a divisão dos dados em conjuntos e as estratégias para imputação dos dados. A configuração de objetos do workflow é estabelecida através da utilização de uma API ou de arquivos de propriedades, onde é possível habilitar os componentes e configurar seus parâmetros.

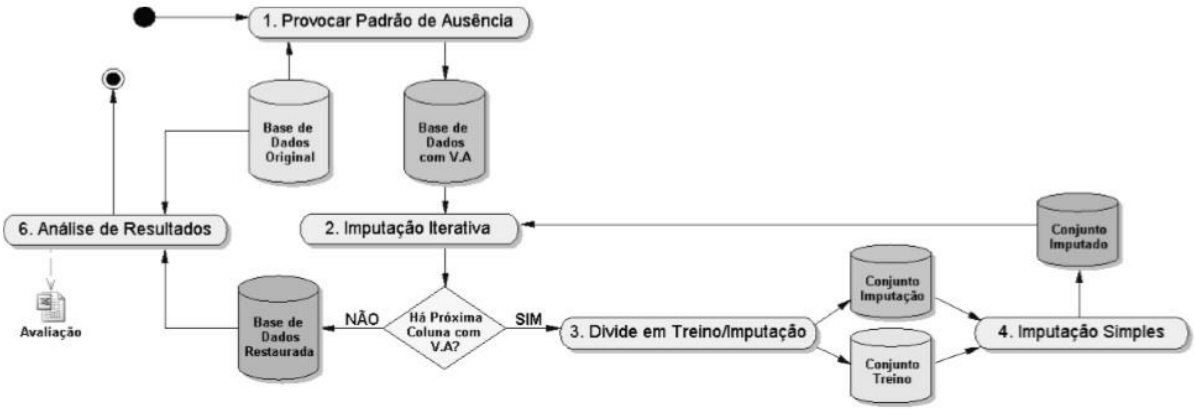


Figura 3: Figura 3: UML processo de experimento de imputação iterativa (CASTANEDA; FERLIN; GOLDSCHMIDT, 2007)

No trabalho corrente, utilizar a solução proposta em (CASTANEDA; FERLIN; GOLDSCHMIDT, 2007) atenderia às necessidades do Appraisal no ponto da imputação por estratégias, combinações de imputação e geração de resultados em arquivo externo. Contudo, o trabalho não prevê integração nem comunicação entre versões e funcionalidades acrescentadas com o tempo. Em outras palavras, o software não permite que as contribuições sejam acumulativas. A ideia do atual

trabalho é utilizar um sistema de workflow já existente, ao contrário da ideia apresentada no artigo citado acima, o que, no caso do Appraisal, possivelmente significaria ‘reinventar a roda’ Outra questão significativa que justifica a escolha de uma reestruturação do sistema e a escolha do padrão arquitetural, é o fato de que o Appraisal e seus pósteros não suportam interação com o usuário, além da sua configuração ser interna (através de um arquivo) e mais técnica e o software acima não tratar isso.

4 DESENVOLVIMENTO

O desenvolvimento desse trabalho se deu através de testes feitos com o Taverna Workbench. A escolha do Taverna Workbench se deu pelas razões de a) o software ser da Apache, sendo esse um nome conceituado b) a linguagem do script padrão ser em Java c) o software ser bem documentado e d) no início dos testes o software ter dado indícios de que atenderia à necessidade apresentada

O objetivo de contribuição do trabalho era conseguir manipular o Appraisal de forma externa e mais simples, deixando seu consumo mais independente. Essa mudança possibilita aos desenvolvedores e cientistas consumirem o Appraisal sem precisar alterar o font ou seu arquivo de configuração. Após esse trabalho, é possível realizar uma imputação composta através do Appraisal sem precisar rodá-lo.

O objetivo dos experimentos era encontrar a melhor forma de manipular o software Appraisal a partir do Taverna. A melhor maneira encontrada foi através de uma requisição get. Essa requisição é passada para um servlet através do TomCat. O servlet é responsável chamar uma classe que é responsável por consumir o projeto do Appraisal. Dessa forma, tanto os desenvolvedores, quanto os cientistas serão beneficiados. Os cientistas, sem conhecimento técnico, podem realizar a imputação através de uma interface e os desenvolvedores podem consumir essa imputação através do servlet. Por exemplo, do ponto de vista técnico, caso deseje-se criar uma estratégia que utilize a imputação ‘propagation’, não há mais a necessidade de reescrevê-la, copiá-la ou modificar o código do Appraisal para utilizá-la, é possível fazer isso através da chamada ao servlet.

Os experimentos foram divididos em ciclos e cada ciclo com n execuções. Os serviços usados nos testes foram o *Text constant* - serviço para guardar uma cadeia de caracteres -, o *SpreadsheetImport*, - serviço para importar bases de dados -, o *Beanshell* - serviço para criar scripts -, e o REST - serviço que possibilita a comunicação através dos métodos http. Além das portas de saída de workflow.

Ao final da realização dos ciclos testes, esperava-se encontrar os pontos positivos e negativos do software testado e as características mais relevantes que embasassem a eleição do mesmo como software escolhido. Como não havia conhecimento prévio prático do software testado, os testes inicializaram-se a partir de tarefas elementares.

Pelo trabalho visar atender à necessidade do software de imputação dados, nos testes foram feitas manipulações com uma mesma base de dados usada na tese (SOARES, 2007). A base de dados utilizada foi a Iris Plants, com 150 tuplas e cinco colunas, sendo quatro atributos numéricos e um classificador. Os atributos da base citada são: *sepal_length, sepal_width, petal_length, petal_width e class,* respectivamente.

Nos testes iniciais foram utilizadas a base original iris_plant e uma base suja salvas em formato de planilha (.xlsx). Para obter a base suja, foram deletados 16 valores da coluna Sepal_Width de forma manual e aleatória. Nos testes que consumiram o Appraisal e, consequentemente, o mySql, foram utilizadas a base original iris_plants e a base suja iris_mcar_sepalwidth_10. Todas as bases foram obtidas através do repositório git do Appraisal. As bases com extensão xlsx encontram-se em uma pasta chamada “Base” e as gerados no mySql são obtidas através de um script.

4.1 CICLOS DE TESTE

Ciclo 1: Importação da base de dados e execução de tarefas elementares

Execução 1: Ler/importar alguma base de dados e manipular algum atributo.

Execução 2: Ler/importar base de dados suja (com valores ausente) e rodar um algoritmo para identificar os valores ausentes. A partir disso gerar uma saída - um array - com as posições das tuplas que contêm os valores ausentes.

Considerações gerais: Após entender um pouco sobre o software, a execução do ciclo se concluiu sem muitas dificuldades. A base foi importada a partir de um caminho local, mas é possível utilizar um caminho online. A manipulação do workflow a partir do diagrama se mostrou interessante e útil por prover uma visão geral, mas bem detalhada do que foi feito.

Para melhor didática, os serviços utilizados (na área do diagrama) estão com seus respectivos nomes, mas não é um padrão obrigatório ou de boas práticas.

No valor do serviço *Text constant*, como mencionado anteriormente, foi inserido o caminho local da base suja. Esse valor foi passado para o *SpreadsheetImport*. Esse, por sua vez, é responsável por entender o valor passado como uma tabela e nele foram configuradas 5 saídas, onde cada uma corresponde a uma coluna da tabela. Essas saídas foram associadas às 5 entradas do criador de script *Beanshell*. Esse por sua vez foi configurado para receber essas entradas como arraylist. A partir disso, de acordo com o estipulado na execução 2, foi implementado um código que manipulasse esses arrays, identificasse os valores ausentes e gerasse uma saída de arraylist. Essa saída foi configurada no *Beanshell* como arraylist (depth 1) e associada à porta de saída ‘SeparaValoresAusentes’, como é possível ver na imagem 4. Após executar o workflow, é possível ver os valores do array de saída na área *workflow results,* na figura 5*.* Essa é basicamente como a estrutura do Taverna trabalha.

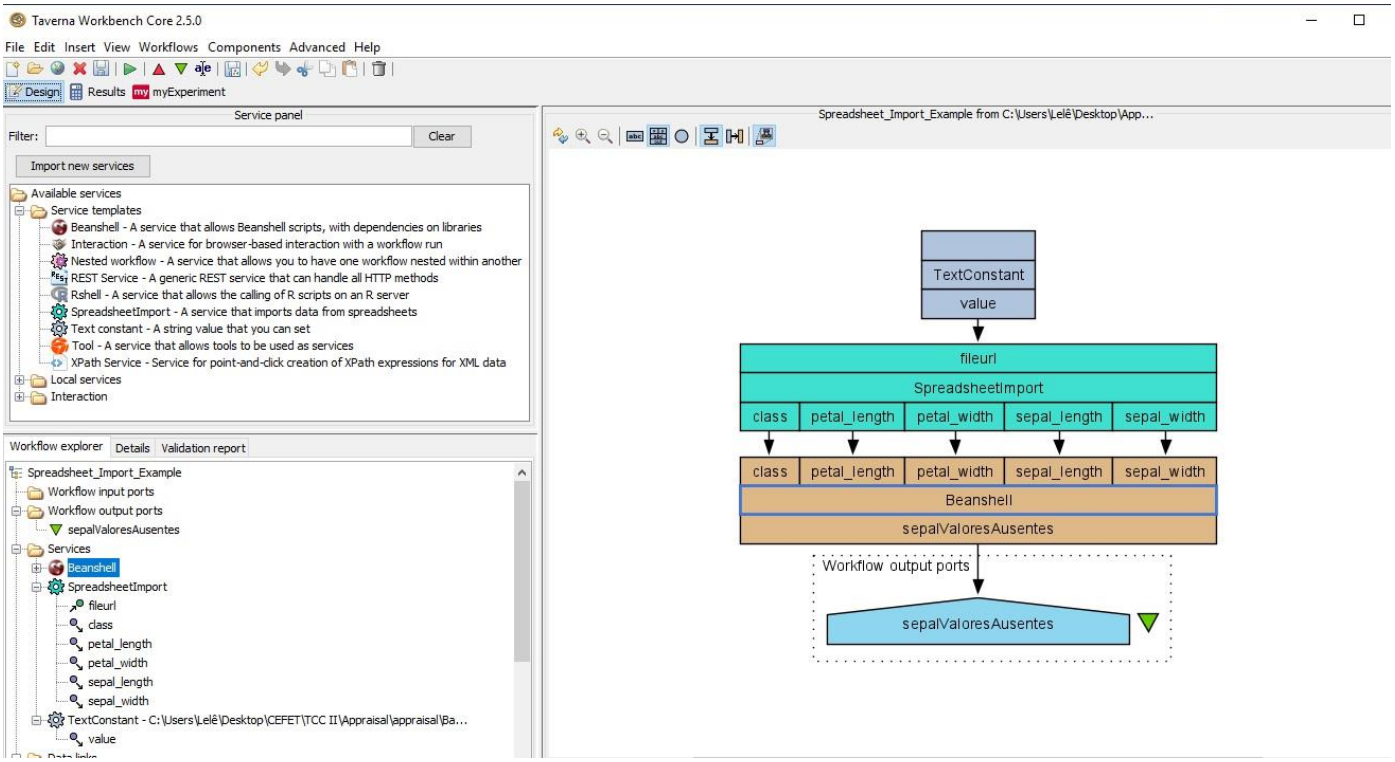


Figura 4: Perspectiva Design e área Workflow Diagram com os serviços e a porta utilizados. Ciclo 1, Execução 2

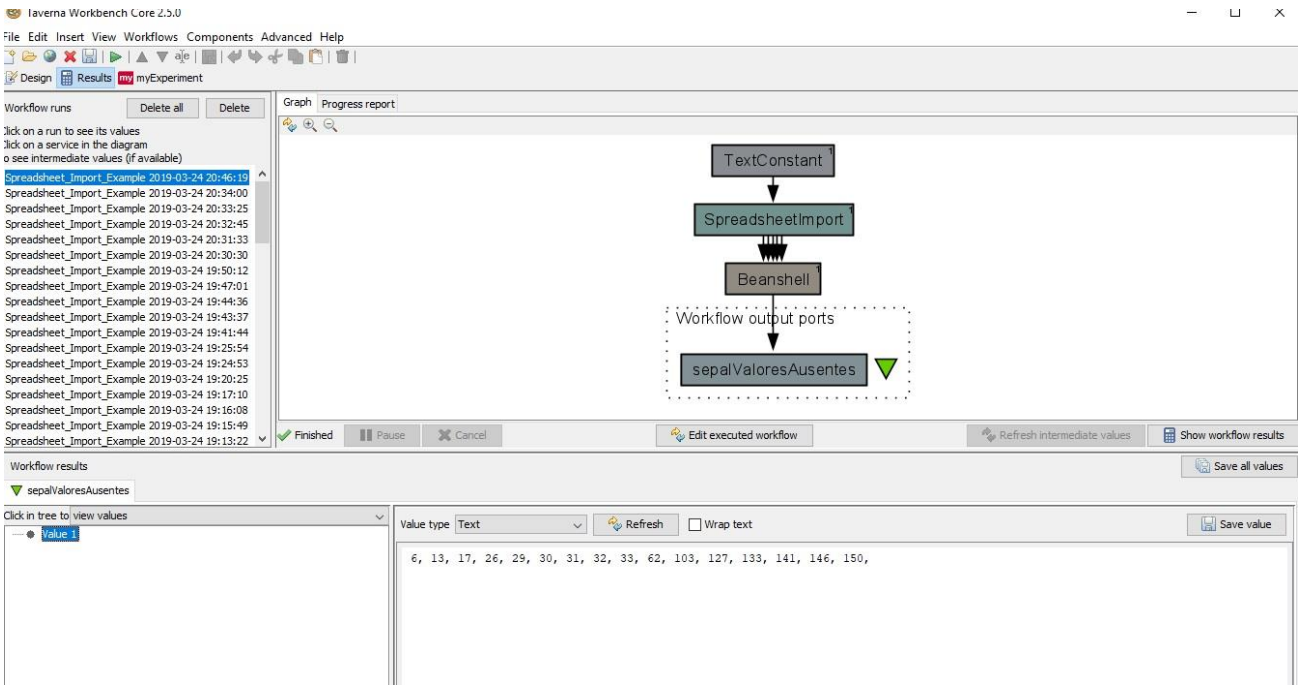


Figura 5: Resultado do Ciclo 1, Execução 2. À direita inferior, os valores da saída: a posição das tuplas com valores ausentes

Ciclo 2: Gerar saídas com valores imputados

Execução 1: A partir da base de dados suja, identificar os valores ausentes de uma tupla e substituí-los pelo valor 20. Gerar uma saída de array correspondente a essa tupla.

Execução 2: A partir da base de dados suja, preencher os valores ausentes com a soma: 20 + o valor da coluna *Petal_Width* correspondente à sua tupla. Gerar uma saída de array com os valores manipulados.

Considerações gerais: O valor “20”, nas execuções acima, foi escolhido pois nenhum dos valores presentes na base de dados ser maior que 10, o que facilita a identificação dos valores manipulados na tela de resultados. Foram definidas 2 saídas na execução 2: uma contendo o array com valores ausentes substituídos e a outra com o tamanho do array.

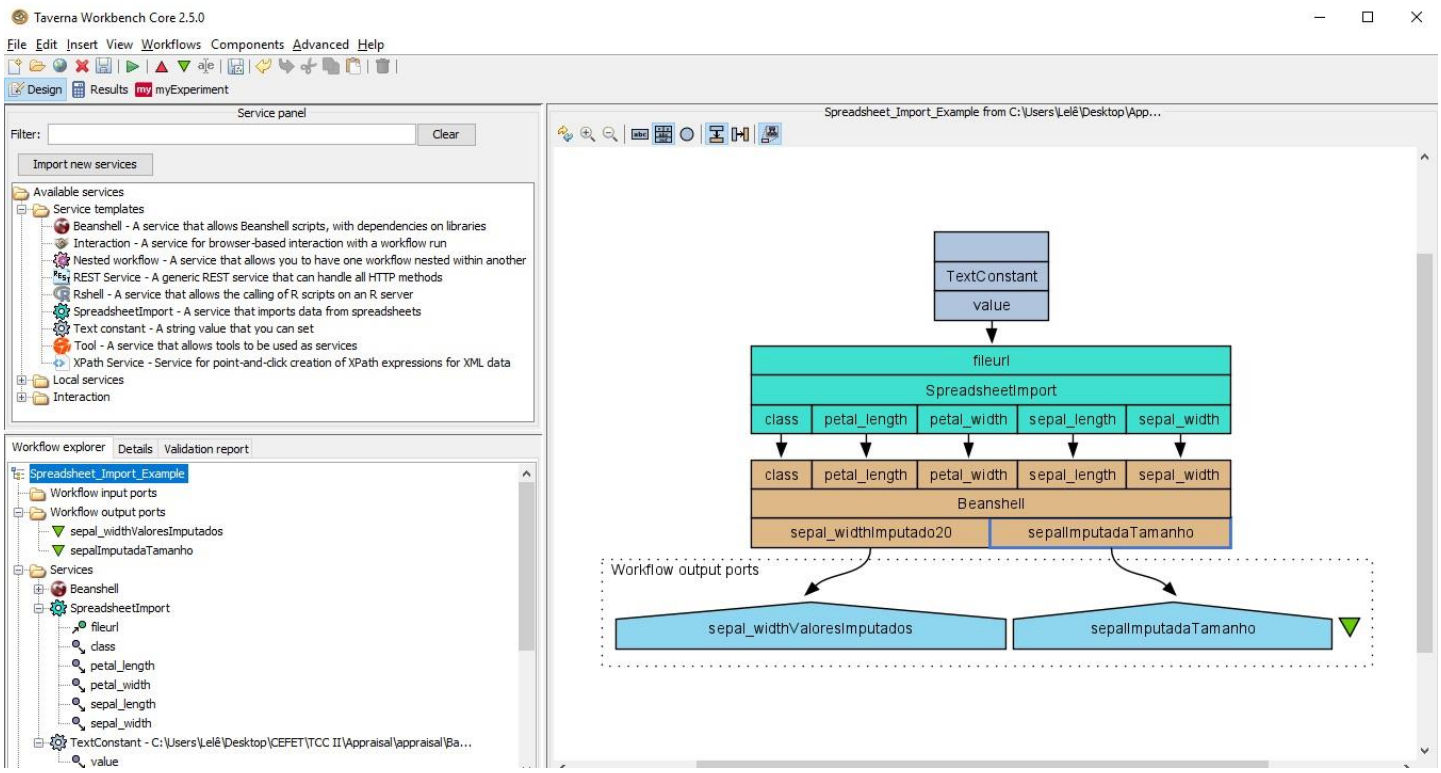


Figura 6: Ciclo 2, Execução 2. Saída da esquerda contendo o

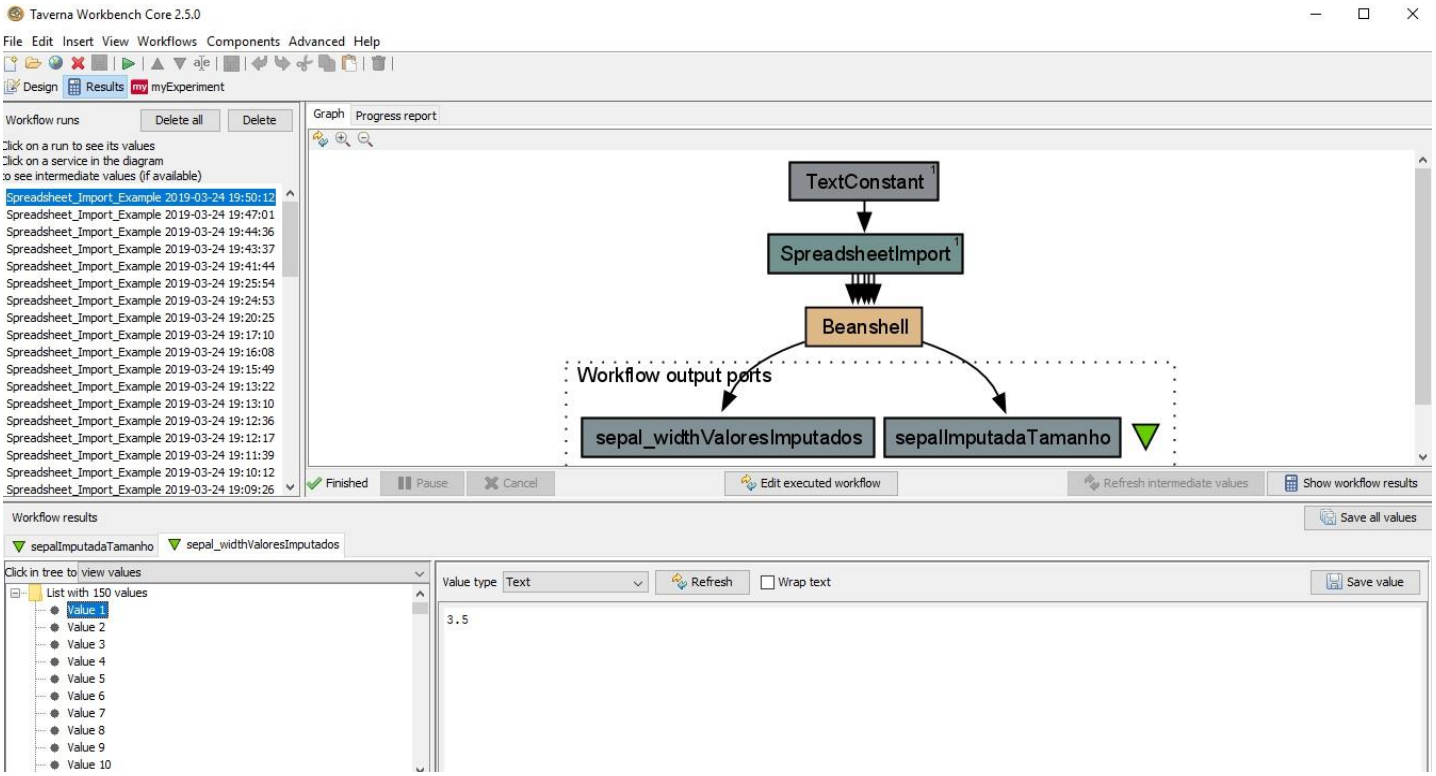


Figura 7: Resultado Ciclo 2, Execução 2. Área workflow results mostrando o array e o valor da sua primeira posição

Ciclo 3: Consumo de serviços externas

Os próximos ciclos de teste têm como objetivo testar e avaliar o desempenho do Taverna trabalhando com classes externas. Esses testes visam analisar como será feito o reaproveitamento das classes e códigos do Appraisal dentro do Taverna.

Após leitura da documentação do Taverna, foram identificadas algumas possíveis formas de se trabalhar com os recursos do Apparaisal: como uma API, a partir do *.jar* do projeto e consumi-lo como *web service*, por exemplo.

Execução 1: Utilização de API: importar o Appraisal como uma API no Taverna e consumir 2 de seus métodos (serviços).

Execução 2: Utilização de *web service*: a partir de um webservice existente, importá-lo com um novo serviço e consumi-lo no workflow.

Execução 3: Consumo de *web service*: a partir da criação de um webservice com endpoint local

Considerações Gerais – execução 1: Para utilizar o serviço API do Taverna é preciso instalar o plug-in 'API Consumer Service'. Após a instalação, aparecerá a opção do plug-in nas importações de novos serviços. Para importar uma API é preciso do *.jar* do projeto que se quer. Porém, o tipo de arquivo aceito no Taverna para utilizar esse serviço é xml. É preciso baixar um software externo relacionado ao Taverna, o 'API Consumer Tool', para gerar esse arquivo xml. Nesse software, informa-se o caminho do *.jar* do projeto e o software gera o xml. Contudo, não foi possível seguir com a execução por conta de impasses técnicos. O API Consumer Tool precisa das tags de documentação *javadoc* e o Appraisal não possui essas tags em seu código. Então, dar continuidade à essa execução seria algo trabalhoso e fugiria ao objetivo do trabalho. Porém, se futuramente essa documentação for feita, a importação do Taverna como API possivelmente será viável.

Considerações Gerais – execução 2: Para a execução 2 foi utilizado um *web service* existente dos Correios (disponível em: http://ws.correios.com.br/calculador/CalcPrecoPrazo.asmx?WSDL). Após importação do WSDL no ambiente do Taverna, os métodos ficam disponíveis e agrupados dentro do serviço WSDL, na área *Service Panel*, junto com os outros serviços. Para a execução, foi adicionado ao workflow o serviço 'ListaServicos', que possui 2 saídas. Foi escolhida uma de suas saídas, a 'parameters', para ser associada a uma porta de saída de workflow, pois se desejava observar o comportamento e conteúdo do serviço. É possível observar o workflow completo na imagem 8. Quando o workflow foi executado, foi retornada uma String formatada como conteúdo xml. O conteúdo dessa String correspondia aos serviços disponíveis dos correios para cálculo de preço e/ou prazo. É possível observar parte da saída na imagem 9.

Considerações Gerais – execução 3: Essa execução visa conseguir importar e consumir no Taverna um *web service* local. A partir disso, analisar a possibilidade de utilizar o Appraisal através de um *web service*.

Para tal, foi criado um programa Java de calculadora, contendo os métodos somar e subtrair. Esse programa foi criado para observar o comportamento do Taverna em relação a *web services* locais. Na importação do *web service* é preciso informar o caminho do arquivo wsdl, correspondente ao web service.

Após a criação do programa Java e seu wsdl, ao colocar o caminho do wsdl na importação do Taverna e confirmar, o Taverna não reconheceu o wsdl e apresentou um erro. Foi identificado que gerar esse arquivo para ser reconhecido pelo Taverna é trabalhoso do ponto de vista técnico, então nesse caso seria necessário um pouco mais de conhecimento técnico por parte da pessoa que fosse reproduzir o teste ou realizar alguma modificação e isso foge ao objetivo do trabalho, além de ser uma tarefa que consome uma quantidade considerável de tempo. Então, por essas 2 razões, a execução não foi levada a frente.

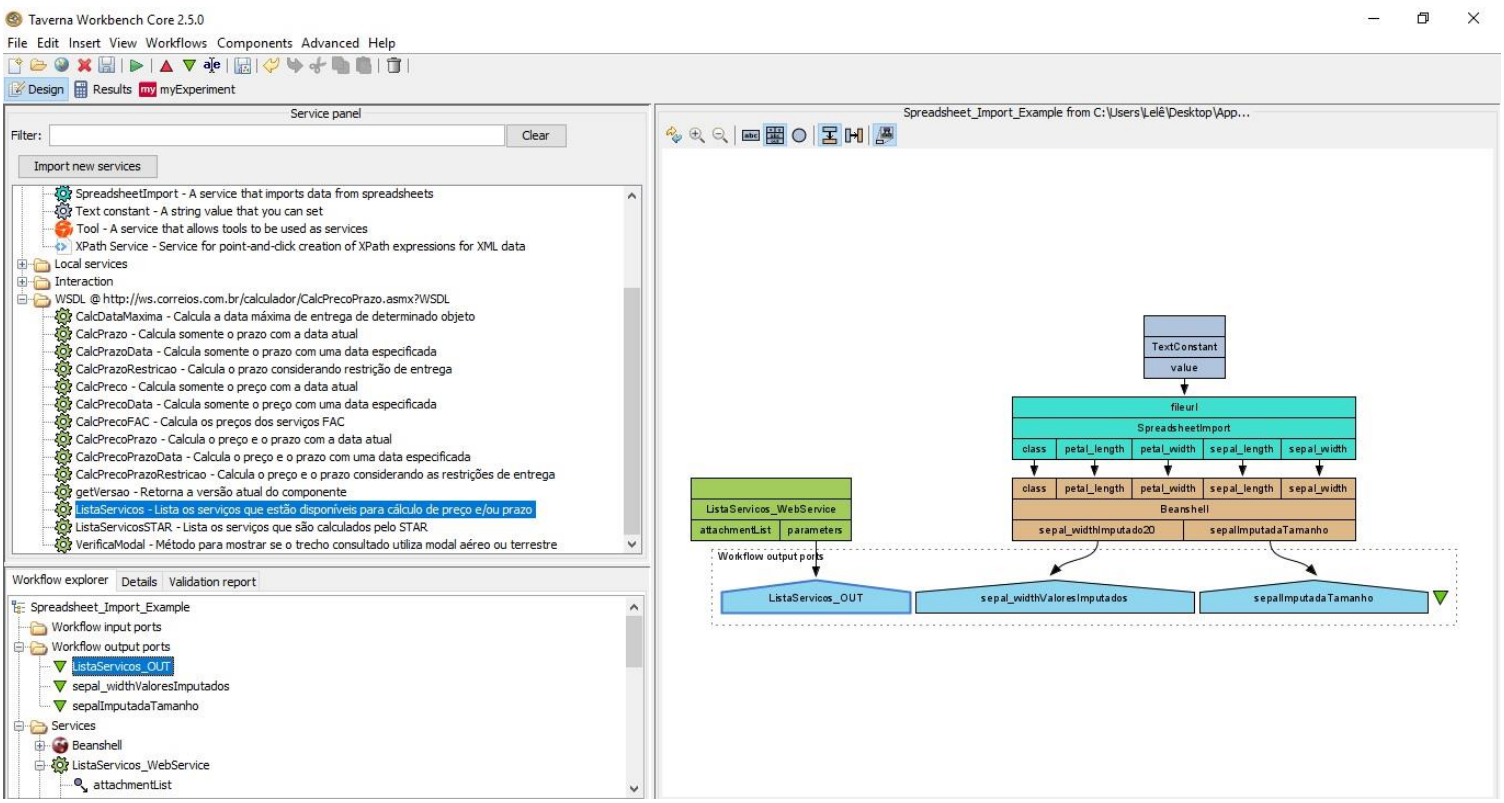


Figura 8: Diagrama do Ciclo 3, Execução 2

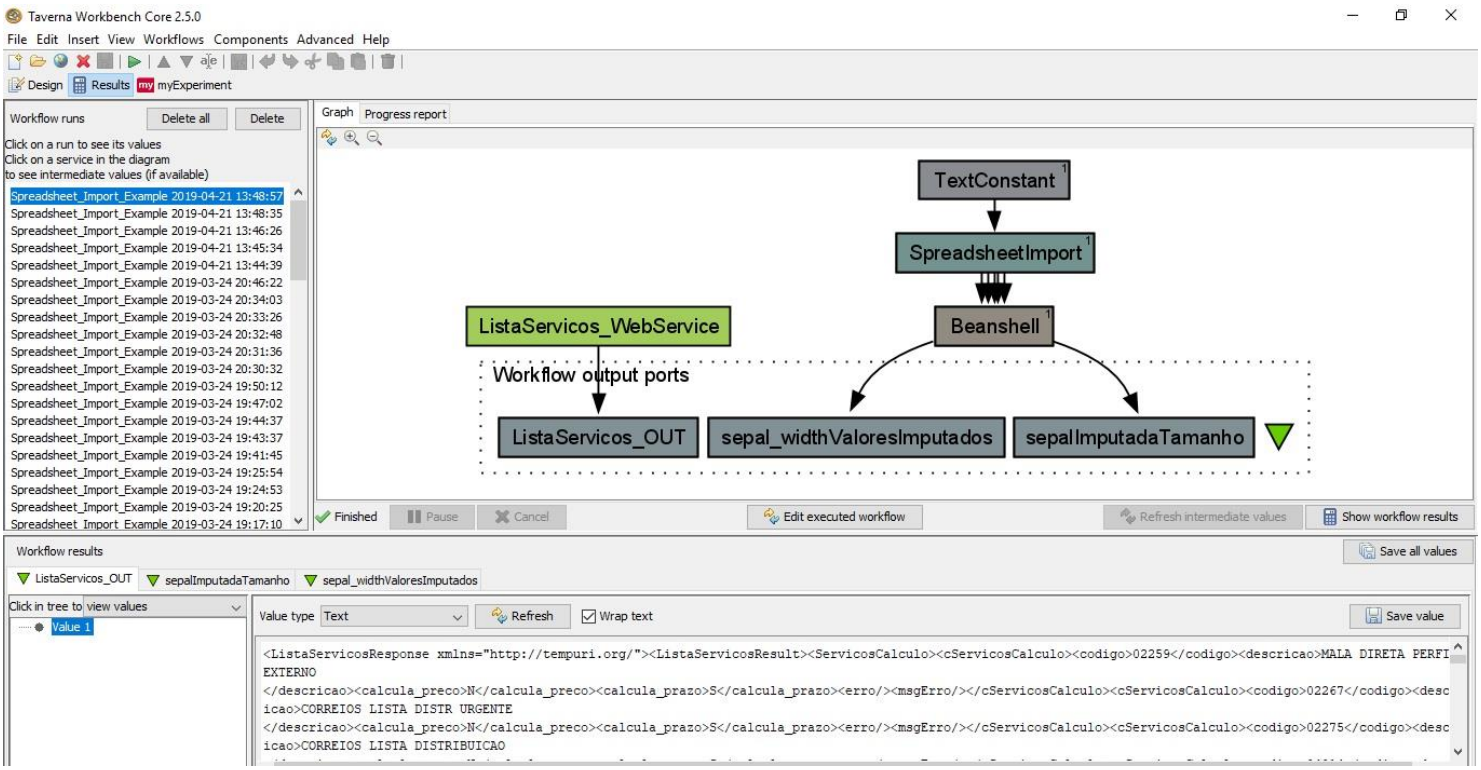


Figura 9: Ciclo 3, Execução 2. Exibição da saída do serviço dos Correios, importado a partir de um wsdl.

Ciclo 4: Consumo da API Rest, através de servlets e protocolo http

Após os ciclos anteriores, foi decido por analisar a possibilidade de utilizar o serviço de API Rest do Taverna. A API Rest é um serviço diferente da API Consumer e uma de suas vantagens é ser um serviço primitivo do Taverna.

As próximas execuções irão avaliar a comunicação da API Rest com um servlet. O objetivo é fazer o servlet se comunicar internamente com o Appraisal e consumi-lo. O servlet criado foi associado ao Tomcat 7.

Execução 1: Através da API Rest, fazer a comunicação do Taverna com um servlet. A comunicação será através do método post.

Execução 2: Através da API Rest, fazer a comunicação do Taverna com o servlet através do método get. O método get se comunicará com o projeto do Appraisal.

Considerações gerais – execução 1: O método post foi escolhido pois não há passagem de parâmetros e o primeiro objetivo era avaliar a comunicação entre os serviços. Foi configurado no serviço Rest qual método ele iria consumir. Nesse caso, o método 'POST', como é possível observar na figura 10. No servlet, o método *doPost* foi configurado para retornar no navegador a mensagem "POST teste". Na figura 11 é possível visualizar essa mensagem na porta de saída.

Considerações Gerais – execução 2: O Taverna passará informações para o servlet através do método doGet. O objetivo era poder modificar esses parâmetros de forma fácil e intuitiva. Foram criados cinco serviços de texto (*Text constant*) para inserir e modificar esses parâmetros. Foram escolhidos 5 parâmetros pois eles irão corresponder a 5 variáveis utilizadas no Appraisal para realizar uma imputação por média e essa foi a primeira comunicação feita entre o Taverna, o Servlet e o Appraisal.

Como enviar 5 parâmetros pelo método get acaba criando uma String grande, foi adicionado ao workflow um serviço *Beanshell* para formatar a url de forma mais legível, organizada e simples de modificar, como é possível visualizar na figura 12.

Para realizar a comunicação entre o projeto que contém o servlet e o projeto do Appraisal, foi preciso adicionar ao projeto do servlet, o projeto do Appraisal pelo Java Build Path. Para consumir o Appraisal através do método http, foi preciso gerar um javadoc do Appraisal e adicioná-lo a pasta lib do TomCat. Na figura 12, também é possível ver 2 portas de saída de workflow, onde a da esquerda "actualUrl" corresponde a url enviada para o servlet e a saída da direita "saída_doGetREST" corresponde ao retorno recebido pelo servlet.

Os parâmetros passados para o servlet foram passados para a classe responsável por manipular o Appraisal, através de seu construtor, e depois atribuídos a atributos do Appraisal. Nesse momento, não foi realizada nenhuma imputação, somente averiguado como seria feita a imputação e se a comunicação tinha sido bem-sucedida e os erros que surgiriam.

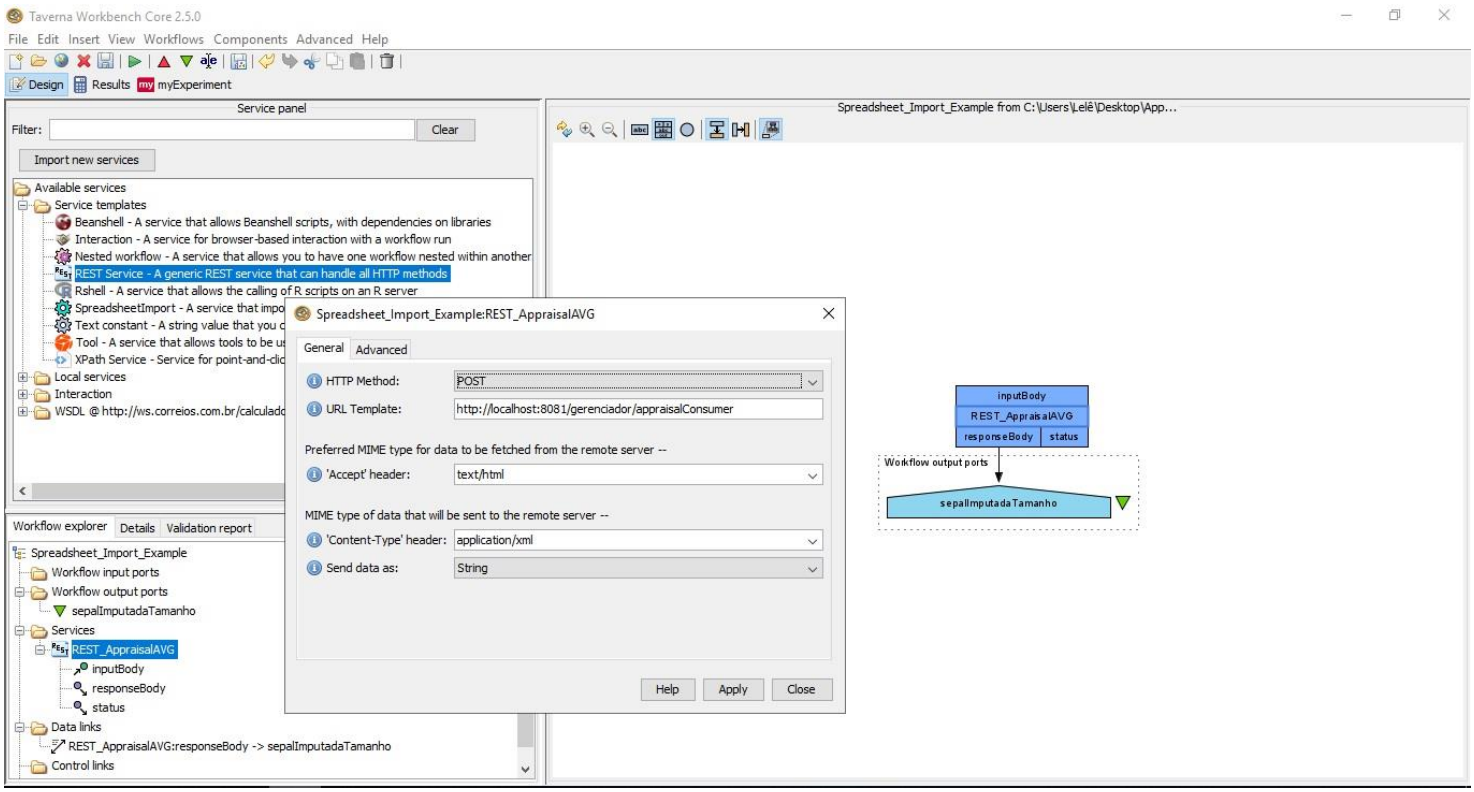


Figura 10: Ciclo 4, Execução 1 com interface de configuração do serviço REST

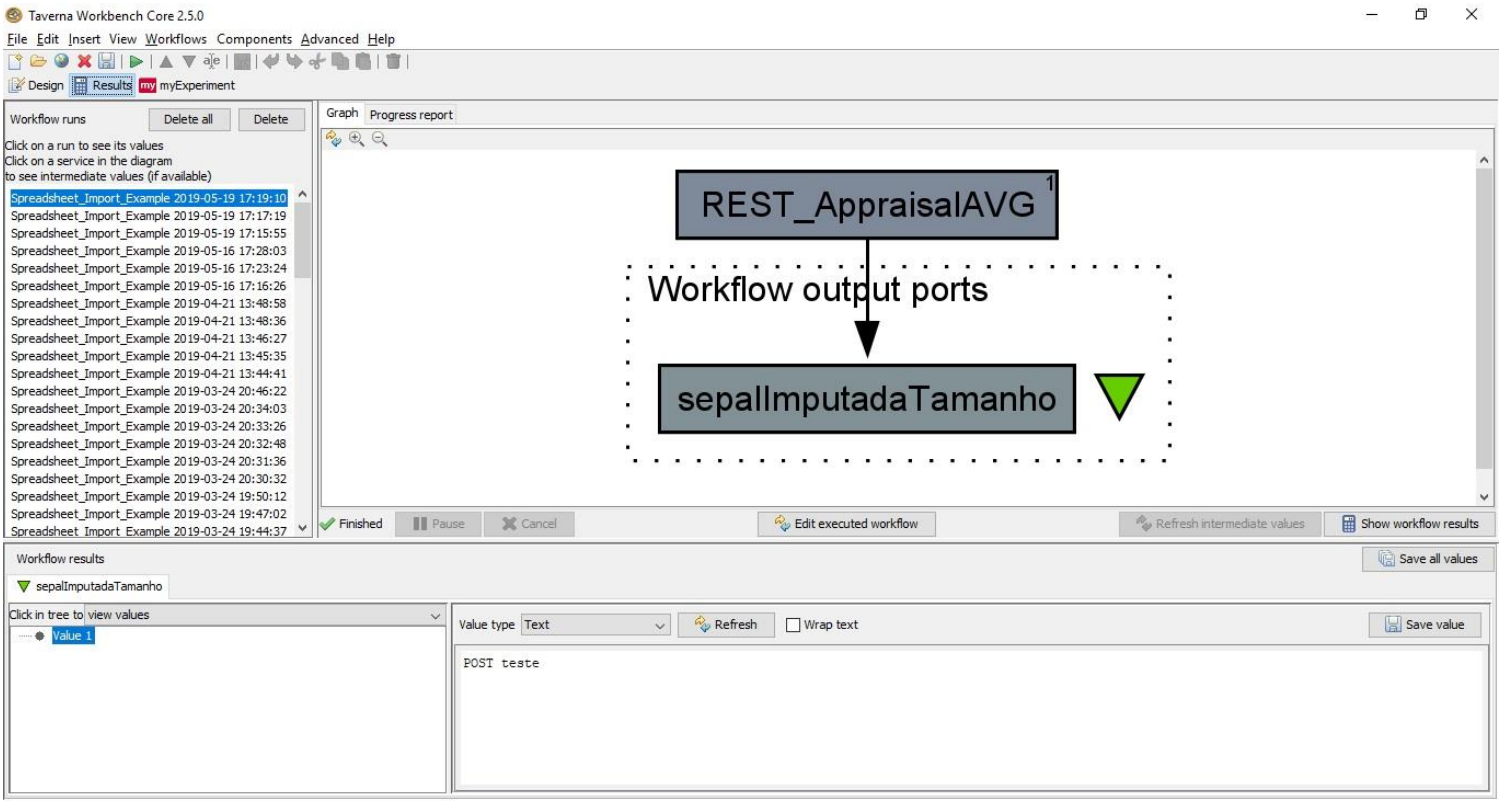


Figura 11: Resultado Ciclo 4, Execução 1 com o retorno do servlet

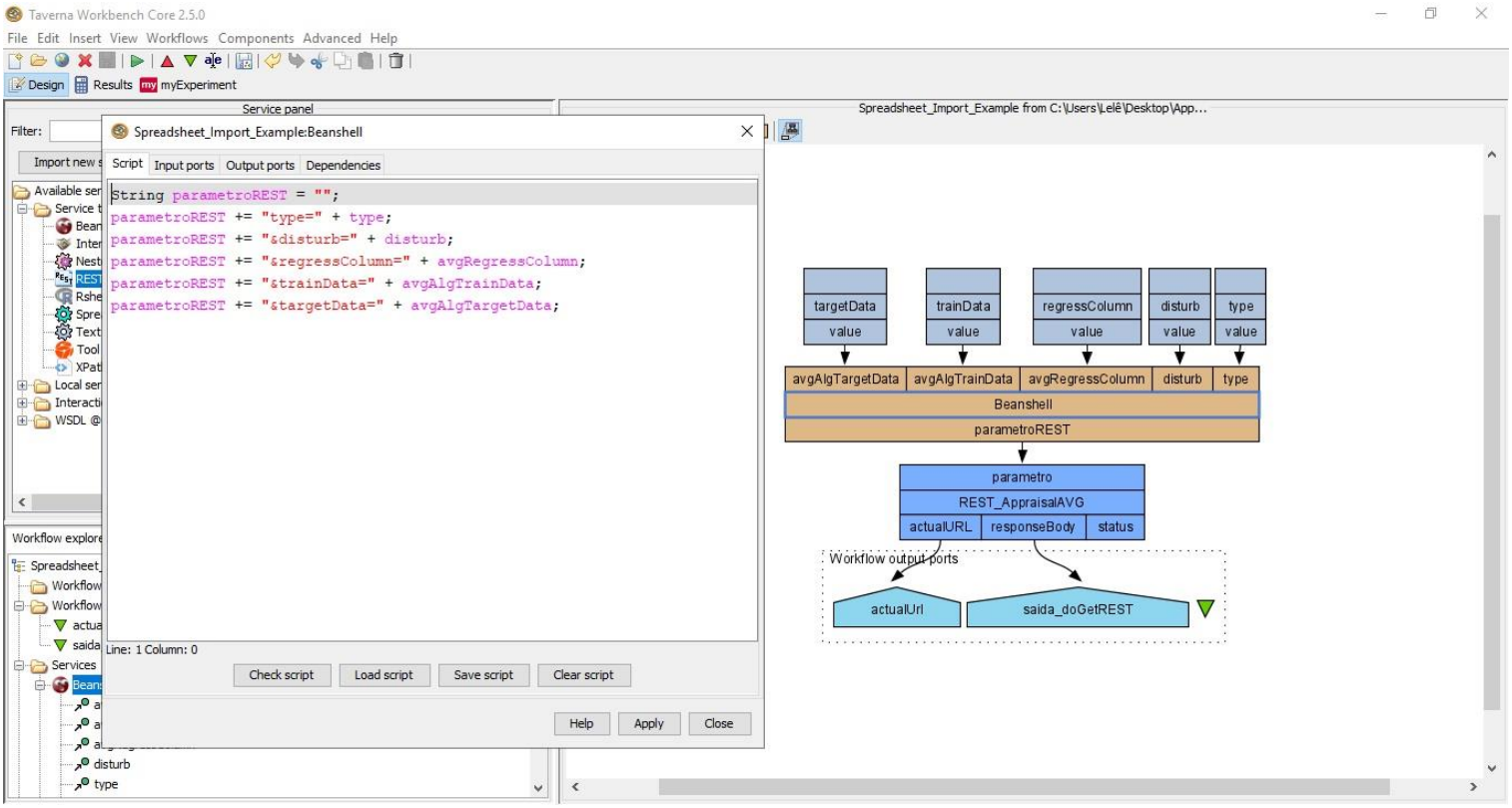


Figura 12: Ciclo 4, Execução 2 com interface do Beanshell montando a url a ser passada.

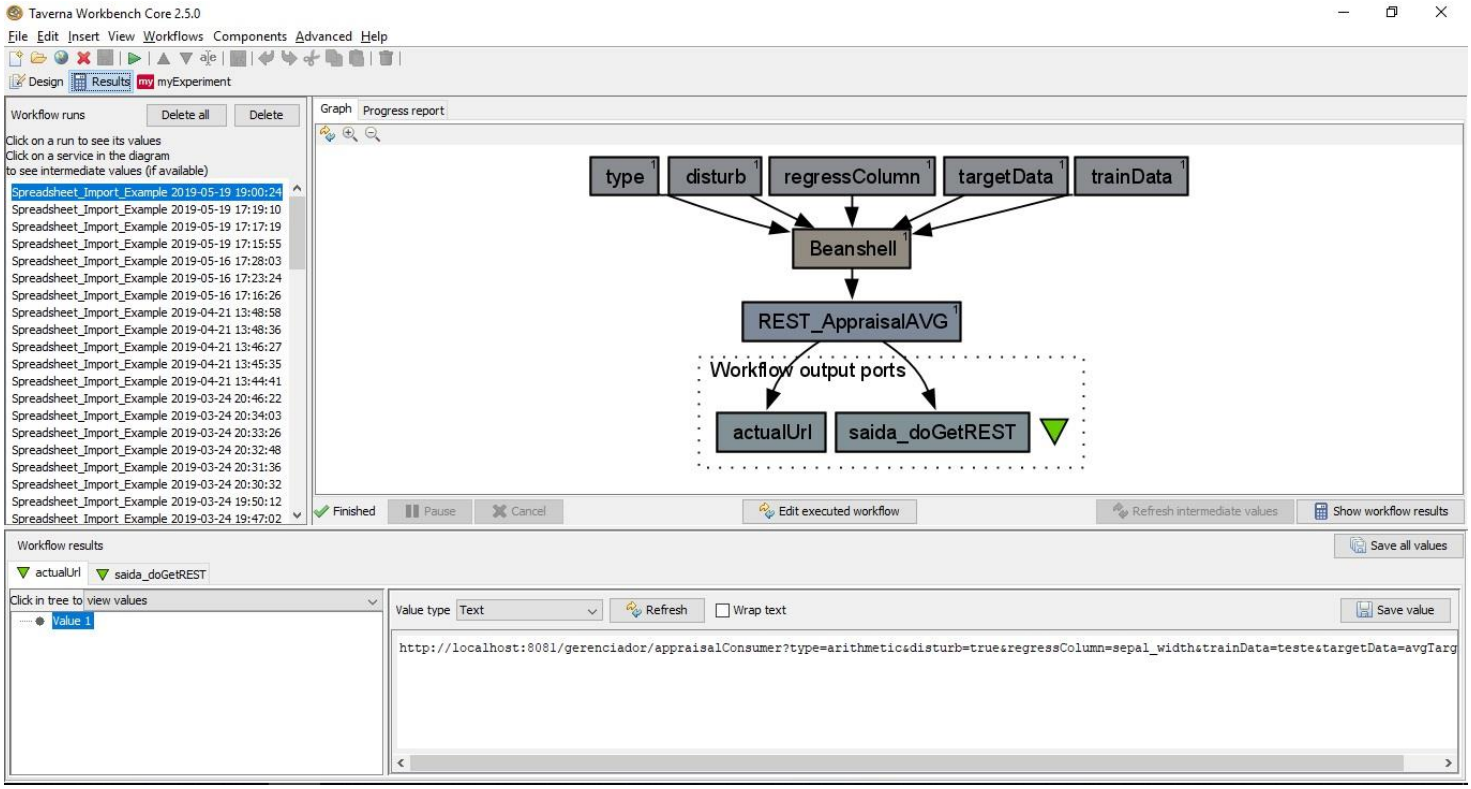


Figura 13: Resultado Ciclo 4, Execução 2. A porta 'actualUrl' mostrando a url enviada ao servlet

Ciclo 5: Realizar imputação através da API Rest e método get do servlet, criados no ciclo 4.

Execução 1: Remover as configurações para que o Appraisal funcionasse a partir do servlet e não mais de arquivos externos.

Execução 2: Realizar uma estratégia de imputação escolhida através do Taverna e com todos os parâmetros necessários recebidos pelo servlet.

Considerações gerais – execução 1: Essa execução foi estabelecida mais como uma pré execução da execução 2. Pois, para ser possível trabalhar com as informações vindas do workflow, os métodos do Appraisal que utilizavam as informações lidas do arquivo fonte de propriedades precisaram ser replicados e adaptados para receber essas informações via parâmetro. Então, o objetivo dessa execução era identificar e replicar esses métodos que utilizavam as configurações vindas do arquivo externo.

Após a identificação desses métodos, eles foram replicados e configurados de acordo com o mencionado acima. A esses métodos foi adicionada uma impressão no console para retornar os parâmetros recebidos, com o objetivo de averiguar se estavam chegando corretamente. Em alguns casos, foram criados construtores para receber as informações e guardá-las em atributo, em vez de passa-las por parâmetro.

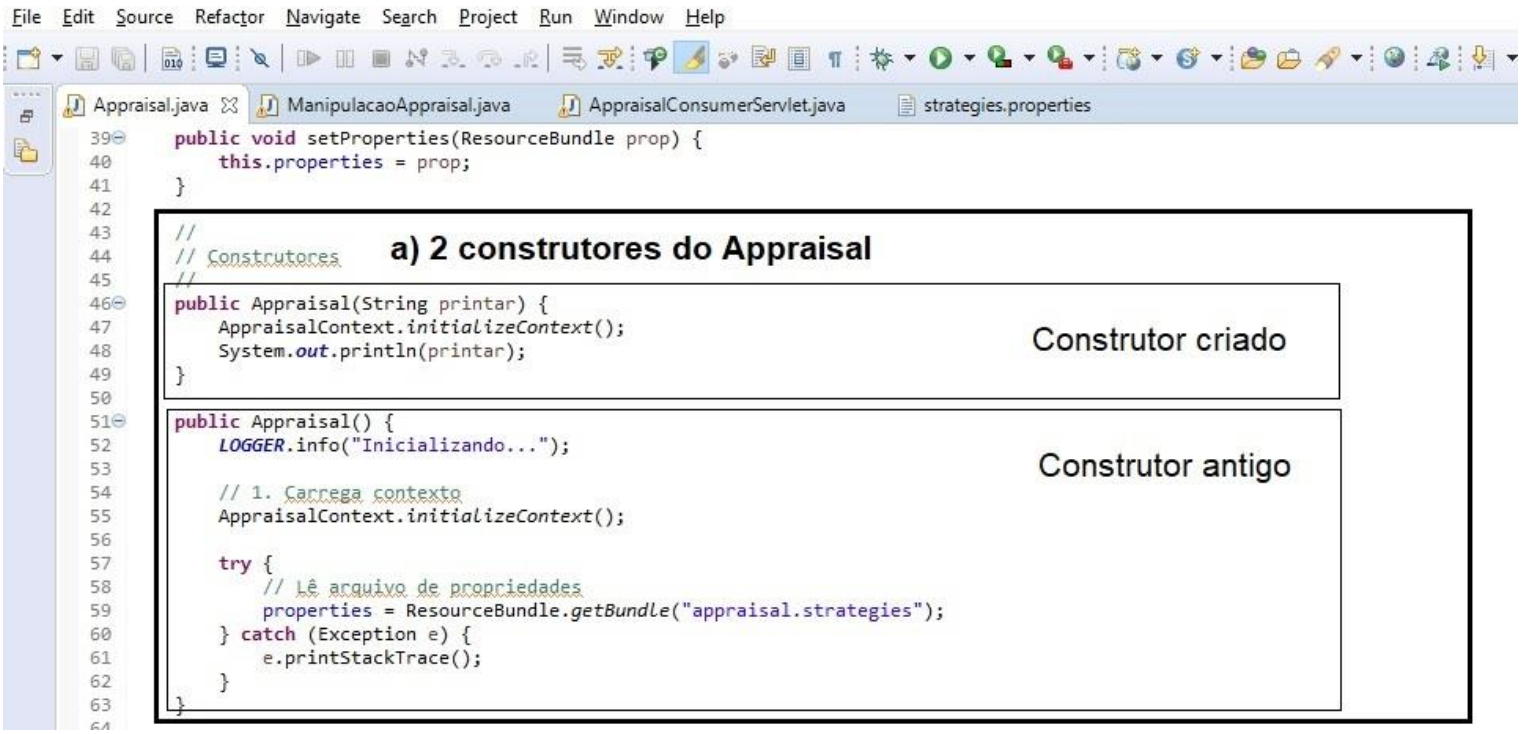
É possível visualizar na figura 14: a) 2 construtores do Appraisal:

i. O antigo sem parâmetros que consome o arquivo externo, o fonte de propriedades
ii. O criado que recebe uma String como parâmetro e a imprime no console.

Considerações gerais – execução 2: Nessa execução, os construtores e métodos criados na execução 1 foram configurados para realizar a imputação de fato. A estratégia de imputação escolhida para realizar o teste final, foi a *regression,* - imputação simples - aplicando os algoritmos de média (*avg*), *backpropagation* e *knn*.

A criação das estratégia era feita dentro do método “*test*” da classe Appraisal a partir da configuração lida no arquivo fonte de propriedades. Na nova implementação, a criação das estratégias é feita no método de manipulação do Appraisal e passado como parâmetro para o método “*testeWorkflow*” criado. Nos dois métodos, as estratégias são criadas e adicionadas a um ArrayList. É possível ver essa modificação de forma mais clara nas figuras 15 e 16. Na figura 15, o método antigo sem parâmetros que realiza a imputação através das configurações lidas do arquivo fonte. Na figura 16, o método criado que recebe as configurações para realizar as imputações via parâmetro.

Para escolher a(s) estratégia(s) através do Taverna é preciso modificar do serviço *Text constant*, nomeado como 'estrategias', no workflow. Caso deseje-se mais de uma estratégia, é preciso separá-las por ponto e vírgula. Ex: regression; selection.regression. Para simplificar a escolha das estratégias, foi colocado um serviço *Text constant* no workflow listando todas as estratégias. Porém, somente a 'regression' e a 'selection.regression' foram configuradas para serem executadas através do servlet e somente a 'regression' testada. Contudo, a partir de como o projeto foi arquitetado e a partir do teste feito, é possível adaptar o projeto para trabalhar com outras estratégias sem modificar a execução, somente adaptar o método de criação de estratégias para trabalhar com as outras. Após esse ciclo, a imputação conseguiu ser realizada com sucesso.




```java
public void setProperties(ResourceBundle prop) {
    this.properties = prop;
}

//
// Construtores
//
public Appraisal(String printar) {
    AppraisalContext.initializeContext();
    System.out.println(printar);
}

public Appraisal() {
    LOGGER.info("Inicializando...");

    // 1. Carrega contexto
    AppraisalContext.initializeContext();

    try {
        // Lê arquivo de propriedades
        properties = ResourceBundle.getBundle("appraisal.strategies");
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Figura 14: Construtores do Appraisal. Ciclo 5, Execução 1.

Appraisal.java | ManipulacaoAppraisal.java | AppraisalConsumerServlet.java | strategies.properties

```java

    public void test() {
        // Infra-Estrutura
        DatabaseHandler databaseHandler = MySQLHandler.getInstance();

        //
        // 1. Recupera parâmetros principais
        //
        String originalDatabase = getStringProperty("original.database");
        String regressionDatabase = getStringProperty("regression.database");
        String regressionColumn = getStringProperty("regression.column");

        LOGGER.info("Base Original: " + originalDatabase);
        LOGGER.info("Base de Regressão: " + regressionDatabase);
        LOGGER.info("Coluna de Regressão: " + regressionColumn);

        Dataset originalDataset;
        Dataset regressionDataset;

        //
        // 2. Recupera a base original
        //
        try {
            databaseHandler.selectDatabase(originalDatabase);
            originalDataset = databaseHandler.toDataset();
        } catch (DatabaseException e) {
            e.printStackTrace();
            return;
        }

        //
        // 3. Recupera a base de regressão
        //
        try {
            databaseHandler.selectDatabase(regressionDatabase);
            regressionDataset = databaseHandler.toDataset();
        } catch (DatabaseException e) {
            e.printStackTrace();
            return;
        }

        //
        // 4. Monta as estratégias a serem executadas
        //
        List<Strategy> strategies = new ArrayList<Strategy>();

        if (getBooleanProperty("regression.enabled")) {
            strategies.add(new RegressionStrategy());
        }
        if (getBooleanProperty("selection.regression.enabled")) {
            strategies.add(new SelectionStrategy());
        }
        if (getBooleanProperty("clustering.regression.enabled")) {
            strategies.add(new ClusteringStrategy(getBooleanProperty("clustering.regression.pso"),
                    getBooleanProperty("clustering.regression.kmeans"), getBooleanProperty("clustering.regression.knn"),
                    getBooleanProperty("clustering.regression.bkprop"),
                    getBooleanProperty("clustering.regression.avg")));
        }
        if (getBooleanProperty("selection.clustering.regression.enabled")) {
            strategies.add(new SelectionClusteringStrategy(getBooleanProperty("selection.clustering.regression.pso"),
                    getBooleanProperty("selection.clustering.regression.kmeans"),
                    getBooleanProperty("selection.clustering.regression.knn"),
                    getBooleanProperty("selection.clustering.regression.bkprop"),
                    getBooleanProperty("selection.clustering.regression.avg")));
        }
        if (getBooleanProperty("clustering.selection.regression.enabled")) {
            strategies.add(new ClusteringSelectionStrategy(getBooleanProperty("clustering.selection.regression.pso"),
                    getBooleanProperty("clustering.selection.regression.kmeans"),
                    getBooleanProperty("clustering.selection.regression.knn"),
                    getBooleanProperty("clustering.selection.regression.bkprop"),
                    getBooleanProperty("clustering.selection.regression.avg")));
        }

        //
        // 5. Roda as estratégias
        //
        Dataset originalColumn = originalDataset.copyColumn(regressionColumn);

        for (Strategy strategy : strategies) {
            // Roda a estratégia atual
            strategy.runStrategy(originalColumn, regressionDataset, regressionColumn);
        }
    }
```

Figura 15: : Método do Appraisal que realiza imputação através da leitura de arquivo externo

Appraisal.java | ManipulacaoAppraisal.java | AppraisalConsumerServlet.java | strategies.properties

```
64
65      public void testeWorkflow(String originalDB, String regressionDB, String regressColumn,
66                                String resultPath, List<Strategy> strategies) {
67
68
69          DatabaseHandler databaseHandler = MySQLHandler.getInstance();
70          String originalDatabase = originalDB;
71          String regressionDatabase = regressionDB;
72          String regressionColumn = regressColumn;
73
74          Dataset originalDataset;
75          Dataset regressionDataset;
76
77          // 2. Recupera a base original
78          //
79          try {
80              databaseHandler.selectDatabase(originalDatabase);
81              originalDataset = databaseHandler.toDataset();
82          } catch (DatabaseException e) {
83              e.printStackTrace();
84              return;
85          }
86          //
87          // 3. Recupera a base de regressão
88          //
89          try {
90              databaseHandler.selectDatabase(regressionDatabase);
91              regressionDataset = databaseHandler.toDataset();
92          } catch (DatabaseException e) {
93              e.printStackTrace();
94              return ;
95          }
96
97          // 4. Monta Estrategia
98          //as estratégias são montadas na classe ManipulacaoAppraisal e passadas por parametro
99
100         // 5. Roda as estratégias
101         //
102         Dataset originalColumn = originalDataset.copyColumn(regressionColumn);
103
104         new ResultWriter(resultPath,regressionDB);
105
106         for (Strategy strategy : strategies) {
107             // Roda a estratégia atual
108             strategy.runStrategy(originalColumn, regressionDataset, regressionColumn);
109         }
110     }
```

Figura 16: Método do Appraisal que realiza imputação através da passagem de parâmetros. Adicionado no Ciclo 5, Execução 2

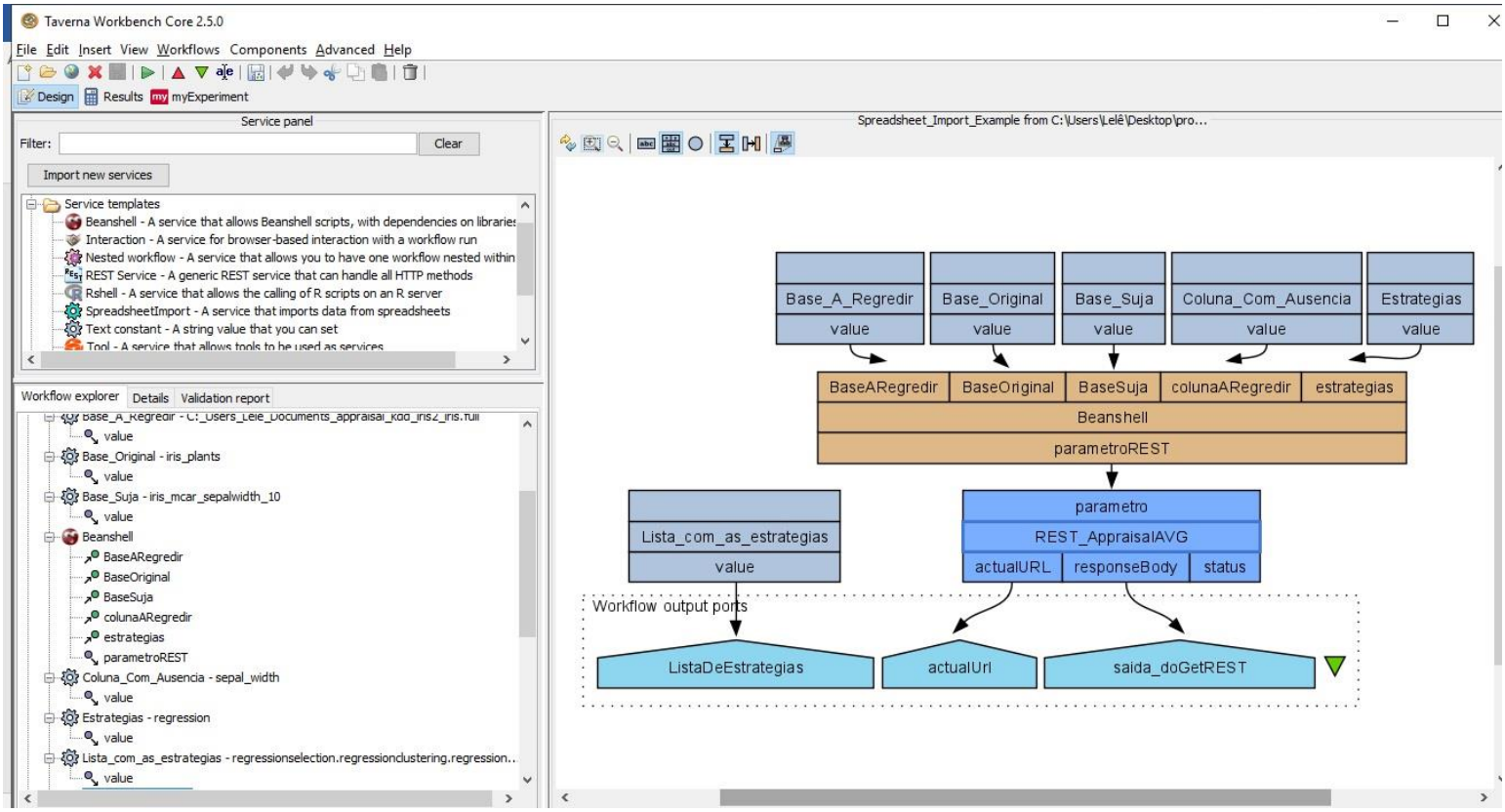


Figura 17: Versão final do wokflow

5 RELATO DE EXPERIÊNCIA

5.1 PONTOS FORTES E FRACOS DA FERRAMENTA TAVERNA PARA REALIZAR A INTEGRAÇÃO

Antes da execução dos ciclos de teste conhecia-se poucas características sobre o Taverna que fossem diretamente relevantes para essas execuções. Dentre essas características, estão os fatos de o software Taverna ser da Apache e possuir uma documentação clara e bem feita. E esses, por sua vez, foram os pontos positivos mais relevantes para dar início aos ciclos de teste com o Taverna.

O fato do Taverna ser da Apache, acaba criando uma expectativa de que o software é um software de qualidade – o que realmente se mostrou verdade – e que há quantidade, ao menos suficiente, de informação disponível sobre ele. Isso se mostrou também verdade e possibilitou um estudo melhor do Taverna, já que foram achados projetos prontos na internet com alguns serviços básicos já implementados e que mostravam como o Taverna poderia trabalhar. A interface intuitiva e seu

Um dos principais pontos fortes da ferramenta é a quantidade e qualidade dos serviços primitivos, como o *Beanshell* e o *SpreadsheetImport.* Isso possibilita ao software ser flexível na criação dos workflows e na escolha dos serviços que serão utilizados. O serviço *Beanshell* é outro ponto forte do software, pois embora a linguagem utilizada seja semelhante ao Java, é um pouco mais simples, o que agiliza o desenvolvimento, por exemplo, não precisar declarar alguns tipos de variáveis, como o arrayList.

Nesse quesito, um dos pontos negativos, é que tanto os serviços, como as portas de entrada e saída de workflow adicionados à área do diagrama de workflow necessariamente precisam estar em um contexto. Ou seja, estar conectada ao workflow. Caso contrário o software gera um erro na execução. Essa característica impossibilita criar serviços ou entradas e saídas e deixá-los armazenados para usar posteriormente. Fazendo uma analogia, é como se não fosse possível comentar códigos.

Outro ponto negativo no que tange os serviços, é que os botões de desfazer e refazer não funcionam na parte da escrita do script do Beanshell. Para assegurar recuperação de algum código, é necessário criar um backup no bloco de notas a

cada modificação, por exemplo. As funções de refazer e desfazer funcionam bem na área de criação do workflow, associando e desassociando serviços.

Em um dos ciclos de teste, foi tentado utilizar a classe Appraisal direto pelo Taverna, a partir do *Beanshell.* Esse relato não se encontra na seção de desenvolvimento, pois não agregou valor ao ciclo, somente foi constatado que a utilização de utilização de bibliotecas externas conta-se um ponto negativo pois a importação do jar é feita de forma manual. É preciso baixar o arquivo .jar e jogá-lo para a pasta correspondente ao *Beanshell.*

No ciclo 3, foi relatado que para utilizar o serviço ApiConsumer do Taverna é preciso instalar o plug-in 'API Consumer Service' antes. Esse plugin possibilita importar um serviço de consumo de APIs. Além desse plug-in, é preciso baixar um software externo para gerar um arquivo xml e é a partir desse arquivo que se importa e consome APIs pelo Taverna. Sendo o consumo de API um tarefa comum em desenvolvimento de software e no Taverna esse consumo não ser trivial., essa característica conta como um ponto negativo da ferramenta.

5.2 DIFICULDADES ENCONTRADAS E SOLUÇÕES DADAS

Parte das dificuldades encontradas está ligada a quesitos técnicos, como versões de .jar. Os erros que mais tomaram tempo para serem corrigidos, foram erros de ambiente. A versão do TomCat, por exemplo, precisava ser superior a 8. Foi utilizada a 8.5. A versão do mysql-connector precisou ser atualizada, pois a utilizada no projeto do Appraisal estava defasada. A versão do mysql-connector precisava ser a partir da versão 8.0.1. Foi utilizada foi a 8.0.11.

Mesmo adicionando a refêrencia do projeto Appraisal ao projeto criado, as modificações feitas no projeto do Appraisal e, não eram refletidas somente buildando os projetos novamente. Foi preciso 1) limpar e buildar o projeto do Appraisal e o projeto criado 2) gerar um novo .jar do Appraisal 3) adicioná-lo a pasta lib do TomCat 4) buildar o projeto criado e 5) reiniar o TomCat. Essa sequência de passos precisou ser seguida a cada mudança no código do Appraisal. Foram gerados ao todo 21 .jars diferentes.

Identificar os erros e as informações que chegavam nos métodos do Appraisal também foi uma tarefa trabalhosa, visto que como o projeto era consumido, foi preciso criar um atributo chamado ‘saida’ na classe Appraisal para ir incrementando seu valor com todas as informações necessárias para, ao final da execução do workflow, as informações serem impressas no console do Eclipse ou no retorno do método doGet do servlet.

Foi possível observar a partir dos testes que o Appraisal não possuir tags de documentação tornou algumas das possíveis soluções não factíveis. Essa foi uma das principais barreiras para encontrar a melhor solução para consumo do Appraisal a partir da interface. Visto que isso contribuiu para aumentar o tempo de teste das possíveis soluções e diminuiu as possibilidades de soluções, a falta de documentação do código foi uma das principais dificuldade encontradas que precisou ser contornada, pois não foi possível solucioná-la.

5.3 CONSIDERAÇÕES TÉCNICAS

Para ser possível consumir a Appraisal a partir do Taverna, é preciso 1) Baixar o software eclipse. 2) Importar o projeto do Appraisal e do servlet disponíveis no Github. 3) Instalar e configurar o TomCat a partir da versão 8. 4) Baixar o Taverna Workbench 5) Baixar e instalar o MySql Workbench a partir da versão 8.

6 CONCLUSÃO

O ponto de partida desse trabalho foi o objetivo de substituir a forma de execução e configuração do software Appraisal, pois a manipulação desse software era feita através de arquivos externos e linhas de comando (CLI), o que exigia conhecimento e esforço técnico adicionais. O presente trabalho teve como foco o processo de migração do Appraisal de CLI para uma interface de sistema de workflow. A partir dessa ideia, foi construído um workflow através do sistema de gerenciamento workflow Taverna Workbench.

Junto ao workflow, foi desenvolvido um projeto Java Web que possibilitou a comunicação entre o software Tavena e o Appraisal. Essa comunicação se deu através de um serviço REST que foi configurado no Taverna para consumir um servlet. Esse servlet criado a partir do projeto Java Web se comunica com uma classe criada para consumir os métodos do Appraisal.

Para chegar ao resultado desse trabalho, foram definidos e executados ciclos de testes para guiarem e embasarem esse resultado. Esse trabalho contém, além do resultado, os relatos desses ciclos de teste. Como não se conhecia o software Taverna a fundo, nem como se seguiriam os testes, os critérios utilizados para avaliar a melhor maneira de criar o workflow foram o tempo médio gasto em cada execução e a complexidade técnica. Esses critérios visaram atender a dois casos: 1) cientistas que pretendiam somente utilizar o Appraisal imputando e verificando a qualidade dos dados e; 2) os cientistas e desenvolvedores que pretendiam melhorar e criar projetos a partir do Appraisal.

No primeiro caso, a ferramenta Taverna se mostrou útil e cumpriu objetivo de forma satisfatória, pois sua interface é intuitiva e não é necessário conhecimento técnico para operá-la.

No segundo caso, o workflow e o servlet possibilitarão aos desenvolvedores gastarem menos tempo na execução e desenvolvimento de novas funcionalidades. A criação do servlet foi essencial para que se pudesse aproveitar as imputações já feitas pelo Appraisal, por exemplo, a imputação simples. Nesse caso, realiza-se essa imputação através da chamada ao doGet do servlet, ou seja, sem precisar reescrever ou replicar qualquer código relacionado à imputação. Uma das tarefas que tornou isso possível foi o ‘desmembramento’ do Appraisal, pois sua arquitetura era amarrada, sem respeitar muitos conceitos de O.O, sendo um exemplo a

montagem das estratégias, que era feita no mesmo método que chama a imputação. Após esse trabalho, essa montagem de estratégias é feita em um método específico para isso, facilitando assim a manutenção, utilização e a criação de novas estratégias.

Como possíveis trabalhos futuros, desmembrar o Appraisal aplicando conceitos de O.O e criar tags de documentação possibilitariam uma aplicação mais ampla do software e um melhor reaproveitamento de seu código.

7 REFERÊNCIAS

BARKER, A.; VAN HEMERT, J. Scientific workflow: A survey and research directions. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 4967 LNCS, p. 746–753, 2008.

BRAGHETTO, K. R.; RESUMO, D. C. Capítulo 1 Introdução à Modelagem e Execução de Workflows Científicos. **Atualizações em Informática. 1ed. Porto Alegre: SBC**, p. 1–40, 2014.

BROWN, J. L. et al. **GridNexus : A Grid Services Scientific Workflow System**. 2005

CASTANEDA, R.; FERLIN, C.; GOLDSCHMIDT, R. Aprimorando Processos de Imputação Multivariada de Dados com Workflows. p. 238–252, 2007.

DEELMAN, E. et al. Workflows and e-Science: An Overview of Workflow System Features and Capabilities. **Future Gener. Comput. Syst.**, v. 25, n. 5, p. 528–540, 2009.

DEELMAN, E. et al. Pegasus, a Workflow Management System for Science Automation. **Future Gener. Comput. Syst.**, v. 46, n. C, p. 17–35, 2015.

ENGELS, J. M.; DIEHR, P. Imputation of missing longitudinal data: a comparison of methods. **Journal of Clinical Epidemiology**, v. 56, n. 10, p. 968–976, 2003.

GUIDE, U. **DRAFT Kepler : An Extensible System for Design and Execution of Scientific Workflows User Guide**. 2004

LIU, G. et al. **Automated realization of business workflow specification**. Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops. **Anais**...2010

MARCIO JOSÉ DE ALMEIDA, T. M. R. COMUNICADO ESPP-CFRH 06/2016. p. 1–4, 2016.

MEDINA, J. V. DE. ALGORITMOS GENÉTICOS E REDES DE KOHONEN NA COMPLEMENTAÇÃO DE DADOS AUSENTES. p. 142, 2012.

MONTEIRO, D. S. Imputação Categórica em Base de Dados. 2008.

OUYANG, C. et al. Workflow management. **Handbook on Business Process Management 1: Introduction, Methods, and Information Systems**, p. 475–506, 2015.

RIBEIRO, L. D. E. S. UTILIZANDO PROVENIÊNCIA PARA A COMPLEMENTAÇÃO DE DADOS NO CONTEXTO DO PROCESSO DE ETL. p. 106, 2010.

RUSINKIEWICZ, M.; SHETH, A. Specification and execution of transactional workflows. **Modern Database Systems**, n. 3652008, p. 592–620, 1995.

SCHAFER, J. L.; OLSEN, M. K. Multiple Imputation for Multivariate Missing-Data Problems: A Data Analyst's Perspective. **Multivariate Behavioral Research**, v. 33, n. 4, p. 545–571, 1998.

SHIELDS, M.; TAYLOR, I. **Programming scientific and distributed workflow with Triana services**. Proceedings of Workflow in Grid Systems Workshop in GGF10. **Anais**...2004

SOARES, J. DE A. Pré-Processamento Em Mineração De Dados: Um Estudo Comparativo Em Complementação. p. 245, 2007.

SOMMERVILLE, I. **Software Engineering**. 5th. ed. [s.l.] Addison-Wesley, 1995.

TAYLOR, I. J. et al. **Workflows for e-Science: scientific workflows for grids**. [s.l.] Springer Publishing Company, Incorporated, 2014.

WEIN, J. D. et al. WORKFLOW SYSTEM AND METHOD. v. 2, n. 12, p. 38, 2008.

WOLSTENCROFT, K. et al. The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. **Nucleic acids research**, v. 41, n. W1, p. W557--W561, 2013.