

UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ALEXANDRE HUFF

ClusterFlow – Um ambiente de apoio à realização de experimentos científicos
para um *cluster* de computadores de alto desempenho

Maringá
2010

Livros Grátis

<http://www.livrosgratis.com.br>

Milhares de livros grátis para download.

ALEXANDRE HUFF

ClusterFlow – Um ambiente de apoio à realização de experimentos científicos para um *cluster* de computadores de alto desempenho

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação

Orientadora: Profa. Dra. Itana Maria de Souza Gimenes

Co-orientador: Prof. Dr. Ronaldo Augusto de Lara Gonçalves

Maringá
2010

Dados Internacionais de Catalogação-na-Publicação (CIP)
(Biblioteca Central - UEM, Maringá – PR., Brasil)

H889c Huff, Alexandre
ClusterFlow - um ambiente de apoio à realização de experimentos científicos para um cluster de computadores de alto desempenho. / Alexandre Huff. -- Maringá, 2010.
146 f. : il. color., figs., list.

Orientadora : Prof.^a Dr.^a Itana Maria de Souza Gimenes.
Co-orientador : Prof. Dr. Ronaldo Augusto de Lara Gonçalves.

Dissertação (mestrado) - Universidade Estadual de Maringá, Centro de Tecnologia, Departamento de Informática, Programa de Pós-Graduação em Ciência da Computação, 2010.

1. Experimento científico. 2. Workflow científico. 3. Serviços Web. 4. Cluster de computadores. 5. WS-BPEL. 6. Linguagem WS-BPEL. 7. ClusterFlow. 8. Atividades humanas - Gerenciamento - Experimento científico - WS-BPEL. 9. Atividades humanas - Gerenciamento - Workflow - WS-BPEL. 10. Sistema gerenciador - Workflow. I. Gimenes, Itana Maria de Souza, orient. II. Gonçalves, Ronaldo Augusto de Lara, co-orient. III. Universidade Estadual de Maringá. Centro de Tecnologia. Departamento de Informática. Programa de Pós-Graduação em Ciência da Computação. IV. Título.

CDD 21.ed. 003.35133

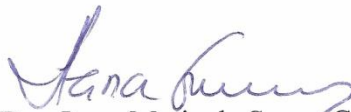
FOLHA DE APROVAÇÃO

ALEXANDRE HUFF

ClusterFlow – Um ambiente de apoio à realização de experimentos científicos para um *cluster* de computadores de alto desempenho

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação pela Banca Examinadora composta pelos membros:

BANCA EXAMINADORA



Profa. Dra. Itana Maria de Souza Gimenes
Universidade Estadual de Maringá – DIN/UEM



Profa. Dra. Luciana Andréia Fondazzi Martimiano
Universidade Estadual de Maringá – DIN/UEM



Profa. Dra. Ellen Francine Barbosa
Universidade de São Paulo – ICMC/USP

Aprovada em: 24 de setembro de 2010.

Local da defesa: Sala 101, Bloco C56, *campus* da Universidade Estadual de Maringá

DEDICATÓRIAS

Dedico este trabalho aos meus pais e avós por sempre estarem me incentivando a lutar pela conquista dos meus sonhos.

AGRADECIMENTOS

Agradeço primeiramente a Deus, pelo dom da vida, por me acompanhar em todos os momentos, me dar forças e auxiliar em minhas decisões.

Aos meus pais Airton e Noeli, meus avós Edmundo e Gertrudes, e demais familiares que sempre estão me apoiando em cada nova jornada de estudos e trabalho para a conquista de meus objetivos.

À minha orientadora Professora Doutora Itana Maria de Souza Gimenes, pela oportunidade, confiança, profissionalismo, dedicação e seus conhecimentos compartilhados.

Ao meu co-orientador Professor Doutor Ronaldo Augusto de Lara Gonçalves, pelos conhecimentos compartilhados.

A todos os professores que de forma direta ou indireta contribuíram com este trabalho.

Aos meus amigos do mestrado, André Barbosa Verona (BV), Rafael Cassolato de Meneses (Cassolato), Gabriel Costa Silva, Marcelo R. Borth (Borth), Carlos A. M. Basso (Carlitos), Alberto B. Biasão (Betão), Henrique Y. Shishido (Shido), Rodrigo T. Pagno (Gaucho), Mauro H. Mulati, Gustavo Sato, Camila Lapazini Leal (Camilinha), Everton F. Barros, Gécen D. De Marchi e demais amigos do mestrado que não pude citar aqui por ser uma lista extensa, pelas madrugadas e finais de semana de estudos, dicas, incentivos, cafés, mates, chás, churrascos e cantorias ao som do meu inseparável azulão (violão), cervejadas quando possível, enfim, momentos de estudos e distrações.

Aos amigos e colegas do ICMC-USP de São Carlos, principalmente àqueles que contribuíram para a realização deste trabalho e aos demais pelos momentos de distração. Aos amigos das repúblicas Mandando Bem e Roliudi S/A, por terem me acolhido durante a minha permanência em São Carlos.

À Maria Inês Davanço, por ser uma excelente profissional, pela paciência, ajuda e compreensão.

A todos os meus demais e incontáveis amigos, pelo incentivo e apoio.

Ao apoio financeiro do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e do Projeto PROCAD.

Enfim, a todas as pessoas que de alguma ou outra forma contribuíram para eu chegar até aqui, o meu mais sincero agradecimento.

EPÍGRAFE

“Quanto mais aumenta nosso conhecimento,
mais evidente fica nossa ignorância.”

(JOHN FITZGERALD KENNEDY)

ClusterFlow – Um ambiente de apoio à realização de experimentos científicos para um *cluster* de computadores de alto desempenho

RESUMO

A simulação de experimentos científicos apoiada por computador tem gerado uma quantidade de dados cada vez maior, o que demanda do uso de *clusters* de computadores que oferecem poder de processamento próximo ao dos *mainframes* com menor custo. No entanto, *clusters* devem ser compartilhados entre os cientistas, pois esses nem sempre estão presentes em laboratórios de pesquisa. Cientistas enfrentam algumas barreiras para o uso de *clusters*, como, falta de conhecimento para sua utilização e a distância geográfica entre os laboratórios e *clusters*. *Workflows* científicos e serviços web se apresentam como soluções viáveis para apoiar os cientistas no planejamento e execução de experimentos em *clusters*. As atividades de um *workflow* científico podem envolver serviços web e atividades humanas. Este trabalho de mestrado propõe o desenvolvimento de um ambiente de apoio à realização de experimentos científicos para um *cluster* de computadores, o *ClusterFlow*. O ambiente visa apoiar pesquisadores na definição, execução, compartilhamento e reuso de *workflows* científicos que são exportados para a linguagem WS-BPEL. As principais contribuições apresentadas neste trabalho são: (i) oferecer o projeto documentado de um ambiente de apoio à realização de experimentos científicos para um *cluster* de computadores baseado em serviços; (ii) propor um mecanismo que apóie o gerenciamento de atividades humanas em *workflows* científicos; e, (iii) oferecer um mecanismo de compartilhamento e reuso desses *workflows*. O projeto do ambiente é avaliado com base no desenvolvimento de um protótipo e um exemplo de aplicação.

Palavras-chave: Experimento Científico. *Workflow* Científico. Serviços Web. *Cluster* de Computadores. WS-BPEL.

ClusterFlow – An environment to support the realization of scientific experiments for a high performance cluster of computers

ABSTRACT

Computer-supported simulation of scientific experiments has been generating an increasing amount of data. This demands the use of computer clusters as they have high processing power whereas lower costs than mainframes. However, clusters need to be shared among scientists as they are not present in every research laboratory. Nevertheless, scientists still face barriers to use clusters such as lack of knowledge to use them and the geographic distance between research laboratories and clusters. Scientific workflows and web services has been studied as feasible solutions to support the planning and execution of experiments in clusters. The activities of scientific workflows may involve web services and human activities. This work presents an environment, named *ClusterFlow*, to support the definition and execution of experiments in a cluster of computers. *ClusterFlow* aims at supporting scientists in the definition, execution, sharing and reuse of scientific workflows exported to the WS-BPEL language. The main contributions of this work are: (i) to present a documented design of an environment to support scientific experiments in clusters based on web services; (ii) to propose a support mechanism for the management of human tasks in scientific workflows; and, (iii) to offer a mechanism for sharing and reuse scientific workflows. The design of *ClusterFlow* was based on a prototype development and an example of application.

Keywords: Scientific Experiment. Scientific Workflow. Web Services. Cluster of Computers. WS-BPEL.

LISTA DE FIGURAS

Figura 1. Ciclo de vida de um experimento científico	23
Figura 2. Principais elementos da arquitetura de serviços web	27
Figura 3. Cenário do ambiente ClusterFlow	47
Figura 4. Arquitetura do ambiente ClusterFlow	48
Figura 5. Subdivisão do Portal Web para Experimentos	49
Figura 6. Subdivisão do Gerenciador de Experimentos	50
Figura 7. Diagrama de atividades do processo de construção e execução de workflows científicos.....	52
Figura 8. Ação Criar Workflow.....	53
Figura 9. Ação Criar Atividade de Software.....	54
Figura 10. Ação Criar Atividade Humana.....	55
Figura 11. Ação Criar SubWorkflow.....	56
Figura 12. Ação Criar Notificação.....	57
Figura 13. Ação Finalizar Workflow.....	58
Figura 14. Ação Compartilhar Workflow e Resultados.....	58
Figura 15. Ação Exportar Workflow.....	59
Figura 16. Ação Implantar Workflow.....	60
Figura 17. Ação Executar Workflow.....	61
Figura 18. Instanciação de um workflow e o ciclo de vida de uma atividade humana	64
Figura 19. Instanciação e criação das atividades humanas.....	66
Figura 20. Ação While para verificar a condição de encerramento das atividades humanas	67
Figura 21. Declaração de propriedade das atividades humanas	68
Figura 22. Liberação de propriedade das atividades humanas	69
Figura 23. Finalização de atividades humanas	70
Figura 24. Evento OnAlarm para lançar o fluxo alternativo de interrupção das atividades humanas.....	71
Figura 25. Ação de captura da exceção deadlineReached	71
Figura 26. Visão geral da abordagem adotada para a captura da proveniência de dados	73
Figura 27. Atividade original empacotada como atividade composta.....	74
Figura 28. Modelo estático para a representação de workflows no ambiente ClusterFlow	76
Figura 29. Exemplo de workflow para adição de valores	83
Figura 30. Modelo estático para a representação da especificação XSD.	83
Figura 31. Modelo estático para a representação da especificação WSDL.....	85
Figura 32. Modelo estático para a representação de workflows em WS-BPEL.....	88
Figura 33. Workflow para realização do exemplo de aplicação.....	100
Figura 34. Tela para criar novos workflows.....	100
Figura 35. Tela que apresenta a definição do workflow.....	101
Figura 36. Definição da atividade de software que configura a aplicação a ser executada....	102
Figura 37. Artefatos antes do mapeamento	103
Figura 38. Mapeamento do artefato problems	103
Figura 39. Artefato problems após o mapeamento	103
Figura 40. Detalhes da atividade de software criada pelo cientista.....	104
Figura 41. Definição da atividade de software que dispara a execução da aplicação remota	105
Figura 42. Tela para definir a atividade de encerramento do workflow.....	105
Figura 43. Tela de compartilhamento, exportação, implantação e instanciação de workflows	106
Figura 44. Tela para instanciar os workflows.....	107

Figura 45. Tela de visualização do estado de execução dos workflows.....	108
Figura 46. Tela para a escolha da instância de execução a ser visualizada.....	110
Figura 47. Tela para a visualização dos detalhes de execução do workflow	110
Figura 48. Detalhes de execução da atividade Configure Ant Application.....	111
Figura 49. Detalhes de execução da atividade Run Ant Application.....	111
Figura 50. Visão do negócio do ambiente ClusterFlow	126
Figura 51. Visão geral do modelo de casos de uso.....	129
Figura 52. Diagrama de atividades completo do workflow TaskManagerProcess	134
Figura 53. Tela de login do ClusterFlow	144
Figura 54. Tela para visualizar todos os workflows compartilhados	144
Figura 55. Página inicial e menu de operações do ambiente.....	145
Figura 56. Tela para reutilizar workflows	145
Figura 57. Tela para definir os artefatos de entrada dos workflows.....	145
Figura 58. Tela para o reuso de atividades de workflows compartilhados.....	145
Figura 59. Tela para selecionar as intâncias de execução de workflows	146

LISTA DE CÓDIGOS

Listagem 1. Estrutura de um documento WS-BPEL.....	29
Listagem 2. Documento XSD gerado pelo ambiente ClusterFlow	84
Listagem 3. Documento WSDL gerado pelo ambiente ClusterFlow	85
Listagem 4. Continuação do documento WSDL gerado pelo ambiente ClusterFlow	86
Listagem 5: Declaração do workflow e dos parceiros envolvidos	88
Listagem 6. Declaração das variáveis do workflow	90
Listagem 7. Atividade de instanciação do workflow em WS-BPEL	90
Listagem 8. Inicialização da variável de entrada da proveniência do serviço web Calculadora	91
Listagem 9. Chamada ao serviço web para a persistência da proveniência de dados	92
Listagem 10. Manipulação de variável e invocação do serviço web Calculadora	93
Listagem 11: Inicialização dos artefatos da mensagem de proveniência de saída do workflow	94
Listagem 12. Manipulação da mensagem da variável de proveniência de saída do workflow	94
Listagem 13. Invocação do serviço web de proveniência de saída do workflow.....	95
Listagem 14. Código WS-BPEL completo do workflow de exemplo CalcWorkflow.....	141

LISTA DE ABREVIATURAS E SIGLAS

BPEL4People	<i>Business Process Execution Language for People</i>
BPEL4WS	<i>Business Process Execution Language for Web Services</i>
BPMN	<i>Business Process Modeling Notation</i>
HTML	<i>Hypertext Markup Language</i>
HTTP	<i>Hypertext Transfer Protocol</i>
J2EE	<i>Java 2 Enterprise Edition</i>
JSON	<i>JavaScript Object Notation</i>
JSP	<i>Java Server Pages</i>
JSTL	<i>Java Standard Tag Library</i>
MIT	<i>Massachusetts Institute of Technology</i>
SGWf	<i>Sistema Gerenciador de Workflow</i>
SGWfC	<i>Sistema Gerenciador de Workflow Científico</i>
SWfMS	<i>Scientific Workflow Management Systems</i>
SMS	<i>Short Message Service</i>
SOA	<i>Service Oriented Architecture</i>
SOAP	<i>Simple Object Access Protocol</i>
SSH	<i>Secure Shell</i>
UDDI	<i>Universal Description Discovery & Integration</i>
URI	<i>Uniform Resource Locator</i>
WS-BPEL	<i>Web Services Business Process Execution Language</i>
WSDL	<i>Web Service Description Language</i>
WS-HumanTask	<i>Web Services Human Task</i>
XML	<i>eXtensible Markup Language</i>
XSD	<i>XML Schema Definition</i>

SUMÁRIO

1. Introdução	17
2. Workflow Científico	21
2.1. Considerações Iniciais	21
2.2. Conceitos de <i>Workflows</i> Científicos.....	21
2.3. <i>Cluster</i> de Computadores	24
2.4. Arquitetura Orientada a Serviço	25
2.5. Serviços Web.....	26
2.6. WS-BPEL	28
2.7. BPEL4People	30
2.8. WS-HumanTask	31
2.9. Princípios para a Construção de <i>Workflows</i> Científicos.....	31
2.9.1. Apoio à Definição de <i>Workflows</i> Abstratos e Concretos.....	32
2.9.2. Execução de <i>Workflows</i> Científicos	32
2.9.3. Apoio à Execução de Atividades Humanas.....	32
2.9.4. Proveniência de Dados	33
2.9.5. Compartilhamento e Reuso de <i>Workflows</i>	34
2.9.6. Recursos Compartilhados	34
2.10. Considerações Finais	34
3. Sistemas Gerenciadores de Workflow Científico.....	36
3.1. Considerações Iniciais	36
3.2. Taverna.....	37
3.3. Kepler	38
3.4. Biowep.....	38
3.5. ^{my} Experiment	39
3.6. Pegasus	39
3.7. GPFlow.....	40
3.8. ActiveVOS	41
3.9. Oracle BPEL <i>Process Manager</i>	41
3.10. Intalio.....	42
3.11. Considerações Finais	42
4. O Ambiente ClusterFlow	43
4.1. Considerações Iniciais	43
4.2. Princípios do Ambiente <i>ClusterFlow</i>	43

4.3.	Cenário do Ambiente <i>ClusterFlow</i>	46
4.4.	Arquitetura do Ambiente	48
4.4.1.	Portal Web para Experimentos	49
4.4.2.	Gerenciador de Experimentos.....	49
4.4.3.	Máquina WS-BPEL.....	50
4.4.4.	Gerenciador de Serviços Web	50
4.4.5.	<i>Cluster</i> de Computadores	51
4.5.	O Processo de Construção e Execução de <i>Workflows</i> Científicos	51
4.5.1.	Criar <i>Workflow</i>	51
4.5.2.	Criar Atividade de Software	53
4.5.3.	Criar Atividade Humana.....	54
4.5.4.	Criar <i>SubWorkflow</i>	55
4.5.5.	Criar Notificação	56
4.5.6.	Finalizar <i>Workflow</i>	57
4.5.7.	Compartilhar <i>Workflow</i> e Resultados	58
4.5.8.	Exportar <i>Workflow</i> e Implantar <i>Workflow</i>	59
4.5.9.	Executar <i>Workflow</i>	60
4.6.	O Gerenciador de Atividades Humanas <i>TaskManagerProcess</i>	62
4.6.1.	Instanciação e Criação das Atividades Humanas	65
4.6.2.	Condição para Encerrar Instâncias de Atividades Humanas	67
4.6.3.	Declaração de Propriedade das Atividades Humanas	68
4.6.4.	Liberação de Propriedade das Atividades Humanas	69
4.6.5.	Finalização de Atividades Humanas.....	69
4.6.6.	Interrupção da Execução de Atividades Humanas	70
4.7.	Proveniência de Dados	72
4.8.	Modelo de Representação de <i>Workflow</i> no Ambiente <i>ClusterFlow</i>	75
4.9.	Considerações Finais	79
5.	Estratégias de Implementação.....	81
5.1.	Considerações Iniciais	81
5.2.	Visão Geral.....	81
5.3.	Mapeamento do Modelo Estático para XSD	83
5.4.	Mapeamento do Modelo Estático para WSDL.....	84
5.5.	Mapeamento do Modelo Estático para WS-BPEL.....	87
5.6.	Considerações Finais	95

6. Exemplo de Aplicação	97
6.1. Considerações Iniciais	97
6.2. Definição do Domínio do Experimento Científico.....	97
6.3. Desenvolvimento do Serviço Web de Acesso ao <i>Cluster</i>	98
6.4. Definição do <i>Workflow</i> para o Experimento Científico	99
6.5. Execução do <i>Workflow</i>	107
6.6. Visualização do Resultado do Experimento.....	109
6.7. Avaliação do Ambiente Proposto.....	112
6.8. Considerações Finais	114
7. Conclusões	116
Referências	121
Apêndice A	126
A.1 Descrição Geral do Ambiente.....	126
A.2 Modelagem e Especificação dos Casos de Uso.....	128
Apêndice B	133
Apêndice C	135
Apêndice D	142
D.1 Principais Atividades Presentes na Linguagem WS-BPEL.....	142
Apêndice E	144

Introdução

A introdução de técnicas de computação para apoiar a comparação e a simulação de experimentos científicos, que por isso também são conhecidos como experimentos *in-silico*, tornou o volume de dados gerado por estes cada vez maior (MATTOSO *et al.*, 2008), o que demanda a utilização de computadores com alto poder de processamento. Neste cenário, a utilização de computadores pessoais se torna inviável devido ao longo tempo de processamento requerido. Por outro lado, a utilização de supercomputadores como *mainframes*, muitas vezes também é inviabilizada, tendo em vista seu alto custo.

Considerando a relação custo-benefício, os *clusters* de computadores se apresentam como soluções que podem oferecer poder de processamento próximo ao dos *mainframes* através de vários computadores interligados por meio de uma rede de interconexão de alta velocidade (UNDERWOOD *et al.*, 2001). Entretanto, esta ainda é uma tecnologia cara que nem sempre está presente em laboratórios de pesquisa. Assim, é necessário que *clusters* sejam compartilhados por vários pesquisadores. Para tal, são necessários mecanismos que permitam o acesso remoto a estes e que experimentos realizados nestes sejam compartilhados e reutilizados. Porém, existem algumas barreiras enfrentadas pelos pesquisadores quando da necessidade de utilização dos *clusters*, dentre elas pode-se destacar: (i) o conhecimento mínimo necessário para a utilização dos mesmos; (ii) as permissões de acesso ao local; e (iii) a distância geográfica entre alguns laboratórios de pesquisa e o local em que os *clusters* estão situados (HUFF *et al.*, 2009). Uma das formas de atender esta demanda é a utilização de *workflows* científicos e serviços web (MATTOSO *et al.*, 2008).

Um *workflow* científico pode ser definido como a automação de um processo científico, no qual as atividades são estruturadas de acordo com o fluxo de dados e as dependências destes (YU *et al.*, 2005) (GIL *et al.*, 2007). As atividades de um *workflow* podem compreender tanto atividades de software, como também atividades realizadas por humanos (HOLLINGSWORTH, 1995). Assim, as atividades de software implementam as soluções computacionais e as atividades humanas representam as ações realizadas pelos cientistas, e que juntas têm por objetivo resolver um problema científico (WANG *et al.*, 2009). Uma vez executadas, estas atividades e seus artefatos de entrada e saída caracterizam um experimento científico (CAVALCANTI *et al.*, 2005).

Uma característica importante em *workflows* científicos é a proveniência de dados, a qual permite que os resultados de atividades do *workflow* possam ser rastreados para auxiliar os cientistas a entender os fatores que conduziram um experimento a determinado resultado. *Workflows* são executados e orquestrados por sistemas específicos, os quais são conhecidos como Sistemas Gerenciadores de *Workflow* (SGWf). Esses sistemas podem apoiar pesquisadores de várias áreas no que diz respeito ao controle de execução de experimentos realizados em *clusters*, bem como na reutilização de parte ou de *workflows* inteiros a partir de experimentos já realizados.

A utilização da Arquitetura Orientada a Serviço (SOA – *Service Oriented Architecture*) e, principalmente, da tecnologia de serviços web, permite que aplicações localizadas remotamente possam ser acessadas por meio da invocação de serviços através da Internet (ALONSO *et al.*, 2004). Estas tecnologias permitem que pesquisadores possam realizar seus experimentos e obter resultados de forma cooperativa. Assim, algumas atividades de um *workflow* científico podem ser realizadas como serviços web. Estes serviços permitem que experimentos realizados em *clusters* de computadores possam ser invocados a partir de atividades de *workflows* científicos. Deste modo, os laboratórios de pesquisa podem disponibilizar serviços web para realizar a comunicação entre os Sistemas Gerenciadores de *Workflow* Científico (SGWfC) e os *clusters* de computadores com o objetivo de realizar experimentos e resolver os problemas de acesso e distância geográfica. Além de resolver estes problemas, os serviços web oferecem uma abstração de comandos necessária para a execução dos experimentos (ou parte deles) que estão contidos no *cluster* sem exigir dos cientistas conhecimentos específicos para a interação com um *cluster*. Esta abstração de comandos é possível, pois os serviços web são invocados a partir de atividades computacionais de *workflows* definidos pelos próprios cientistas.

A execução de um *workflow* requer uma linguagem capaz de descrever formalmente como as atividades serão orquestradas (ALONSO *et al.*, 2004). A linguagem WS-BPEL (*Web Services - Business Process Execution Language*) tem sido amplamente utilizada para compor *workflows* de serviços web. Ela permite definir um modelo e uma gramática para descrever o comportamento de *workflows* baseada nas interações entre os mesmos e seus parceiros na forma de serviços web (OASIS, 2007). Embora WS-BPEL seja considerada a melhor candidata para descrever a orquestração de *workflows* científicos e esteja se tornando uma linguagem de programação completa expressa em XML (*eXtensible Markup Language*) (W3C, 2010b) (DEELMAN *et al.*, 2009), ainda existem lacunas, pois ela não oferece apoio às interações com humanos. Desta forma, implementações proprietárias surgem para contornar esta deficiência, o que leva as execuções de experimentos dos usuários tornarem-se dependentes de um SGWf específico.

Este trabalho de mestrado propõe o desenvolvimento de um ambiente de apoio à realização de experimentos científicos para um *cluster* de computadores, denominado *ClusterFlow*. Este ambiente foi definido com base em pesquisas em diversos SGWfC existentes, tais como Taverna (OINN *et al.*, 2002), Kepler (ALTINTAS *et al.*, 2004), Biowep (ROMANO *et al.*, 2007) e Pegasus (DEELMAN *et al.*, 2002), no ambiente de pesquisa virtual ^{my}Experiment (GOBLE *et al.*, 2007) e no portal web do MIT *Process Handbook*¹ (CORPORATION, 2001). Também foram investigados SGWf empresariais que utilizam a linguagem WS-BPEL para compor os seus *workflows*, tais como ActiveBPEL (ENDPOINTS, 2010a), ActiveVOS (ENDPOINTS, 2010b), Oracle BPEL *Process Manager* (ORACLE, 2010) e Intalio (INTALIO, 2010a), pois eles possuem características semelhantes às do ambiente proposto.

O ambiente *ClusterFlow* visa auxiliar pesquisadores de várias áreas, tais como geografia, engenharia química, biologia e ciência da computação na definição, execução e compartilhamento de *workflows* científicos. Serviços web são utilizados para oferecer meios de acesso aos experimentos contidos em um *cluster* de computadores. Uma interface gráfica a partir da qual os pesquisadores podem gerenciar seus *workflows* é oferecida. Esta interface permite que os cientistas não se preocupem com as tecnologias relacionadas à implementação das atividades. Foi proposto um mecanismo de gerenciamento de *workflow* que é capaz de apoiar a interação de humanos na realização de atividades manuais em *workflows* científicos,

¹ Manual de Processos do Instituto de Tecnologia de Massachusetts

bem como gerenciar as suas instâncias de execução, que é independente de códigos de fabricantes.

As contribuições apresentadas neste trabalho são: (i) oferecer o projeto documentado de um ambiente de apoio à realização de experimentos científicos para um *cluster* de computadores por meio de serviços web; (ii) propor um mecanismo que apóie o gerenciamento das instâncias de execução das atividades humanas em *workflows* científicos, as quais são executadas por ações manuais realizadas por pesquisadores; (iii) oferecer um mecanismo que permita o compartilhamento e reuso de *workflows* científicos e suas atividades entre os pesquisadores. O projeto deste ambiente é avaliado com base no desenvolvimento de um protótipo e um exemplo de aplicação.

Esta dissertação está organizada em 7 (sete) capítulos. O Capítulo 2 apresenta a revisão bibliográfica e a contextualização do assunto deste trabalho. Os principais trabalhos relacionados são descritos no Capítulo 3. O Capítulo 4 apresenta o projeto do ambiente *ClusterFlow* desenvolvido neste trabalho de mestrado, a arquitetura do ambiente e o processo de construção de *workflows* científicos. A proveniência de dados, bem como o mecanismo proposto para a interação e o gerenciamento das instâncias das atividades humanas entre o ambiente e os pesquisadores também são descritos neste capítulo. As estratégias de implementação e as tecnologias utilizadas são descritas no Capítulo 5. Um exemplo de aplicação, o protótipo do projeto do ambiente e a avaliação qualitativa são explicados no Capítulo 6. Por fim, o Capítulo 7 apresenta as conclusões e os trabalhos futuros.

Workflow Científico

2.1. Considerações Iniciais

Este capítulo apresenta a revisão bibliográfica e a conceituação necessária para o entendimento desta dissertação. *Workflows* científicos e sua importância na realização de experimentos científicos são descritos inicialmente. A tecnologia de *cluster* de computadores é abordada com o enfoque na realização de experimentos que necessitem de alto poder de processamento. Na sequência, a arquitetura orientada a serviços e a tecnologia de serviços web são descritas seguidas da linguagem WS-BPEL. Esta linguagem é usada para descrever a forma de execução dos serviços web em um *workflow*. Contudo, atividades humanas em *workflows* necessitam da linguagem BPEL4People em caso da utilização de WS-BPEL. A fundamentação da linguagem BPEL4People antecede a descrição da linguagem WS-HumanTask, que por sua vez define as formas de implementação e apresentação das atividades humanas em *workflows*.

2.2. Conceitos de *Workflows* Científicos

A maioria dos cientistas conduz análises e executam modelos em diversos e diferentes ambientes de hardware e software, coordenando manualmente a importação e exportação de dados de um ambiente para o outro. Este procedimento é suscetível a erros, principalmente quando os experimentos são grandes e complexos a ponto de aumentar a dificuldade do processo de gerenciamento e a troca de dados de um ambiente para o outro. Dessa forma, o

termo *e-science* surge da necessidade de realizar pesquisa científica no desenvolvimento e execução de soluções de alto desempenho. Ele se caracteriza pelo apoio dado ao cientista na realização de seus experimentos científicos utilizando uma infraestrutura computacional adequada. Para apoiar os cientistas no gerenciamento e realização destes experimentos, surge a necessidade da utilização de *workflows* científicos (MATTOSO *et al.*, 2008).

Um *workflow* científico pode ser definido como a automação de um processo científico, em que suas atividades são organizadas de acordo com o fluxo de dados e as dependências entre eles (YU *et al.*, 2005) (GIL *et al.*, 2007). Estas atividades podem envolver atividades de software e atividades realizadas por humanos (HOLLINGSWORTH, 1995). As atividades de software implementam as soluções computacionais e as atividades humanas representam as ações realizadas pelos cientistas, que juntas têm por objetivo resolver um problema científico (WANG *et al.*, 2009). Vale ressaltar que neste trabalho o foco é o apoio à realização de experimentos científicos, e por isto, a definição de *workflow* científico. Outro foco bastante difundido, e não abordado aqui, é o de *workflows* empresariais que tratam da lógica de negócios entre organizações. O conceito de *workflows* empresariais não foi abordado, pois existem muitas divergências na literatura no que se refere à definição e comparação destes dois tipos de *workflows*.

Os *workflows* científicos têm crescido na literatura como um paradigma para a representação e gerenciamento de computações científicas distribuídas. Atualmente, análises científicas complexas necessitam de um grande esforço humano e coordenação manual. Os dados crescem exponencialmente, assim, os cientistas necessitam de ferramentas mais eficientes para auxiliá-los. Ambientes de gerenciamento de *workflows* que apóiam e melhoram os processos científicos são imprescindíveis para manter a velocidade de crescimento rápida dos dados e processamento (GIL *et al.*, 2007).

Com a utilização de *workflows* científicos, os pesquisadores podem definir quais são as suas suposições e assim criar seus experimentos partindo de um conjunto de atividades que devem ser executadas seguindo uma determinada sequência. As informações referentes ao experimento, tais como os parâmetros de entrada, programas que foram utilizados na execução e os resultados encontrados, devem ser armazenadas em uma base de dados para garantir a confiabilidade e a validação dos experimentos (CRUZ *et al.*, 2008).

Técnicas da engenharia de software foram utilizadas para diminuir o tempo de construção dos *workflows* científicos, o que permite o reuso de atividades existentes em outros *workflows*. Assim, um *workflow* científico pode ser totalmente novo, criado a partir de algumas atividades, ou, apenas construído por meio de reutilização de *workflows* existentes.

Neste caso, o *workflow* reutilizado também pode ser visualizado como um *subworkflow*, e assim, realizar o papel semelhante ao de uma atividade pertencente a outro *workflow* em construção ou execução. Desse modo, o entendimento de todo o ciclo de vida do projeto de *workflow*, prototipação, produção, gerenciamento, publicação e descoberta é fundamental para o desenvolvimento de sistemas que possam dar apoio ao trabalho dos cientistas e não apenas na execução dos mesmos (ROURE *et al.*, 2007).

As facilidades para descrição e execução de *workflows* representam um novo modo de compartilhamento e gerenciamento de informação, no qual processos inteiros podem ser capturados eletronicamente e compartilhados para o reuso e referências futuras. Cientistas devem ser estimulados a compartilhar descrições de suas análises científicas e computações tornando-as formais e explícitas. No entanto, não existe um consenso sobre representações comuns na comunidade científica. É importante que informações detalhadas sobre os processos de análises sejam incorporadas às representações de *workflow* para apoiar a descoberta, criação, união, e execução de *workflows*. Assim, essas atividades se tornam um modo natural de conduzir experimentos e compartilhar metodologias científicas por todas as comunidades científicas (GIL *et al.*, 2007).

No contexto desta dissertação, considera-se que um *workflow* científico compreende o processo desde a captura, modelagem, execução, até a análise de experimentos científicos. Assim, quando o experimento científico segue este processo, torna-se reproduzível com maior facilidade. A proveniência de dados registra quais foram as entradas que levaram a um resultado em específico, o que auxilia o processo de análise e compreensão do *workflow* científico (MATTOSO *et al.*, 2008). A Figura 1 apresenta as etapas do ciclo de vida de um experimento científico de acordo com a visão de (OINN *et al.*, 2002).

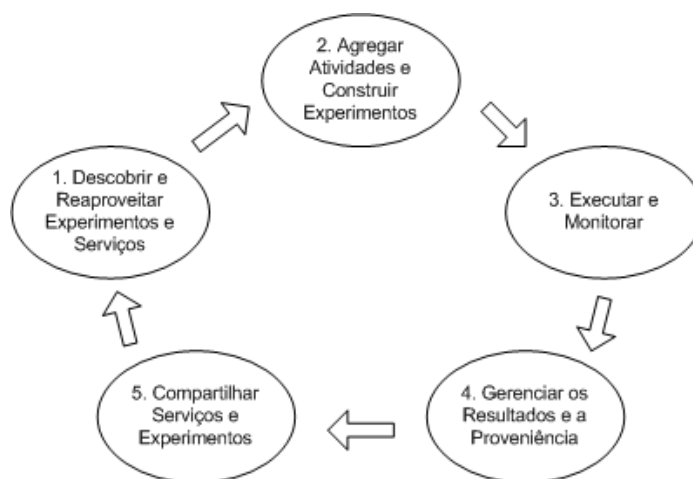


Figura 1. Ciclo de vida de um experimento científico

A “etapa 1” do ciclo de vida de um experimento científico corresponde à fase de busca e descoberta de *workflows* ou serviços existentes para o reuso. Nesta etapa, o cientista consulta repositórios de *workflows* com o objetivo de encontrar algum que atenda às suas necessidades, o que evita o trabalho de recriação de um mesmo *workflow*. Na “fase 2”, o cientista define a sua versão do *workflow*. Esta definição compreende a construção de experimentos a partir da reutilização de *workflows* encontrados na “etapa 1”, bem como a definição de novas atividades. Uma vez o *workflow* definido, este pode ser executado e monitorado pelos pesquisadores. A execução e o monitoramento correspondem à “etapa 3” do ciclo de vida de um experimento científico. Logo que a execução do *workflow* encerrar, seus dados de resultado são analisados e armazenados para consultas futuras. Isto permite que cientistas possam interpretar o resultado gerado em cada atividade a partir de seus dados ancestrais. A execução e o monitoramento representam a “fase 4” do ciclo de vida de um experimento científico. A “etapa 5” corresponde ao compartilhamento do *workflow* e de seu resultado de execução. Uma vez compartilhado, o *workflow* pode ser reutilizado em outros experimentos e assim acelerar o processo de pesquisa científica.

2.3. Cluster de Computadores

Os procedimentos computacionais são fortes aliados de pesquisadores de todas as áreas. No entanto, da mesma forma com que a ciência evolui, a quantidade de informações geradas para realizar as análises destes procedimentos cresce proporcionalmente. As informações geradas por experimentos continuam sendo obtidas em laboratórios, porém, análises estão sendo feitas com auxílio de computador por meio de modelagem, simulações e experimentos computacionais com o apoio de *workflows* científicos (MATTOSO *et al.*, 2008). Para realizar simulações e experimentos é necessária uma infraestrutura computacional com alto poder de processamento, a qual pode ser oferecida por supercomputadores ou *clusters* de computadores.

Um *cluster* de computadores é um tipo de arquitetura de processamento paralelo e distribuído, o qual consiste de uma coleção de computadores interconectados que trabalham juntos com a ilusão de ser apenas um (BELL *et al.*, 2002). Um nó pode ser um computador com um único ou vários processadores, memória, periféricos de entrada e saída e um sistema operacional. Geralmente, um *cluster* consiste de dois ou mais nós de computadores interconectados. Esses nós podem estar localizados em um único gabinete ou estarem separados fisicamente e interconectados por meio de uma rede de alta velocidade.

Embora um *cluster* seja constituído de computadores com poder de processamento e custo menor que um supercomputador, um *cluster* continua sendo um recurso caro. À medida que se deseja aumentar o desempenho do *cluster* são necessários equipamentos de melhor qualidade (UNDERWOOD *et al.*, 2001). O custo destes equipamentos é adicionado ao custo do *cluster*. Assim, com o intuito de aproveitar da melhor maneira o desempenho e o custo da infraestrutura computacional de um *cluster* de computadores, é desejável que este seja disponibilizado para vários grupos de pesquisadores interessados em realizar experimentos que necessitem de alto poder de processamento.

2.4. Arquitetura Orientada a Serviço

A disseminação e a disponibilização da Internet para as pessoas e empresas em geral popularizaram o uso das redes de computadores, o que levou a Internet a se tornar um meio de troca de informações tanto para pessoas como para empresas. O avanço da arquitetura *Cliente-Servidor* gerou a necessidade de interação entre aplicações pertencentes a diferentes organizações. Dessa forma, um novo modelo de computação surgiu para possibilitar a troca de informações e a integração de sistemas existentes, a Computação Orientada a Serviços. Este modelo facilitou a interação entre as aplicações na Internet por meio de serviços eletrônicos (ALONSO *et al.*, 2004).

A computação orientada a serviços é um paradigma computacional que utiliza serviços para apoiar o desenvolvimento de aplicações distribuídas em ambientes heterogêneos de forma rápida e com baixo custo. Os serviços são módulos de software independentes que podem ser descritos, publicados, localizados, orquestrados e desenvolvidos em plataformas computacionais distintas. Os desafios na integração destes serviços pertencentes a diferentes organizações, desenvolvimento de novas abordagens e arquiteturas para tratar os requisitos de baixo acoplamento, padronização e aplicações computacionais que utilizavam protocolos de comunicação independentes culminaram na introdução da Arquitetura Orientada a Serviço (SOA²).

SOA é uma abordagem arquitetural que apóia o desenvolvimento de serviços fracamente acoplados para permitir flexibilidade e interoperabilidade em sistemas de software independentemente da tecnologia usada para o desenvolvimento destes. Os recursos e funcionalidades de software em SOA são disponibilizados através de serviços que são acessados e executados baseados em interfaces de descrições de serviços. Esta abordagem

² Do inglês *Service Oriented Architecture*.

arquitetural permite que os desenvolvedores superem muitas complexidades de implementação encontradas no desenvolvimento de aplicações distribuídas, integração de aplicações e múltiplas plataformas de execução. Assim, o conceito de SOA oferece diversas características interessantes, tais como:

- redução da complexidade dos serviços por meio do encapsulamento, o que permite a invocação de aplicações legadas, disparar recursos e até outros serviços;
- interfaces de serviço que permitem a interoperabilidade entre os sistemas e as arquiteturas por meio do uso de padrões abertos;
- os serviços permitem um modelo fracamente acoplado, no qual os clientes podem fazer o mapeamento da terminação dos serviços em tempo de execução;
- todas as funcionalidades em SOA são definidas como serviços; e
- todos os serviços são aplicações de software independentes (PAPAZOGLU, 2008).

2.5. Serviços Web

Os serviços web representam um tipo de serviço eletrônico. Os motivos que atraem o uso de serviços web são a sua interoperabilidade com outros sistemas e a simplicidade pelo fato de usar a invocação remota das operações utilizando tecnologias padronizadas como, HTTP³ (W3C, 2010a) e XML⁴ (W3C, 2010b) (ALCHIERI *et al.*, 2007).

Serviços web podem ser vistos como uma forma de expor as funcionalidades de um sistema de informação e torná-las disponíveis por meio de tecnologias web padronizadas (ALONSO *et al.*, 2004). Um serviço web é uma aplicação de software que pode ser acessada por meio de um URI⁵, no qual as interfaces podem ser definidas, descritas e descobertas por meio de arquivos XML e serem trocadas utilizando protocolos baseados na Internet. Uma sintaxe comum baseada em XML é adotada por todos os padrões da tecnologia de serviços web (PAPAZOGLU, 2008).

As arquiteturas de serviços web são baseadas em três elementos: o solicitador de serviços, o provedor de serviços e o servidor de registros (ALONSO *et al.*, 2004) (PAPAZOGLU, 2008). Essencialmente existe um protocolo de comunicação SOAP⁶ (W3C, 2007), uma linguagem WSDL⁷ (W3C, 2001) para descrever serviços, e, um diretório de

³ Do inglês *Hypertext Transfer Protocol*.

⁴ Do inglês *eXtensible Markup Language*.

⁵ Do inglês *Uniform Resource Identifier*.

⁶ Do inglês *Simple Object Access Protocol*.

⁷ Do inglês *Web Service Description Language*.

serviços UDDI⁸ (OASIS, 2004). SOAP, WSDL e UDDI, são o centro dos serviços web e os principais conceitos relacionados a eles são:

- SOAP: visa oferecer um modo padronizado de codificar os diferentes protocolos e os mecanismos de interação em documentos XML, os quais podem ser facilmente trocados na Internet;
- WSDL: é uma linguagem que descreve os mecanismos de interação com um serviço web. Essencialmente, ela especifica uma interface que controla a forma de realizar as interações entre o solicitador e o provedor de serviços. O WSDL é descrito por meio de uma interface de descrição de serviços baseada em XML. Assim, quando uma aplicação cliente necessita conhecer a interface oferecida por um serviço web, é o WSDL quem fornece a descrição; e
- UDDI: é a especificação do servidor de registro para os serviços web. Esses servidores são utilizados pelos provedores de serviços para publicar e anunciar os serviços disponíveis, bem como por clientes para consultar por serviços oferecidos.

A Figura 2 mostra os principais elementos da arquitetura de serviços web e a forma de interação entre os mesmos.

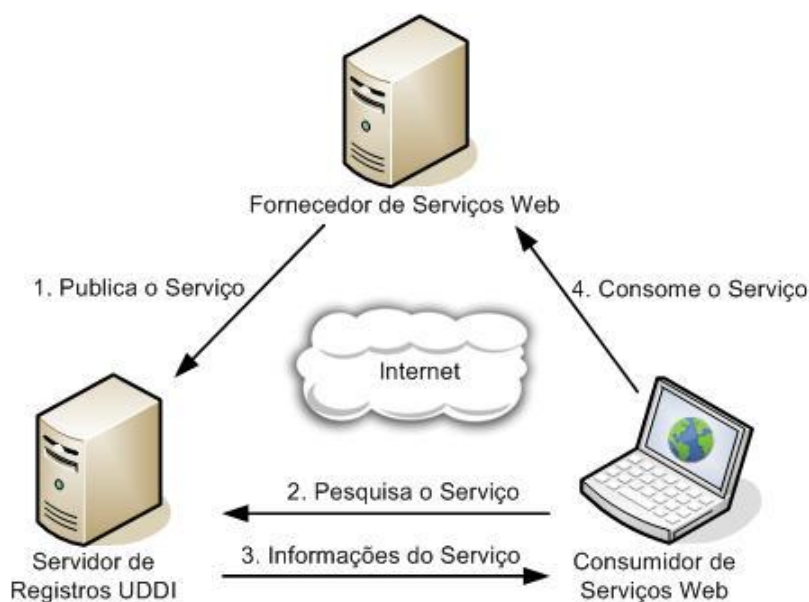


Figura 2. Principais elementos da arquitetura de serviços web

A interação dos principais elementos da arquitetura de serviços web basicamente acontece por meio de 4 (quatro) grandes operações. Primeiramente o Fornecedor de Serviços Web publica a descrição do seu serviço, o WSDL, em um Servidor de

⁸ Do inglês *Universal Description Discovery & Integration*.

Registros UDDI, que também é conhecido como repositório de serviços web. Em seguida, o Consumidor de Serviços Web pesquisa os serviços desejados no repositório de serviços e seleciona o mais apropriado ao seu interesse. Logo, o Servidor de Registros UDDI devolve as informações do serviço selecionado. Por fim, o Consumidor de Serviços Web pode consumir o serviço encontrado por meio da troca de mensagens utilizando o protocolo SOAP.

2.6. WS-BPEL

A complexidade, o inesperado e a interdependência dos componentes em um *workflow* frequentemente exigem flexibilidade em uma linguagem para apoiar a manipulação de exceção, recuperação de situações duvidosas, dinamismo para se adaptar à troca de ambiente e suporte à seleção dinâmica de serviços. Além disso, os serviços expostos como serviços web (tais como aqueles que empacotam e expõem códigos legados) podem ser integrados em *workflows* complexos que podem atravessar múltiplos domínios e organizações. A situação se torna mais complexa quando são combinados múltiplos serviços web geograficamente dispersos, que formam um único serviço composto na forma de *workflow* (ou *subworkflow*). Entretanto, para compor um único serviço web na forma de um *workflow*, é imprescindível que seja utilizada uma linguagem capaz de orquestrar este conjunto de serviços web. A especificação chamada de Linguagem de Execução de Processos de Negócio para Serviços Web (BPEL4WS⁹), atualmente referenciada como WS-BPEL¹⁰, define um padrão a esta linguagem com o objetivo de compor *workflows* de serviços web. No contexto deste trabalho, ela é a responsável por modelar o comportamento de serviços web em uma interação de um experimento científico. Também oferece uma estrutura baseada em XML para descrever o controle lógico exigido para coordenar os serviços web participantes em um conjunto de atividades. Um motor de execução é utilizado para interpretar essa linguagem, coordenar as atividades e estabilizar o *workflow* inteiro caso ocorra algum erro (PELTZ, 2003).

A especificação WS-BPEL oferece apoio para atividades básicas e estruturadas. Uma atividade básica é uma instrução que interage com algo externo ao próprio *workflow*. Exemplos disso são as atividades responsáveis por manipular o recebimento, resposta ou a invocação de serviços web. Por outro lado, as atividades estruturadas são as responsáveis por gerenciar o fluxo inteiro do *workflow*, especificando a sequência de execução para os serviços

⁹ Do inglês *Business Process Execution Language for Web Services*.

¹⁰ Do inglês *Web Services Business Process Execution Language*.

web referenciados. Estas atividades também suportam *loops* condicionais e desvios dinâmicos (OASIS, 2007). O detalhamento das principais atividades da especificação WS-BPEL pode ser visto no Apêndice D.

A WS-BPEL foi construída sobre a especificação WSDL. Assim, o documento WSDL define as operações específicas permitidas durante a execução do *workflow*, e o documento WS-BPEL define como estas devem ser orquestradas. A interface WSDL descreve as operações públicas de instanciação e os tipos de dados que trafegam nas mensagens trocadas pelos *workflows* e serviços web externos (PELTZ, 2003).

A estrutura dos documentos WS-BPEL está dividida em quatro importantes seções detalhadas a seguir:

- *partnerLinks*: define os diferentes parceiros que interagem com o *workflow* durante a sua execução. Cada *partnerLink* é caracterizado por um *partnerLinkType*, que por sua vez, pode ter um ou dois papéis atribuídos. Cada papel identifica a funcionalidade que o *workflow* e o parceiro envolvido devem exercer para aquela interação.
- *variables*: define as variáveis usadas pelo *workflow* durante a execução;
- *faultHandlers*: seção que contém os tratadores de falha que definem as atividades a serem executadas caso alguma exceção ocorra durante a execução; e
- *process*: define as configurações iniciais do *workflow* referenciando os documentos externos necessários ao WS-BPEL e que devem ser importados, bem como define o nome do *workflow*. A Listagem 1 mostra a estrutura de um documento em WS-BPEL (ANDREWS *et al.*, 2003).

```

1 <bpel:process>
2   <bpel:partnerLinks>
3     ...
4   </bpel:partnerLinks>
5   <bpel:variables>
6     ...
7   </bpel:variables>
8   <bpel:faultHandlers>
9     ...
10  </bpel:faultHandlers>
11  <bpel:sequence>
12    <bpel:receive />
13    <bpel:flow>
14      <bpel:invoke />
15      <bpel:invoke />
16    </bpel:flow>
17    <bpel:reply />
18  </bpel:sequence>
19</bpel:process>

```

Listagem 1. Estrutura de um documento WS-BPEL

O surgimento da especificação WS-BPEL como aplicação padrão para a coordenação de serviços web é muito significativo, pois permite que pesquisadores projetem e executem seus *workflows* científicos com a utilização de ferramentas comerciais ou de código aberto. Atualmente, a linguagem WS-BPEL é considerada a melhor candidata para coordenar serviços científicos (AKRAM *et al.*, 2006).

2.7. BPEL4People

Embora a especificação WS-BPEL seja considerada a melhor candidata para coordenar os serviços web de *workflows* científicos, ela não oferece apoio às interações com humanos. Porém, *workflows* geralmente exigem a interação humana durante sua execução, e não prover isso é um problema. Como resultado, surgem dependências de códigos de fabricantes de SGWf específicos para a execução de *workflows*. Para evitar essa dependência e apoiar a variedade de cenários que envolvem pessoas em *workflows*, foi necessária a criação de uma extensão à especificação WS-BPEL, a BPEL4People (OASIS, 2010a). Esta especificação foi definida sobre a especificação WS-BPEL na forma de extensão, de tal modo que suas características podem ser compostas com as características da linguagem WS-BPEL.

A partir da especificação da linguagem BPEL4People, pessoas podem ser envolvidas em *workflows* como um tipo especial de implementação de atividade. Com a finalidade de facilitar essa implementação foi criada uma atividade de extensão na linguagem WS-BPEL. Esta atividade foi chamada de `peopleActivity`. No entanto, o motor de execução WS-BPEL deve processar as atividades das pessoas diferentemente das atividades de invocação de serviços web. Assim, quando uma `peopleActivity` é instanciada pelo motor WS-BPEL, este cria itens de trabalho desta atividade, que são distribuídos para as pessoas adequadas a executá-la com base nos papéis que essas pessoas exercem no *workflow* (KLOPPMANN *et al.*, 2005). Assim sendo, quando uma pessoa resolve trabalhar na atividade, ela pode ler os dados de entrada, executar as ações necessárias, informar o resultado da atividade por meio de uma interface de usuário e enviá-lo ao motor de execução, o qual continuará com a execução do fluxo do *workflow* (HUFF *et al.*, 2009).

Existem algumas formas de submissão de atividades aos usuários as quais podem compreender um diálogo de uma interface gráfica, um formulário HTML¹¹, uma planilha eletrônica, uma mensagem de voz ou SMS¹². Assim, a utilização de BPEL4People para a

¹¹ Do inglês *Hypertext Markup Language*.

¹² Do inglês *Short Message Service*.

construção e execução de *workflows* científicos deixa explícita que a implementação de uma determinada atividade manual deve ser realizada por uma pessoa. Deste modo, as atividades realizadas por pessoas são interpretadas pelo motor de execução de *workflows* como serviços (ações) que são implementados por humanos (KLOPPMANN *et al.*, 2005).

2.8. WS-HumanTask

Apesar da especificação BPEL4People definir que uma atividade humana deve ser realizada em um *workflow* WS-BPEL, a BPEL4People não especifica os detalhes de como a atividade será disponibilizada a uma pessoa. Estes detalhes são definidos por meio da especificação da linguagem WS-HumanTask, a qual permite especificar atividades locais, tais como a aprovação de um resultado, ou, referenciar atividades que estão fora da definição do *workflow* em WS-BPEL, que podem compreender atividades humanas orquestradas por outro *workflow*. Desta forma, a extensão denominada `task` foi introduzida na especificação BPEL4People, a qual referencia os detalhes de implementação das atividades humanas e representa as ações que os usuários devem executar.

A especificação WS-HumanTask permite a integração de seres humanos com aplicações orientadas a serviço definindo duas interfaces: uma expõe o serviço oferecido pela atividade, como a aprovação de algum resultado parcial do experimento, e outra, para os cientistas interagirem com as atividades humanas. Esta oferece consultas às atividades não realizadas, bem como permite que cientistas informem os dados de resultado da atividade humana. Além disso, a especificação WS-HumanTask permite que as pessoas designadas a trabalhar em uma determinada atividade sejam definidas e alocadas em uma determinada ordem, como também podem ser definidos grupos de possíveis proprietários da atividade que são responsáveis por executá-la. WS-HumanTask também permite que notificações sejam definidas para que informações referentes ao *workflow* em execução sejam enviadas aos humanos. Estas notificações são um tipo especial de atividade humana, pois as mesmas são enviadas às pessoas sem aguardar que elas retornem alguma confirmação de recebimento, são assíncronas (OASIS, 2010b).

2.9. Princípios para a Construção de *Workflows* Científicos

Uma série de princípios deve ser levada em consideração na concepção de *workflows* científicos. Os princípios apresentados nesta seção foram extraídos de pesquisas em diversos

trabalhos já realizados na área, tais como (GIL *et al.*, 2007), (BECO *et al.*, 2005), (RYGG *et al.*, 2006), (CRUZ *et al.*, 2008), (ROURE *et al.*, 2007) e (FREIRE *et al.*, 2008).

2.9.1. Apoio à Definição de *Workflows* Abstratos e Concretos

Para que cientistas possam criar seus experimentos, *workflows* necessitam ser definidos de forma abstrata ou concreta. Muitas vezes cientistas desejam definir suas ideias de experimentos como uma forma de documentação e deixar a implementação das atividades de software para o futuro. *Workflows* abstratos permitem que cientistas definam um fluxo lógico de dados e uma sequência de execução das atividades sem preocupar-se com detalhes tecnológicos, como por exemplo, qual software executará em seu *workflow*. Por outro lado, *workflows* concretos são considerados a representação final de *workflows* com detalhes tecnológicos definidos. *Workflows* concretos podem ser instanciados pelos cientistas e executados pelo motor de execução de *workflows*. Um *workflow* é considerado concreto a partir do momento em que todos os mapeamentos do fluxo de dados entre as atividades estiverem corretamente definidos e as atividades de software estiverem relacionadas com suas respectivas implementações computacionais (GIL *et al.*, 2007) (BECO *et al.*, 2005).

2.9.2. Execução de *Workflows* Científicos

A execução de *workflows* científicos é uma forma de automatizar a coordenação de experimentos científicos que geralmente são conduzidos de forma manual em diversos ambientes de hardware e software. Esta coordenação é realizada por máquinas de execução de *workflow* capazes de interpretar, coordenar e executar os *workflows* definidos pelos cientistas. Porém, é comum que SGWf usem sua própria linguagem de definição de *workflow*. A falta de padronização gera uma dificuldade adicional aos pesquisadores, pois *workflows* criados usando uma tecnologia não podem ser traduzidos para outros SGWf, o que gera a necessidade da utilização de uma linguagem padronizada internacionalmente para contornar este obstáculo.

2.9.3. Apoio à Execução de Atividades Humanas

É comum que em *workflows* científicos existam atividades nas quais há a necessidade de interação com os pesquisadores. Experimentos científicos muitas vezes superam execuções de aplicações de software, de forma que podem existir ocasiões nas quais um cientista aprove o

resultado de uma atividade executada por um software ou, este receba uma atividade que deve ser realizada por meio de uma ação humana. Desta forma, cientistas podem participar das execuções de *workflows* introduzindo novos aspectos em experimentos científicos, tais como, a interação do pesquisador com o *workflow* durante sua execução através de uma interface gráfica, que permite informar o resultado da atividade humana conforme os dados resultantes da ação realizada (RYGG *et al.*, 2006) (OASIS, 2010a).

2.9.4. Proveniência de Dados

A proveniência de dados pode auxiliar os cientistas a interpretar e entender os resultados de experimentos científicos, pois é possível realizar uma variedade de análises nestes, tais como: (i) avaliar a sequência de atividades que levaram a um determinado resultado; (ii) compreender o motivo pelo qual um cientista utilizou uma sequência de atividades; (iii) verificar se um experimento foi executado de acordo com procedimentos aceitáveis; (iv) verificar a partir de dados ancestrais se o resultado obtido é confiável ou não; e, (v) identificar os artefatos de entrada dos experimentos que conduziram a um determinado resultado (FREIRE *et al.*, 2008) (MARINHO *et al.*, 2009) (SIMMHAN *et al.*, 2005).

A captura de proveniência pode ser feita a partir de três níveis: sistema operacional, *workflow* e atividade. Quando a proveniência é capturada a partir do primeiro nível, os mecanismos e funcionalidades utilizados para a coleta dos dados são do próprio sistema operacional, tais como, sistema de arquivos e rastreamento de chamadas ao sistema. Este tipo de captura de dados tem a vantagem de ser independente do SGWfC. Entretanto, tem a desvantagem de que os dados coletados são bem específicos e geralmente precisam de processamento posterior. No nível de *workflow*, o responsável por coletar as informações de proveniência é o SGWfC. Embora seja o meio mais usado para realizar este tipo de trabalho, este nível tem a desvantagem de ser dependente do SGWfC. Existe ainda o mecanismo de captura de proveniência em nível de atividade, no qual cada atividade do *workflow* é responsável por coletar suas próprias informações de proveniência. Além disso, este nível de captura tenta reunir as melhores características dos dois níveis anteriores. Assim, a vantagem é que ele é independente do SGWfC, como no nível de sistema operacional e oferece a possibilidade de capturar as informações de forma mais precisa, igualmente a captura em nível de *workflow*. Porém, existe uma desvantagem que é a adaptação das atividades que já existiam anteriormente para incorporar os mecanismos de proveniência (FREIRE *et al.*, 2008).

2.9.5. Compartilhamento e Reuso de *Workflows*

O compartilhamento e o reuso de *workflows* visam acelerar o processo de pesquisa para que cientistas reutilizem atividades de outros *workflows* sem a necessidade de recriar todo o experimento. Uma iniciativa para o compartilhamento de *workflows* que visa a reutilização destes foi desenvolvida por (STEVENS *et al.*, 2003), os quais criaram o ambiente de pesquisa virtual ^{my}Experiment (ROURE *et al.*, 2007). Este ambiente torna possível que *workflows* sejam compartilhados e reutilizados em experimentos similares ao original.

2.9.6. Recursos Compartilhados

Compartilhar recursos computacionais em experimentos científicos pode ser uma forma de diminuir custos dos laboratórios de pesquisa. Além disso, o compartilhamento de recursos permite que *workflows* de experimentos científicos sejam definidos com maior velocidade, pois pesquisadores não necessitam desenvolver ou instalar todos os recursos computacionais para a execução de seus *workflows*. Estes recursos podem compreender a tecnologia de serviços web, *clusters* de computadores, computação em *Grid*, dentre outros.

Para o recurso de *clusters* de computadores, por exemplo, pesquisadores geralmente desconhecem ou possuem pouco conhecimento para a realização de experimentos computacionais. Este pouco conhecimento somado à dificuldade e permissões de acesso ao local onde se encontram os *clusters*, à distância geográfica entre alguns laboratórios de pesquisa e aos locais onde os *clusters* estão situados, oferecem barreiras aos pesquisadores quando experimentos que necessitem de alto poder de processamento devem ser realizados. Para contornar estes obstáculos devem existir mecanismos capazes de abstrair os comandos necessários para a realização de experimentos em *clusters* (HUFF *et al.*, 2009).

2.10. Considerações Finais

Este capítulo apresentou a revisão bibliográfica necessária para o entendimento desta dissertação. *Workflows* científicos podem ser usados para apoiar a automação de experimentos científicos que necessitem de alto poder de processamento obtido por meio de *clusters* de computadores. Contudo, pesquisadores geralmente não possuem conhecimentos suficientes para executar experimentos em *clusters*, o que gera a necessidade da abstração de comandos usados nestes. Os serviços web podem oferecer esta abstração por meio de suas operações definidas em suas interfaces WSDL. Com o apoio que a linguagem WS-BPEL

pode oferecer aos *workflows* científicos, é possível que a execução destes serviços seja orquestrada. No entanto, esta linguagem apenas apóia a orquestração de serviços web que são representados pelas atividades de software de um *workflow* científico. Dessa forma, para definir atividades humanas em *workflows* baseados em WS-BPEL surge a necessidade da utilização das linguagens BPEL4People e WS-HumanTask. Além disso, com base na pesquisa de diversos trabalhos na área, foi levantada uma série de princípios que devem ser levados em consideração para apoiar a construção de *workflows* científicos. No próximo capítulo, alguns trabalhos relacionados mais significativos e as suas principais peculiaridades são descritas.

Sistemas Gerenciadores de *Workflow* Científico

3.1. Considerações Iniciais

Este capítulo visa apresentar os SGWfCs e suas peculiaridades mais relevantes para o desenvolvimento deste trabalho de mestrado. O projeto do ambiente *ClusterFlow* foi definido a partir de estudos destes SGWfCs, bem como da utilização destes para fins de testes e avaliação. São apresentados os SGWfs Taverna (HULL *et al.*, 2006) (OINN *et al.*, 2004), Kepler (ALTINTAS *et al.*, 2004), Biowep (ROMANO *et al.*, 2007), ^{my}Experiment (ROURE *et al.*, 2007), Pegasus (LEE *et al.*, 2009), GPFlow (RYGG *et al.*, 2006), ActiveVOS (ENDPOINTS, 2010b), Oracle BPEL *Process Manager* (ORACLE, 2010) e Intalio (INTALIO, 2010a).

A escolha destes SGWfs teve influência devido às características julgadas interessantes para o *ClusterFlow*. Os SGWfCs Taverna e Kepler são vastamente utilizados pela comunidade científica e possuem implementações de serviços para conceber seus *workflows*. O Taverna possui mais de 3.000 (três mil) serviços disponíveis aos cientistas. O Biowep teve influência com relação à interface gráfica apresentada ao usuário. O ^{my}Experiment apresenta um portal web com a visão de compartilhamento de *workflow* e também por ser vastamente utilizado no meio científico. Os SGWfC Pegasus e Kepler tratam especificamente do gerenciamento e execução de *workflows* que necessitem de alto poder de

processamento. O SGWfC GPFlow é mostrado por se tratar de ambiente que utiliza portal web para acesso aos *workflows* e utiliza a linguagem WS-BPEL para compor seus experimentos. Por fim, os SGWf ActiveVOS, Oracle BPEL *Process Manager* e Intalio tratam do gerenciamento de *workflows* com WS-BPEL e buscam apoiar o gerenciamento de atividades humanas em seus *workflows*.

3.2. Taverna

Taverna¹³ é um SGWfC de código fonte livre desenvolvido pelo projeto myGrid¹⁴, que oferece um conjunto de ferramentas que auxiliam os cientistas a realizar os seus experimentos científicos. Inicialmente, o SGWfC Taverna foi concebido para a comunidade de ciências biológicas tendo como principal meta fazer o projeto e a execução dos *workflows* acessível para os bioinformáticos, os quais não necessariamente são especialistas em programação e serviços web.

O Taverna consiste de uma coleção de programas com dados e *links* de controle entre os mesmos. Esses programas são denominados no Taverna como processadores, podem ter múltiplas entradas e saídas, de modo que um *link* estabelece uma dependência entre a saída de um processador e a entrada de outro. Um *link* de controle indica que um processador apenas pode iniciar a sua execução depois que outro processador executou com sucesso a sua operação. Os processadores são implementados com classes Java caso a execução do experimento seja local, ou por meio de serviços web se a execução for remota. Esta última opção é a mais comum, tendo em vista que as portas de entrada e saída dos processadores correspondem às operações definidas na interface de serviço WSDL. Para realizar a execução de experimentos, o motor de execução escalona a invocação das operações de serviços e gerencia o fluxo de dados entre os processadores (TURI *et al.*, 2007) (OINN *et al.*, 2006).

Os fluxos de execução dos *workflows* no Taverna são escritos em uma linguagem de *workflow* conceitual simplificada chamada *Scufl* (OINN *et al.*, 2002), que é interpretada pelo motor de execução *Freeflow* (WROE *et al.*, 2007). A maioria dos dados de saída é representada na forma textual, entretanto, existe apoio para a utilização de *plugins* nas ferramentas de visualização. Este SGWfC é executado localmente nos computadores dos cientistas. Logo, é necessário que este seja instalado em cada computador que se deseja executar experimentos.

¹³ <http://www.mygrid.org.uk/tools/taverna/>

¹⁴ <http://www.mygrid.org.uk/>

3.3. Kepler

O SGWfC Kepler¹⁵ foi projetado para executar *workflows* em diversas áreas de pesquisa, tais como a biologia, ecologia, geologia, astrofísica e química. Ele possui uma interface gráfica para o projeto e a execução dos *workflows* com um paradigma de modelagem orientada a atores. Cada ator representa uma atividade do *workflow*. Os *workflows* do Kepler são representados em XML por meio da linguagem de modelagem de marcação conhecida como MoML (ALTINTAS *et al.*, 2004).

Algumas características adicionais do Kepler incluem: a criação de protótipos de *workflows*; execução distribuída; e, acesso e consulta a banco de dados. Para a criação de protótipos de *workflows*, o Kepler permite que os cientistas definam as atividades nestes *workflows* antes de implementar o código necessário para a execução (*workflow* abstrato). A execução distribuída pode utilizar como base os serviços web e em *Grid*. Para isso, os atores de serviço web e *Grid* permitem que os cientistas utilizem recursos computacionais disponíveis na Internet, tais como, conexões a outros laboratórios de pesquisa através de serviços web, e conexões a ambientes computacionais em *Grid*. O ator de serviço web oferece ao usuário a possibilidade de executar serviços definidos em interfaces WSDL. O acesso e consultas a banco de dados incluem atores que emitem sinais de conexão com a finalidade realizar consultas.

3.4. Biowep

O Biowep¹⁶ é um ambiente que apenas apóia a execução de *workflows*. É uma aplicação cliente baseada em portal web que permite a seleção e execução de um conjunto pré-definido de *workflows*. A arquitetura do ambiente inclui um gerenciador de *workflows*, uma interface de usuário e dois motores de execução de *workflows*. Os *workflows* executados no Biowep podem ser criados tanto pelo ambiente Taverna como também pelo BioWMS¹⁷ e executados respectivamente pelos motores de execução *FreeFluo* (ROMANO *et al.*, 2007) e *BioAgent/Hermes* (BARTOCCI *et al.*, 2007). A interface gráfica do Biowep oferece apoio à autenticação de usuários e à criação de perfis. Assim, os *workflows* podem ser selecionados por meio do repositório de *workflows* do perfil de um usuário, como também podem ser

¹⁵ <https://kepler-project.org/>

¹⁶ <http://bioinformatics.istge.it/biowep/>

¹⁷ <http://litbio.unicam.it:8080/biowms/>

pesquisados por meio de anotações que foram associadas a eles. Opcionalmente os resultados das execuções podem ser salvos para consultas futuras.

Para a execução dos *workflows*, uma página web é gerada com os artefatos de entrada do *workflow* e sugestões de possíveis valores para estes artefatos. No caso da execução de *workflows* com o motor de execução *Freefluo*, os resultados são armazenados no banco de dados do Biowep e disponibilizados aos usuários. Quando a execução de um *workflow* é longa, este é executado em segundo plano de modo que o usuário possa realizar outras operações no portal web e analisar os resultados posteriormente. Por outro lado, os resultados apenas retornam por e-mail se os *workflows* forem executados com o motor de execução do *BioWMS*.

3.5. ^{my}Experiment

O ^{my}Experiment¹⁸ é uma iniciativa do projeto ^{my}Grid para criar um ambiente de pesquisa virtual (STEVENS *et al.*, 2003), que permite que usuários possam compartilhar, reaproveitar, comentar e discutir experimentos sem levar em consideração qual o sistema de gerenciamento de *workflows* está sendo utilizado. Os cientistas que utilizam o ^{my}Experiment são capazes de trocar *workflows* e outros objetos científicos tão facilmente como as pessoas podem compartilhar documentos, fotos e vídeos na Internet, uma vez que o ^{my}Experiment se assemelha mais com os sites de redes sociais, tais como o *MySpace*¹⁹ e o *Youtube*²⁰, do que com os portais tradicionais de computação em *Grid*, sendo dessa forma mais familiar para a nova geração de cientistas (GOBLE *et al.*, 2007). O reuso de *workflows* está basicamente ligado ao desejo de que eles possam ser compartilhados pela comunidade como o protocolo científico de “melhor prática”, o qual pode ser reutilizado exatamente como projetado ou variando por meio de simples substituições de dados, parâmetros ou serviços equivalentes (WROE *et al.*, 2007).

3.6. Pegasus

O SGWfC Pegasus é capaz de mapear descrições estruturadas de *workflows* abstratos em alto nível para recursos capazes de serem executados em ambientes em *Grid* (LEE *et al.*, 2009). Assim, este SGWfC utiliza-se de *workflows* abstratos para descrever a topologia lógica e as

¹⁸ <http://www.myexperiment.org/>

¹⁹ <http://www.myspace.com/>

²⁰ <http://www.youtube.com/>

funcionalidades da aplicação e executa as atividades do *workflow* usando o sistema Condor-G. Este sistema é o responsável pelo gerenciamento das atividades capazes de alocar recursos, enviar, cancelar, bem como monitorar as atividades que estão sendo executadas na *Grid* em favor do usuário (FREY *et al.*, 2002) (KEE *et al.*, 2008).

Um *workflow* no ambiente Pegasus passa por três fases até a sua execução final, que são: a criação do *workflow*; o planejamento da execução do *workflow*; e, a execução propriamente dita. Na primeira fase, o *workflow* é criado de forma abstrata. Na segunda fase, o Pegasus realiza uma série de refinamentos para transformar o *workflow* abstrato em concreto de modo que este possa ser executado utilizando os recursos em *Grid*. Os resultados deste refinamento são armazenados na forma de um arquivo de mapeamento que serve como entrada e que pode ser interpretado pelo sistema Condor-DagMan. O Condor-DagMan é o motor de execução dos *workflows* responsável por escalonar, recuperar e relatar o resultado da execução do conjunto de atividades enviadas ao sistema Condor-G. Na terceira fase é realizada a execução dos *workflows*.

3.7. GPFlow

O SGWfC GPFlow permite a composição de atividades por meio de uma portal web. O modelo de *workflow* foi inspirado em planilhas eletrônicas permitindo que eles possam ser apresentados, instanciados de forma assíncrona de tal modo que o navegador web pode ser fechado e o *workflow* pode ser conferido novamente mais tarde. Os *workflows* também podem ser reiniciados e parados a qualquer momento. Assim, um *workflow* pode ser alterado enquanto estiver executando, de modo que, qualquer recálculo necessário será realizado automaticamente surtindo efeito por todo o *workflow*.

A arquitetura foi construída no topo do BizTalk²¹, o qual é um servidor de integração da Microsoft que utiliza WS-BPEL para descrever e executar seus *workflows*. Foi utilizado o componente de *workflow* dinâmico do BizTalk chamado de *Human Workflow Services* (HWS), o qual permite a orquestração de *workflows* que se adaptam aos requisitos dinâmicos que oferecem o recálculo automático das execuções. Seus *workflows* são gravados em arquivos XML e disponibilizados como modelos de *workflows* que podem ser reutilizados (RYGG *et al.*, 2006).

²¹ <http://www.microsoft.com/biztalk>

3.8. ActiveVOS

O ActiveVOS (ENDPOINTS, 2010b) é um SGWf utilizado para compor e executar *workflows* no meio empresarial e é uma evolução do SGWf ActiveBPEL (ENDPOINTS, 2010a). Possui uma interface gráfica para a definição dos *workflows* baseada no projeto da ferramenta de desenvolvimento Eclipse (ECLIPSE, 2010). Os *workflows* são criados a partir de componentes previamente definidos, os quais correspondem às atividades presentes na especificação WS-BPEL 2.0. Além disso, o ActiveVOS permite a definição de atividades humanas baseadas nas especificações BPEL4People e WS-HumanTask em seus *workflows* e às interpreta em seu motor de execução.

Este SGWf oferece uma interface gráfica voltada para a web para instanciar os *workflows* definidos. As atividades humanas definidas nos *workflows* também podem ser visualizadas e manipuladas através desta interface gráfica. Contudo, a sua ferramenta para a definição de *workflows* é executada localmente, não oferece meios de compartilhamento e reutilização destes, bem como necessita de licença comercial para o seu uso (ENDPOINTS, 2010b).

3.9. Oracle BPEL Process Manager

Este SGWf é utilizado para definir e executar *workflows* na área empresarial. Utiliza-se da especificação WS-BPEL para compor seus *workflows* e não existe forma de importação ou exportação explícita de código WS-BPEL. A ferramenta gráfica usada para a definição de *workflows* possui componentes que representam as atividades definidas na especificação da linguagem WS-BPEL, o que permite que os usuários não necessitem de conhecimentos da linguagem para definir os seus *workflows*. O gerenciamento das atividades humanas deste SGWf acontece por meio de um software proprietário desenvolvido especificamente para apoiar a execução de *workflows* destas atividades, o qual atualmente não é baseado nas linguagens BPEL4People e WS-HumanTask. Além disso, o SGWf oferece uma interface web que permite a instanciação e a visualização da execução dos *workflows*, bem como a interação dos usuários com as atividades humanas que devem ser realizadas.

O Oracle BPEL *Process Manager* é disponibilizado em duas versões. A versão não paga possui restrições de hardware para a execução de *workflows*, tais como o número de processadores e a quantidade de memória que podem ser utilizadas. A versão completa necessita de licença comercial para a utilização (ORACLE, 2010).

3.10. Intalio

O SGWf Intalio também visa apoiar o gerenciamento de *workflows* pertencentes a organizações empresariais. Sua interface gráfica para a definição de *workflows* é baseada na ferramenta de desenvolvimento Eclipse (ECLIPSE, 2010) e, por isso, necessita ser executada localmente. Ela não permite o compartilhamento e reuso de *workflows*. Os *workflows* definidos por meio desta interface gráfica são baseados na linguagem BPMN²² (OMG, 2010), que não é abordada nesta dissertação. Estes *workflows* são exportados para a linguagem WS-BPEL e são orquestrados pela máquina de execução de *workflows* Apache ODE (APACHE, 2010a). Contudo, esta máquina de execução de *workflows* não interpreta as especificações BPEL4People e WS-HumanTask. Dessa forma, este SGWf possui as principais funcionalidades destas especificações implementadas por meio de um *workflow* desenvolvido com a linguagem WS-BPEL. Assim, ao final da execução do *workflow* definido pelo usuário, existirá uma instância do *workflow* de gerenciamento de atividades humanas para cada atividade humana definida. As instâncias das atividades humanas são gerenciadas por meio de uma interface web que é apoiada pelo *framework* TEMPO (INTALIO, 2010b) (INTALIO, 2010a).

3.11. Considerações Finais

Neste capítulo foram apresentados os principais trabalhos relacionados ao projeto do ambiente *ClusterFlow*. Esses trabalhos tiveram influência no desenvolvimento deste trabalho de mestrado, uma vez que, foram realizados estudos e testes de utilização com a visão do usuário final em alguns deles, tais como, Taverna, Kepler, Biowep, ^{my}Experiment, ActiveVOS, Oracle BPEL *Process Manager* e Intalio, a fim de levantar as características que pudessem ser aproveitadas para o *ClusterFlow*. As deficiências que estes apresentam com relação ao ciclo de vida de um experimento científico e aos principais princípios para a construção de *workflows* apresentados no capítulo anterior também foram examinados.

²² Do inglês *Business Process Modeling Notation*

O Ambiente *ClusterFlow*

4.1. Considerações Iniciais

Este capítulo visa apresentar o projeto do ambiente *ClusterFlow* desenvolvido neste trabalho de mestrado. Inicialmente os princípios do ambiente são apresentados seguidos da descrição do cenário no qual o *ClusterFlow* está inserido e da arquitetura inicial do ambiente. O processo para conceber os *workflows* a partir do *ClusterFlow*, bem como a execução e o compartilhamento dos resultados dos *workflows* são apresentados na sequência. O processo proposto para apoiar o gerenciamento das instâncias de execução das atividades humanas também é descrito. O capítulo inclui a definição do mecanismo de proveniência de dados das instâncias dos *workflows*, além de descrever o relacionamento entre o modelo conceitual e a especificação WS-BPEL.

4.2. Princípios do Ambiente *ClusterFlow*

Para conceber o ambiente *ClusterFlow*, foram adotados os princípios que devem ser levados em consideração para a construção de *workflows* científicos, bem como foram analisados alguns dos SGWfs existentes descritos no capítulo anterior. Os principais princípios e sua importância para o contexto do ambiente *ClusterFlow* são explanados a seguir:

- *workflows* abstratos e concretos: o apoio à definição de *workflows* abstratos é opcional. No entanto, os SGWfs apresentados no capítulo anterior implementam esta funcionalidade para permitir que os cientistas representem inicialmente suas

definições de *workflow* sem se preocuparem com detalhes de execução, tais como, o mapeamento dos artefatos e a configuração da atividade a ser executada. No contexto do ambiente *ClusterFlow*, um *workflow* é considerado concreto a partir do momento em que os mapeamentos do fluxo de dados entre as atividades tanto de software como humanas estejam corretamente definidos, bem como as atividades de software estejam relacionadas com suas respectivas implementações computacionais. Dessa forma, cientistas que utilizarem o ambiente *ClusterFlow* poderão definir as hipóteses de seus experimentos de forma abstrata, o que permite que estes experimentos estejam documentados antes da implementação de suas atividades, caso estas venham a ter que ser desenvolvidas;

- execução de *workflows* científicos: SGWfCs como Taverna, Kepler e Pegasus utilizam linguagens próprias para a definição de seus *workflows*, o que gera uma dificuldade adicional para os cientistas, pois *workflows* criados por estes SGWfCs não podem ser traduzidos para outros SGWfCs que não tenham exatamente a mesma máquina de execução de *workflows*. Muitas vezes a dificuldade em abstrair a execução em determinada tecnologia, tal como as execuções em *Grid* que o Pegasus apóia, conduzem os desenvolvedores a criar a sua própria linguagem de definição de *workflow*. O SGWfC Taverna foi inicialmente projetado para interpretar a linguagem WS-BPEL (OINN *et al.*, 2002), porém na época esta linguagem era incipiente. O ambiente *ClusterFlow* foi desenvolvido com base na linguagem definida pela especificação WS-BPEL, que é uma linguagem padronizada internacionalmente e que pode ser executada em qualquer SGWf que implementa esta especificação. Isto permite que *workflows* criados pelos cientistas sejam executados e orquestrados automaticamente, o que elimina a necessidade de conduzir a realização de experimentos em diversos ambientes de hardware e software manualmente;
- apoio à execução de atividades humanas: os SGWfCs Taverna, Kepler, Biowep e Pegasus não oferecem apoio às atividades com interação humana. Por outro lado, os SGWf ActiveVOS, GPFlow, Oracle BPEL *Process Manager* e Intalio oferecem tal apoio. Os SGWf que oferecem este apoio são baseados na especificação WS-BPEL, pois suas máquinas de execução implementam tal linguagem. Entretanto, a linguagem WS-BPEL não apóia a interação humana com *workflows*. Assim, é necessária a utilização de outras especificações que são extensões da WS-BPEL. As especificações BPEL4People (OASIS, 2010a) e WS-HumanTask (OASIS, 2010b) oferecem este

apoio. No entanto, apenas a máquina de execução do SGWf ActiveVOS implementa essas extensões de WS-BPEL. Porém, o ActiveVOS é um SGWf comercial e muitas vezes não interessa aos laboratórios de pesquisa por elevar os custos de pesquisa. Desta forma, o motor de execução do ActiveVOS não foi utilizado no ambiente *ClusterFlow*. Os demais SGWfs possuem suas próprias implementações de interação com humanos, o que restringe os cientistas a utilizarem aqueles SGWfs. O ambiente Intalio se apoia no *framework* TEMPO (INTALIO, 2010b) para realizar essas interações e implementa um *workflow* para gerenciar as atividades humanas, porém, este *framework* está ligado à implementação da máquina de execução WS-BPEL Apache ODE (APACHE, 2010a). O ambiente *ClusterFlow* propõe uma solução baseada no SGWf Intalio para apoiar o gerenciamento das atividades humanas. No entanto, as dependências de classes e comandos específicos da máquina WS-BPEL Apache ODE não são utilizados, garantindo com isso, que se esta máquina WS-BPEL for descontinuada, a migração para outra exigirá que apenas os serviços web cliente presentes no *ClusterFlow* que se conectam ao Apache ODE sejam alterados para se comunicarem com a nova máquina WS-BPEL;

- proveniência de dados: os SGWfs apresentados nesta dissertação implementam proveniência de dados em nível de *workflow*. O ambiente *ClusterFlow* oferece este apoio em nível de atividades, o que permite que as mesmas possuam seus artefatos de entrada e saída rastreados no momento de sua execução. O método de proveniência adotado e que está detalhado na Seção 4.7 segue algumas características do modelo desenvolvido por (MARINHO *et al.*, 2009);
- compartilhamento e reuso de *workflows*: assim como a proveniência de dados, os SGWfC apresentados implementam de alguma forma o compartilhamento e reuso de atividades de *workflows*. Contudo, essas atividades são previamente definidas por desenvolvedores das aplicações de software. Este é o caso dos SGWfC Taverna, Kepler, Pegasus, ActiveVOS, Oracle BPEL *Process Manager* e Intalio. Nesses ambientes, não é possível que *workflows* definidos pelos próprios cientistas sejam compartilhados com outros pesquisadores. Para tal, necessitariam utilizar o ambiente de pesquisa virtual ^{my}Experiment. O ambiente *ClusterFlow* permite que, uma vez o serviço web que representa a atividade de software esteja adicionado ao repositório de serviços, todos os usuários do ambiente possam utilizá-lo. O mesmo acontece para as atividades humanas definidas no ambiente. Dessa forma, pesquisadores que utilizam o

ambiente *ClusterFlow* podem definir e compartilhar os seus *workflows* com os demais cientistas, e assim disseminar o conhecimento sobre determinado experimento com os demais pesquisadores, além de permitir que *workflows* e atividades sejam compartilhadas e reutilizadas em outros experimentos sem a necessidade de outros ambientes, tais como o ^{my}Experiment; e

- recursos compartilhados: para permitir que *workflows* de experimentos científicos sejam definidos com maior velocidade e diminuir os custos de pesquisa, o ambiente *ClusterFlow* atende o princípio de recursos compartilhados. Estes recursos são representados por meio de serviços web, os quais podem acessar, por exemplo, um *cluster* de computadores que é um recurso caro. Para contornar os obstáculos enfrentados na realização de experimentos que utilizam *clusters* de computadores, o ambiente *ClusterFlow* permite que as aplicações de alto desempenho sejam invocadas por meio destes serviços web, abstraindo os comandos necessários para a execução destas. Os serviços web são invocados por meio das atividades de software do *workflow* criado pelo usuário do ambiente *ClusterFlow*. SGWfCs como Taverna, Kepler e Pegasus exigem que o cientista possua conhecimentos em *clusters* ou nas tecnologias responsáveis para acessá-los, dificultando, desta maneira, que estes experimentos sejam executados de forma transparente ao pesquisador, de modo que apenas haja a preocupação com o fluxo dos dados do experimento definido no *workflow*, assim como acontece no ambiente *ClusterFlow*.

4.3. Cenário do Ambiente *ClusterFlow*

O ambiente *ClusterFlow* permite que os pesquisadores possam definir, manter, executar, compartilhar e reutilizar seus *workflows*. Ele oferece aos pesquisadores uma interface gráfica na qual eles podem gerenciar seus *workflows* sem a necessidade de se preocupar com as questões tecnológicas de implementação das atividades. Dessa forma, os pesquisadores apenas se preocupam com o próprio experimento que pretendem realizar. Para criar um *workflow*, o conjunto de atividades necessárias para alcançar o objetivo do experimento deve ser definido, bem como o respectivo fluxo de dados por meio de artefatos de entrada e saída. A Figura 3 mostra o cenário do ambiente *ClusterFlow*.

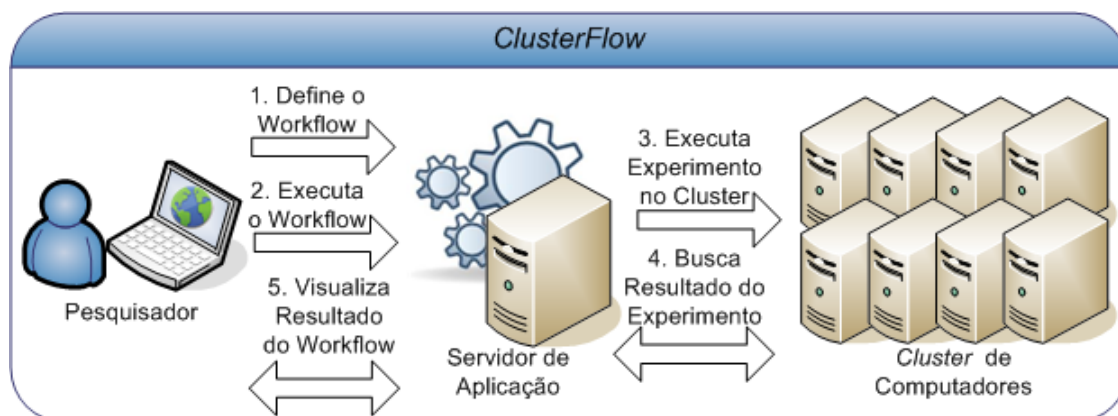


Figura 3. Cenário do ambiente ClusterFlow

Neste cenário, pode-se visualizar que o Pesquisador inicialmente define o seu *workflow* por meio de uma interface gráfica oferecida pelo Servidor de Aplicação do ambiente *ClusterFlow* e que pode ser acessada a partir de um navegador web (passo 1). Assim que o *workflow* estiver completamente definido, ele pode ser exportado pelo Servidor de Aplicação e implantado na máquina de execução de *workflows* que se encontra neste mesmo servidor. A instanciação dos *workflows* ocorre de modo assíncrono para que o navegador web possa ser fechado sem a paralisação do experimento (passo 2). A execução de um *workflow* pode ser acessada novamente a qualquer momento. As aplicações de alto desempenho localizadas no *Cluster* de Computadores são acessadas através de serviços web desenvolvidos por algum especialista na área de serviços web, e estes são disponibilizados no repositório de serviços do Servidor de Aplicação. Desta forma, durante a execução de um *workflow*, aplicações de alto desempenho no *Cluster* de Computadores são invocadas por meio destes serviços, abstraindo com isso os detalhes tecnológicos intrínsecos à execução de cada aplicação (passo 3). Os resultados das execuções destas aplicações são capturados assim que cada atividade do *workflow* encerrar a sua execução. Estes resultados são enviados ao mecanismo de proveniência de dados para armazenamento, que também está presente no Servidor de Aplicação (passo 4). O Pesquisador pode acompanhar a execução do seu *workflow* visualizando os resultados parciais de cada atividade, ou aguardar o encerramento da execução do *workflow* para obter e analisar todos os resultados da instância de execução (passo 5).

Com base neste cenário, nos princípios para a construção de *workflows* científicos e nos princípios para a construção do ambiente *ClusterFlow*, a arquitetura deste ambiente foi concebida para apoiar a realização de experimentos científicos principalmente para *clusters* de computadores.

4.4. Arquitetura do Ambiente

A Figura 4 apresenta a arquitetura do ambiente *ClusterFlow*. Os pacotes da arquitetura estão agrupados em três camadas lógicas: Pesquisador, Servidor de Aplicação e *Cluster*. As camadas são divididas pelos traços horizontais pontilhados. Na primeira camada está o Portal Web para Experimentos, que é acessado a partir dos computadores de pesquisadores. A segunda camada compreende os elementos da arquitetura que necessitam de um servidor de aplicação como base para a sua execução. Estes elementos podem ser executados tanto em um ou mais servidores de aplicação, embora exista dependência entre os elementos, estes podem ser executados separadamente em servidores de aplicação distintos. Além disso, encontra-se nesta camada a Máquina WS-BPEL responsável por orquestrar os *workflows* criados pelos cientistas. A terceira camada corresponde ao Cluster de Computadores, que possui as aplicações de alto desempenho representadas pelas atividades de software no *workflow* científico.

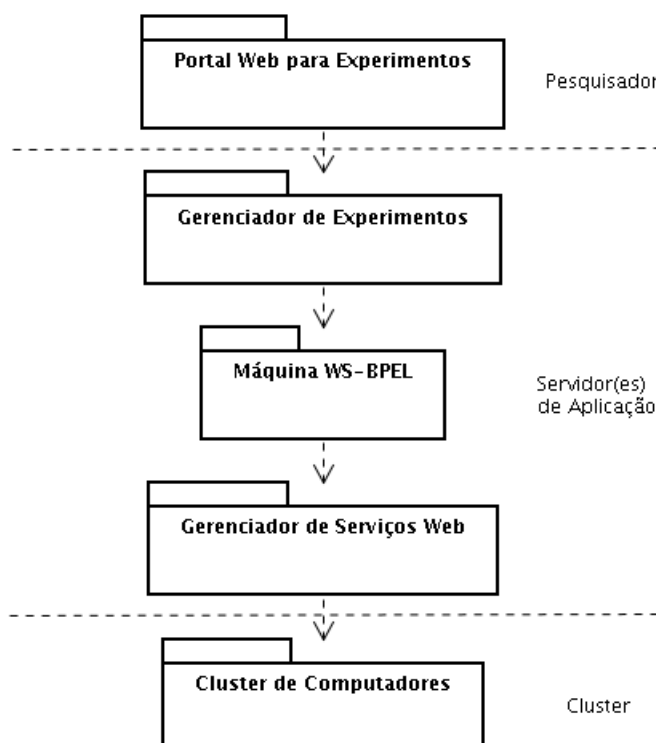


Figura 4. Arquitetura do ambiente *ClusterFlow*

Os pacotes definidos para a arquitetura do ambiente *ClusterFlow* são descritos nas próximas subseções.

4.4.1. Portal Web para Experimentos

O Portal Web para Experimentos é a fronteira entre os pesquisadores e o ambiente *ClusterFlow*. Os *workflows* são descritos e apresentados em uma estrutura de hipertexto inspirada no Manual de Processos do Instituto de Tecnologia de Massachusetts (*MIT Process Handbook*), no qual os *workflows* são estruturados por meio de uma árvore hierárquica. O portal web permite ao pesquisador consultar o estado de execução do seu *workflow* independentemente de sua localização. Do mesmo modo, o reuso de uma atividade ou *workflow* inteiro é oferecido por meio de uma interface gráfica na qual o pesquisador pode consultar e selecionar os *workflows* e suas respectivas atividades. O pesquisador pode fechar o seu navegador sem interromper a execução do *workflow* e posteriormente acessar o portal web para visualizar o andamento da execução do mesmo. O Portal Web para Experimentos possui duas interfaces gráficas com o usuário, conforme mostra a Figura 5.

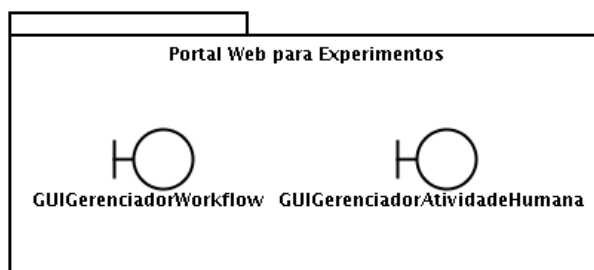


Figura 5. Subdivisão do Portal Web para Experimentos

A primeira interface gráfica permite o gerenciamento dos *workflows*, de forma que os cientistas possam criar, manter e reutilizar *workflows* e atividades com relação à fase de definição dos *workflows* científicos. A segunda interface gráfica é usada para o gerenciamento das instâncias das atividades humanas, que pode compreender: (a) visualização das atividades humanas escalonadas para o papel que o usuário exerce no *workflow*; (b) declaração de execução de atividades humanas; (c) liberação da execução de atividades humanas, que permite que o pesquisador deixe de realizar tarefas voltando-as para o estado inicial antes de assumi-las; e, (d) conclusão das atividades humanas informando seus dados de resultado.

4.4.2. Gerenciador de Experimentos

A Figura 6 mostra o pacote Gerenciador de Experimentos responsável pela lógica de negócio das operações oferecidas aos pesquisadores por meio do portal web. Este pacote permite que o pesquisador seja capaz de armazenar as informações para a criação e reuso dos

workflows em um banco de dados relacional que outros pesquisadores também tenham acesso. Entretanto, este acesso só é concedido caso o pesquisador tenha compartilhado os dados do *workflow*. A opção de exportação oferecida pelo portal fará com que o Gerenciador de Experimentos busque as informações do *workflow* no banco de dados e faça a tradução para a especificação WS-BPEL. A exportação é necessária em cada alteração do *workflow*, pois uma nova versão do arquivo da especificação WS-BPEL deve ser gerada para a execução da nova versão do experimento. O Gerenciador de Experimentos também é responsável por implantar os *workflows* na Máquina WS-BPEL.

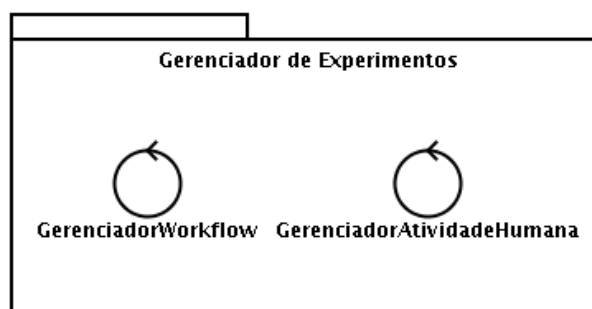


Figura 6. Subdivisão do Gerenciador de Experimentos

4.4.3. Máquina WS-BPEL

A Máquina WS-BPEL é um software responsável por interpretar e orquestrar os *workflows* criados pelos cientistas. Ela comunica-se com os serviços web enviando e recebendo mensagens e trata da manipulação dos dados e recuperação de falhas de acordo com as definições dos *workflows* criados pelos cientistas.

4.4.4. Gerenciador de Serviços Web

O pacote Gerenciador de Serviços Web representa os serviços necessários para realizar a invocação das aplicações de alto desempenho no *cluster*. Além disso, possui um repositório de serviços web utilizado para consultar os serviços disponíveis para conceber atividades de software. Os serviços web atuam como interfaces responsáveis pela comunicação com o *cluster*, o qual disparará a execução de aplicações de alto desempenho já existentes. Para cada aplicação contida no *cluster* são necessários um ou mais serviços web. Esses serviços não necessitam estar fisicamente implantados no Gerenciador de Serviços Web, contudo, as interfaces de descrição de serviço (WSDL) devem

obrigatoriamente estar disponíveis no mesmo. Essas interfaces são pesquisadas e recuperadas do repositório de serviços pelo Gerenciador de Experimentos, que disponibiliza os serviços no Portal Web para Experimentos para que o cientista possa escolher o mais apropriado para a realização da atividade de seu *workflow*.

4.4.5. Cluster de Computadores

O pacote Cluster de Computadores representa o *cluster* que é o responsável pela execução de aplicações computacionais que necessitem de alto poder de processamento. Estas aplicações de alto desempenho são executadas a partir dos serviços web que geralmente estão implantados no Gerenciador de Serviços Web do ambiente *ClusterFlow*. As aplicações que são disponibilizadas no *cluster* são desenvolvidas por cientistas que possuem conhecimento em *clusters* e aplicações de alto desempenho. Além disso, as aplicações devem estar prontas e disponíveis aos serviços web no momento da execução do *workflow*.

4.5. O Processo de Construção e Execução de Workflows Científicos

O ambiente *ClusterFlow* propõe um processo para que os cientistas criem, executem e compartilhem seus experimentos. Este processo é apresentado na Figura 7 por meio do diagrama de atividades. As atividades mostradas no diagrama se apóiam em diferentes pacotes da arquitetura definida para o ambiente *ClusterFlow*. No entanto, para não haver divergência com relação às atividades de um *workflow*, usaremos o termo “ação” para referenciar o diagrama de atividades. Assim, as ações que o cientista deve realizar no processo de construção e execução de *workflows* científicos é descrito a seguir.

4.5.1. Criar Workflow

A primeira ação que um cientista deve realizar é a Criar Workflow. Nesta ação, o cientista define somente as características referentes à atividade inicial do seu *workflow*, atribuindo um nome, o objetivo, a descrição geral do que o mesmo se propõe a realizar e os artefatos de entrada do *workflow*. A partir desta ação é definido o fluxo lógico do conjunto de atividades que compreendem o *workflow* científico que os pesquisadores pretendem construir. A Figura 8 mostra o detalhamento desta ação e os elementos da arquitetura envolvidos. O

diagrama desta figura está dividido em raias horizontais que representam os pacotes da arquitetura que estão presentes na realização da ação Criar Workflow. As raias verticais estão divididas em Pesquisador e ClusterFlow. A primeira representa as operações que os pesquisadores realizam com os *workflows* através do portal, e a segunda mostra as ações realizadas automaticamente pelo ambiente *ClusterFlow* em cada pacote da arquitetura. As figuras seguintes e semelhantes a esta seguem a mesma notação.

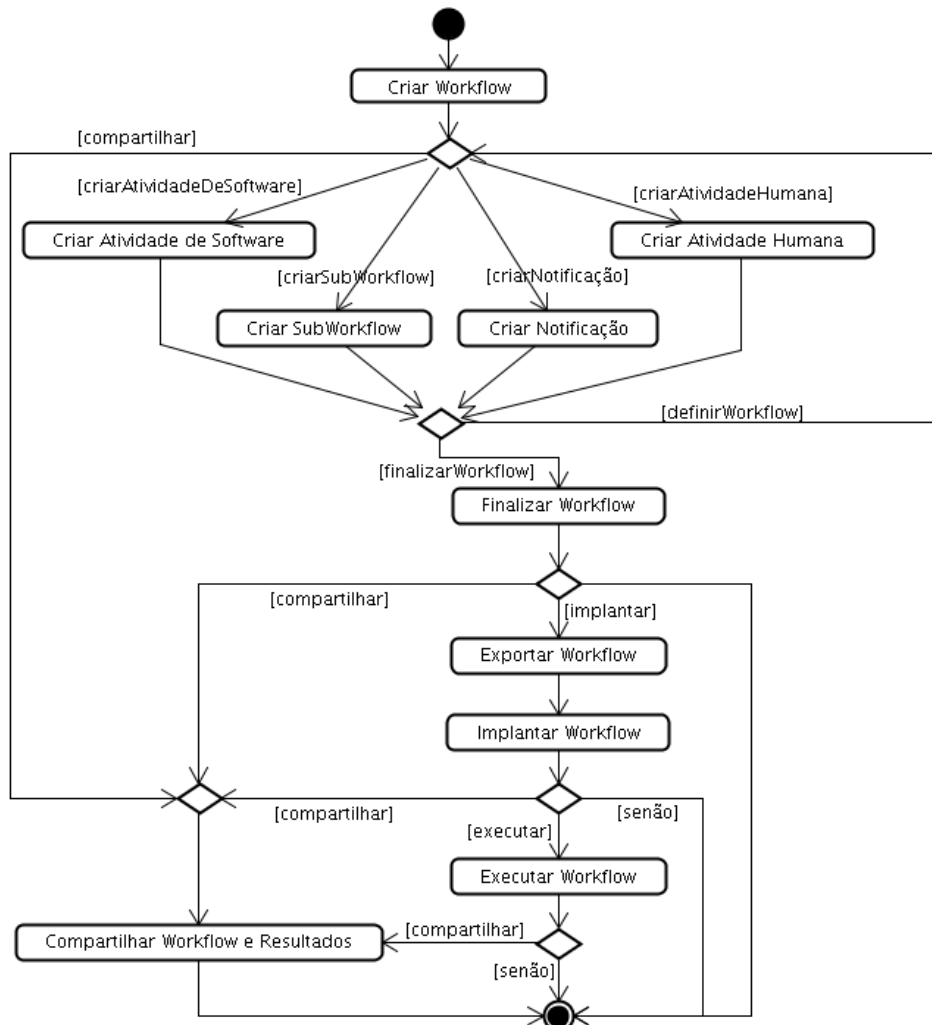


Figura 7. Diagrama de atividades do processo de construção e execução de workflows científicos

A ação Criar Workflow utiliza os pacotes Portal Web para Experimentos no qual os pesquisadores definem seus *workflows* e artefatos de entrada, bem como do Gerenciador de Experimentos, que realiza as validações necessárias e atualiza o banco de dados com as informações do novo *workflow* criado.

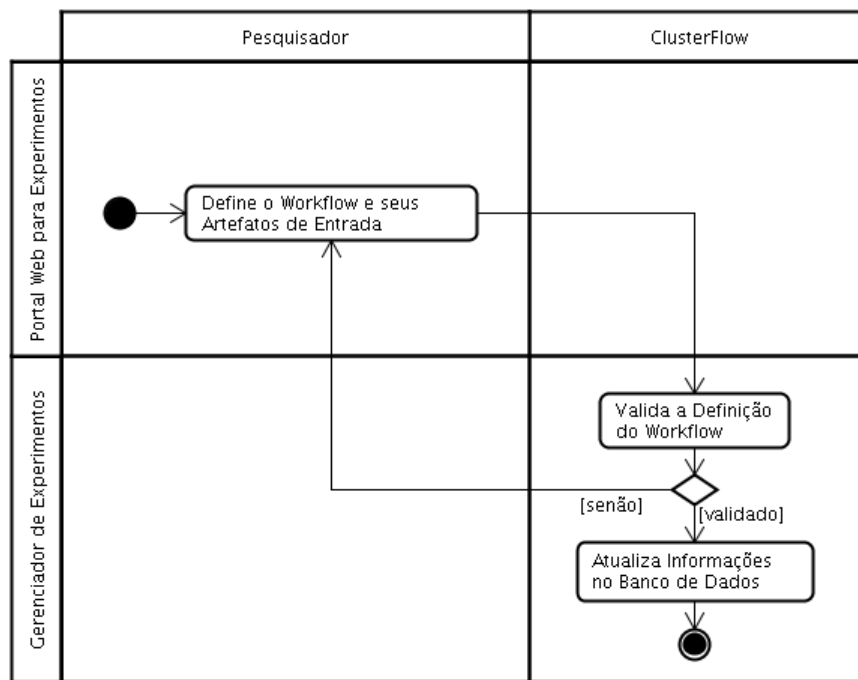


Figura 8. Ação Criar Workflow

Para dar continuidade à construção do *workflow* os cientistas devem definir a sequência em que as atividades devem ser realizadas para obter os resultados do experimento. Essas atividades podem compreender atividades de software, atividades humanas, *subworkflows* e notificações.

4.5.2. Criar Atividade de Software

As atividades computacionais executadas nos experimentos científicos são representadas por meio da ação Criar Atividade de Software. Este tipo de atividade define os serviços web que serão invocados e orquestrados no momento da execução do *workflow*, que por sua vez invocam as aplicações de alto desempenho localizadas no *cluster* de computadores. O detalhamento da ação e os pacotes da arquitetura presentes são apresentados na Figura 9.

Para criar uma atividade de software são necessários os elementos: Portal Web para Experimentos, para o pesquisador interagir com o ambiente; o Gerenciador de Experimentos, que busca atividades de software, artefatos, valida as informações da atividade e atualiza o banco de dados; e, o Gerenciador de Serviços Web, que tem seu repositório de serviços consultado.

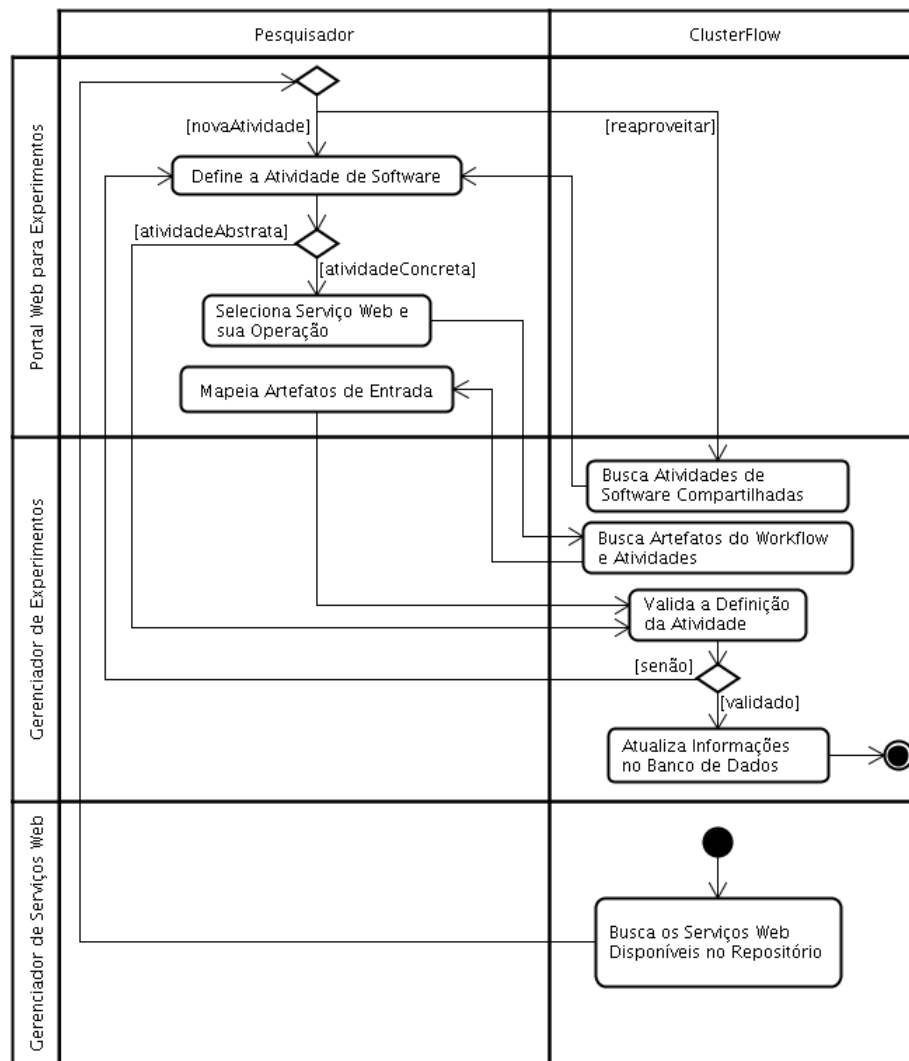


Figura 9. Ação Criar Atividade de Software

4.5.3. Criar Atividade Humana

A definição das atividades humanas é representada por meio da ação Criar Atividade Humana, a qual permite que pesquisadores possam definir as instruções que os possíveis proprietários das atividades humanas devem realizar. Além disso, permite que os cientistas definam o conjunto de artefatos que está sendo esperado como resultado da mesma. As definições desta atividade serão enviadas à lista de atividades humanas dos seus possíveis proprietários no momento da sua execução. A Figura 10 detalha a ação para criar uma atividade humana.

Nesta figura, o pesquisador pode escolher se deseja reutilizar uma atividade humana definida em outro *workflow* científico que está compartilhado. Para reutilizar a atividade humana, o ambiente *ClusterFlow* faz a busca de todos os *workflows* que contenham atividades

humanas e às apresenta ao pesquisador agrupadas por *workflow*. Por outro lado, o cientista pode definir uma nova atividade humana. Para isto, ele a descreve, define seus artefatos, prazos e cria os papéis dos possíveis proprietários que serão comunicados no momento de sua execução. Além disso, o cientista mapeia os artefatos de entrada da atividade com os artefatos de saída de atividades anteriores. Os artefatos de saída são buscados por meio do Gerenciador de Experimentos. Quando o cientista submeter a definição da atividade ao *ClusterFlow*, este realiza a sua validação e atualiza o banco de dados caso a definição estiver correta.

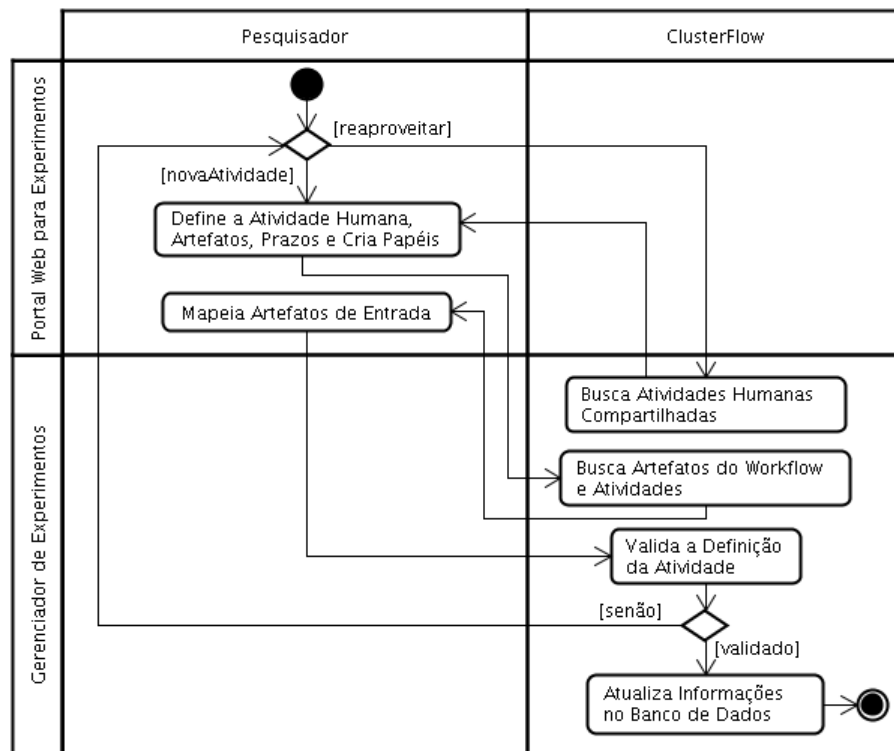


Figura 10. Ação Criar Atividade Humana

4.5.4. Criar SubWorkflow

A ação Criar SubWorkflow permite que cientistas possam definir com base no reuso, que uma atividade do *workflow* que está sendo criado represente um *workflow* já definido. Desta forma, esta atividade do *workflow* é capaz de encapsular outro *workflow* e permitir que o cientista não se preocupe com as atividades internas do mesmo. A Figura 11 apresenta os pacotes da arquitetura presentes que apóiam esta funcionalidade no ambiente *ClusterFlow*. Esta ilustração mostra o fluxo de atividades executadas pelo pesquisador e o *ClusterFlow* para a busca e o reuso de um *workflow* compartilhado.

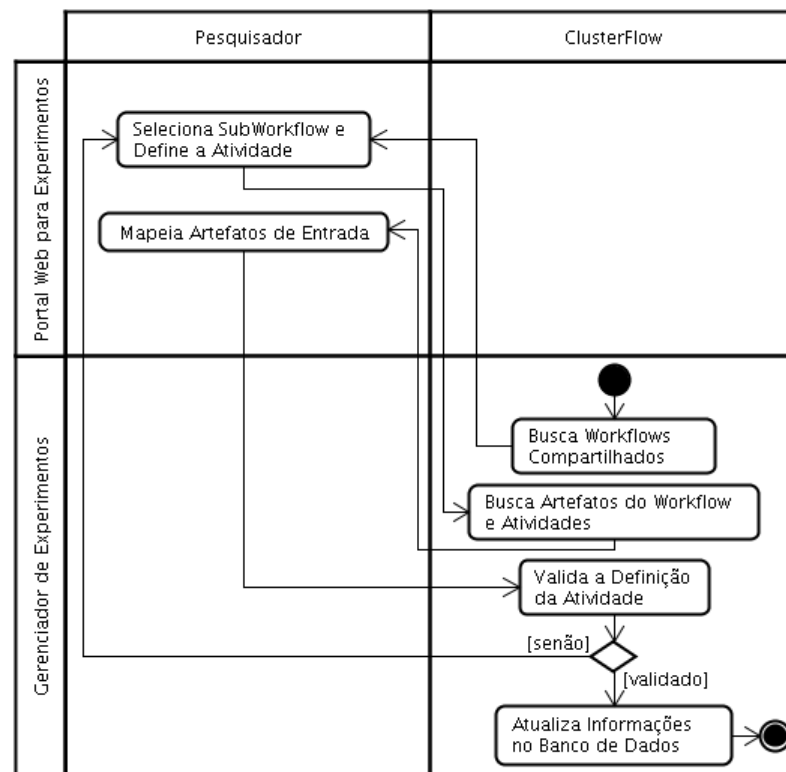


Figura 11. Ação Criar SubWorkflow

Para criar um *subworkflow*, o pesquisador deve selecionar o *workflow* que será reutilizado. Logo, o *ClusterFlow* busca as definições do *workflow* selecionado e importa para a ação Criar SubWorkflow, que é semelhante à criar uma atividade de software. Em seguida, o pesquisador mapeia os artefatos de entrada do *subworkflow* com os artefatos de saídas das atividades anteriores. Por fim, o *ClusterFlow* realiza as validações necessárias e grava a definição do *subworkflow* no banco de dados relacional.

4.5.5. Criar Notificação

A ação Criar Notificação permite que cientistas criem notificações que são enviadas aos pesquisadores no momento da execução do *workflow*. Estas podem compreender notificações estáticas que apresentam apenas informações definidas pelo próprio cientista, como também notificações dinâmicas, que informam resultados parciais do *workflow* ao término da execução das atividades desejadas. A Figura 12 detalha esta ação.

Para definir as notificações lançadas pelo *workflow* científico, o pesquisador define um título e a descrição para a mesma, bem como seleciona os papéis dos pesquisadores que receberão esta notificação. Caso o cientista deseje criar uma notificação dinâmica para informar o resultado de uma execução de atividade, ele também deve selecionar a atividade

desejada. Em seguida, os dados são validados e armazenados no banco de dados pelo *ClusterFlow*.

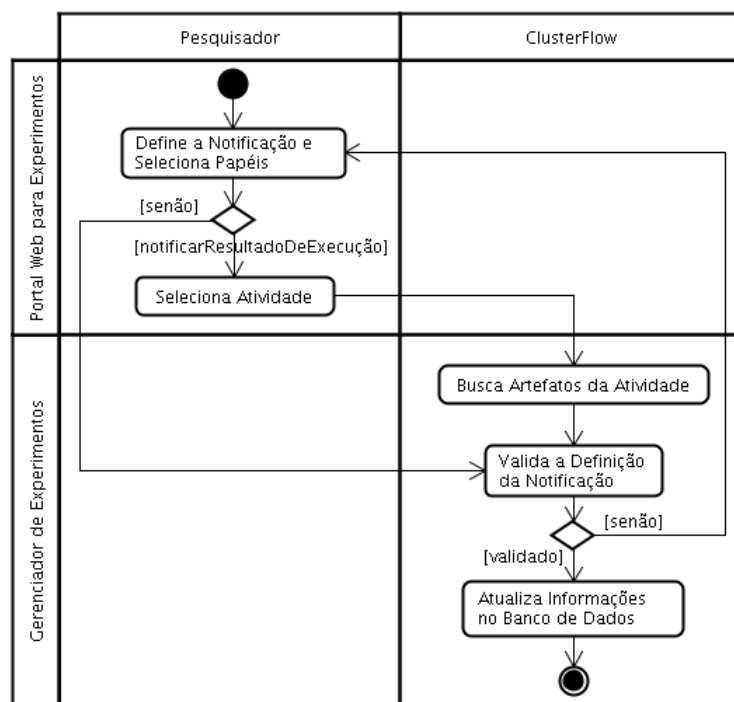


Figura 12. Ação Criar Notificação

4.5.6. Finalizar Workflow

A ação *Finalizar Workflow* representa o ponto de encerramento do *workflow*. Nesta ação, o cientista pode definir quais artefatos serão apresentados como resultado do experimento que foi realizado. Esta ação corresponde à última atividade a ser definida no *workflow*, pois ela depende dos dados de saída de atividades anteriores para que seus artefatos sejam mapeados. A Figura 13 apresenta o seu detalhamento e os pacotes da arquitetura envolvidos.

Logo após a descrição da finalização do *workflow* e a definição dos seus artefatos de resultado, os artefatos de saída das atividades são buscados pelo ambiente automaticamente. Isto permite que o cientista possa mapear os artefatos de saída das atividades para os artefatos de saída (resultado) do *workflow*. Ao final do mapeamento, a atividade é validada e a definição é armazenada em banco de dados pelo *ClusterFlow*.

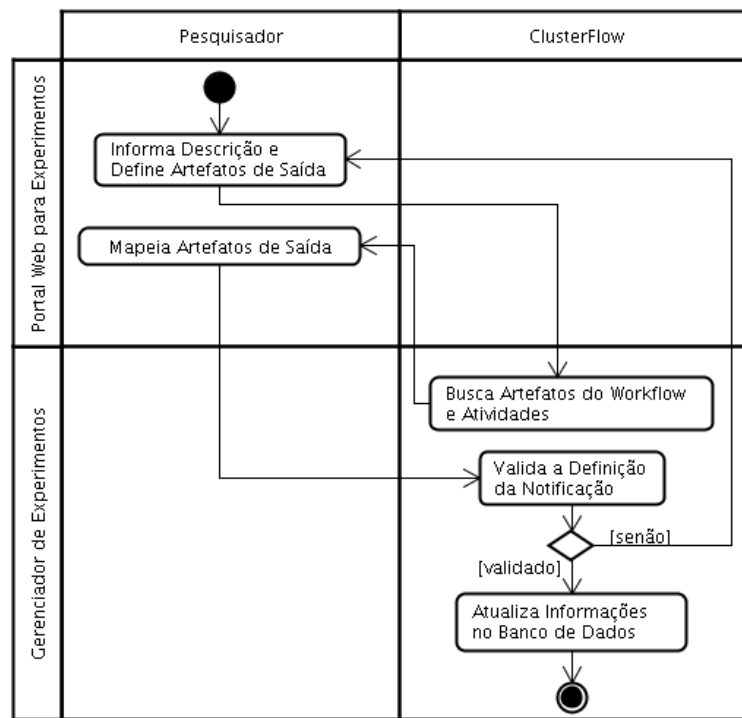


Figura 13. Ação Finalizar Workflow

4.5.7. Compartilhar Workflow e Resultados

O cientista pode optar por compartilhar seus *workflows* com outros pesquisadores utilizando a ação Compartilhar Workflow e Resultados, conforme mostra a Figura 14.

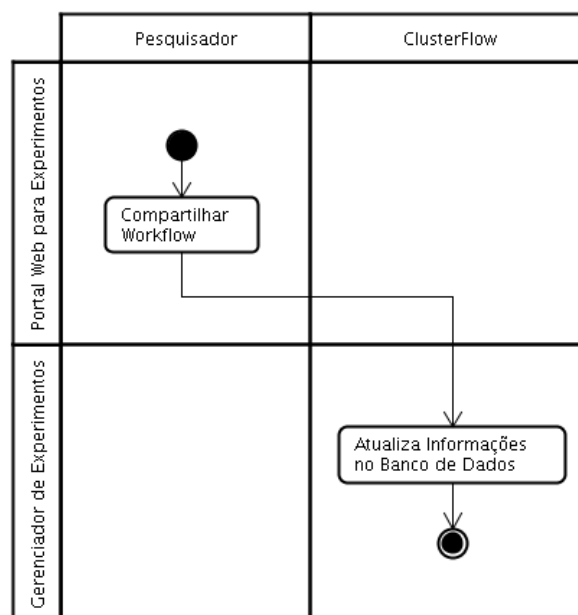


Figura 14. Ação Compartilhar Workflow e Resultados

Nesta ação, além de o pesquisador compartilhar os *workflows*, ele também compartilha os resultados dos experimentos. Para isto, o cientista seleciona a opção de compartilhamento e o ambiente *ClusterFlow* atualiza as informações no banco de dados relacional. Caso o pesquisador tenha a intenção de retirar o compartilhamento, esta opção deve ser desmarcada no portal web. Além disso, a ação de compartilhamento do experimento pode ser realizada antes ou depois da execução o *workflow*.

4.5.8. Exportar *Workflow* e Implantar *Workflow*

Após definir as atividades do *workflow*, o cientista pode exportar e implantar o *workflow* na Máquina WS-BPEL para a execução. Entretanto, a exportação somente é permitida para *workflows* concretos. Assim, a exportação e a implantação dos *workflows* são representadas, respectivamente, pelas ações Exportar *Workflow* e Implantar *Workflow*. Embora essas atividades apareçam separadas no diagrama que define o processo de construção de *workflows* científicos, apresentado na Figura 7, elas são executadas com apenas uma interação do cientista no Portal Web para Experimentos. Os pacotes da arquitetura usados para realizar as ações Exportar *Workflow* e Implantar *Workflow* são mostrados, respectivamente, pelas Figura 15 e Figura 16.

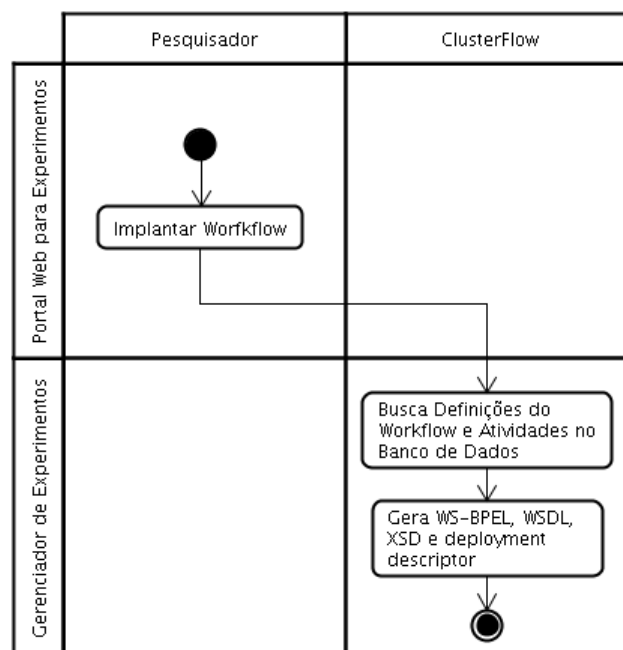


Figura 15. Ação Exportar *Workflow*

Nesta ação, o pesquisador seleciona a operação de implantação do *workflow* e o ambiente *ClusterFlow* se encarrega de buscar as definições do *workflow* selecionado e de suas

atividades. Com esses dados, o Gerenciador de Experimentos do *ClusterFlow* gera os arquivos que representam o *workflow* científico e um arquivo descritor de serviços para implantação na Máquina WS-BPEL. Os pacotes da arquitetura que estão envolvidos na realização desta ação são o Portal Web para Experimentos e o Gerenciador de Experimentos.

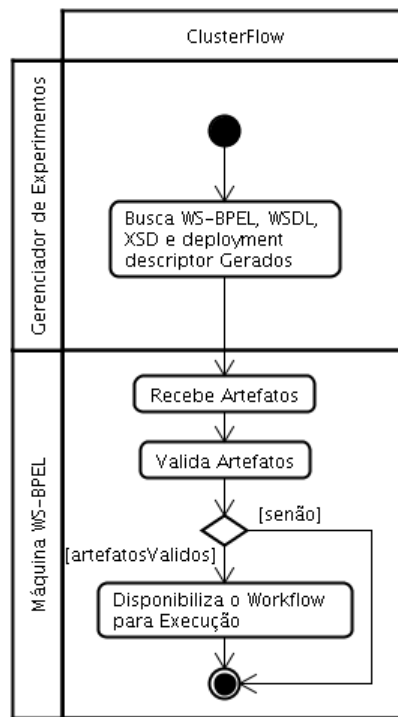


Figura 16. Ação Implantar Workflow

A ação Implantar Workflow é apoiada pelos pacotes Gerenciador de Experimentos e Máquina WS-BPEL. Quando a exportação do *workflow* é encerrada, o Gerenciador de Experimentos busca os artefatos do *workflow* gerados pela ação anterior, e os envia para a Máquina WS-BPEL por meio de um serviço web. Esta, por sua vez, recebe os artefatos, faz a validação destes e disponibiliza o *workflow* para a execução na forma de um serviço web composto.

4.5.9. Executar Workflow

A última atividade do processo é a execução do *workflow* na Máquina WS-BPEL e está representada no diagrama por meio da ação Executar Workflow. Esta ação é responsável por instanciar o *workflow* a partir do Portal Web para Experimentos, conforme mostra a Figura 17. O ciclo de execução dos *workflows* também está ilustrado nesta figura.

Assim, pode-se perceber o relacionamento dos pacotes da arquitetura com as ações realizadas durante o processo de orquestração dos *workflows*.

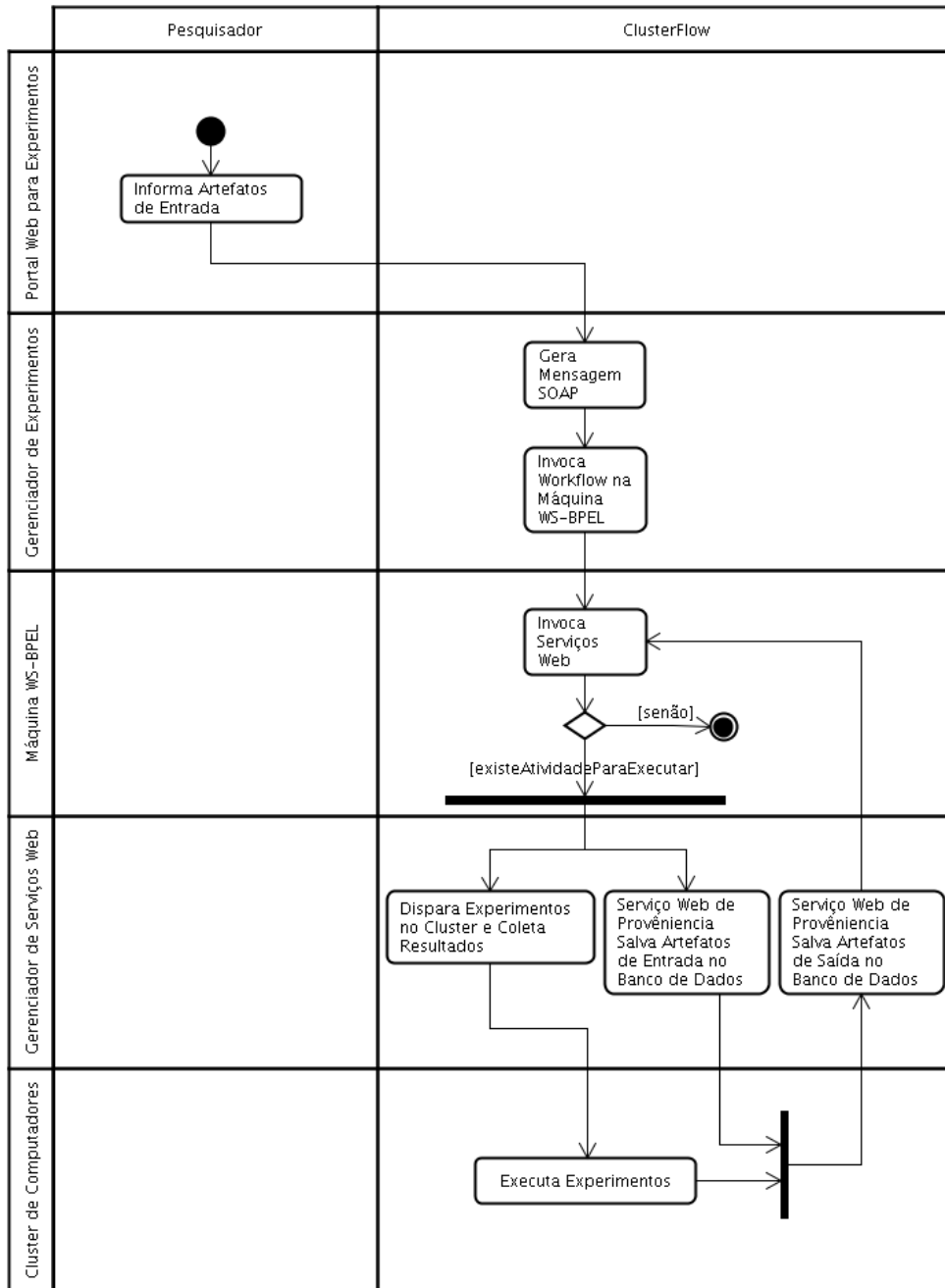


Figura 17. Ação Executar Workflow

A ação Executar Workflow se apoia em todos os pacotes da arquitetura do ambiente. Para instanciar o *workflow* e informar os seus artefatos de entrada o pacote Portal Web para Experimentos é utilizado. Quando o cientista disparar a execução do *workflow* no portal, o Gerenciador de Experimentos gera uma mensagem SOAP a partir dos artefatos de entrada e invoca o *workflow* na Máquina WS-BPEL para iniciar a

sua execução. Uma vez o *workflow* em execução, a Máquina WS-BPEL executa as atividades e invoca os serviços web definidos para cada uma delas até que todas sejam executadas. Os serviços web invocados ou pelo menos a sua definição em WSDL estão presentes no Gerenciador de Serviços Web. As aplicações de alto desempenho são disparadas no *cluster* paralelamente à proveniência dos artefatos de entrada da atividade. A proveniência de saída somente é invocada pela Máquina WS-BPEL após o término da execução das ações paralelizadas. Desta forma, encerra-se o processo para a construção e execução de *workflows* científicos no ambiente *ClusterFlow*.

Para que a execução dos *workflows* com as aplicações contidas no *cluster* e o apoio às atividades realizadas por humanos possam ser realizadas em um mesmo *workflow*, a proposta de um processo capaz de gerenciar as instâncias das atividades humanas foi imprescindível. O processo proposto para o gerenciamento das atividades humanas foi concebido para dar apoio à interação dos pesquisadores com as execuções dos *workflows* científicos (o que é muito comum em experimentos científicos), pois a máquina de execução de *workflows* utilizada neste trabalho não oferece tal apoio de forma nativa. Este processo está descrito na seção seguinte.

4.6. O Gerenciador de Atividades Humanas

TaskManagerProcess

Devido ao fato da maioria das máquinas WS-BPEL bem como da utilizada neste trabalho não oferecerem apoio à execução de atividades realizadas por humanos, foi necessário propor um processo capaz de gerenciar as instâncias de atividades humanas. Dentre os SGWf baseados na especificação WS-BPEL pesquisados, apenas o ActiveVOS (ENDPOINTS, 2010b) implementa e interpreta explicitamente as especificações BPEL4People (OASIS, 2010a) e WS-HumanTask (OASIS, 2010b) no *workflow* criado. Os demais SGWf baseados em WS-BPEL utilizam-se de suas próprias implementações para manipular as atividades humanas e não seguem os padrões das especificações BPEL4People e WS-HumanTask. Dessa forma, o *TaskManagerProcess* foi concebido na forma de um *workflow* baseado na especificação WS-BPEL, o que permite que todos os SGWfs que implementam esta especificação possam tanto orquestrar os *workflows* criados pelos cientistas, como também gerenciar a demanda das instâncias de atividades humanas que muitos *workflows* científicos possuem.

Atualmente, existem pesquisas que buscam resolver este problema. Uma delas é o projeto Apache Hise (APACHE, 2010), que visa a implementação da especificação WS-HumanTask utilizada para definir as características de uma atividade humana, além de apoiar a implementação de um motor de execução que corresponde à implementação do serviço de interações humanas. Este serviço faz a leitura das definições das atividades humanas providas em arquivo XML baseado na especificação WS-HumanTask, bem como as distribui aos seus proprietários baseando-se nos papéis atribuídos a eles. Contudo, este projeto não foi utilizado neste trabalho de mestrado, pois ainda está em fase de incubação.

O gerenciador das atividades humanas projetado para o ambiente *ClusterFlow* foi projetado a partir de ideias da implementação do *framework* Tempo (INTALIO, 2010b), que é um projeto relacionado ao SGWf Intalio (INTALIO, 2010a). Contudo, não foi possível a utilização deste *framework* devido ao seu acoplamento com a Máquina WS-BPEL Apache ODE (Orchestration Director Engine) (APACHE, 2010a), pois o funcionamento do *framework* depende da implementação desta máquina de orquestração de *workflows*.

A Figura 18 mostra o diagrama de sequência que representa a instanciação de um *workflow* e o ciclo de vida de uma atividade humana no mesmo. Este diagrama também apresenta as tecnologias e os principais métodos que podem ser usados para apoiar esta característica do ambiente *ClusterFlow*.

Nesta figura, pode-se perceber que ao encontrar uma atividade humana no *workflow* científico, a Máquina WS-BPEL invoca de maneira assíncrona o *workflow* TaskManagerProcess, que é responsável por criar e gerenciar as instâncias das atividades humanas. Logo que a nova atividade humana é criada, o *workflow* TaskManagerProcess invoca o serviço web TaskManagementService para salvar a nova atividade criada no banco de dados. Depois que esta foi persistida no banco, os proprietários podem recuperar suas informações a partir do serviço web TaskManagementService. As operações do TaskManagementService para declarar que uma atividade humana será executada por um cientista e a de liberar uma atividade para que outros cientistas possam vir a executá-la, também são invocadas pelo TaskManagerProcess. Assim que o cientista terminar de realizar a sua atividade, o resultado é enviado ao *workflow* científico para continuar a execução do experimento. O envio do resultado para o TaskManagerProcess é realizado por meio de uma operação oferecida no portal web. Em seguida, o TaskManagerProcess faz uma chamada assíncrona de retorno ao *workflow* científico e repassa o resultado da atividade humana.

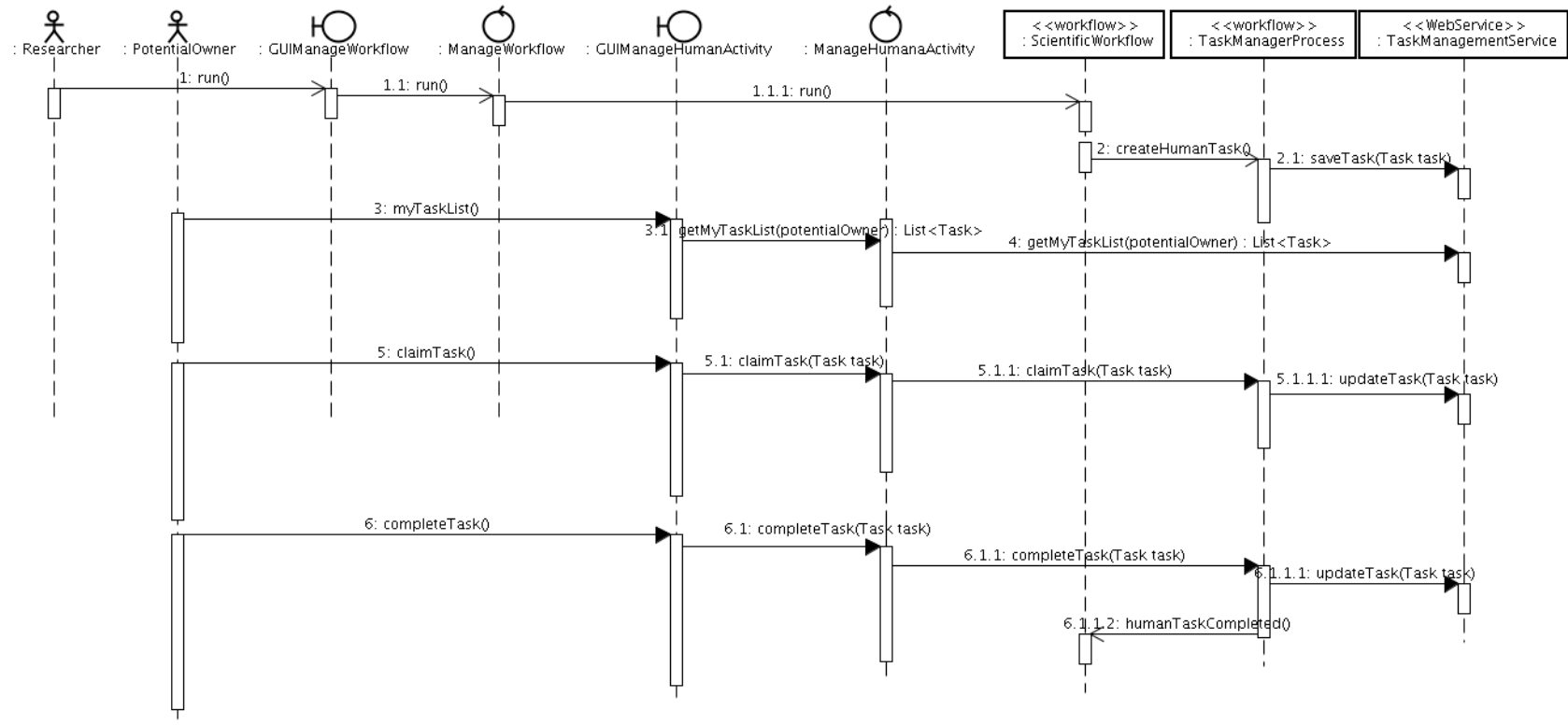


Figura 18. Instanciação de um workflow e o ciclo de vida de uma atividade humana

Nas próximas subseções é apresentado o processo proposto para o gerenciamento das instâncias das atividades humanas no ambiente *ClusterFlow*. Este processo segue a nomenclatura da especificação WS-BPEL, pois o mesmo pode ser concebido na forma de *workflow*. Assim, o diagrama de atividades do processo proposto para o gerenciamento das instâncias de atividades humanas foi dividido em partes com o objetivo de facilitar a compreensão do mesmo. Diversas atividades de manipulações de variáveis foram omitidas para simplificar as ilustrações. O Apêndice B mostra o diagrama completo.

4.6.1. Instanciação e Criação das Atividades Humanas

Para cada interação humana de um *workflow* científico é necessária a criação de uma instância do *workflow* de gerenciamento de atividades humanas, o `TaskManagerProcess`. Desta forma, se um *workflow* científico possuir, por exemplo, 5 (cinco) atividades humanas, ao final de sua execução foram criadas 5 (cinco) instâncias do `TaskManagerProcess`.

Ao encontrar uma atividade humana na definição do *workflow* científico, o Gerenciador de Experimentos exporta a chamada desta atividade como uma invocação de serviço web. O *workflow* `TaskManagerProcess` depois de implantado corresponde a um serviço web composto disponibilizado pela Máquina WS-BPEL. Assim, as chamadas às atividades humanas são realizadas por meio da invocação do `TaskManagerProcess`.

A Figura 19 mostra de forma resumida apenas as ações necessárias para criar a instância de uma atividade humana. O termo “ações” corresponde às atividades do diagrama apresentado. A notação do diagrama está em inglês, pois segue a nomenclatura da linguagem WS-BPEL, o que facilita a compreensão de cada atividade presente no `TaskManagerProcess`. As figuras seguintes seguem o mesmo padrão. Assim, a ação `While !Completed` aparece ao final desta ilustração com o intuito de apresentar uma visão geral do *workflow* e mostrar a finalização do mesmo. Contudo, ela empacota diversas atividades WS-BPEL responsáveis por manipular a atividade humana do *workflow* científico, que são explicadas nas subseções seguintes.

A ação `Receive CreateTask` permanece aguardando uma invocação de serviço para dar início à execução do *workflow* e com isso criar uma nova instância dele. Após o *workflow* instanciado, a ação `Assign CreateTask` cria a atividade humana em nível de

workflow, pois apenas copia os parâmetros de entrada recebidos por `Receive CreateTask` para as variáveis pertencentes à atividade humana.

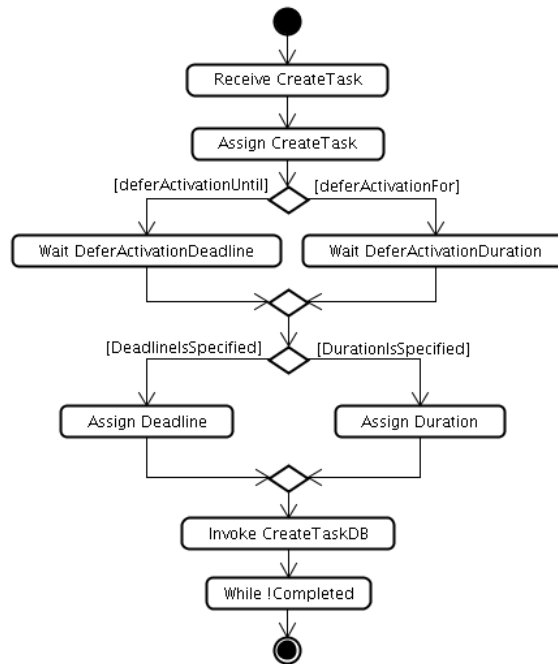


Figura 19. Instanciação e criação das atividades humanas

Cientistas podem necessitar que atividades humanas apenas sejam criadas e executadas a partir de determinado momento ou data. Para que isto ocorra, eles devem informar os valores deste parâmetro no momento da definição da atividade humana. Desta forma, o *workflow* científico repassa o parâmetro para o *workflow* `TaskManagerProcess` e este faz a verificação para constatar se o parâmetro informado satisfaz uma data em específico ou se corresponde a um tempo limite. As condições de verificação podem ser `deferActivationUntil`, caso o cientista tenha informado uma data e hora em específico ou `deferActivationFor`, no caso da ação ter que aguardar um determinado tempo para continuar a sua execução. Assim, a ação `Wait DeferActivationDeadline` aguarda até que a data e hora limites sejam atingidas e `Wait DeferActivationDuration` aguarda o encerramento do tempo para continuar com a execução. Contudo, de acordo com a especificação WS-BPEL, apenas é possível informar uma destas duas condições.

Da mesma forma, atividades humanas podem ter uma data ou tempo limite para o encerramento de execução. Igualmente às condições de ativação, os prazos de conclusão são informados no momento da definição da atividade humana e são repassados ao `TaskManagerProcess` em tempo de execução. As condições `DeadlineIsSpecified`

e `DurationIsSpecified` são utilizadas, respectivamente, para verificar se uma data limite foi definida para que a ação seja encerrada e verificar se um tempo limite de finalização foi configurado. As ações `Assign Deadline` e `Assign Duration` realizam os cálculos baseados nos prazos informados pelo cientista durante o processo de criação da atividade humana para o *workflow* científico. Apenas uma das duas formas do prazo de encerramento da atividade humana pode ser informada.

A atividade humana neste momento está armazenada apenas nas variáveis de instância do *workflow* e ainda não foi armazenada na base de dados do ambiente *ClusterFlow*, o que não permite que a mesma seja visualizada pelos possíveis proprietários encarregados de realizá-la. Desta forma, a ação `Invoke CreateTaskDB` invoca o serviço `web TaskManagementService` e os dados de entrada da atividade humana são armazenados no banco de dados. Assim, ela torna-se disponível aos possíveis donos para a declaração de posse e início da realização da mesma.

4.6.2. Condição para Encerrar Instâncias de Atividades Humanas

A Figura 20 mostra a ação `While !Completed` que é responsável por verificar a condição de encerramento das instâncias das atividades humanas.

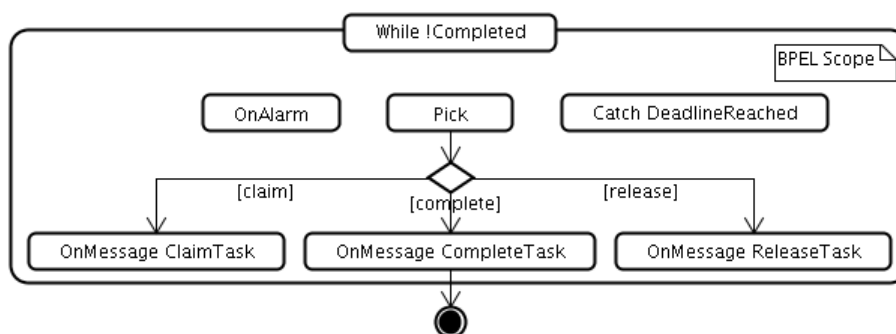


Figura 20. Ação *While* para verificar a condição de encerramento das atividades humanas

A ação `While` possui um escopo BPEL (`BPEL Scope`) para permitir que o *workflow* possa lançar uma exceção pelo evento `OnAlarm`, de forma que esta possa ser capturada pela ação `Catch`. A utilização da ação `Pick` permite que o *workflow* aguarde o recebimento de mensagens específicas. De acordo com a especificação WS-BPEL (OASIS, 2007), os eventos `OnMessage` da atividade `Pick` em WS-BPEL podem ser relacionados às atividades do tipo `Receive`, pois são atividades que permanecem bloqueadas e não serão completadas enquanto uma mensagem adequada não for recebida pela instância do *workflow*.

Usamos o termo atividade na frase anterior, pois a especificação WS-BPEL trata-as como atividades de *workflow*. Entretanto, o recebimento de uma mensagem adequada pela ação *Pick* não é o suficiente para encerrar a execução da atividade humana. É necessário que o estado da atividade no *TaskManagerProcess* seja alterado para *Completed*, caso contrário a mesma continua sendo executada. Isto é possível por meio da utilização da ação *While* que não permite a passagem para a ação seguinte sem a atividade humana ter alcançado o estado *Completed*. No caso do *workflow TaskManagerProcess*, a próxima ação a ser executada é o encerramento da atividade humana, permitindo com isso que o *workflow* científico criado pelo pesquisador continue o seu fluxo de execução.

Foram definidas três operações imprescindíveis para a manipulação de atividades humanas por meio dos eventos *OnMessage*. Estas operações foram baseadas em definições que estão presentes nas especificações *BPEL4People* e *WS-HumanTask*. As operações são as seguintes: (i) declaração de propriedade para a realização de atividades humanas (*claimTask*); (ii) a operação para completar uma atividade humana, retornar os dados e a execução para o *workflow* científico (*completeTask*); e, (iii) a liberação de uma atividade humana (*releaseTask*).

4.6.3. Declaração de Propriedade das Atividades Humanas

A declaração de posse de atividade permite que somente um dos possíveis proprietários tenha a capacidade de realizar a mesma. Esta declaração é apoiada no *TaskManagerProcess* pelo evento *OnMessage ClaimTask* da ação *Pick*, conforme é mostrado na Figura 21.

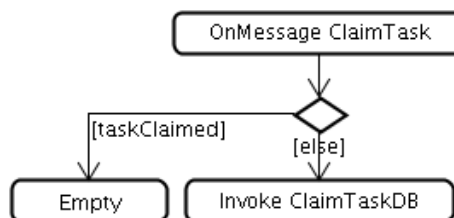


Figura 21. Declaração de propriedade das atividades humanas

No início da operação *claimTask*, o estado da atividade humana é consultado para verificar se a mesma já foi declarada que seria executada por outro pesquisador. Caso esteja declarada, a atividade *Empty* da especificação WS-BPEL é executada. Esta atividade pode ser utilizada quando existe a necessidade de não realizar algum tipo de ação, como exemplo, pode-se usar uma atividade *Empty* quando uma falha ou exceção precisa ser capturada e

omitida (OASIS, 2007). Contudo, se a atividade ainda não foi declarada ela passa a ser processada. Desta forma, o estado da atividade humana é alterado para `Claimed` e a ação `Invoke ClaimTaskDB` entra em execução, que por sua vez, invoca o serviço web `TaskManagementService` para as alterações no banco de dados. Com isso, na próxima consulta às atividades humanas a serem realizadas a atividade alterada apenas será visualizada pelo proprietário que declarou a posse da mesma.

4.6.4. Liberação de Propriedade das Atividades Humanas

A liberação de posse das atividades humanas permite que os possíveis proprietários das atividades possam consultar e visualizá-las novamente. Este processo de liberação das atividades é mostrado na Figura 22.

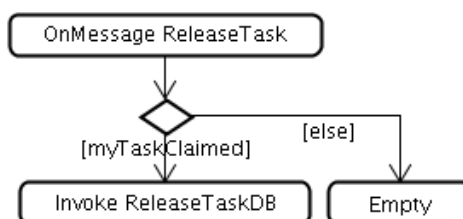


Figura 22. Liberação de propriedade das atividades humanas

A operação `releaseTask` possui uma verificação do estado atual em que a atividade humana se encontra, que é responsável por constatar se ela foi declarada que seria executada pelo pesquisador que invocou a operação. Assim, caso esta verificação retorne verdadeira, o estado da atividade humana é alterado para `Ready`. Logo, a ação `Invoke ReleaseTaskDB` é acionada e invoca o serviço web `TaskManagementService` para realizar as operações necessárias no banco de dados do ambiente *ClusterFlow*. Por outro lado, se a verificação realizada no início da operação não for verdadeira, o *workflow* gerenciador de atividades humanas não deve executar nenhum tipo de ação e, portanto, a atividade `Empty` é utilizada para este caso.

4.6.5. Finalização de Atividades Humanas

O encerramento de uma atividade humana, bem como o retorno dos artefatos e da execução ao *workflow* científico são realizados por meio da operação `completeTask`. Esta operação é apresentada na Figura 23.

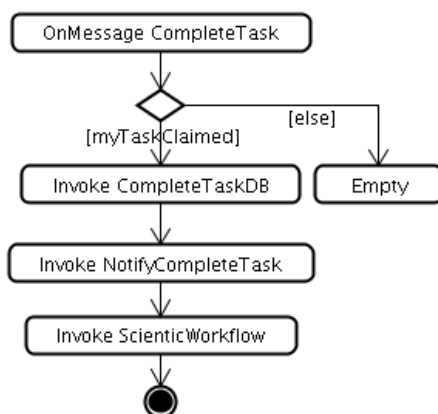


Figura 23. Finalização de atividades humanas

Quando o evento `OnMessage CompleteTask` for executado, a condição `myTaskClaimed` é verificada. Esta condição garante que o usuário que estiver invocando a operação seja o proprietário da atividade humana. Caso não seja o proprietário, a atividade `Empty` é executada. Por outro lado, se a condição for verdadeira, a operação do evento altera o estado da atividade humana para `Completed` e invoca o serviço web `TaskManagementService` para propagar as alterações no banco de dados por meio da ação `Invoke CompleteTaskDB`. Logo que as alterações forem realizadas, os cientistas pertencentes aos papéis da atividade humana serão notificados com o resultado da mesma. Assim, os dados desta notificação são enviados à ação `Invoke NotifyCompleteTask` que também invoca o serviço web `TaskManagementService`. Para esta operação, o serviço web adiciona as notificações referentes ao resultado da atividade no banco de dados, permitindo assim, que cientistas pertencentes aos papéis declarados para a atividade humana acompanhem o seu resultado. Da mesma forma, é necessário que os artefatos de resultado da atividade sejam enviados ao *workflow* científico que está aguardando a finalização da mesma. Estes artefatos são definidos no momento em que a atividade humana é criada no *workflow* científico. A ação `Invoke ScientificWorkflow` submete os artefatos ao *workflow* científico para dar continuidade à execução do experimento.

4.6.6. Interrupção da Execução de Atividades Humanas

Para que a interrupção das atividades humanas por exceder o prazo de execução seja possível, os tratadores de falha (`Fault Handlers`) podem ser utilizados. Esses tratadores de falha podem ser analisados em WS-BPEL como uma forma de alterar o processamento normal de um escopo. Tratadores de falha explícitos podem ser anexados a um escopo com o objetivo de

oferecer um modo de definir um conjunto de atividades capazes de manipular uma exceção gerada, e são definidas pelos conceitos de captura `Catch` e `CatchAll` (OASIS, 2007). Assim, o resultado das ações responsáveis por calcular o tempo e data limites para a conclusão de uma atividade humana (`DealineIsSpecified` ou `DurationIsSpecified`) é utilizado como artefato de entrada para o evento que trata da interrupção das atividades humanas. Para que estas possam ser interrompidas foi adicionado outro evento no escopo BPEL criado. Este evento é executado em paralelo com o *workflow* `TaskManagerProcess` e é capaz de capturar o momento exato em que a atividades humanas devem ter a execução interrompida. De acordo com a especificação WS-BPEL, o evento `OnAlarm` de uma atividade `Pick` é capaz de satisfazer esta necessidade. Assim, este evento também foi adicionado ao escopo criado e está ilustrado na Figura 24.

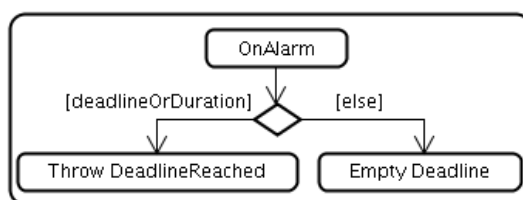


Figura 24. Evento `OnAlarm` para lançar o fluxo alternativo de interrupção das atividades humanas

O evento `OnAlarm` é acionado no momento que a condição verificada por ele (o tempo limite for excedido para a execução da atividade humana) retornar verdadeira, executando assim, a ação `Throw DeadlineReached`. Esta ação é encarregada de lançar uma exceção programada pelo próprio *workflow* e paralisar a execução do mesmo, bem como direcionar a execução para o fluxo alternativo capturado pelo nome da exceção lançada (`deadlineReached`). As exceções lançadas pelo evento `OnAlarm` com o nome de `deadlineReached` são capturadas por meio da ação `Catch DeadlineReached`, mostradas pela Figura 25.

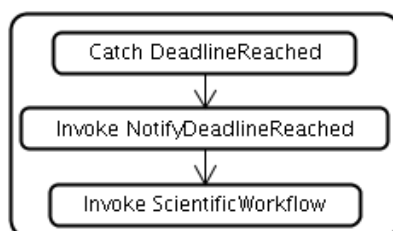


Figura 25. Ação de captura da exceção `deadlineReached`

A ação `Catch DeadlineReached` é utilizada neste *workflow* para dar início a um fluxo de notificação. Assim, na medida em que a falha é capturada o estado da atividade humana é alterado para `Failed`. Esta notificação é enviada aos possíveis proprietários por meio da ação `Invoke NotifyDeadlineReached`, que invoca o serviço `web TaskManagementService`. Em seguida, o *workflow* científico retorna à execução por meio da ação `Invoke ScientificWorkflow`. A visão completa do *workflow* `TaskManagerProcess` se encontra no Apêndice B. Dessa forma, encerra-se todo o fluxo de execução do *workflow* de gerenciamento de instâncias de atividades humanas.

O ambiente *ClusterFlow* também possui o rastreamento dos artefatos das atividades durante a execução dos *workflows* científicos, conhecido em ambientes deste tipo como proveniência de dados. A proveniência de dados do ambiente *ClusterFlow* é descrita na seção seguinte.

4.7. Proveniência de Dados

A proveniência de dados do ambiente *ClusterFlow* é realizada em nível de atividade, garantindo com isso que a mesma possa ser utilizada em diferentes máquinas de execução de *workflows* baseados na linguagem WS-BPEL, além de permitir que os dados de proveniência sejam coletados de forma precisa, exigindo pouco processamento posterior à execução para a visualização dos artefatos capturados. Estes artefatos são armazenados em um banco de dados relacional. Assim, as informações são centralizadas, o que facilita o seu compartilhamento entre os pesquisadores, proporciona um modo eficiente para realizar consultas complexas e oferece segurança no armazenamento.

Foi utilizado um sistema independente capaz de gerenciar a persistência destas informações. A decisão da adoção deste sistema também foi baseada no trabalho de (MARINHO *et al.*, 2009), que implementa proveniência para outros SGWfC, tais como, Kepler e Taverna que são SGWfC muito utilizados. Dessa forma, para tornar essa comunicação e a independência de SGWf praticável, algumas pesquisas realizadas por (MUNROE *et al.*, 2006) e (LIN *et al.*, 2008) propõem um conjunto de adaptações manuais na definição de um *workflow* e de suas atividades. Estas adaptações compreendem o desenvolvimento manual de software para a captura da proveniência em cada atividade nova ou existente no *workflow*. Entretanto, a maioria dos cientistas não possui conhecimento computacional suficiente para realizar estas adaptações nas atividades, pois estas podem ser provenientes de software de terceiros o que torna a adaptação difícil de ser realizada. No

entanto, a adaptação da proveniência em nível de atividade do *workflow* no ambiente *ClusterFlow* permite que elas sejam ajustadas de forma automática e sem a intervenção do cientista. Além disso, esta tarefa de adaptação não pertence ao objetivo do desenvolvimento do *workflow* científico. Assim, o cientista não tem a necessidade de se preocupar com a proveniência, mas apenas em criar o *workflow* para o experimento desejado.

Contudo, esta medida tomada possui algumas desvantagens, pois a persistência dos dados é realizada por um mecanismo externo e o motor de orquestração necessita realizar diversas chamadas a este sistema, além de enviar os dados a serem persistidos. Desta forma, uma sobrecarga pode ser gerada na rede de comunicação, o que pode atrasar a execução do *workflow* caso os dados utilizados ou gerados pelas atividades sejam na ordem de Gigabytes. A Figura 26 apresenta a visão geral da abordagem adotada.

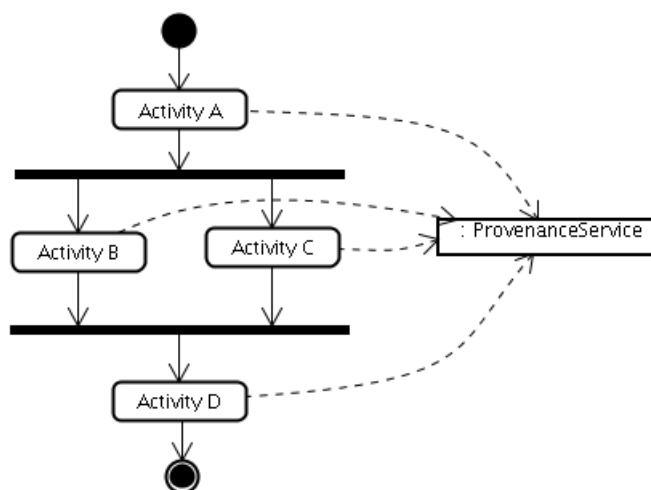


Figura 26. Visão geral da abordagem adotada para a captura da proveniência de dados

Cada atividade pertencente ao *workflow* científico é responsável pela captura dos dados de entrada como também dos dados de saída. Estes dados são armazenados na base de dados e relacionados com a instância da atividade que está sendo executada, fornecendo assim, a proveniência dos resultados baseados em cada parâmetro de entrada.

Como o ambiente *ClusterFlow* gerencia os *workflows* criados a partir da especificação WS-BPEL, cada atividade que é realizada representa um serviço web. Para oferecer a proveniência em nível de atividade e considerando que o serviço de proveniência também é um serviço web, foi necessária a utilização do conceito de empacotadores que são capazes de encapsular as atividades originais em atividades compostas. A Figura 27 mostra um *workflow* contendo uma atividade original sendo empacotada como uma atividade composta.

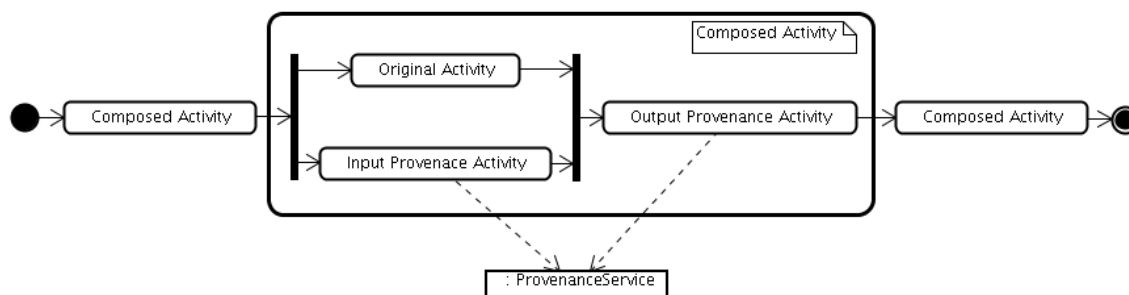


Figura 27. Atividade original empacotada como atividade composta

A solução de encapsular automaticamente uma atividade original para outra composta realizada pelo ambiente *ClusterFlow* agrega atividades adicionais, formando assim, um *subworkflow* a partir da atividade original. Porém, este *subworkflow* é visualizado em nível conceitual, uma vez que não é necessário criar um *workflow* para cada atividade, mas apenas um fluxo lógico internamente ao *workflow* científico que foi modelado pelo cientista. Estas duas atividades adicionais inseridas no contexto da atividade original (*Input Provenance Activity* e *Output Provenance Activity*) são as responsáveis por realizar a coleta dos dados de proveniência e posteriormente invocar o serviço web *ProvenanceService* para realizar a persistência dos dados. Além disso, pode-se perceber que as atividades *Original Activity* e *Input Provenance Activity* são executadas paralelamente, o que pode diminuir o tempo de execução do *workflow* com a adoção desta estratégia.

Esta tática de execução em paralelo da atividade original e a atividade de proveniência diferem da estratégia proposta em (LIN *et al.*, 2008), pois o ambiente *ClusterFlow* permite a execução paralela destas duas atividades, enquanto a proposta dos autores apenas permite a execução em paralelo das atividades de proveniência. Outra diferença da proveniência do *ClusterFlow* é que apenas existe uma atividade responsável por capturar todos os dados em cada momento de captura, diferindo também da proposta de (LIN *et al.*, 2008). A proposta dos autores utiliza uma atividade para cada artefato nas fases de captura, o que exige um número maior de requisições ao mecanismo de proveniência.

Entretanto, com o intuito de garantir a aplicabilidade da estratégia de proveniência do ambiente *ClusterFlow*, ainda não foram realizados estudos empíricos do quanto que a mesma implicará no desempenho do ambiente, bem como no alto consumo da banda de rede com a grande quantidade de dados transmitidos. Porém, ela pode apresentar diversos benefícios aos cientistas que utilizam o ambiente, uma vez que os mesmos apenas necessitam se preocupar com a construção dos seus *workflows* científicos (LIN *et al.*, 2008) (MARINHO *et al.*, 2009).

4.8. Modelo de Representação de *Workflow* no Ambiente *ClusterFlow*

Os *workflows* criados pelos pesquisadores no ambiente *ClusterFlow* podem ser armazenados e recuperados posteriormente com a possibilidade de fazer o uso da técnica de reutilização para acelerar o processo de pesquisa científica. O armazenamento das definições dos *workflows* é apoiado por um banco de dados relacional, o que garante que eles sejam armazenados de modo centralizado e em local que todos os pesquisadores possam ter acesso. Além disso, a definição em um modelo estático foi concebida baseada nos padrões que podem ser interpretados pela Máquina WS-BPEL. Assim, o modelo estático para o armazenamento dos dados dos *workflows* para o ambiente *ClusterFlow* é ilustrado na Figura 28, e teve como base as especificações WS-BPEL, WSDL, XSD²³ (W3C, 2009), BPEL4People e WS-HumanTask.

Este modelo possui elementos preenchidos com tonalidade cinza e elementos sem preenchimento. Os elementos preenchidos representam àqueles que estão implementados no ambiente *ClusterFlow* e são utilizados para apoiar o gerenciamento de experimentos científicos. Por outro lado, os elementos sem preenchimento representam àqueles que ainda necessitam de implementação futura. Além disso, o modelo estático não possui elementos que definem a modelagem dos dados que devem ser armazenados durante as execuções das instâncias das atividades humanas e notificações. Estes dados representam àqueles que serão visualizados no protótipo do portal web, pelos possíveis proprietários das atividades humanas (*People Activity*) a serem executadas. O mesmo ocorre para as notificações (*Notification*), pois a mesma notificação pode ser enviada para vários pesquisadores e estes podem excluir apenas as notificações pertencentes ao seu usuário do ambiente. A exclusão de notificação não pode afetar a visualização da mesma instância de notificação enviada a outro pesquisador. Esta funcionalidade ainda precisa ser analisada, modelada e implementada no protótipo do ambiente *ClusterFlow*.

²³ Do inglês XML Schema Definition.

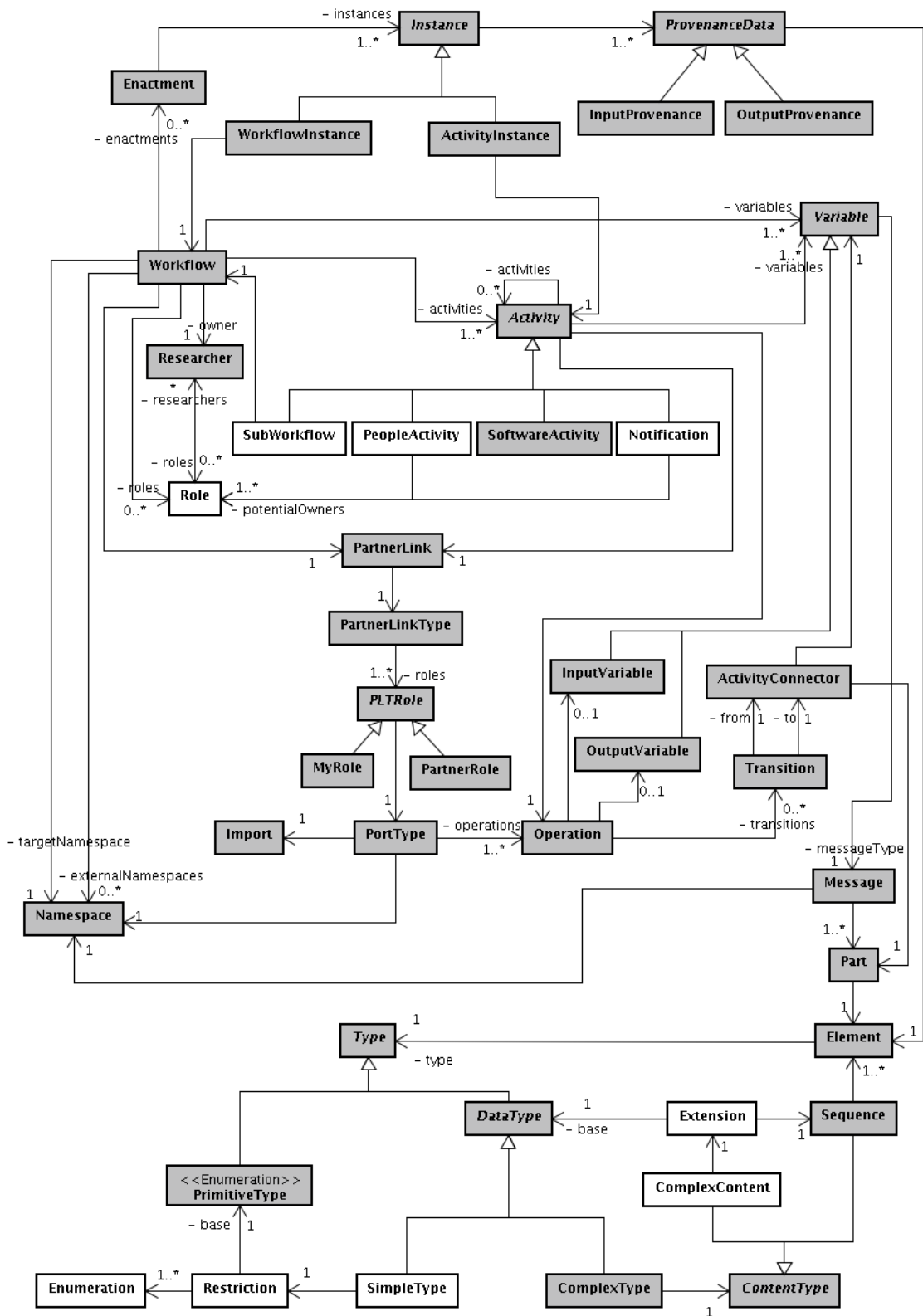


Figura 28. Modelo estático para a representação de workflows no ambiente ClusterFlow

O modelo estático da Figura 28 representa os elementos envolvidos na definição de um *workflow* no ambiente *ClusterFlow*. Com base nesses elementos, *workflows* podem ser

exportados para a Máquina WS-BPEL e executados a partir do Portal Web para Experimentos. Os elementos principais são descritos a seguir:

- Workflow: é o elemento base no processo de construção de um *workflow* científico. A partir dele é que todas as definições do conjunto de atividades e instâncias de execuções envolvidas em um experimento são referenciadas;
- Activity: representa a abstração do conjunto de atividades envolvidas no *workflow* científico que correspondem a `SoftwareActivity`, `PeopleActivity`, `Notification` e `SubWorkflow`. Cada atividade pode conter uma ou mais subatividades que formam o fluxo lógico da execução do experimento. Caso a atividade possua subatividades, estas serão executadas em paralelo na Máquina WS-BPEL;
- Role: este elemento representa os papéis que os possíveis proprietários (`PotentialOwner`) das atividades humanas (`PeopleActivity`) exercem no *workflow* que está sendo criado. É com base neste papel que as atividades humanas e notificações são enviadas aos pesquisadores. Cada papel pode conter vários pesquisadores (`Researcher`) relacionados, bem como um pesquisador pode estar relacionado a diversos papéis (`Role`) em diferentes *workflows*;
- Operation: cada atividade definida no *workflow* deve estar relacionada à apenas uma operação, pois o elemento `PortType` da especificação WSDL pode conter diversas operações. Assim, é necessário definir a qual das operações cada atividade está relacionada;
- Transition: este elemento determina as transições de dados entre uma atividade e outra. É por meio deste que o fluxo lógico das trocas de dados entre as atividades é definido. Todas as transições definidas devem possuir o relacionamento com o elemento `ActivityConnector`. O mapeamento de cada artefato de entrada e saída das operações representa uma transição;
- ActivityConnector: este elemento representa os conectores de atividades que definem explicitamente os mapeamentos entre os artefatos do *workflow* e de suas atividades. Cada conector possibilita que um dos artefatos de saída de uma atividade possa ser mapeado com um dos artefatos de entrada da próxima atividade a ser executada. Este elemento está relacionado diretamente com um elemento `Element`,

que por sua vez é uma parte (Part) de uma mensagem (Message) trocada entre as atividades;

- Element: este elemento do modelo estático define os elementos da especificação XSD. Elementos desta especificação definem os tipos de dados que podem ser trocados em XML. Além disso, os elementos XSD definem tipos de dados primitivos (PrimitiveType), tipos complexos (ComplexType) e tipos simples (SimpleType), todos representados no modelo estático;
- Type: representa a abstração do tipo do elemento (Element);
- PrimitiveType: este elemento do modelo estático refere-se aos tipos de dados primitivos definidos por meio da especificação XSD, tais como, string, int, decimal, base64Binary, dentre outros. Os tipos primitivos representam o último nível da representação de um elemento da especificação XSD;
- Restriction: o elemento Restriction do modelo estático representa uma restrição de um elemento do tipo simples (SimpleType), que é do tipo primitivo. As restrições são utilizadas para definir o tipo de dado que deve ser utilizado pelo elemento Enumeration do modelo estático;
- Enumeration: este elemento representa um conjunto de possíveis valores que um pesquisador define na criação de uma atividade humana, obrigando que os artefatos devam satisfazer a lista de valores cadastrada neste elemento. Um exemplo de utilização dele é a configuração de uma resposta de aprovação da execução de alguma atividade, por exemplo, “Aprovado” ou “Desaprovado”. Enumeration é um elemento pertencente à especificação XSD;
- ComplexContent: este elemento também pertence à especificação XSD e no ambiente ClusterFlow é utilizado para representar a herança de outro tipo de dado. A utilização de herança para definir os artefatos de um workflow no ambiente ClusterFlow pode permitir que as definições de artefatos sejam reutilizadas em diferentes atividades no mesmo workflow;
- Extension: define o tipo do elemento a ser reutilizado (DataType) e o conjunto de novos elementos específicos pertencentes àquele artefato;
- Enactment: este elemento é utilizado para armazenar a identificação de todas as execuções dos workflows, bem como o estado de execução no qual cada workflow se

encontra. Com este elemento todas as execuções de *workflows* (Enactment) podem ser diferenciadas umas das outras;

- Instance: representa as instâncias de execução. Estas instâncias podem ser de *workflows*, bem como de atividades. Cada instância possui dados que são relevantes para a proveniência de dados que ocorre durante a execução do *workflow*. Estes dados podem compreender a data e hora de início e finalização da execução da atividade ou do próprio *workflow*, bem como a lista de proveniência dos dados; e
- ProvenanceData: o elemento ProvenanceData representa os artefatos e seus valores capturados durante a execução do *workflow*. Para cada artefato do *workflow* e de suas atividades é necessário uma instância do elemento ProvenanceData, que por sua vez está relacionado com o elemento Element, o que possibilita representar a proveniência de cada artefato pertencente ao *workflow* científico.

Os demais elementos do modelo estático são relevantes para a geração das versões concretas e executáveis dos *workflows* criados a partir do ambiente *ClusterFlow*. Contudo, não foram comentados, pois são utilizados para representar as estruturas das especificações responsáveis por permitirem a execução dos *workflows* na Máquina WS-BPEL.

4.9. Considerações Finais

Este capítulo apresentou o projeto do ambiente *ClusterFlow*. O projeto foi baseado principalmente na necessidade de atender os princípios para a construção de *workflows* científicos, e da necessidade de um ambiente que oferecesse uma interface gráfica aos cientistas com um maior nível de abstração para estes definirem *workflows* que automatizassem seus experimentos científicos. Esta abstração inclui a execução de aplicações de alto desempenho em *clusters* de computadores.

O cenário em que o ambiente *ClusterFlow* está inserido foi apresentado para oferecer uma visão geral da forma de realização de experimentos científicos no mesmo. Com base neste cenário, nos princípios do ambiente e no levantamento de requisitos mostrado no Apêndice A, a arquitetura inicial do ambiente foi proposta e descrita. Além disso, o processo de construção de *workflows* científicos que foi apresentado oferece uma visão em alto nível das ações que devem ser realizadas pelos cientistas durante a realização de seus experimentos.

Da mesma maneira, foi proposto um processo capaz de apoiar o gerenciamento das instâncias das atividades humanas, uma vez que apenas o SGWf ActiveVOS oferece tal apoio

em sua implementação. Os demais SGWf que implementam a especificação WS-BPEL usam soluções proprietárias. Além disso, os SGWfCs que apóiam a execução de experimentos que necessitem de alto poder de processamento não oferecem apoio ao gerenciamento das atividades humanas, lacuna que é preenchida com a implementação do processo de gerenciamento de atividades humanas proposto.

A proveniência de dados do ambiente permite a coleta dos artefatos em nível de atividade. Para isto, as atividades do *workflow* são empacotadas em atividades compostas de forma automática pelo *ClusterFlow*. Estas atividades podem ser comparadas a *subworkflows*, pois cada atividade composta possui o seu próprio fluxo de execução. O mecanismo desenvolvido para receber os artefatos de proveniência é um serviço web que armazena as informações em banco de dados relacional.

O modelo conceitual usado para realizar o mapeamento das definições de *workflows* a partir do banco de dados para a especificação WS-BPEL também foi apresentado. Este modelo representa os elementos envolvidos na definição de *workflows* no ambiente *ClusterFlow*. O capítulo seguinte apresenta as principais estratégias de implementação usadas para o desenvolvimento do ambiente *ClusterFlow*.

Estratégias de Implementação

5.1. Considerações Iniciais

Este capítulo apresenta as principais estratégias de implementação do protótipo do ambiente *ClusterFlow*. Uma visão geral das tecnologias utilizadas para o desenvolvimento do protótipo é apresentada inicialmente. Em seguida é mostrado um diagrama de atividades que representa um *workflow* de exemplo usado para facilitar a descrição e o entendimento das estratégias de mapeamento e exportação dos *workflows* definidos no *ClusterFlow*. A descrição dos mapeamentos foi dividida em 3 (três) seções. A primeira mostra parte do modelo estático apresentado na Figura 28 que corresponde à definição da especificação XSD. O mapeamento do modelo estático para a especificação XSD também é descrito. O documento XML gerado com esse mapeamento representa os tipos de dados do documento WSDL do *workflow*. O mapeamento do modelo estático para este documento WSDL é descrito na seção seguinte. Por fim, é apresentado e descrito o mapeamento do modelo estático que representa as definições da especificação WS-BPEL.

5.2. Visão Geral

O protótipo do ambiente *ClusterFlow* foi desenvolvido com base em diversas tecnologias, tais como, J2EE (*Java2 Enterprise Edition*), JSP (*Java Server Pages*), JSTL (*Java Standard Tag Library*), Struts2 (APACHE, 2010b), JSON (*JavaScript Object Notation*) (APACHE, 2010e), Ajax (*Asynchronous Javascript And XML*), JQuery (JQUERY, 2010), Toplink (ORACLE,

2010a), Javascript, banco de dados relacional PostgreSQL versão 8.3, Apache Axis2 (APACHE, 2010c), Apache Tomcat (APACHE, 2010d) versão 6.0.26, Apache ODE (*Orchestrator Director Engine*) (APACHE, 2010a) versão 1.3.4, usado para a execução dos *workflows* baseados em WS-BPEL, e o sistema operacional Debian GNU/Linux 5.0 “Lenny”.

O *cluster* de computadores utilizado pertence ao Laboratório Experimental de Computação de Alto Desempenho (LECAD) do Departamento de Informática da UEM, e é formado por 1 (um) nó servidor de acesso e 6 (seis) nós escravos homogêneos que são conectados por meio de um *switch gigabit*. A configuração das máquinas é a seguinte:

- servidor de acesso: CPU Core2Quad 2.4Ghz, 4 núcleos, memória RAM de 2048 MB, conexão de rede de 1Gbit e capacidade de armazenamento de 400GB; e
- nós escravos: CPU AMD Opteron 1218 2.6Ghz, 2 núcleos, memória RAM de 4096 MB, conexão de rede de 1Gbit e capacidade de armazenamento de 1TB.

O protótipo foi desenvolvido com base no processo para a construção de *workflows* científicos que foi descrito na Seção 4.5. Contudo, atualmente o protótipo não apóia todo o processo de construção de *workflows*, pois a sua implementação é extensa e complexa. O foco pretendido foi implementar as funcionalidades mínimas para demonstrar a viabilidade do ambiente. O fato da interface gráfica estar em nível de protótipo não permite que a geração de código WS-BPEL, WSDL e XSD não seja realizada de acordo com os padrões propostos por estas especificações. Assim, para que os *workflows* criados pelos cientistas possam ser executados na Máquina WS-BPEL, foi necessário o desenvolvimento de um mecanismo de exportação de *workflows* que atendesse os padrões definidos nas especificações WS-BPEL, WSDL e XSD. Atualmente, as definições e exportações de *workflows* realizadas pelo ambiente implementam o fluxo de atividades de software definidas no processo de construção de *workflows* científicos. O processo necessário para o gerenciamento de atividades humanas, bem como as principais tecnologias relacionadas à definição, exportação e execução destas atividades foram definidos no projeto do ambiente, porém, a implementação das definições de exportação das atividades humanas, notificações e *subworkflow* foram deixadas como trabalhos futuros.

As interfaces gráficas criadas para que pesquisadores possam definir, executar e analisar seus *workflows* são apresentadas em detalhes no Capítulo 6, em que é descrito um exemplo de aplicação desenvolvido no *ClusterFlow*. As seções seguintes descrevem a exportação de *workflows*, que é realizada com base no mapeamento do modelo estático para as especificações WS-BPEL, WSDL e XSD. É utilizado um exemplo de *workflow* que contém

apenas uma atividade de software. Esta atividade invoca um serviço web que realiza uma operação de adição de 2 (dois) valores inteiros. Este exemplo está sendo usado pois simplifica o processo de descrição da tradução do modelo estático para as especificações supracitadas.

A Figura 29 mostra o diagrama de atividades do *workflow* de exemplo. Os estereótipos que aparecem na figura representam as atividades da especificação WS-BPEL que são utilizadas para compor o *workflow*. Este *workflow* foi definido por meio do ambiente *ClusterFlow* e os códigos das especificações que são apresentados foram gerados pelo mecanismo de exportação do ambiente.

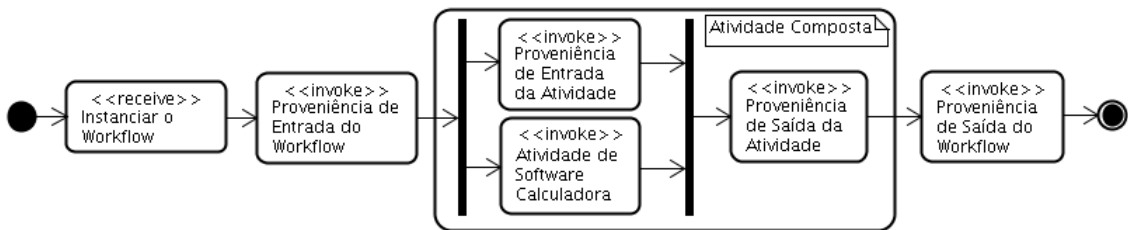


Figura 29. Exemplo de *workflow* para adição de valores

5.3. Mapeamento do Modelo Estático para XSD

Os elementos do modelo estático não oferecem apoio a todas as definições da especificação XSD (W3C, 2009). Contudo, as principais estruturas da especificação necessárias para o ambiente *ClusterFlow* foram modeladas. A Figura 30 mostra parte do diagrama do modelo estático que está relacionada à tradução das definições de *workflows* armazenados em banco de dados para a especificação XSD. Este diagrama, assim como os seguintes, é parte do diagrama da Figura 28.

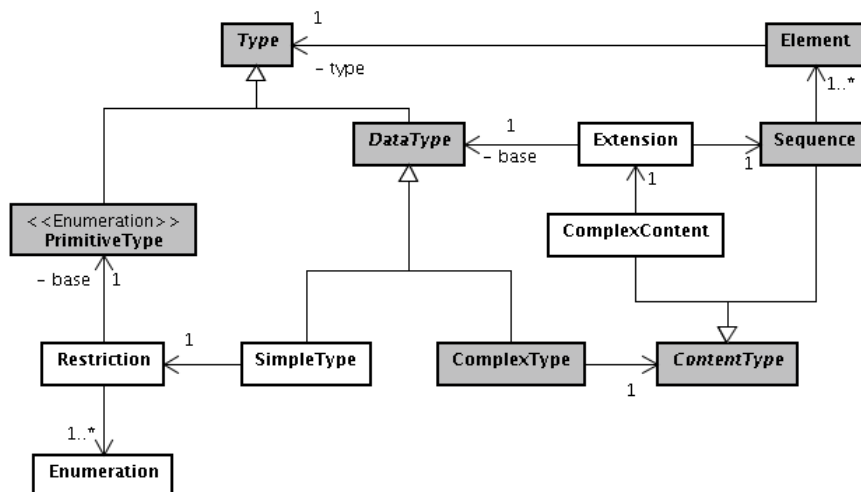


Figura 30. Modelo estático para a representação da especificação XSD.

Com base no exemplo do *workflow* de adição gerado pelo *ClusterFlow*, obtém-se o código da especificação XSD ilustrado na Listagem 2. Quando os elementos da estrutura do documento XML forem referenciados, é utilizado o termo “*tag*”. Para as classes do modelo estático é utilizado o termo “elemento”.

```

1 <schema xmlns="http://www.w3.org/2001/XMLSchema"
2   attributeFormDefault="unqualified" elementFormDefault="qualified"
3   targetNamespace="http://les.din.uem.br/clusterflow/process/CalcWorkflow">
4   <element name="CalcWorkflowProcessRequest" type="tns:inputWorkflowType" />
5   <element name="CalcWorkflowProcessResponse" type="tns:oututWorkflowType" />
6   <complexType name="inputWorkflowType">
7     <sequence>
8       <element name="artefact2" type="int" />
9       <element name="artefact1" type="int" />
10    </sequence>
11  </complexType>
12  <complexType name="oututWorkflowType">
13    <sequence>
14      <element name="workflowResult" type="int" />
15    </sequence>
16  </complexType>
17 </schema>

```

Listagem 2. Documento XSD gerado pelo ambiente *ClusterFlow*

Comparando o modelo estático (Figura 30) e a Listagem 2, pode-se notar o relacionamento entre os elementos do modelo e o código XML gerado. É importante ressaltar que este código é apenas a definição dos artefatos de entrada e saída do *workflow*, uma vez que os artefatos das atividades de software são importados dos documentos WSDL presentes no repositório de serviços web. As *tags* declaradas nas linhas 4 e 5 referenciam os tipos complexos presentes nas linhas 6 e 12, cujas relações ocorrem com os elementos `Element`, `Type`, `DataType` e `ComplexType` do modelo estático. Os tipos complexos representados pelas *tags* `complexType` nas linhas 6 e 12 possuem uma sequência de *tags* (linhas 8; 9; e 14) que são do tipo primitivo ilustrados respectivamente pelos elementos `ComplexType`, `Sequence`, `Element` e `PrimitiveType` do modelo estático.

5.4. Mapeamento do Modelo Estático para WSDL

O mecanismo de geração da interface WSDL do ambiente *ClusterFlow* se apóia em diversos elementos do modelo estático. Além disso, alguns dos elementos do modelo estático fazem a ligação entre as especificações. No caso de XSD e WSDL, o elemento do modelo que tem esse papel é o `Element`. A Figura 31 exhibe o modelo estático usado pelo mecanismo de geração da interface WSDL do *ClusterFlow*.

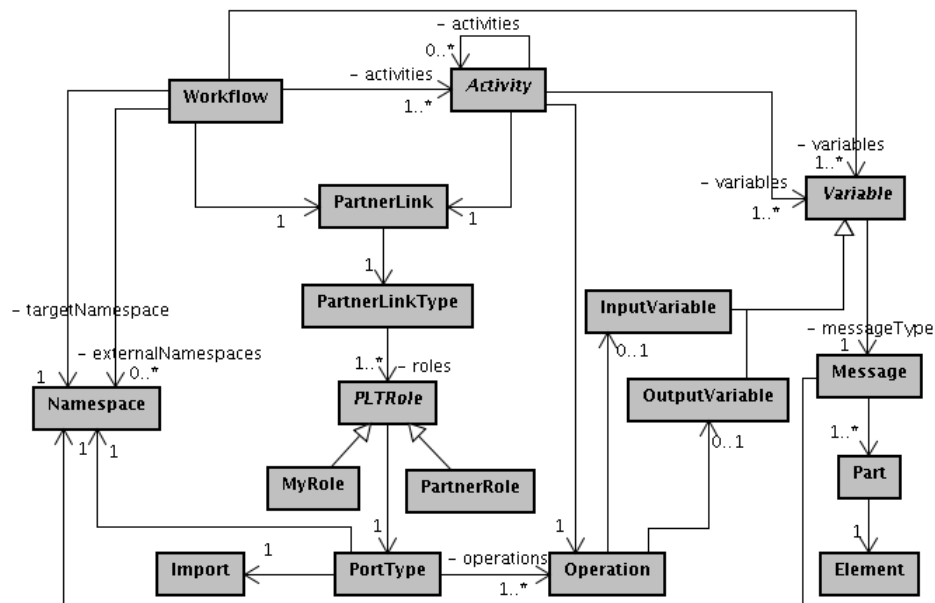


Figura 31. Modelo estático para a representação da especificação WSDL

Com base neste modelo e no *workflow* de exemplo, o documento da especificação WSDL gerado pelo mecanismo de exportação do ambiente *ClusterFlow* é mostrado na Listagem 3.

```

1 <definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
2   name="CalcWorkflowProcess"
3   xmlns:impl="http://huff.calc.les.din.uem.br"
4   xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
5   xmlns:provenanceService="http://provenance.clusterflow.din.uem.br"
6   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
7   xmlns:tns="http://les.din.uem.br/clusterflow/process/CalcWorkflow"
8   targetNamespace="http://les.din.uem.br/clusterflow/process/CalcWorkflow">
9   <plnk:partnerLinkType name="CalcWorkflowPLT">
10    <plnk:role name="CalcWorkflowProvider" portType="tns:CalcWorkflowPortType" />
11  </plnk:partnerLinkType>
12  <plnk:partnerLinkType name="CalculadoraPLT">
13    <plnk:role name="CalculadoraProvider" portType="impl:Calculadora" />
14  </plnk:partnerLinkType>
15  <plnk:partnerLinkType name="ProvenancePortTypePLT">
16    <plnk:role name="ProvenancePortTypeProvider" portType="provenanceService:ProvenancePortType" />
17  </plnk:partnerLinkType>
18  <import location="Calculadora.wsdl" namespace="http://huff.calc.les.din.uem.br" />
19  <import location="ProvenanceService.wsdl" namespace="http://provenance.clusterflow.din.uem.br" />
20  <types>
21    <schema>
22      ...
23    </schema>
24  </types>
25  ...
26  ...
27 </definitions>
  
```

Listagem 3. Documento WSDL gerado pelo ambiente *ClusterFlow*

Conforme a Listagem 3, o documento WSDL exportado e gerado pelo ambiente *ClusterFlow* possui a *tag* *definitions* que define o início e fim do documento. Nesta *tag* pode-se notar a declaração de diversos *namespaces* junto de seus prefixos (linha 3 a 8) que servem para eliminar ambiguidades no momento da referência a um artefato ou outro

documento que pode ser importado por este documento WSDL. Esses *namespaces* são representados no modelo estático pelo elemento `Namespace`. O atributo `name` (linha 2) recebe o valor que pertence ao atributo `name` do elemento `Workflow` no modelo estático, porém, não representado nas figuras.

A especificação WSDL também conta com a declaração dos tipos de parceiros representada pela *tag* `partnerLinkType` (linhas 9 a 17), responsável por definir quais papéis cada parceiro pode estar realizando no *workflow*. Um `partnerLinkType` pode ter os papéis de consumidor e fornecedor de serviço em um mesmo *workflow*. O modelo estático representa as *tags* `partnerLinkType`, `role` e `portType`, respectivamente, por meio dos elementos `PartnerLinkType`, `PLTRole` e `PortType`. A *tag* `types` (linha 20) representa o elemento de extensão no qual é declarado o documento XSD da definição WSDL que foi descrito na seção anterior. As *tags* `import` (linhas 18 e 19) definem os documentos que devem ser importados para a definição deste documento WSDL. Elas são mapeadas a partir do elemento `Import` do modelo estático.

```

22 <definitions>
23   ...
24   <message name="CalcWorkflowProcessRequestMessage">
25     <part element="tns:CalcWorkflowProcessRequest" name="payload" />
26   </message>
27   <portType name="CalcWorkflowPortType">
28     <operation name="process">
29       <input message="tns:CalcWorkflowProcessRequestMessage" />
30     </operation>
31   </portType>
32   <binding name="CalcWorkflowSOAPBinding" type="tns:CalcWorkflowPortType">
33     <soap:binding style="document"
34       transport="http://schemas.xmlsoap.org/soap/http" />
35     <operation name="process">
36       <soap:operation
37         soapAction="http://les.din.uem.br/clusterflow/process/CalcWorkflow/process" />
38       <input>
39         <soap:body use="literal" />
40       </input>
41       <output>
42         <soap:body use="literal" />
43       </output>
44     </operation>
45   </binding>
46   <service name="CalcWorkflowService">
47     <port binding="tns:CalcWorkflowSOAPBinding" name="CalcWorkflowPort">
48       <soap:address
49         location="http://localhost:8080/ode/processes/CalcWorkflowProcess" />
50     </port>
51   </service>
52 </definitions>

```

Listagem 4. Continuação do documento WSDL gerado pelo ambiente ClusterFlow

O relacionamento entre os documentos WSDL e XSD ocorre por meio do atributo `element` pertencente a uma *tag* `part` (linha 25) de uma mensagem do documento WSDL (linha 24). A *tag* `part` referencia o elemento declarado na especificação XSD (Listagem 2, linha 4), que por sua vez define os tipos de dados dos artefatos trocados entre os parceiros na

execução do *workflow*. Esses são mapeados pelos elementos `Message`, `Part` e `Element` do modelo estático.

O conjunto de operações do *workflow* é representado por um `portType` no documento WSDL (linha 27). Para este exemplo temos apenas uma operação denominada `process`, a qual é obrigatória em todos os *workflows*, pois se refere à operação de instanciação do *workflow*. Para invocações de serviços web assíncronos são declaradas outras operações, pois os serviços web necessitam retornar a chamada ao *workflow* para continuar a execução do experimento. A `tag portType` e suas `tags` filhas são representadas pelos elementos `PortType`, `Operation`, `InputMessage` e `OutputMessage` do modelo estático.

Assim como a `tag portType`, a `tag binding` (linha 32) é mapeada pelos mesmos elementos do modelo estático. Entretanto, `portType` define as operações abstratas do *workflow* enquanto `binding` define as operações concretas. Ela também informa o protocolo de comunicação que deve ser utilizado (linhas 36, 39 e 42), neste exemplo, protocolo SOAP, bem como se a mensagem é orientada a documento ou a procedimento (linha 33) e o meio de transporte a ser usado (linha 34). Além disso, o documento WSDL também define o serviço web (linha 46) e a localização da implementação deste serviço (linha 48). Com isso a descrição da interface de serviço do *workflow* está definida e pode ser utilizada pelos serviços web cliente que desejam instanciar o *workflow*.

5.5. Mapeamento do Modelo Estático para WS-BPEL

A Figura 32 apresenta os elementos do modelo estático que são usados para mapear e gerar o *workflow* baseado na especificação WS-BPEL. O mapeamento do modelo estático para a linguagem WS-BPEL possui alguns elementos de ligação entre as especificações WSDL e WS-BPEL, que são representados no modelo por `PartnerLinkType`, `PLTRole`, `Message` e `PortType`.

O *workflow* utilizado como exemplo neste capítulo e gerado pelo *ClusterFlow* com base neste modelo estático é apresentado em fragmentos para facilitar a descrição do mapeamento realizado. A proveniência de dados também é mostrada nas listagens de código com o fluxo de execução do *workflow* científico. O *workflow* completo é apresentado no Apêndice C. A Listagem 5 mostra a declaração do *workflow* e dos parceiros envolvidos.

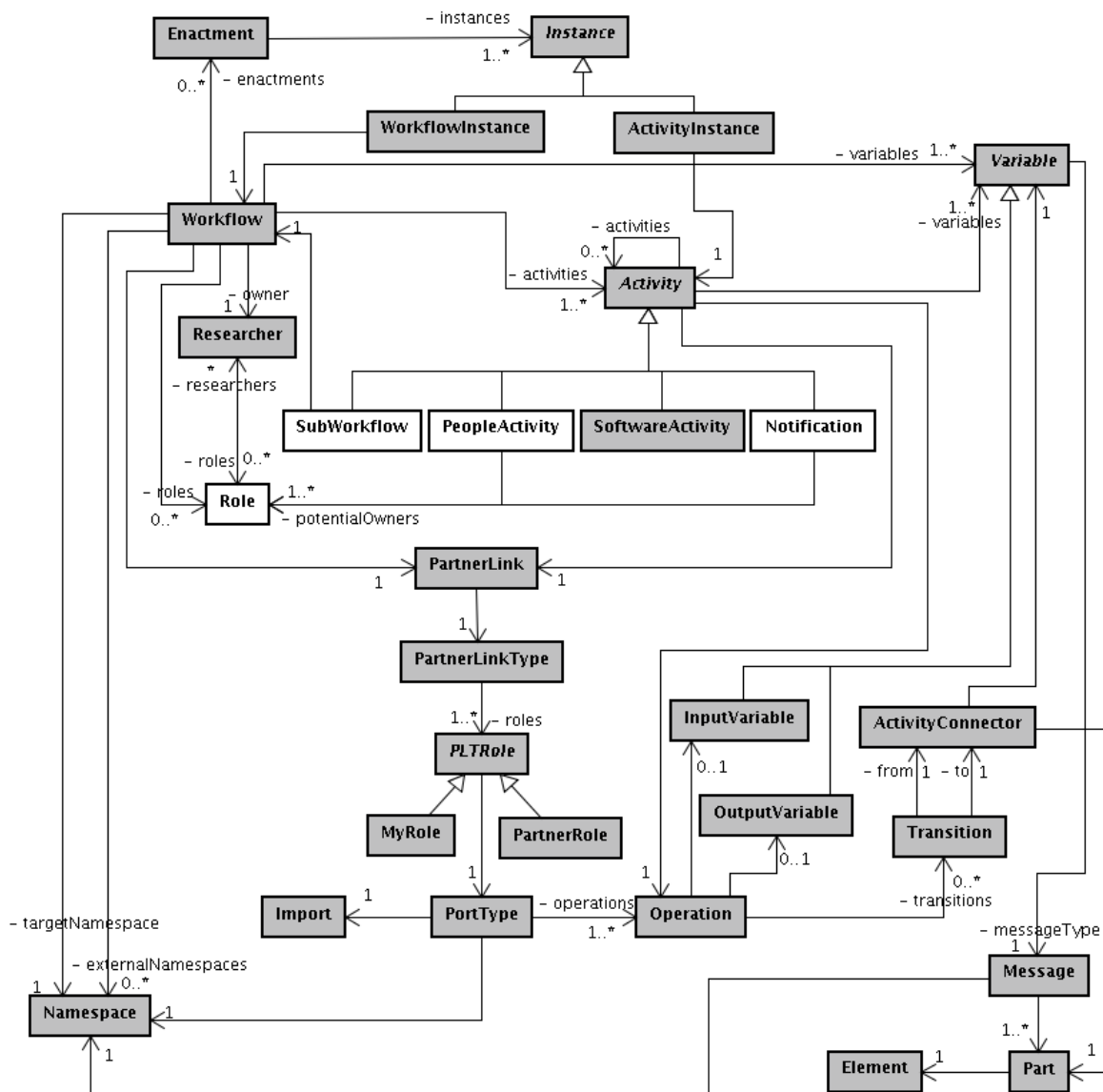


Figura 32. Modelo estático para a representação de workflows em WS-BPEL

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <bpel:process exitOnStandardFault="no" name="CalcWorkflow"
3   suppressJoinFailure="yes"
4   targetNamespace="http://les.din.uem.br/clusterflow/process/CalcWorkflow"
5   xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
6   xmlns:implCalculadora="http://huff.calc.les.din.uem.br"
7   xmlns:provenanceServiceprovenance="http://provenance.clusterflow.din.uem.br"
8   xmlns:tns="http://les.din.uem.br/clusterflow/process/CalcWorkflow">
9   <bpel:import importType="http://schemas.xmlsoap.org/wsdl/" location="CalcWorkflow.wsdl"
10    namespace="http://les.din.uem.br/clusterflow/process/CalcWorkflow" />
11   <bpel:import importType="http://schemas.xmlsoap.org/wsdl/" location="ProvenanceService.wsdl"
12    namespace="http://provenance.clusterflow.din.uem.br" />
13   <bpel:import importType="http://schemas.xmlsoap.org/wsdl/"
14    location="Calculadora.wsdl" namespace="http://huff.calc.les.din.uem.br" />
15   <bpel:partnerLinks>
16     <bpel:partnerLink myRole="CalcWorkflowProvider" name="client"
17      partnerLinkType="tns:CalcWorkflowPLT" />
18     <bpel:partnerLink name="ProvenancePortTypePL"
19      partnerLinkType="tns:ProvenancePortTypePLT" partnerRole="ProvenancePortTypeProvider" />
20     <bpel:partnerLink name="CalculadoraPL"
21      partnerLinkType="tns:CalculadoraPLT" partnerRole="CalculadoraProvider" />
22   </bpel:partnerLinks>

```

Listagem 5: Declaração do workflow e dos parceiros envolvidos

A *tag* `process` da Listagem 5 representa a definição do *workflow* em WS-BPEL criado pelo cientista. `CalcWorkflow` é o nome do *workflow*, representado por meio do atributo `name` (linha 1), seguido dos *namespaces* dos parceiros envolvidos (linhas 3 a 7). O mapeamento da declaração do *workflow* e dos *namespaces* é realizado, respectivamente, pelos elementos `Workflow` e `Namespace` do modelo estático. Em seguida as importações dos documentos WSDL dos parceiros envolvidos é apresentada pelas *tags* `import` (linhas 8 a 12) e que são mapeadas a partir do elemento `Import` do modelo estático. O atributo `location` da *tag* `import` define os documentos a serem importados, `CalcWorkflow.wsdl` e `ProvenanceService.wsdl`.

Os parceiros envolvidos na execução do *workflow* são agrupados pela *tag* `partnerLinks` (linha 14). Cada parceiro possui uma declaração denominada `partnerLink` (linhas 15, 17 e 19) identificada pelo atributo `name`, bem como da referência ao `partnerLinkType` definido no documento WSDL mostrado na Listagem 3. Os parceiros do *workflow* são `client` (o próprio *workflow*), `ProvenancePortTypeProvider` (serviço web de proveniência de dados) e `CalculadoraProvider` (serviço web da calculadora). O papel do parceiro é caracterizado pelos atributos `myRole` ou `partnerRole`. O atributo `myRole` (linha 15) representa o papel do *workflow* exercido com o parceiro. Por outro lado, o papel do parceiro envolvido no *workflow* é representado pelo atributo `partnerRole` (linhas 18 e 20). As declarações dos parceiros são mapeadas a partir dos elementos `PartnerLink`, `PartnerLinkType` e `PLTRole` do modelo estático.

A declaração das variáveis que o *workflow* utiliza para trocar as mensagens com os seus parceiros é mostrada na Listagem 6. As variáveis referentes a cada parceiro do *workflow* são declaradas pela *tag* `variable`. Cada variável possui um nome único de identificação provida pelo atributo `name` e a declaração do tipo de sua mensagem por meio do atributo `messageType`. Neste exemplo, temos 3 (três) variáveis que pertencem: (i) à atividade de instanciação do *workflow* (`input` linha 23); e (ii) à invocação do serviço web que calcula a soma de dois valores inteiros (`CalculadoraPLRequest` e `CalculadoraPLResponse`, linhas 46 e 48). As demais variáveis apresentadas no *workflow* estão relacionadas à proveniência de dados. A variável que representa o resultado do *workflow* é a mesma de captura da proveniência de saída do *workflow*, `ProvenancePortTypeoutputProvenancePLRequest` (linha 52). Esta variável

corresponde a uma das mensagens de entrada para a operação do serviço web de proveniência de dados. Desta forma, não é necessário declarar uma variável específica para retornar o resultado do *workflow*. As variáveis do *workflow* declaradas em WS-BPEL são mapeadas a partir dos elementos `Variable` e `Message` do modelo estático.

```

22 <bpel:variables>
23   <bpel:variable messageType="tns:CalcWorkflowProcessRequestMessage"
24     name="input" />
25   <bpel:variable
26     messageType="provenanceServiceprovenance:inputProvenanceResponseMessage"
27     name="workflowInstanceIdVariable" />
28   <bpel:variable
29     messageType="provenanceServiceprovenance: faultProvenanceRequestMessage"
30     name="ProvenancePortTypefaultProvenancePLRequest" />
31   <bpel:variable
32     messageType="provenanceServiceprovenance: faultProvenanceResponseMessage"
33     name="ProvenancePortTypefaultProvenancePLResponse" />
34   <bpel:variable
35     messageType="provenanceServiceprovenance: startProvenanceRequestMessage"
36     name="ProvenancePortTypestartProvenancePLRequest" />
37   <bpel:variable
38     messageType="provenanceServiceprovenance: startProvenanceResponseMessage"
39     name="ProvenancePortTypestartProvenancePLResponse" />
40   <bpel:variable
41     messageType="provenanceServiceprovenance: inputProvenanceRequestMessage"
42     name="ProvenancePortTypeinputProvenancePLRequest" />
43   <bpel:variable
44     messageType="provenanceServiceprovenance: inputProvenanceResponseMessage"
45     name="ProvenancePortTypeinputProvenancePLResponse" />
46   <bpel:variable messageType="implCalculadora:addRequest"
47     name="CalculadoraaddPLRequest" />
48   <bpel:variable messageType="implCalculadora:addResponse"
49     name="CalculadoraaddPLResponse" />
50   <bpel:variable
51     messageType="provenanceServiceprovenance: outputProvenanceRequestMessage"
52     name="ProvenancePortTypeoutputProvenancePLRequest" />
53   <bpel:variable
54     messageType="provenanceServiceprovenance: outputProvenanceResponseMessage"
55     name="ProvenancePortTypeoutputProvenancePLResponse" />
56 </bpel:variables>

```

Listagem 6. Declaração das variáveis do workflow

A instanciação de um *workflow* ocorre através da *tag* `receive`, que é responsável por aguardar uma requisição de um serviço web cliente conforme mostra a Listagem 7.

```

89 <bpel:sequence>
90   <bpel:receive createInstance="yes" name="CalcWorkflowReceive"
91     operation="process" partnerLink="client" portType="tns:CalcWorkflowPortType"
92     variable="input" />
93   ...

```

Listagem 7. Atividade de instanciação do workflow em WS-BPEL

A atividade `receive` da linguagem WS-BPEL tem seus atributos mapeados a partir dos elementos `Workflow`, `Variable`, `PortType`, `Operation` e `PartnerLink`. Esta atividade sempre é a primeira a ser executada por qualquer *workflow*, pois ela cria uma nova instância do mesmo por meio do atributo `createInstance` configurado com o valor `yes`. Em seguida é adicionada a atividade de proveniência dos artefatos de entrada do *workflow*, que representa uma atividade de invocação de serviço web expresso em WS-BPEL por meio da *tag* `invoke`. Entretanto, não é apresentada neste exemplo por se assemelhar à

proveniência de uma atividade de software do *workflow*. A proveniência de entrada do *workflow* pode ser vista no Apêndice C.

A Listagem 8 mostra a inicialização da variável de proveniência para a atividade de software configurada com serviço web Calculadora responsável pela soma de dois inteiros.

```

245 <bpel:assign validate="no">
246   <bpel:copy>
247     <bpel:from>
248       <bpel:literal>
249         <provenanceService:inputProvenanceRequest
250           xmlns:provenanceService="http://provenance.clusterflow.din.uem.br"
251           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
252           <provenanceService:entity />
253           <provenanceService:entityId />
254           <provenanceService:enactmentId />
255           <provenanceService:inputProvenance>
256             <provenanceService:key />
257             <provenanceService:value />
258           </provenanceService:inputProvenance>
259           <provenanceService:inputProvenance>
260             <provenanceService:key />
261             <provenanceService:value />
262           </provenanceService:inputProvenance>
263         </provenanceService:inputProvenanceRequest>
264       </bpel:literal>
265     </bpel:from>
266     <bpel:to part="parameters" variable="ProvenancePortTypeinputProvenancePLRequest" />
267   </bpel:copy>
268   <bpel:copy>
269     ...
270   </bpel:copy>
271 </bpel:assign>

```

Listagem 8. Inicialização da variável de entrada da proveniência do serviço web Calculadora

Nesta listagem o cabeçalho da mensagem de entrada da proveniência é representado pela *tag* `inputProvenanceRequest` (linha 249). Esta *tag* é mapeada a partir do elemento `Element` do modelo estático, que deve ser do tipo complexo, pois este contém os demais artefatos para invocar o serviço de proveniência. Esta mensagem possui os artefatos descritos a seguir:

- entity: identifica se a proveniência é oriunda do *workflow* ou de uma atividade deste *workflow*. Mapeado dos elementos `Workflow` ou `Activity` do modelo estático;
- entityId: caracteriza o valor do identificador do artefato `entity`. Mapeado do elemento `Workflow` ou `Activity` do modelo estático;
- enactmentId: caracteriza o valor do identificador de execução da instância do *workflow* e é mapeado a partir do elemento `Enactment` do modelo estático;

- key: representa o identificador do artefato de entrada da atividade que será realizada a proveniência. O elemento `Element` do modelo estático é mapeado durante a exportação do *workflow*.
- value: recebe o valor que deverá ser armazenado no banco de dados.

Nesta inicialização apenas são definidos os artefatos que compõem a mensagem (linhas 249 a 263). Esta mensagem é atribuída à variável de entrada do serviço web de proveniência denominada `ProvenancePortTypeinputProvenancePLRequest` (linha 266). A atribuição de valores aos artefatos desta variável pode ser vista no Apêndice C. A Listagem 9 mostra a chamada ao serviço web responsável pela persistência da proveniência de dados. A variável de entrada do serviço de proveniência que contém os artefatos inicializados e seus respectivos valores atribuídos é referenciada pelo atributo `inputVariable` na linha (326).

```
326 <bpel:invoke inputVariable="ProvenancePortTypeinputProvenancePLRequest"
327     name="InputProvenanceInvoke" operation="inputProvenance"
328     outputVariable="ProvenancePortTypeinputProvenancePLResponse"
329     partnerLink="ProvenancePortTypePL" portType="provenanceServiceprovenance:ProvenancePortType" />
```

Listagem 9. Chamada ao serviço web para a persistência da proveniência de dados

Em paralelo à proveniência de dados é realizada a chamada do serviço web Calculadora. Este paralelismo representa a atividade composta mostrada na Figura 27. Assim, a manipulação da variável de entrada e a invocação do serviço web Calculadora é realizada conforme a Listagem 10. Nesta listagem a atividade de software definida pelo cientista durante criação do *workflow* corresponde à atividade `invoke` mostrada na linha 247. Primeiramente é realizada a inicialização da variável de entrada do serviço web Calculadora (linhas 210 a 221), que é baseada na descrição deste serviço por meio de seu documento WSDL. Em seguida, a ilustração mostra a atribuição de valores para os artefatos da atividade.

As *tags from* (linhas 223 e 235) representam as origens dos artefatos e são mapeadas pelos elementos `Variable`, `Part` e `Element` do modelo estático. Além destes, o elemento `Transition` define qual é o elemento `Element` de origem do artefato (linhas 225 e 237). Por outro lado, as *tags to* (linhas 228 e 240) definem os destinos dos artefatos de origem, que são mapeados pelos elementos `Variable`, `Part` e `Element` do modelo estático. Para as *tags to*, o elemento `Transition` define os destinos dos artefatos (linhas 230 e 242). Após a manipulação da variável de entrada o serviço web Calculadora é invocado (linha 247).

O mapeamento para a *tag* `invoke` que envia uma mensagem de requisição para o serviço web `Calculadora` parte do elemento `SoftwareActivity` do modelo estático, e além deste possui os elementos `Variable`, `PortType`, `Operation` e `PartnerLink` mapeados durante a exportação e geração do *workflow*. O resultado da execução deste serviço web é retornado por meio da variável `CalculadoraaddPLResponse` declarada pelo atributo `outputVariable` (linha 249). Este atributo somente é utilizado para os serviços web que executam de maneira síncrona, o que exige que a execução do *workflow* aguarde o resultado do serviço web para então continuar a sua execução.

```

209 <bpel:assign validate="no">
210   <bpel:copy>
211     <bpel:from>
212       <bpel:literal>
213         <impl:add xmlns:impl="http://huff.calc.les.din.uem.br"
214           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
215           <impl:value1 />
216           <impl:value2 />
217         </impl:add>
218       </bpel:literal>
219     </bpel:from>
220     <bpel:to part="parameters" variable="CalculadoraaddPLRequest" />
221   </bpel:copy>
222   <bpel:copy>
223     <bpel:from part="payload" variable="input">
224       <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
225         <![CDATA[tns:artefact1]]>
226       </bpel:query>
227     </bpel:from>
228     <bpel:to part="parameters" variable="CalculadoraaddPLRequest">
229       <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
230         <![CDATA[implCalculadora:value1]]>
231       </bpel:query>
232     </bpel:to>
233   </bpel:copy>
234   <bpel:copy>
235     <bpel:from part="payload" variable="input">
236       <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
237         <![CDATA[tns:artefact2]]>
238       </bpel:query>
239     </bpel:from>
240     <bpel:to part="parameters" variable="CalculadoraaddPLRequest">
241       <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
242         <![CDATA[implCalculadora:value2]]>
243       </bpel:query>
244     </bpel:to>
245   </bpel:copy>
246 </bpel:assign>
247 <bpel:invoke inputVariable="CalculadoraaddPLRequest"
248   name="AddSoftwareActivityInvoke" operation="add"
249   outputVariable="CalculadoraaddPLResponse" partnerLink="CalculadoraPL"
250   portType="implCalculadora:Calculadora" />

```

Listagem 10. Manipulação de variável e invocação do serviço web `Calculadora`

A proveniência de dados do resultado da atividade de software ilustrada pela Figura 29 e representada por meio da ação `Proveniência de Saída da Atividade`, ocorre assim que as execuções em paralelo das atividades anteriores estiverem completadas. A manipulação dos artefatos da variável de entrada desta atividade e a invocação do serviço web correspondente não são mostradas neste capítulo. Entretanto, podem ser vistas no *workflow* mostrado no Apêndice C.

O fato de o *workflow* ser instanciado de forma assíncrona e não possuir variável que informe o seu resultado, exige que a proveniência de dados seja aplicada para representar os artefatos de saída do *workflow*. A Listagem 11 mostra a inicialização dos artefatos que compõem a mensagem de proveniência de saída do *workflow*.

```

387 <bpel:assign validate="no">
388   <bpel:copy>
389     <bpel:from>
390       <bpel:literal>
391         <provenanceService:outputProvenanceRequest
392           xmlns:provenanceService="http://provenance.clusterflow.din.uem.br"
393           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
394           <provenanceService:instanceId />
395           <provenanceService:outputProvenance>
396             <provenanceService:key />
397             <provenanceService:value />
398           </provenanceService:outputProvenance>
399         </provenanceService:outputProvenanceRequest>
400       </bpel:literal>
401     </bpel:from>
402     <bpel:to part="parameters" variable="ProvenancePortTypeoutputProvenancePLRequest" />
403   </bpel:copy>

```

Listagem 11: Inicialização dos artefatos da mensagem de proveniência de saída do *workflow*

Nesta listagem pode-se observar que a mensagem que será enviada ao serviço web de proveniência foi definida nas linhas 391 a 399. Os artefatos que compõem a mensagem e seus mapeamentos já foram abordados anteriormente. A linha 402 mostra a variável `ProvenancePortTypeoutputProvenancePLRequest` que recebe a mensagem inicializada e foi declarada por meio do atributo `variable`. A Listagem 12 apresenta a manipulação dos valores desta mensagem.

```

404   <bpel:copy>
405     <bpel:from part="parameters" variable="workflowInstanceIdVariable">
406       <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:subLang:xpath1.0">
407         <![CDATA[provenanceServiceprovenance:instanceId]]>
408       </bpel:query>
409     </bpel:from>
410     <bpel:to part="parameters" variable="ProvenancePortTypeoutputProvenancePLRequest">
411       <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:subLang:xpath1.0">
412         <![CDATA[provenanceServiceprovenance:instanceId]]>
413       </bpel:query>
414     </bpel:to>
415   </bpel:copy>
416   <bpel:copy>
417     <bpel:from>
418       <bpel:literal>90</bpel:literal>
419     </bpel:from>
420     <bpel:to part="parameters" variable="ProvenancePortTypeoutputProvenancePLRequest">
421       <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:subLang:xpath1.0">
422         <![CDATA[provenanceServiceprovenance:outputProvenance[1]/provenanceServiceprovenance:key]]>
423       </bpel:query>
424     </bpel:to>
425   </bpel:copy>
426   <bpel:copy>
427     <bpel:from part="parameters" variable="CalculadoraaddPLResponse">
428       <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:subLang:xpath1.0">
429         <![CDATA[implCalculadora:addReturn]]>
430       </bpel:query>
431     </bpel:from>
432     <bpel:to part="parameters" variable="ProvenancePortTypeoutputProvenancePLRequest">
433       <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:subLang:xpath1.0">
434         <![CDATA[provenanceServiceprovenance:outputProvenance[1]/provenanceServiceprovenance:value]]>
435       </bpel:query>
436     </bpel:to>
437   </bpel:copy>
438 </bpel:assign>

```

Listagem 12. Manipulação da mensagem da variável de proveniência de saída do *workflow*

Os artefatos desta mensagem recebem seus respectivos valores de acordo com a definição dos artefatos de saída do *workflow*. Assim, o artefato `instanceId` definido na linha 394 da Listagem 11 recebe o valor contido na variável `workflowInstanceIdVariable` na linha 405 da Listagem 12. Este valor é atribuído ao artefato `instanceId` da variável de entrada da proveniência apresentada na linha 410. O valor deste artefato é mapeado a partir do elemento `Instance` do modelo estático. A linha 418 mostra o valor “90” que será atribuído ao artefato `key` na linha 422. O valor atribuído a este artefato é mapeado por meio do identificador do elemento `Element` do modelo estático. O valor do dado da proveniência de saída do *workflow* é atribuído a partir da variável de saída `addReturn` do serviço web `Calculadora`, e é apresentado na linha 429. Este valor é atribuído ao artefato `value` que é mostrado na linha 434, e pertence à variável de entrada da proveniência definida na linha 432. Assim que a mensagem da variável de proveniência estiver com seus dados corretamente preenchidos, o serviço web de proveniência é invocado. O trecho do *workflow* responsável por esta funcionalidade é mostrado na Listagem 13.

```

439     <bpel:invoke inputVariable="ProvenancePortTypeoutputProvenancePLRequest"
440                 name="OutputProvenanceInvoke" operation="outputProvenance"
441                 outputVariable="ProvenancePortTypeoutputProvenancePLResponse"
442                 partnerLink="ProvenancePortTypePL"
443                 portType="provenanceServiceprovenance:ProvenancePortType" />
444   </bpel:sequence>
445</bpel:process>

```

Listagem 13. Invocação do serviço web de proveniência de saída do workflow

Este trecho de código do *workflow* apresenta a última atividade (linha 439) que todos os *workflows* exportados a partir do ambiente *ClusterFlow* executam, a proveniência dos dados de saída do *workflow*. Além disso, pode-se perceber que a linha 445 encerra a definição do *workflow* iniciada na linha 1 da Listagem 5.

5.6. Considerações Finais

Este capítulo apresentou as principais estratégias de implementação do protótipo do ambiente *ClusterFlow*. Essas estratégias foram utilizadas para implementar o núcleo do protótipo do ambiente. O modelo estático definido para o ambiente foi subdividido de acordo com as especificações XSD, WSDL e WS-BPEL, que são geradas pelo protótipo do *ClusterFlow*. Os modelos estáticos apresentados neste capítulo apóiam a modelagem dos dados e permitem que os *workflows* armazenados no banco de dados relacional sejam exportados para a execução.

Um *workflow* de exemplo com apenas uma atividade (Calculadora) definida pelo cientista foi usado para facilitar a descrição e o entendimento da relação entre o modelo estático e essas especificações, o que não é trivial. O modelo estático não oferece apoio a todas as definições das especificações XSD, WSDL e WS-BPEL. Contudo, ele está de acordo com as necessidades do ambiente *ClusterFlow*. A exportação dos *workflows* é realizada apenas para definições de atividades de software, porém, foram definidos alguns elementos no modelo estático que permitem a definição de atividades humanas, notificações, e reuso de *workflows* como um tipo especial de atividade (*SubWorkflow*). O modelo estático também apóia o reuso da definição de tipos complexos de dados. No entanto, esta funcionalidade não foi desenvolvida devido à complexidade e o tempo necessário para a sua implementação.

Exemplo de Aplicação

6.1. Considerações Iniciais

Este capítulo visa apresentar um exemplo de aplicação do ambiente *ClusterFlow* que explora suas funcionalidades bem como as estratégias de implementação apresentadas no Capítulo 5. Além disso, o capítulo apresenta o nível de abstração de comandos que o pesquisador necessita para executar experimentos no *cluster* de computadores. Inicialmente, é apresentada a definição do experimento alvo que pode ser aplicado à resolução dos problemas de automação para a escala de trabalho de funcionários em grandes organizações. Na sequência, é descrito o serviço web desenvolvido para acessar o computador com a aplicação de alto desempenho que será executada. O diagrama de atividades que representa o *workflow* concebido a partir do protótipo bem como o apoio computacional que o *ClusterFlow* oferece aos cientistas para conceberem, executarem e visualizarem os resultados dos experimentos, são apresentados na sequência. A avaliação qualitativa do apoio que o *ClusterFlow* oferece aos cientistas com base no ciclo de vida de um experimento científico também é discutida. Por fim, as considerações finais do capítulo são apresentadas.

6.2. Definição do Domínio do Experimento Científico

O domínio do experimento científico usado como exemplo de aplicação é o de investigação da meta-heurística de Otimização por Colônia de Formigas Artificiais (*Ant Colony Optimization* - ACO), que tem o objetivo de obter algoritmos mais eficientes para a resolução

de problemas NP-Difíceis. Meta-heurística pode ser definida como uma estrutura genérica para o desenvolvimento e aplicação de algoritmos heurísticos. Esses algoritmos não garantem uma solução ótima, porém, avaliações empíricas mostram que eles oferecem uma solução boa o suficiente para determinados problemas em um tempo de computação aceitável (MULATI, 2009). Assim, este experimento visa contribuir na implementação de uma aplicação de alto desempenho e avaliação de resultados sobre o problema de cobertura de conjunto (PCC), o qual pertence aos problemas que não se modificam enquanto estão sendo resolvidos.

6.3. Desenvolvimento do Serviço Web de Acesso ao Cluster

O desenvolvimento dos serviços web que farão o acesso ao *cluster* é uma fase necessária para que os *workflows* científicos e aplicações de alto desempenho possam ser executados de forma conjunta. Os serviços web devem ser desenvolvidos para cada nova atividade de software que se deseja disponibilizar aos pesquisadores. Para isso, estes serviços devem obedecer às notações exigidas pelas aplicações de alto desempenho, tais como, parâmetros de entrada e saída, ordem de execução de comandos, bem como possuírem uma conta no *cluster* com permissões de acesso e execução dos comandos necessários. Geralmente as configurações e resultados dessas aplicações são acessados por meio de arquivos texto.

Assim, o experimento do problema de cobertura de conjunto por colônia de formigas artificiais desenvolvido por (MULATI, 2009) e utilizado neste exemplo de aplicação é realizado em duas etapas. Na primeira, é realizada a configuração da aplicação na qual os parâmetros de entrada são inseridos em um arquivo texto. Um conjunto de instruções de execução é gerado a partir deste arquivo e são armazenadas em outro arquivo que possui permissões para a execução. A execução do arquivo com o conjunto de instruções corresponde à segunda etapa do experimento. O arquivo executável é responsável por disparar as diferentes instâncias de execução as quais correspondem aos problemas que se deseja resolver, que estão definidos na base de dados da *OR Library* (BEASLEY, 1990). Além disso, a segunda etapa também corresponde à coleta dos resultados da execução realizada, que são retornados em formato de arquivo texto.

O serviço web desenvolvido para abstrair os comandos do *cluster* e o conjunto de instruções de execução necessárias segue a lógica das duas etapas definidas para o experimento. Cada etapa definida corresponde a uma operação do serviço web. Assim, temos

uma operação que recebe os artefatos de entrada da Máquina WS-BPEL e cria o arquivo de configuração com base nestes artefatos. Esta mesma operação executa um programa que tem a função de criar um arquivo executável com o conjunto de instruções para a execução do experimento. O nome deste arquivo criado é retornado por esta operação do serviço web. A segunda operação do serviço web desenvolvido recebe como artefato de entrada o nome do arquivo retornado pela operação anterior, e inicia a execução do experimento. Assim que a execução encerrar os dados armazenados no arquivo texto são retornados pela segunda operação do serviço web e enviados à Máquina WS-BPEL do ambiente *ClusterFlow*. Logo, a Máquina WS-BPEL invoca o serviço web de proveniência de dados para armazenar o resultado da execução no banco de dados. O serviço web desenvolvido conecta-se ao computador responsável por executar o experimento através de uma conexão SSH (*Secure Shell*).

Para a utilização deste experimento pelos pesquisadores foi necessário a adição do serviço desenvolvido ao repositório de serviços web do ambiente *ClusterFlow*. Este processo de disponibilização do serviço é realizado acrescentando o documento WSDL do serviço em um diretório específico do *ClusterFlow*, que é usado como repositório de serviços web. Todos os serviços que os pesquisadores necessitarem para conceberem seus *workflows* devem estar presentes neste diretório. Futuramente este repositório de serviços pode ser desenvolvido para fazer o uso do repositório de serviços UDDI, que permite oferecer um mecanismo padronizado de disponibilização e consulta de serviços web.

6.4. Definição do *Workflow* para o Experimento Científico

Uma vez que todos os serviços web necessários para conceber o *workflow* estejam disponíveis no repositório de serviços, o cientista pode iniciar a construção do *workflow*. O *workflow* exemplo é apresentado na Figura 33 e consiste de quatro atividades. As atividades de manipulação de variáveis, tratamento de falhas e proveniência de dados não foram incluídas nesta ilustração, exceto na última atividade do *workflow* que armazena o resultado obtido no banco de dados. Esta atividade foi incluída nesta figura, pois sempre será a última atividade a ser executada em todos os *workflows* criados a partir do *ClusterFlow* e corresponde à atividade de encerramento do mesmo.

O ciclo seguido para a construção do *workflow* no *ClusterFlow* é definido de acordo com o processo para a construção de *workflows* científicos descrito na Seção 4.5. Assim, a primeira ação que um cientista deve realizar é criar um novo *workflow* científico. Deste modo,

a tela do protótipo que corresponde à ação de criar um novo *workflow* científico é mostrada na Figura 34.

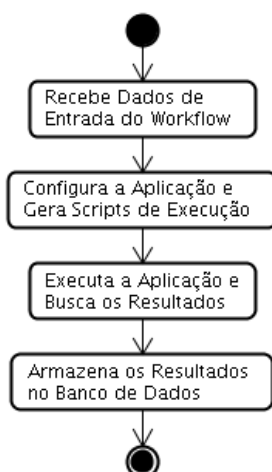


Figura 33. Workflow para realização do exemplo de aplicação

Figura 34. Tela para criar novos workflows

A tela para criar novos *workflows* corresponde à definição da primeira atividade do *workflow* ilustrado na Figura 33, que aguarda uma requisição de execução e os artefatos de entrada do *workflow*. Com esta tela o pesquisador pode tanto criar um *workflow* totalmente novo, como também pode fazer o reuso das definições de seus próprios *workflows* ou daqueles que foram compartilhados por outros cientistas. As telas utilizadas para reutilizar *workflows* e definir os artefatos de entrada do *workflow* são mostradas no Apêndice E. Para este exemplo de aplicação considera-se que o pesquisador deseja criar um *workflow* totalmente novo. Assim, o cientista define um nome para o *workflow* por meio do campo Name. Em seguida, os objetivos da execução do *workflow* são descritos no campo Goals com o intuito de fornecer uma descrição breve e concisa do que o *workflow* se propõe a realizar. Na sequência, o pesquisador pode informar a descrição geral do *workflow* no campo

Description. Na Figura 34 aparecem os artefatos necessários para a invocação dos serviços web que executam as atividades de software deste experimento. As atividades de software são: (i) criar arquivo de configuração com base nos artefatos informados no *workflow* e gerar o conjunto de instruções de execução; e, (ii) executar o experimento e buscar o resultado obtido nesta execução. A Figura 35 mostra a tela que apresenta a atividade inicial do *workflow* já definida pelo cientista.

Workflow Details ✎ ✕

Name: Ant Experiment

Goals: The workflow's goal is to abstract the commands to run the Ant Application

Description: The first step is to create the variables from this workflow. In the second, these artefacts created must to be forwarded to the Configure Ant Application activity. The last activity should run the script returned by the Configure Ant Application.

Input Artefacts: [problemsToSolve](#) [numberOfAttempts](#) [alpha](#) [numberOfAnts](#) [beta](#) [rho](#)

Next Activities

New Activity
[Software Activity - People Activity - SubWorkflow](#)

Actions
[Create Notification - Define Workflow's End](#)

Workflow properties

Property	Value
Owner	A. Huff
Creation date	Monday, August 23, 2010 5:20:06 PM BRT
Last modified	Monday, August 23, 2010 9:29:43 PM BRT

Figura 35. Tela que apresenta a definição do *workflow*

Nesta tela pode-se perceber que o atributo `Next Activities` não possui atividade configurada. Este atributo representa a próxima atividade que deve ser executada, ou se existir mais de uma atividade, estas serão executadas paralelamente. Todas as telas que apresentam as definições de atividades seguem este mesmo padrão.

As atividades de um *workflow* devem ser definidas em sequência. A atividade inicial do exemplo de aplicação é a geração do arquivo de configuração e do conjunto de instruções de execução da aplicação alvo, neste caso *Ant Application*. Assim, uma atividade de software deve ser definida para realizar a invocação do serviço web que foi descrito na Seção 6.3. A tela para a definição de uma atividade de software pode ser acessada por meio do *link Software Activity* na tela que apresenta a definição do *workflow*. A definição desta atividade é ilustrada na Figura 36.

Create Software Activity

Name: Activity Reuse

Goals:

Description:

Software Activity Description:
 This service run in a cluster of computers. Its artefacts are:
 * problems: can be from scp401 to scp410 and they must be separated by a space
 * antAmount: number of ants
 * alpha: reveal the pheromone importance on a path construction by an ant
 * beta: reveal the heuristic information importance on the path construction by an ant
 * rho: shows the pheromone rate evaporation
 * attemptsNumber: number of attempts to get better results
 * antApplicationScriptName: the script name to run the Ant Application

Input Artefacts: **problems alpha antAmount beta rho attemptsNumber**
 Output Artefacts: **antApplicationScriptGenerated**

Activity's Input Data Binding

Artefacts to bind:

- **problems** < Workflow Input / problemsToSolve ✗
- **alpha** < Workflow Input / alpha ✗
- **antAmount** < Workflow Input / numberOfAnts ✗
- **beta** < Workflow Input / beta ✗
- **rho** < Workflow Input / rho ✗
- **attemptsNumber** < Workflow Input / numberOfAttempts ✗

Figura 36. Definição da atividade de software que configura a aplicação a ser executada

Esta tela possui alguns campos que devem ser configurados para que o serviço web desejado seja invocado. Duas caixas de seleção suspensa são usadas para selecionar, respectivamente, o serviço web desejado e a operação deste serviço web. Com base nestas seleções, o ambiente apresenta ao cientista uma descrição do que o serviço se propõe a realizar. Esta descrição está definida no documento WSDL de cada serviço utilizado. Neste caso pode-se observar que as descrições dos artefatos de entrada são apresentadas ao pesquisador através do campo `Software Activity Description`. Em atividades de software o pesquisador não necessita criar os artefatos de entrada e saída da mesma, pois o *ClusterFlow* importa automaticamente do documento WSDL do serviço web selecionado. Entretanto, os mapeamentos dos artefatos de entrada devem ser realizados manualmente. O campo `Artefacts to bind` apresenta todos os artefatos a serem mapeados. Para explicar o mapeamento de um artefato usaremos como exemplo o artefato `problems`. A Figura 37 mostra a visualização dos artefatos antes do mapeamento.

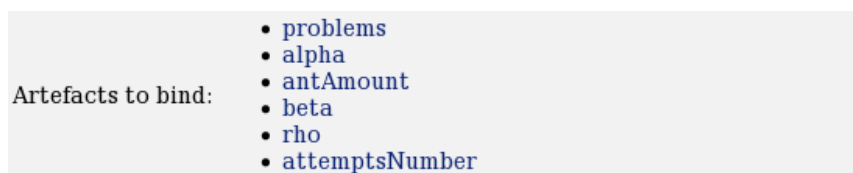


Figura 37. Artefatos antes do mapeamento

Para mapear o artefato `problems`, o pesquisador deve clicar sobre o artefato e realizar o mapeamento da origem dos dados conforme mostra a Figura 38.

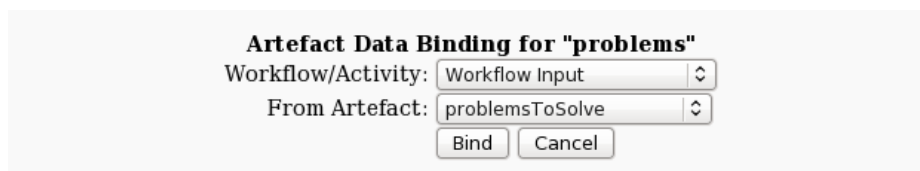


Figura 38. Mapeamento do artefato `problems`

A partir da tela para o mapeamento do artefato `problems` o pesquisador tem a possibilidade de escolher a origem do artefato. Esta origem pode ser tanto dos artefatos de instanciação do *workflow* bem como dos artefatos de saída das atividades deste *workflow*. A escolha da origem é realizada por meio do campo `Workflow/Activity`. O campo `From Artefact` é completado com os artefatos da atividade ou do *workflow* que foi selecionado no campo anterior. Contudo, este campo será preenchido apenas com os artefatos que possuírem o mesmo tipo de dados definidos, por exemplo, para mapear o artefato `problems` são buscados os artefatos do tipo “texto”, pois `problems` está definido como tipo texto no documento WSDL. Por fim, o cientista seleciona o artefato de origem. Neste exemplo o artefato `problems` da atividade de software é mapeado com o artefato `problemsToSolve` que pertence aos artefatos de entrada do *workflow*. A Figura 39 mostra o artefato `problems` após o mapeamento.

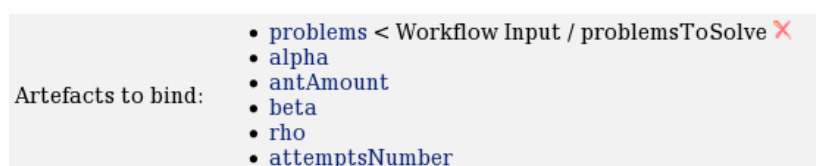


Figura 39. Artefato `problems` após o mapeamento

Após o mapeamento do artefato pode-se visualizar a origem e o artefato mapeado. Nesta ilustração o artefato de entrada `problems` da atividade de software criada está sendo mapeado a partir do artefato de entrada `problemsToSolve`, que foi definido no *workflow*. A visualização deste mapeamento está sendo ilustrado na Figura 39 por meio da representação

textual “problems < Workflow Input / problemsToSolve”. A Figura 40 mostra a tela que apresenta os detalhes da atividade de software Configure Ant Application que foi criada pelo cientista.

Software Activity Details ✕

Name: Configure Ant Application

Goals: This activity configures the Ant Application

Description: This activity generates the configuration file to the Ant Application and the execution script based on this configuration file

Input Artefacts: beta rho antAmount alpha attemptsNumber problems

Output Artefacts: antApplicationScriptGenerated

Activity's Input Data Binding	
Artefacts Bound:	<ul style="list-style-type: none"> problems < Ant Experiment / problemsToSolve attemptsNumber < Ant Experiment / numberOfAttempts alpha < Ant Experiment / alpha rho < Ant Experiment / rho beta < Ant Experiment / beta antAmount < Ant Experiment / numberOfAnts

Next Activities

New Activity
[Software Activity - People Activity - SubWorkflow](#)

Actions
[Create Notification](#)

Activity properties

Property	Value
Creation date	Monday, August 23, 2010 8:06:23 PM BRT
Last modified	

Figura 40. Detalhes da atividade de software criada pelo cientista

Esta tela mostra as informações definidas após a criação da atividade de software, porém, ela não oferece possibilidade de alteração. Caso seja necessária alguma alteração, esta deve ser excluída e uma nova atividade deverá ser configurada. A alteração de atividades de software ainda não foram implementadas. Os artefatos de entrada são apresentados junto de seus mapeamentos.

De acordo com o *workflow* definido na Figura 33, as atividades devem ser realizadas sequencialmente. Neste momento, já foi definida a configuração da aplicação que o experimento executará, mas é necessário definir a próxima atividade do *workflow* que representa a operação do serviço web responsável por disparar a execução da aplicação alvo remotamente. A Figura 41 mostra a tela com esta atividade já configurada.

Create Software Activity

Name: Activity Reuse

Goals:

Description:

AntExperimentService

runAntExperiment

Software Activity Description:
 This service run in a cluster of computers. Its artefacts are:
 * problems: can be from scp401 to scp410 and they must be separated by a space
 * antAmount: number of ants
 * alpha: reveal the pheromone importance on a path construction by an ant
 * beta: reveal the heuristic information importance on the path construction by an ant
 * rho: shows the pheromone rate evaporation
 * attemptsNumber: number of attempts to get better results
 * antApplicationScriptName: the script name to run the Ant Application

Software Activity:

Input Artefacts:

Output Artefacts:

Activity's Input Data Binding

Artefacts to bind:

- < Configure Ant Application / antApplicationScriptGenerated ✖

Figura 41. Definição da atividade de software que dispara a execução da aplicação remota

A configuração desta atividade ocorre de maneira análoga à atividade de software definida anteriormente. Porém, para iniciar a execução da aplicação remota o cientista deve escolher a operação `runAntExperiment` do serviço web. Esta operação possui apenas um artefato de entrada denominado `antApplicationScriptName`, o qual recebe o nome do arquivo com as instruções de execução da aplicação remota gerado na atividade anterior. Além disso, o mapeamento deste artefato foi realizado a partir da saída da atividade anterior como pode ser observada no atributo `Artefacts to bind`. Os outros detalhes de configuração desta atividade são semelhantes a todas as atividades de software. Por fim, deve ser configurado o encerramento do *workflow* que é ilustrado pela Figura 42.

Workflow's End Definition

Description:

Output Artefacts: ✖

Workflow's Output Data Binding

Artefacts to bind:

- < Run Ant Application / result ✖

Figura 42. Tela para definir a atividade de encerramento do workflow

Esta tela representa a última atividade a ser realizada pelo *workflow*. Nela são definidos os artefatos de saída mais relevantes para o *workflow*, embora os resultados de cada atividade possam ser visualizados separadamente devido à realização da proveniência de dados durante a execução do *workflow*. A tela permite que o cientista defina alguma descrição que possa ser relevante para o entendimento da finalização do *workflow*, bem como possibilita que o pesquisador crie os artefatos de saída e os mapeie com os resultados de atividades anteriores. Os artefatos de saída criados nesta tela são usados para gerar a proveniência que corresponde à última atividade do diagrama definido na Figura 33. Após a atividade de encerramento do *workflow*, este pode ser opcionalmente compartilhado, exportado e implantado na Máquina WS-BPEL.

O compartilhamento de *workflows* oferecido pelo *ClusterFlow* permite que cientistas publiquem seus *workflows* de maneira rápida e simplificada. Uma vez o *workflow* compartilhado, seus dados capturados pelo mecanismo de proveniência também se tornam visíveis aos outros pesquisadores. Este compartilhamento dos dados de proveniência unido com o compartilhamento do *workflow* foi adotado, pois *workflows* com execução de longa duração não precisam ser executados novamente com os mesmos artefatos, permitindo que o cientista visualize antecipadamente os resultados obtidos a partir de determinados artefatos de entrada. A Figura 43 mostra a tela usada para realizar o compartilhamento, exportação, implantação e instanciação de *workflows*.

My Workflow List			
Workflow	Goals	Owner	Options
Ant Experiment	The workflow's goal is to abstract the commands to run the Ant Application	A. Huff	<input checked="" type="checkbox"/> Share Deploy
Calc Workflow	The Calc Workflow's goal is to sum two integer numbers	A. Huff	<input type="checkbox"/> Share Undeploy Run

Figura 43. Tela de compartilhamento, exportação, implantação e instanciação de *workflows*

A tela desta figura também é usada para visualizar todos os *workflows* pertencentes ao cientista. Neste exemplo, observa-se que a caixa de verificação denominada Share da coluna Options aparece selecionada para o *workflow* Ant Experiment e significa que este *workflow* está compartilhado com os demais pesquisadores. Além disso, pode-se perceber que existem mais 3 (três) botões que são descritos a seguir:

- Deploy: usado para realizar a exportação e implantação do *workflow* na Máquina WS-BPEL. Este botão somente está habilitado enquanto o *workflow* não foi exportado e implantado na máquina de execução;

- Undeploy: utilizado para realizar o processo inverso ao botão Deploy. Sua funcionalidade é retirar o *workflow* da Máquina WS-BPEL. Somente está habilitado enquanto o *workflow* estiver implantado na máquina de execução; e
- Run: usado para instanciar os *workflows* implantados na máquina de execução de *workflows* e somente está habilitado enquanto o *workflow* estiver implantado e pronto para a execução na Máquina WS-BPEL.

6.5. Execução do *Workflow*

Para iniciar a execução do *workflow* é necessário que o cientista informe os artefatos de entrada requeridos e crie uma nova instância de execução do mesmo. Para isto, foi desenvolvida uma tela no ambiente *ClusterFlow* a qual permite que cientistas realizem esta ação. Como cada *workflow* implantado na Máquina WS-BPEL corresponde a um serviço web, a instanciação do *workflow* ocorre através da invocação destes por meio de um serviço web cliente, que foi desenvolvido de forma genérica e adaptável em tempo de execução a todos os *workflows* criados no *ClusterFlow*. A tela que realiza a instanciação do *workflow* através do serviço web cliente é ilustrada na Figura 44.

Run Workflow

Workflow Name: Ant Experiment

Input Artefacts

alpha:	<input type="text" value="1.0"/>	double
problemsToSolve:	<input type="text" value="scp401 scp405 scp407 scp410"/>	string
beta:	<input type="text" value="5.0"/>	double
numberOfAnts:	<input type="text" value="10"/>	int
numberOfAttempts:	<input type="text" value="2"/>	int
rho:	<input type="text" value="0.3"/>	double

Figura 44. Tela para instanciar os *workflows*

Esta tela mostra o exemplo de instanciação do *workflow* que foi criado e descrito na seção anterior. Os valores dos artefatos de entrada foram extraídos da literatura e de testes realizados por (MULATI, 2009). Os artefatos definidos na tela para criar novos *workflows* apresentada na Figura 34 e também presentes na tela para instanciar os *workflows* são descritos a seguir:

- alpha: indica a importância do feromônio na construção de um caminho por uma formiga;

- problemToSolve: define quais problemas a aplicação tenta resolver e podem variar de scp401 à scp410;
- beta: indica a importância da informação heurística na construção de um caminho por uma formiga;
- numberOfAnts: número de formigas usadas para resolver os problemas;
- numberOfAttempts: número de tentativas de execução da aplicação para obter melhor resultado; e
- rho: indica a taxa de evaporação do feromônio (MULATI, 2009).

Os artefatos apresentados na tela de instanciação de *workflow* são seguidos de sugestões, tais como, *string*, *double* e *int*, que informam ao cientista qual é o tipo de dado esperado para cada artefato de entrada. Uma vez que os campos dos artefatos estejam corretamente preenchidos, o *workflow* pode ser instanciado. Esta instanciação ocorre de maneira assíncrona e desta forma caso exista necessidade, o cientista pode fechar o navegador sem interferir na execução do *workflow*.

O estado de execução em que o *workflow* se encontra também pode ser visualizado pelos pesquisadores. Existem 3 (três) possíveis estados: (i) *In Progress*: indica que o *workflow* está em execução; (ii) *Completed*: informa que o *workflow* teve sua execução encerrada com sucesso; e, (iii) *Failed*: indica que ocorreu alguma falha durante a execução e o fluxo alternativo foi executado. Este fluxo alternativo é responsável apenas por alterar o estado do *workflow* para *Failed* e encerrar a execução do mesmo. A captura da causa da falha bem como da sua visualização no portal web ainda não foram implementadas no protótipo. A Figura 45 mostra a tela na qual o cientista pode visualizar o estado de execução em que o *workflow* se encontra.

Workflow Results					
Name: Ant Experiment					
Execution Times	Enactment Id	Status	Input Artefacts	Output Artefacts	Result Options
Start: Monday, August 23, 2010 10:17:34 PM BRT End:	46	In Progress...	alpha: 1.0 rho: 0.3 problemsToSolve: scp401 scp405 scp407 scp410 numberOfAnts: 10 numberOfAttempts: 2 beta: 5.0		<input type="button" value="Discard"/>

Figura 45. Tela de visualização do estado de execução dos workflows

Esta tela pode ser usada tanto para a visualização do estado de execução do *workflow* como também para a visualização detalhada da execução das atividades do mesmo. Para o

primeiro caso, a coluna *Status* representa o estado no qual o *workflow* se encontra no momento. Neste exemplo o *workflow* está em execução. Além do estado de execução existem outras informações que podem ser visualizadas a partir desta tela, tais como, informações que foram capturadas durante a execução e que correspondem à proveniência de dados. A visualização dessas informações é descrita na seção seguinte.

6.6. Visualização do Resultado do Experimento

A visualização do resultado da execução do *workflow* pode ser realizada enquanto o mesmo se encontra em qualquer um dos 3 (três) possíveis estados, pois o mecanismo de proveniência do *ClusterFlow* atua durante todo o processo de execução e é acionado no início e término da execução de cada atividade. Assim, resultados parciais da execução do *workflow* também podem ser visualizados e experimentos com *workflows* de longa duração podem ser avaliados durante a sua execução. A Figura 46 mostra a tela que permite a escolha da instância de execução do *workflow* a ser visualizada, bem como o exemplo do *workflow* com a execução encerrada com sucesso e outro em execução.

Nesta ilustração, pode-se perceber que o identificador da primeira instância de execução do *workflow* (*Enactment Id*) possui o mesmo valor (46) se comparado com a Figura 45, no entanto, o seu estado (*Status*) foi alterado, pois a execução foi encerrada. Além disso, esta tela mostra os instantes de início e fim da execução do *workflow* por meio da coluna *Execution Times*. Os artefatos de entrada representados pela coluna *Input Artefacts* bem como os artefatos de saída representados pela da coluna *Output Artefacts* também são mostrados. O pesquisador pode excluir as instâncias de execução dos *workflows*, tal operação é acionada pelo botão *Discard* da coluna *Result Options*. Para visualizar os detalhes de execução de uma instância de *workflow* o cientista necessita apenas navegar sobre a linha que corresponde à instância desejada e selecioná-la.

A Figura 47 mostra a visualização dos resultados da instância de execução do *workflow* que foram coletados pelo mecanismo de proveniência de dados. A visualização dos resultados oferecida por esta tela permite que o cientista avalie os resultados obtidos de forma simples, com navegação rápida em todas as atividades e com o uso de apenas uma tela. Além dos resultados, dos detalhes de início e fim dos tempos de execução e o identificador da instância do *workflow*, são mostrados os artefatos de entrada que o *workflow* recebeu. Esses artefatos foram definidos pelo cientista no momento da criação do *workflow*.

Workflow Results					
Name: Ant Experiment					
Execution Times	Enactment Id	Status	Input Artefacts	Output Artefacts	Result Options
Start: Monday, August 23, 2010 10:17:34 PM BRT End: Monday, August 23, 2010 10:18:10 PM BRT	46	Completed.	alpha: 1.0 rho: 0.3 problemsToSolve: scp401 scp405 scp407 scp410 numberOfAnts: 10 numberOfAttempts: 2 beta: 5.0	workflowResult: 431 514 432 514	Discard
Start: Monday, August 23, 2010 10:36:26 PM BRT End:	47	In Progress...	alpha: 2.0 problemsToSolve: scp402 scp403 scp406 numberOfAttempts: 2 rho: 0.5 beta: 4.0 numberOfAnts: 20		Discard

Figura 46. Tela para a escolha da instância de execução a ser visualizada

Instance Execution Details				
Name: Ant Experiment				
Execution Times	Enactment Id	Workflow Input	Workflow Output	Status
Start: Monday, August 23, 2010 10:17:34 PM BRT End: Monday, August 23, 2010 10:18:10 PM BRT	46	alpha: 1.0 rho: 0.3 problemsToSolve: scp401 scp405 scp407 scp410 numberOfAnts: 10 numberOfAttempts: 2 beta: 5.0	workflowResult: 431 514 432 514	Completed.
Workflow Activities Execution Details Configure Ant Application Run Ant Application				

Figura 47. Tela para a visualização dos detalhes de execução do workflow

Como resultado, tem-se o artefato `workflowResult` que foi definido por meio da tela de finalização do *workflow*. Os resultados obtidos com a execução desta instância de *workflow* (431, 514, 432 e 514) correspondem, respectivamente, aos problemas resolvidos (scp401, scp405, scp407 e scp410). O resultado de cada problema pode variar de acordo com os valores dos demais artefatos informados na instanciação do *workflow*. Além disso, o melhor resultado obtido em execuções de experimentos para resolver o problema de cobertura de conjunto e disponibilizado na base de dados da *OR Library* foi o valor 429, que corresponde ao resultado ótimo (MULATI, 2009).

Outra facilidade que a tela oferece é a análise detalhada dos artefatos de cada atividade executada no *workflow*. O conjunto de atividades que compõem o *workflow* é mostrado de acordo com a sua ordem de execução por meio do campo `Workflow Activities Execution Details`. Desta forma, o pesquisador pode navegar com facilidade à atividade desejada e obter uma visão geral da execução das atividades do *workflow*. Para

visualizar os detalhes de execução de cada atividade o cientista somente necessita acessar o *link* que corresponde ao nome da atividade criada no *workflow*. A Figura 48 ilustra os detalhes de execução da atividade *Configure Ant Application*.

Configure Ant Application		
Execution Times	Input Provenance	Output Provenance
Start: 2010-08-23T22:17:34 End: 2010-08-23T22:17:38	antAmount: 10 alpha: 1.0 rho: 0.3 beta: 5.0 attemptsNumber: 2 problems: scp401 scp405 scp407 scp410	antApplicationScriptGenerated: C00,025.sh
Close		

Figura 48. Detalhes de execução da atividade Configure Ant Application

Assim como na visualização dos detalhes de execução do *workflow*, os detalhes das atividades possuem o momento de início e fim de suas execuções. A execução da atividade *Configure Ant Application*, neste exemplo, teve a sua execução realizada em 4 segundos. Com base nos artefatos de entrada desta atividade que foram mapeados a partir dos artefatos de entrada do *workflow*, a atividade *Configure Ant Application* gerou um arquivo contendo as instruções de execução com o nome de *C00,025.sh*, o qual é mostrado na coluna *Output Provenance*. Os nomes dos artefatos de entrada e saída desta atividade representados nas colunas *Input Provenance* e *Output Provenance*, foram importados pelo *ClusterFlow* a partir do documento WSDL do serviço web que executa esta atividade. O artefato de saída *antApplicationScriptGenerated* é usado pela próxima atividade do *workflow*, *Run Ant Application*. A Figura 49 mostra os detalhes de execução da atividade *Run Ant Application*.

Run Ant Application		
Execution Times	Input Provenance	Output Provenance
Start: 2010-08-23T22:17:38 End: 2010-08-23T22:18:10	antApplicationScriptName: C00,025.sh	result: 431 514 432 514
Close		

Figura 49. Detalhes de execução da atividade Run Ant Application

Os detalhes de execução apresentados e capturados pelo mecanismo de proveniência de dados mostram que a atividade recebeu como artefato de entrada o nome do arquivo de execução *C00,025.sh* gerado pela atividade anterior, e seus resultados foram os valores 431, 514, 432 e 514. Estes valores correspondem àqueles que foram capturados pela

proveniência na finalização do *workflow*, que também podem ser vistos na Figura 47, pois os artefatos foram mapeados na tela de finalização de *workflows* durante a sua definição.

6.7. Avaliação do Ambiente Proposto

Após a execução do experimento apoiado pelo *workflow* Ant Experiment e definido através do protótipo do ambiente *ClusterFlow*, foi realizada uma análise qualitativa do ambiente proposto. Esta análise foi desempenhada por meio da comparação do ciclo de vida clássico de um experimento científico, dos princípios que devem ser levados em consideração para a construção de *workflows* científicos, e das características presentes no ambiente *ClusterFlow*. As características avaliadas são descritas a seguir:

- definição de *workflows* abstratos e concretos: esta característica apóia a fase 2 (dois) do ciclo de vida de um experimento científico. A implementação do protótipo do ambiente *ClusterFlow* não oferece apoio à definição de *workflows* abstratos. Todos os *workflows* definidos no protótipo estão na forma concreta. Para que *workflows* abstratos possam ser definidos no ambiente é necessário que o cientista seja capaz de criar uma atividade para simular uma atividade concreta, de forma que, os artefatos possam ser definidos, alterados, excluídos e as atividades de software sejam descritas textualmente ao invés da seleção de serviços web. Um benefício que o ambiente *ClusterFlow* oferece se comparado com os SGWfCs apresentados, é a definição das atividades em alto nível que permite que os cientistas não necessitem de conhecimentos de serviços web ou alguma linguagem de programação. Para definir um *workflow*, apenas é necessário que o cientista possua bom conhecimento do fluxo de dados do *workflow* que será criado. O conhecimento do fluxo de dados é exigido, pois os protótipos para definir os *workflows* induzem o cientista a defini-lo pensando na transferência de artefatos entre as atividades que o compõem;
- execução de *workflows* científicos: este aspecto oferece apoio à fase 3 (três) do ciclo de vida de um experimento científico. A execução de *workflows* no ambiente *ClusterFlow* oferece vantagem no tocante à padronização da linguagem de definição do *workflow* se comparado com SGWfCs tais como, Taverna, Kepler e Pegasus. Além disso, o pesquisador tem a possibilidade de disparar a execução de seu *workflow* e fechar o navegador web sem parar a execução do experimento, o que é imprescindível para *workflows* com execução de longa duração. A execução ocorre por meio da

exportação do *workflow* a partir de banco de dados relacional para a especificação WS-BPEL, que por sua vez deve ser implantada na Máquina WS-BPEL;

- execução de atividades humanas: esta característica também oferece apoio à fase 3 (três) do ciclo de vida de um experimento científico. Dos SGWfs investigados, o GPFlow, ActiveVOS, Oracle *Bpel Process Manager* e Intalio oferecem apoio à interação com humanos em seus *workflows*. Contudo, estes sistemas, com exceção do ActiveVOS, usam implementações proprietárias de atividades humanas que não seguem os padrões das especificações BPEL4People e WS-HumanTask. Desta forma, limitam o uso a seus SGWf, pois existe a dependência da máquina de execução de *workflows*. Embora a implementação do protótipo do ambiente *ClusterFlow* não ofereça apoio à execução de atividades humanas, foi projetado um *workflow* capaz de gerenciar as instâncias de execução desse tipo de atividade. Além deste *workflow*, as tecnologias necessárias para a execução das atividades humanas no *ClusterFlow* também foram estudadas e propostas. Desta forma, com a implementação desta proposta será possível que atividades humanas sejam executadas em *workflows* científicos gerenciados pelo ambiente *ClusterFlow*;
- proveniência de dados: a proveniência de dados apóia a fase 4 (quatro) do ciclo de vida de um experimento científico. Os SGWfCs apresentados nesta dissertação implementam a proveniência de dados em nível de *workflow*, pois o próprio SGWfC é o responsável por capturar estes dados. No entanto, este nível de captura de proveniência tem a desvantagem de ser dependente da máquina de execução de *workflow* do SGWf. Assim, a proveniência no ambiente *ClusterFlow* é capturada em nível de atividade, pois garante que a substituição da Máquina WS-BPEL seja possível caso a mesma seja descontinuada e permite a visualização clara e detalhada dos artefatos capturados em cada atividade do *workflow*. A tela para a visualização dos resultados das instâncias de execução dos *workflows* permite aos pesquisadores ter uma visão geral do fluxo de atividades que foram executadas, de forma que a proveniência de dados das atividades possa ser analisada em qualquer ordem e a qualquer momento;
- compartilhamento e reuso de *workflows*: o compartilhamento de *workflows* apóia a fase 5 (cinco) do ciclo de vida de um experimento científico e o reuso apóia a fase 1 (um) deste ciclo. Esta característica se torna diferenciada diante dos SGWfCs analisados, pois estes não oferecem apoio aos pesquisadores com relação ao

compartilhamento das ideias dos experimentos, que por sua vez são representadas na forma de *workflows*. Tais sistemas apenas permitem que as implementações das atividades de software sejam compartilhadas, o que faz necessário a utilização de um ambiente de pesquisa virtual como o ^{my}Experiment para prover o compartilhamento das definições do *workflow*. No ambiente *ClusterFlow*, a funcionalidade de compartilhamento e reuso se apresenta no próprio protótipo do ambiente e não exige a utilização de um ambiente como o ^{my}Experiment para prover esta funcionalidade. Dessa forma, o conhecimento sobre os experimentos que um cientista possui pode ser disseminado com os demais pesquisadores, acelerando com isso, o processo de pesquisa científica; e

- recursos compartilhados: o ambiente *ClusterFlow* atende este princípio para a realização de experimentos científicos, pois os recursos são compartilhados por meio de serviços web, os quais podem acessar, por exemplo, um *cluster* de computadores que é um recurso caro. A execução de experimentos em *cluster* através do protótipo do ambiente *ClusterFlow* oferece vantagem ao cientista no que se refere à abstração dos comandos necessários para executar a aplicação de alto desempenho, bem como permite que cientistas com pouco conhecimento da tecnologia de *cluster* de computadores tenham acesso a este recurso de forma compartilhada e de maneira transparente. Contudo, a distribuição de tarefas aos computadores do *cluster* e demais configurações necessárias para a execução da aplicação de alto desempenho fica a cargo do cientista responsável por desenvolver a aplicação no *cluster*. SGWfCs tais como, Taverna, Kepler e Pegasus exigem que o pesquisador possua conhecimento em *cluster*, pois as configurações de execução da aplicação são realizadas nas definições do *workflow*.

6.8. Considerações Finais

Neste capítulo foi apresentado um exemplo de aplicação a fim de validar o protótipo e suas funcionalidades desenvolvidas para o ambiente *ClusterFlow*. O exemplo de aplicação compreende a criação de um *workflow* capaz de apoiar a execução da aplicação de alto desempenho que trata o problema de cobertura de conjunto por colônia de formigas artificiais, a qual pode ser usada para definir escalas de trabalho de funcionários de grandes organizações (MULATI, 2009). O capítulo também apresentou todas as etapas para conceber um *workflow*

científico, inclusive o desenvolvimento do serviço web responsável por abstrair os comandos necessários para executar a aplicação no computador remoto.

A aplicação de alto desempenho executada no experimento realizado foi desenvolvida com o paradigma de programação sequencial e apenas um computador foi usado para sua execução. Contudo, é possível que as diferentes instâncias dos problemas resolvidos por ela sejam executadas em um *cluster* de computadores, paralelizando com isso, a execução de cada problema do algoritmo sequencial. Uma nova versão do algoritmo está sendo desenvolvido utilizando o paradigma de programação paralela, o que possibilita diminuir o tempo de execução das aplicações. A utilização de um *cluster* não dificultaria o desenvolvimento do serviço web e não afetaria a construção do *workflow*, pois apenas os comandos que o serviço web enviaria ao *cluster* seriam modificados.

Pode-se perceber que as telas desenvolvidas para o ambiente *ClusterFlow* permitem que os cientistas não necessitem de conhecimentos em programação ou de *cluster* de computadores, porém o conhecimento do experimento que se deseja conceber deve estar consolidado. A discussão e análise das funcionalidades providas pelo *ClusterFlow* levando em consideração o ciclo de vida de um experimento científico e os princípios para a construção de *workflows* científicos também foram apresentadas. Nesta análise percebe-se que o protótipo não implementa todas essas funcionalidades, porém o estudo e o projeto do ambiente deixam claros os caminhos a serem seguidos para que todos os princípios sejam satisfeitos e implementados.

No Capítulo 7 são apresentadas as conclusões e os trabalhos futuros para o ambiente *ClusterFlow*.

Conclusões

Esta dissertação apresentou um ambiente de apoio à realização de experimentos científicos para um *cluster* de computadores por meio de serviços web e *workflows* científicos, o *ClusterFlow*. O ambiente foi concebido a partir dos princípios que devem ser levados em consideração na construção de *workflows* científicos com o objetivo de apoiar as fases do ciclo de vida clássico de um experimento científico.

O princípio de apoio à definição de *workflows* abstratos e concretos definido no ambiente *ClusterFlow* apóia a fase 2 do ciclo de vida de um experimento científico. Para isso, foi concebido um mecanismo para o ambiente *ClusterFlow* no qual cientistas podem definir *workflows* concretos a partir de uma interface gráfica que é acessada através de navegadores web. Além disso, este mecanismo permite que as atividades de software sejam definidas através de serviços web selecionados a partir do repositório de serviços do ambiente *ClusterFlow*.

A definição e a navegação das atividades dos *workflows* foram baseadas no portal web do MIT *Process Handbook* (CORPORATION, 2001). As atividades definidas nos *workflows* do *ClusterFlow* são estruturadas por meio de uma árvore hierárquica que tem o seu último nó definido pela atividade de encerramento do *workflow*. Assim, o benefício que o ambiente *ClusterFlow* oferece diante dos SGWfC apresentados é permitir que os cientistas definam as atividades de *workflows* em alto nível, de tal que forma que não demande o conhecimento em alguma linguagem de programação específica por parte do cientista. No entanto, este deve possuir o conhecimento claro do fluxo de dados entre as atividades do *workflow*, pois o

ambiente *ClusterFlow* induz o cientista a conceber os *workflows* pensando na transferência de dados entre as atividades necessárias para resolver os problemas científicos representados pelos *workflows*.

O apoio à execução de *workflows* científicos concebido para o *ClusterFlow* atende à fase 3 do ciclo de vida de um experimento científico. Uma vez que os *workflows* são definidos no ambiente, estes são exportados para a execução a partir das definições armazenadas em banco de dados. Para a exportação dos *workflows* foram utilizadas as definições das especificações XSD, WSDL e WS-BPEL, que são executadas na máquina de execução Apache ODE. A vantagem do uso destas especificações é a padronização da linguagem de execução de *workflows*, pois ela permite que *workflows* do ambiente *ClusterFlow* sejam executados em qualquer máquina de execução que implemente a especificação WS-BPEL, o que possibilita a troca desta máquina caso esta seja descontinuada. Outra vantagem do *ClusterFlow* com relação aos SGWfC Taverna, Kepler e Pegasus é a possibilidade de iniciar a execução de um *workflow* e fechar o navegador web sem interferir na execução em andamento, bem como visualizar os resultados da execução a qualquer momento.

A execução de atividades humanas em *workflows* científicos também oferece apoio à fase 3 do ciclo de vida de experimentos científicos. Para apoiar a interação de humanos em *workflows* científicos e o gerenciamento das instâncias de execução destas atividades, o mecanismo proposto para o *ClusterFlow* foi baseado na definição de um *workflow* que utiliza a linguagem WS-BPEL. Este *workflow* foi projetado para implementar algumas das funcionalidades apontadas nas especificações BPEL4People e WS-HumanTask, uma vez que a máquina de execução de *workflows* Apache ODE usada neste trabalho não implementa as duas últimas especificações citadas. Além disso, a interação das tecnologias necessárias para a execução destas atividades também foram descritas, o que permite que com a implementação desta proposta, as atividades humanas possam ser executadas em *workflows* científicos gerenciados pelo *ClusterFlow*. A vantagem de utilizar este mecanismo é a independência de códigos de fabricante existentes em SGWfs como o Oracle BPEL *Process Manager* e Intalio.

A proveniência de dados presente no ambiente *ClusterFlow* apoia a fase 4 do ciclo de vida de experimentos científicos e permite que: (i) os artefatos de entrada e saída dos *workflows* e suas atividades sejam rastreados, o que garante maior confiabilidade no resultado obtido; (ii) identificar os artefatos de entrada que conduziram a determinado resultado de um experimento científico; e, (iii) compreender o motivo que levou um cientista a utilizar determinada sequência de atividades. Além disso, a proveniência de dados do *ClusterFlow* foi concebida em nível de atividade, o que tem a vantagem de ser independente de sistema

operacional, bem como oferece a possibilidade de captura de forma mais precisa, e com isso, exige menos processamento posterior que ocorre durante o processo de visualização dos resultados. A implementação da proveniência é realizada automaticamente pelo *ClusterFlow* durante a exportação do *workflow*, o que permite que o cientista não se preocupe com esta funcionalidade no processo de construção do *workflow* científico. Na execução, cada atividade do *workflow* é responsável por coletar os seus artefatos de entrada e saída. Estes artefatos são enviados ao serviço web responsável por armazená-los no banco de dados do *ClusterFlow*. Para isto, cada atividade do *workflow* é empacotada como uma atividade composta, análoga a um *subworkflow*.

O compartilhamento e reuso de *workflows* apóia respectivamente as fases 5 e 1 do ciclo de vida de um experimento científico. Este princípio concebido para o ambiente *ClusterFlow* permite o uso de *subworkflows* que podem ser incorporados em *workflows* na forma de um tipo especial de atividade. Estes *subworkflows* podem ser reutilizados a partir de *workflows* já definidos no ambiente, porém, o reuso de *subworkflows* somente é permitido para *workflows* que foram compartilhados pelos cientistas. O compartilhamento e reuso destes é possível, pois os dados das definições dos *workflows* são armazenados em um banco de dados relacional que permite o armazenamento centralizado no qual todos os cientistas que utilizam o ambiente tenham acesso. A vantagem do *ClusterFlow* com relação aos SGWfC apresentados é permitir que o compartilhamento e reuso de *workflows* esteja centralizado em um único ambiente, o que difere dos demais, pois estes exigem da utilização do ambiente de pesquisa virtual ^{my}Experiment. Além disso, o mecanismo de compartilhamento e reuso de *workflows* e atividades permite que cientistas que utilizam o *ClusterFlow* disseminem o conhecimento sobre experimentos com os demais pesquisadores, o que acelera o processo de pesquisa científica e o avanço da ciência.

O princípio de recursos compartilhados atendido pelo *ClusterFlow* é apoiado pela utilização de serviços web. Estes serviços web podem acessar, por exemplo, um *cluster* de computadores que é um recurso caro e por isso deve ser compartilhado com diversos cientistas. A vantagem que o *ClusterFlow* oferece com relação a este princípio é a abstração dos comandos necessários para a execução de aplicações de alto desempenho, permitindo que cientistas com pouco conhecimento da tecnologia de *clusters* de computadores tenham o acesso a este recurso de forma compartilhada e transparente.

Para avaliar o projeto proposto do ambiente *ClusterFlow* foi desenvolvido um protótipo e um exemplo de aplicação. O protótipo foi desenvolvido com o objetivo de oferecer uma interface gráfica para permitir que pesquisadores definam seus *workflows* sem a

preocupação com relação às tecnologias relacionadas à implementação das atividades. Com o intuito de apoiar a definição de *workflows* a partir do protótipo, um processo que define as atividades de construção e execução de *workflows* científicos foi concebido. O *workflow* de exemplo de aplicação para resolver o problema de cobertura de conjunto foi construído e executado com o apoio do protótipo do ambiente *ClusterFlow* e seguiu este processo de construção.

Na avaliação, foi realizada uma análise qualitativa do ambiente proposto, comparando-se o ciclo de vida de um experimento científico, os princípios para a construção e execução de *workflows* científicos e o ambiente *ClusterFlow*. No entanto, foram implementadas as funcionalidades mínimas no protótipo para demonstrar a viabilidade do ambiente proposto. Como resultado da avaliação, o ambiente *ClusterFlow* se mostrou viável, pois apóia todas as fases do ciclo de vida de um experimento científico. Contudo, nem todos os princípios foram atendidos, mas o estudo e o projeto do ambiente deixaram claros os caminhos a serem adotados para que todos os princípios sejam satisfeitos e implementados.

As contribuições deste trabalho compreendem: (i) oferecer a documentação do projeto de um ambiente de apoio à realização de experimentos científicos para um *cluster* de computadores utilizando *workflows* científicos e serviços web; (ii) a proposta de um mecanismo de gerenciamento de atividades humanas e suas instâncias de execução definidas em *workflows* científicos; e, (iii) oferecer um mecanismo que permita o compartilhamento e reuso de *workflows* científicos e suas atividades entre os pesquisadores.

Assim, este ambiente é capaz de apoiar pesquisadores de diversas áreas, tais como, geografia, engenharia química, biologia e ciência da computação com relação à definição, execução, visualização, compartilhamento e reuso de *workflows* científicos. Além disso, este ambiente pode beneficiar outros laboratórios específicos, como àqueles pertencentes à Central Analítica da UEM, local onde se concentram equipamentos de alto custo e, portanto, únicos.

Como o desenvolvimento de um SGWfC é uma tarefa que demanda muitos homens-hora de implementação e o projeto do *ClusterFlow* responde a várias questões de pesquisa, ainda existem trabalhos futuros que incluem:

- a implementação de novas funcionalidades no protótipo, pois ele não apóia todo o processo de construção e execução de *workflows* científicos. As principais funcionalidades a serem implementadas são: (i) a definição de *workflows* abstratos; (ii) o mecanismo de apoio à definição e execução das atividades humanas; (iii) a utilização de *subworkflows*; (iv) a utilização do repositório de serviços web UDDI; e, (v) mecanismos de segurança para os dados que trafegam na rede;

- realizar estudos empíricos com relação à execução do mecanismo de proveniência de dados, principalmente para artefatos com tamanho na ordem de Gigabytes, o que pode congestionar a rede de comunicação e prejudicar a execução de outras atividades do *workflow* científico;
- realizar estudos de mecanismos para controlar as versões de *workflows* definidos pelos cientistas, o que possibilita o acompanhamento da evolução dos *workflows* científicos;
- realizar estudos e avaliações do ambiente proposto com pesquisadores de outras áreas do conhecimento, com o objetivo de melhorar a usabilidade e abstração das definições de *workflows* científicos com a utilização do ambiente *ClusterFlow*; e
- realizar estudos com relação ao escalonamento da execução de experimentos no *cluster* de computadores, pois experimentos que necessitam de alto poder de processamento podem ter seus resultados prejudicados caso exista outra aplicação usando os recursos computacionais do *cluster*.

Referências

- AKRAM, A.; MEREDITH, D.; ALLAN, R. Evaluation of BPEL to Scientific Workflows. 2006, p. 269–274.
- ALCHIERI, E.; BESSANI, A.; DA SILVA FRAGA, J. Infra-estrutura com Segurança de Funcionamento para Cooperação de Serviços Web. *Anais do 25º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, v. 25, Belém, PA, 2007.
- ALONSO, G.; CASATI, F.; KUNO, H.; MACHIRAJU, V. *Web services: Concepts, architectures and Applications*. Springer Verlag, 2004.
- ALTINTAS, I.; BERKLEY, C.; JAEGER, E.; JONES, M.; LUDASCHER, B.; MOCK, S. Kepler: An Extensible System for Design and Execution of Scientific Workflows. In: *16th International Conference on Scientific and Statistical Database Management (SSDBM04)*, 2004.
- ANDREWS, T.; CURBERA, F.; DHOLAKIA, H.; GOLAND, Y.; KLEIN, J.; LEYMAN, F.; LIU, K.; ROLLER, D.; SMITH, D.; THATTE, S.; *ET AL.* Business Process Execution Language for Web Services, version 1.1. *Standards proposal by BEA Systems, International Business Machines Corporation, and Microsoft Corporation*, 2003.
- APACHE. Apache Hise, 2010. Disponível em: <<http://incubator.apache.org/hise/index.html>>. Acesso em maio de 2010.
- APACHE. Apache ODE (Orchestration Director Engine), 2010a. Disponível em: <<http://ode.apache.org/>>. Acesso em maio de 2010.
- APACHE. Struts 2, 2010b. Disponível em: <<http://struts.apache.org/>>. Acesso em Julho de 2010.
- APACHE. Apache Axis2/Java, 2010c. Disponível em: <<http://ws.apache.org/axis2/>>. Acesso em Julho de 2010.
- APACHE. Apache Tomcat, 2010d. Disponível em: <<http://tomcat.apache.org/>>. Acesso em Julho de 2010.
- APACHE. Json (JavaScript Object Notation), 2010e. Disponível em: <<http://www.json.org/>>. Acesso em Agosto de 2010.
- BARTOCCI, E.; CORRADINI, F.; MERELLI, E.; SCORTICHINI, L. BioWMS: a Web-Based Workflow Management System for Bioinformatics. *BMC bioinformatics*, v. 8, n.1, 2007.

- BEASLEY, J. OR-Library: Distributing Test Problems by Electronic Mail. *Journal of the Operational Research Society*, p. 1069–1072, 1990.
- BECO, S.; CANTALUPO, B.; GIAMMARINO, L.; MATSKANIS, N.; SURRIDGE, M. OWL-WS: A Workflow Ontology for Dynamic Grid Service Composition. in: *e-Science and Grid Computing, 2005. First International Conference on*, 2005, p. 8 pp. –155.
- BELL, G.; GRAY, J. What's Next in High-Performance Computing? *Commun. ACM*, v. 45, n. 2, p. 91–95, 2002.
- CAVALCANTI, M. C.; TARGINO, R.; BAIÃO, F.; RÖSSLE, S. C.; BISCH, P. M.; PIRES, P. F.; CAMPOS, M. L. M.; MATTOSO, M. Managing Structural Genomic Workflows Using Web Services. *Data & Knowledge Engineering*, v. 53, n. 1, p. 45 – 74, biological Data Management, 2005.
- CORPORATION, P. MIT Process Handbook, 2001. Disponível em: <<http://process.mit.edu/>>. Acesso em: Abril 2010.
- CRUZ, S. DA; CHIRIGATI, F.; DAHIS, R.; CAMPOS, M.; MATTOSO, M. Controles de Fluxo Explícitos em Workflows Científicos. *II e-Science Workshop, Campinas - SP. Anais do II e-Science Workshop/SBES. Porto Alegre - RS : SBC*, p. 10, 2008.
- DEELMAN, E.; BLYTHE, J.; GIL, Y.; KESSELMAN, C. Pegasus: Planning for Execution in Grids. *GriPhyN*, v. 20, p. 2002, 2002.
- DEELMAN, E.; GANNON, D.; SHIELDS, M.; TAYLOR, I. Workflows and e-Science: An Overview of Workflow System Features and Capabilities. *Future Generation Computer Systems*, v. 25, n. 5, p. 528 – 540, 2009.
- ECLIPSE. Eclipse.org, 2010. Disponível em: <<http://www.eclipse.org>>. Acesso em Agosto de 2010.
- ENDPOINTS, A. The ActiveBPEL Engine, 2010a. Disponível em: <<http://www.activevos.com/community-open-source.php>>. Acesso em: Abril 2010.
- ENDPOINTS, A. ActiveVOS Business Process Management, 2010b. Disponível em: <<http://www.activevos.com/>>. Acesso em: Abril 2010.
- FREIRE, J.; KOOP, D.; SANTOS, E.; SILVA, C. Provenance for Computational Tasks: A Survey. *Computing in Science and Engineering*, v. 10, n. 3, p. 11, 2008.
- FREY, J.; TANNENBAUM, T.; LIVNY, M.; FOSTER, I.; TUECKE, S. Condor-G: A Computation Management Agent for Multi-Institutional Grids. *Cluster Computing*, v. 5, n. 3, p. 237–246, 2002.
- GIL, Y.; DEELMAN, E.; ELLISMAN, M.; FAHRINGER, T.; FOX, G.; GANNON, D.; GOBLE, C.; LIVNY, M.; MOREAU, L.; MYERS, J. Examining the Challenges of Scientific Workflows. *IEEE Computer*, v. 40, n. 12, p. 24–32, 2007.
- GOBLE, C. A.; DE ROURE, D. C. myExperiment: Social Networking for Workflow-Using e-Scientists. In: *WORKS '07: Proceedings of the 2nd workshop on Workflows in support of large-scale science*, New York, NY, USA: ACM, 2007, p. 1–2.

- HOLLINGSWORTH, D. Workflow management coalition: The workflow reference model. *Workflow Management Coalition*, 1995.
- HUFF, A.; GIMENES, I. M. D. S.; GONÇALVES, R. A. D. L. *ClusterFlow: um Ambiente de Apoio a Experimentos Científicos em um Cluster de Computadores. III e-Science Workshop, Fortaleza - CE. Anais do III e-Science Workshop/SBES. Porto Alegre - RS : SBC, v. 1, p. 33–40, 2009.*
- HULL, D.; WOLSTENCROFT, K.; STEVENS, R.; GOBLE, C.; POCOCK, M.; LI, P.; OINN, T. Taverna: A Tool for Building and Running Workflows of Services. *Nucleic acids research*, v. 34, 2006.
- INTALIO. Intalio Community Edition, 2010a. Disponível em: <<http://community.intalio.com/>>. Acesso em: Abril 2010.
- INTALIO. Intalio Workflow Tempo, 2010b. Disponível em: <<http://intaliotempo.wordpress.com/>>. Acesso em maio de 2010.
- JQUERY. JQuery, 2010. Disponível em: <<http://jquery.com/>>. Acesso em Julho de 2010.
- KEE, Y.-S.; BYUN, E.; DEELMAN, E.; VAHI, K.; KIM, J.-S. Pegasus on the Virtual Grid: A Case Study of Workflow Planning Over Captive Resources. In: *Workflows in Support of Large-Scale Science, 2008. WORKS 2008. Third Workshop on*, 2008, p. 1–9.
- KLOPPMANN, M.; KOENIG, D.; LEYMANN, F.; PFAU, G.; RICKAYZEN, A.; von Riegen, C.; Schmidt, P.; Trickovic, I. WS-BPEL Extension for People–BPEL4People. *Joint white paper, IBM and SAP*, 2005.
- LEE, K.; PATON, N.; SAKELLARIOU, R.; DEELMAN, E.; FERNANDES, A.; MEHTA, G. Adaptive Workflow Processing and Execution in Pegasus. *Concurrency and Computation: Practice and Experience*, v. 21, n. 16, p. 1965–1981, 2009.
- LIN, C.; LU, S.; LAI, Z.; CHEBOTKO, A.; FEL, X.; HUA, J.; FOTOUHI, F. Service-Oriented Architecture for View: A Visual Scientific Workflow Management System. 2008, p. 335–342.
- MARINHO, A.; WERNER, C.; DA CRUZ, S. M. S.; MATTOSO, M.; BRAGANHOLO, V.; MURTA, L. A Strategy for Provenance Gathering in Distributed Scientific Workflows. *Services, IEEE Congress on*, v. 0, p. 344–347, 2009.
- MATTOSO, M.; WERNER, C.; TRAVASSOS, G.; BRAGANHOLO, V.; MURTA, L.; CAMPOS, J.; HORNSBY, K. Gerenciando Experimentos Científicos em Larga Escala. *SBC*, p. 121–135, 2008.
- MULATI, M. H. Investigação da Meta-Heurística de Otimização por Colônia de Formigas Artificiais Aplicada ao Problema de Cobertura de Conjunto. Dissertação de mestrado, Universidade Estadual de Maringá, Departamento de Informática, 2009.
- MUNROE, S.; MILES, S.; MOREAU, L.; VÁZQUEZ-SALCEDA, J. Prime: A Software Engineering Methodology for Developing Provenance-Aware Applications. In: *SEM '06: Proceedings of the 6th international workshop on Software engineering and middleware*, New York, NY, USA: ACM, 2006, p. 39–46.

- OASIS. UDDI Version 3.0.2, 2004. Disponível em: <http://uddi.org/pubs/uddi_v3.htm>. Acesso em abril de 2010.
- OASIS. Web Services Business Process Execution Language Version 2.0, 2007. Disponível em: <<http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>>. Acesso em maio de 2010.
- OASIS. WS-BPEL Extension for People (BPEL4People) Specification Version 1.1, 2010a. Disponível em: <<http://docs.oasis-open.org/bpel4people/bpel4people-1.1-spec-cd-08.html>>. Acesso em abril de 2010.
- OASIS. Web Services - Human Task (WS-HumanTask) Specification Version 1.1, 2010b. Disponível em: <<http://docs.oasis-open.org/bpel4people/ws-humantask-1.1.html>>. Acesso em maio de 2010.
- OINN, T.; GREENWOOD, M.; ADDIS, M.; ALPDEMIR, M.; FERRIS, J.; GLOVER, K.; GOBLE, C.; GODERIS, A.; HULL, D.; MARVIN, D.; *ET AL.* Taverna: Lessons in Creating a Workflow Environment for the Life Sciences. *Journal of Concurrency and Computation: Practice and Experience*, 2002.
- OINN, T.; ADDIS, M.; FERRIS, J.; MARVIN, D.; GREENWOOD, M.; CARVER, T.; POCOCK, M.; WIPAT, A.; LI, P. Taverna: A Tool for the Composition and Enactment of Bioinformatics Workflows. *Bioinformatics*, 2004.
- OINN, T.; GREENWOOD, M. F.; ADDIS, M.; ALPDEMIR, M.; FERRIS, J.; GLOVER, K.; GOBLE, C.; GODERIS, A.; HULL, D.; MARVIN, D.; LI, P.; LORD, P.; POCOCK, M.; SENGER, M.; STEVENS, R.; WIPAT, A.; WROE, C. Taverna: Lessons in Creating a Workflow Environment for the Life Sciences. *Concurrency Computation Practice and Experience*, v. 18, n. 10, p. 1067–1100, 2006.
- OMG. Object Management Group/Business Process Management Initiative, 2010. Disponível em: <<http://www.bpmn.org>>. Acesso em Agosto de 2010.
- ORACLE. Oracle BPEL Process Manager, 2010. Disponível em: <<http://www.oracle.com/technology/products/ias/bpel/index.html>>. Acesso em: Abril 2010.
- ORACLE. Oracle Toplink, 2010a. Disponível em: <<http://www.oracle.com/technology/products/ias/toplink/index.html>>. Acesso em Julho de 2010.
- PAPAZOGLU, M. P. *Web services: Principles and Technology*. Pearson Prentice Hall, p. 784, 2008.
- PELTZ, C. Web Services Orchestration and Choreography. *Computer*, v. 36, n. 10, p. 46–52, 2003.
- ROMANO, P.; BARTOCCI, E.; BERTOLINI, G.; DE PAOLI, F.; MARRA, D.; MAURI, G.; MERELLI, E.; MILANESI, L. Biowep: A Workflow Enactment Portal for Bioinformatics Applications. *BMC bioinformatics*, v. 8, n. 1, 2007.

- DE ROURE, D.; GOBLE, C.; STEVENS, R. Designing the myExperiment Virtual Research Environment for the Social Sharing of Workflows. In: *e-Science and Grid Computing, IEEE International Conference on*, 2007, p. 603–610.
- RYGG, A.; ROE, P.; WONG, O. GPFlow: An Intuitive Environment for Web Based Scientific Workflow. In: *Grid and Cooperative Computing Workshops, 2006. GCCW '06. Fifth International Conference on*, 2006, p. 204–211.
- SIMMHAN, Y. L.; PLALE, B.; GANNON, D. A Survey of Data Provenance in e-Science. *SIGMOD Rec.*, v. 34, n. 3, p. 31–36, 2005.
- STEVENS, R.; ROBINSON, A.; GOBLE, C. myGrid: Personalised Bioinformatics on the Information Grid. *Bioinformatics*, v. 19, n. 90001, p. 302–304, 2003.
- TURI, D.; MISSIER, P.; GOBLE, C.; DE ROURE, D.; OINN, T. Taverna Workflows: Syntax and Semantics. In: *IEEE International Conference on e-Science and Grid Computing*, 2007, p. 441–448.
- UNDERWOOD, K. D.; SASS, R. R.; LIGON, III, W. B. Cost Effectiveness of an Adaptable Computing Cluster. In: *Supercomputing '01: Proceedings of the 2001 ACM/IEEE conference on Supercomputing (CDROM)*, New York, NY, USA: ACM, 2001, p. 54–54.
- W3C. Web Services Description Language, 2001. Disponível em: <<http://www.w3.org/TR/wsdl>>. Acesso em abril de 2010.
- W3C. SOAP Version 1.2, 2007. Disponível em: <<http://www.w3.org/TR/soap12-part1/>>. Acesso em abril de 2010.
- W3C. XML Schema Definition Language (XSD) 1.1, 2009. Disponível em: <<http://www.w3.org/TR/xmlschema11-1/>>. Acesso em julho de 2009.
- W3C. HTTP - Hypertext Transfer Protocol, 2010a. Disponível em: <<http://www.w3.org/Protocols/>>. Acesso em abril de 2010.
- W3C. Extensible Markup Language, 2010b. Disponível em: <<http://www.w3.org/XML/>>. Acesso em abril de 2010.
- WANG, F.; DENG, H.; LIANG, B.; GUO, L.; JI, K. A Study on a Lightweight Scientific Workflow System for Astronomical e-Science Service. In: *2009 Third International Symposium on Intelligent Information Technology Application*, IEEE, 2009, p. 69–72.
- WROE, C.; GOBLE, C.; GODERIS, A.; LORD, P.; MILES, S.; PAPAY, J.; ALPER, P.; MOREAU, L. Recycling Workflows and Services Through Discovery and Reuse. *Concurrency and Computation: Practice and Experience*, v. 19, n. 2, p. 181–194, 2007.
- YU, J.; BUYYA, R. A Taxonomy of Scientific Workflow Systems for Grid Computing. *SIGMOD Officers, Committees, and Awardees*, v. 34, n. 3, p. 44, 2005.

Apêndice A

Este apêndice contém os principais artefatos levantados durante a fase de análise de requisitos para o desenvolvimento do ambiente *ClusterFlow*. Desta forma, é apresentada uma descrição geral do ambiente, o levantamento dos principais requisitos funcionais e não funcionais, o modelo de casos de uso e a descrição dos mesmos.

A.1 Descrição Geral do Ambiente

O ambiente *ClusterFlow* visa auxiliar pesquisadores de várias áreas, tais como geografia, engenharia química, biologia e ciência da computação na criação, gerenciamento da execução e compartilhamento de *workflows* científicos. O ambiente utiliza serviços web para oferecer acesso aos experimentos contidos em um *cluster* de computadores. A Figura 50 mostra a visão do negócio do ambiente *ClusterFlow*.

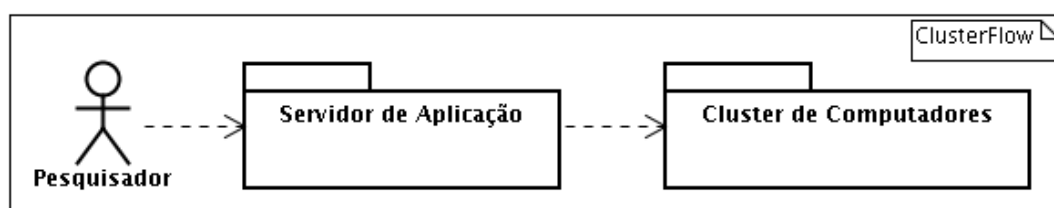


Figura 50. Visão do negócio do ambiente *ClusterFlow*

Na visão do negócio, o Pesquisador interage com o Servidor de Aplicação com o objetivo de realizar experimentos e visualizar seus resultados. O pacote Servidor de Aplicação é responsável por oferecer uma interface web para que os cientistas criem seus *workflows* de experimentos, acompanhem suas execuções e visualizem seus resultados. Este pacote também permite que as aplicações de alto desempenho disponibilizadas no *cluster* sejam executadas a partir de atividades definidas no *workflow*. O pacote Cluster de Computadores representa as aplicações de alto desempenho que estão disponíveis em um *cluster* e prontas para execução. Desse modo, as execuções das

aplicações de alto desempenho disponíveis no *cluster* e a execução do *workflow* definido pelo pesquisador são considerados dois processos computacionais distintos.

Para que os cientistas possam desenvolver suas ideias e criar seus *workflows* no ambiente, foi realizado o levantamento das principais funcionalidades necessárias. Estas funcionalidades são descritas na forma de requisitos funcionais e não funcionais. Os principais requisitos funcionais são os seguintes:

- gerenciamento de *workflows* científicos: a partir do ambiente *ClusterFlow*, os pesquisadores podem descrever textualmente seus *workflows* e atividades incluindo as definições e objetivos das mesmas, criar *workflows* totalmente novos, bem como mantê-los. O ambiente oferece apoio para os cientistas definirem *workflows* concretos. Além disso, o ambiente permite que o pesquisador possa verificar o estado de execução em que seus *workflows* se encontram, bem como visualizar os dados dos *workflows* em execução e dos que foram executados;
- compartilhamento e reutilização de *workflows* científicos: as definições dos *workflows* realizadas pelos cientistas são armazenadas em banco de dados para serem compartilhadas e reutilizadas. A operação de compartilhamento pode ser realizada para *workflows* abstratos e concretos. Assim, cientistas podem fazer o reuso de *workflows* ou de atividades. Além da reutilização, os pesquisadores podem readaptar os *workflows* e atividades de acordo com as necessidades de cada experimento. Isto pode compreender os atributos de parâmetros de entrada e sua variação dos valores, bem como a descrição da atividade a ser realizada;
- gerenciamento de atividades humanas: as atividades de um *workflow* científico podem ser computacionais e humanas. As atividades humanas correspondem àquelas que necessitem de interação dos cientistas para a sua realização. O ambiente deve oferecer aos cientistas a possibilidade de criar e manter estas atividades;
- exportação de *workflows* para a linguagem WS-BPEL: os *workflows* científicos definidos de forma concreta podem ser exportados para a linguagem WS-BPEL baseando-se nas definições armazenadas em banco de dados;
- execução de *workflows* científicos: os *workflows* que foram exportados podem ser instanciados pelos cientistas e executados no ambiente *ClusterFlow*. O ambiente possui um software responsável por orquestrar e gerenciar o conjunto de atividades dos *workflows*. Somente *workflows* concretos podem ser orquestrados por este software;

- armazenamento da proveniência de dados: os dados utilizados e gerados pelos *workflows* de experimentos científicos são armazenados em uma base de dados para consultas futuras. Desta forma, pesquisadores podem tomar conhecimento antecipado dos dados de entrada responsáveis por gerar um resultado em específico; e
- comunicação com o cluster de computadores: o ambiente *ClusterFlow* deve oferecer mecanismos que permitam que *workflows* criados pelos cientistas se comuniquem com o *cluster* de computadores a fim de executar os experimentos contidos neste.

Os principais requisitos não-funcionais são:

- acesso aos workflows por meio de um portal web: o ambiente *ClusterFlow* deve oferecer acesso aos *workflows* por meio de um portal web, assim evita que pesquisadores necessitem da instalação de aplicações locais em cada computador utilizado para a manipulação de *workflows*;
- execução de workflows independente do portal web: é desejável que o pesquisador tenha a capacidade de iniciar a execução do seu *workflow* e posteriormente possa fechar o navegador sem a interrupção do experimento. Além disso, é desejável que ele possa acessar o portal em outra oportunidade para verificar a execução e os resultados do *workflow* que havia instanciado anteriormente;
- desconhecimento de linguagem de programação: os usuários do ambiente não devem precisar de conhecimentos profundos de alguma linguagem de programação ou tecnologias de *middleware*; e
- controle de acesso: apenas usuários autorizados devem ter acesso ao ambiente e aos seus dados.

A.2 Modelagem e Especificação dos Casos de Uso

A Figura 51 apresenta o modelo de casos de uso do ambiente *ClusterFlow*. Os atores *Researcher* e *WS-BPEL Engine* representam, respectivamente, o pesquisador responsável por definir os *workflows* e o motor de execução de *workflows*. O ator *PotentialOwner* representa os possíveis proprietários (pesquisadores) das atividades humanas que interagem com o ambiente no momento da execução do *workflow*; e o *TaskManagerProcess* representa o *workflow* responsável por implementar o gerenciamento das instâncias das atividades humanas.

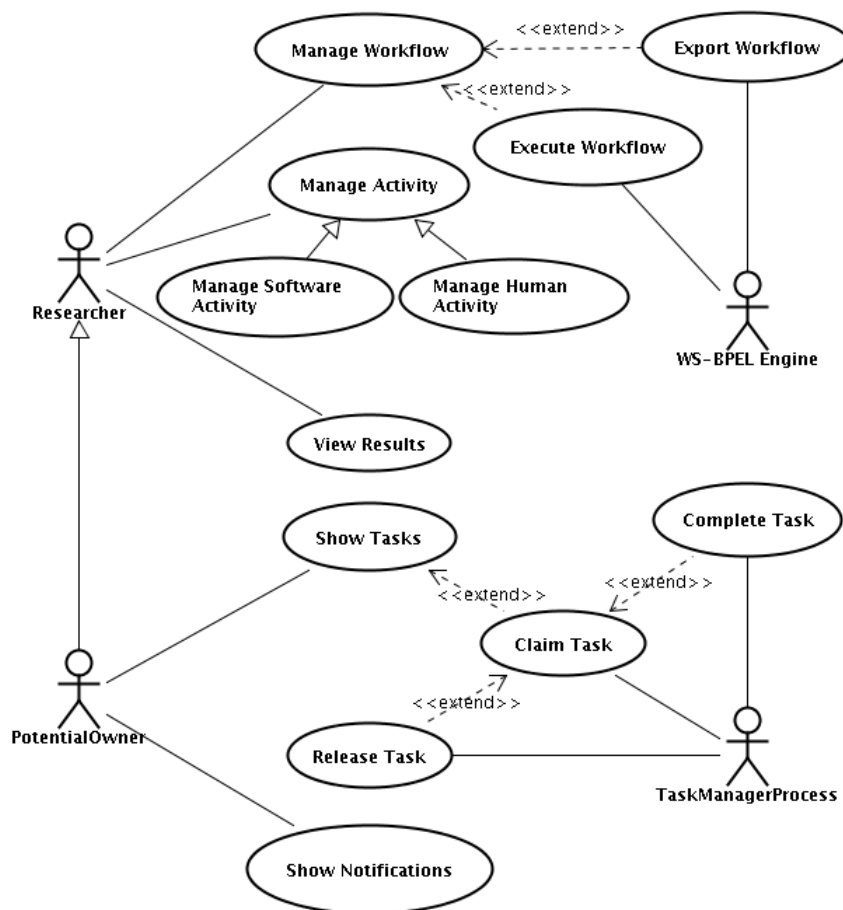


Figura 51. Visão geral do modelo de casos de uso

Com base no modelo de casos de uso apresentado, a descrição geral destes é apresentada a seguir:

- Manage Workflow: responsável por manipular as ações que possam ocorrer com um *workflow* no ambiente. O pesquisador pode criar novos *workflows*, compartilhar, reaproveitar aqueles anteriormente definidos, reeditá-los, definir quais serão os artefatos necessários para dar início à execução de uma nova instância, bem como determinar os artefatos de saída. Para configurar o encerramento do *workflow*, o pesquisador deve definir os artefatos que informarão o resultado do *workflow* ao cientista. Para isto, o pesquisador faz o mapeamento das saídas das atividades retornando apenas os resultados relevantes do experimento realizado;
- Manage Activity: gerencia as atividades criadas no ambiente para a realização dos experimentos. As atividades podem ser realizadas por humanos (cientistas) ou por software que compreendem as atividades computacionais invocadas por meio de serviços web. As atividades podem ser reutilizadas a partir de atividades de outros *workflows* já definidos. Para que uma atividade seja executada corretamente, seus

artefatos de entrada devem ser mapeados a partir dos artefatos de saída de atividades anteriores. Com isso, o ciclo de mapeamento do fluxo dos dados das atividades é realizado gradativamente, pois apenas ocorre para os artefatos de entrada e permite que a saída seja mapeada por atividades posteriores;

- Manage Human Activity: este caso de uso é realizado quando o pesquisador necessitar que uma interação humana seja configurada. As atividades humanas devem possuir no mínimo um papel atribuído a elas que é usado para definir os responsáveis por executá-las. Dessa forma, os pesquisadores que são cadastrados em um papel serão comunicados quando uma determinada atividade humana deverá ser realizada. Este aviso ocorre por meio de uma mensagem eletrônica enviada à lista de atividades a serem realizadas. Os artefatos de entrada e saída destas atividades devem ser definidos pelo cientista.
- Manage Software Activity: trata das atividades computacionais pertinentes aos *workflows*. Estas atividades são invocadas pela máquina de execução de *workflows* utilizando a tecnologia de serviços web. Elas compreendem dentre outros, os experimentos computacionais contidos no *cluster* de computadores. Os artefatos deste tipo de atividade não necessitam ser definidos pelo cientista, pois o documento WSDL fornece essas definições;
- Export Workflow: responsável por traduzir os dados armazenados no banco de dados para a especificação WS-BPEL. É requerido quando o cientista desejar executar uma instância de *workflow*, pois as definições dos *workflows* criados estão armazenadas em banco de dados e o motor de execução de *workflows* somente interpreta a descrição formal dos mesmos. Esta descrição ocorre por meio de arquivos XML baseados na especificação WS-BPEL. Após a exportação, o *workflow* é implantado na máquina de execução de *workflows* e é disponibilizado para que o pesquisador possa instanciá-lo a partir de um navegador web. Este caso de uso estende o *Manage Workflow* e é opcional, pois o pesquisador tem a possibilidade de definir *workflows* e exportá-los em outra oportunidade;
- Execute Workflow: trata da instanciação dos *workflows*. Este deve iniciar, pausar, reiniciar e parar a execução dos *workflows*. Deve também oferecer recursos de consulta para informar ao cientista o estado de execução em que um determinado *workflow* se encontra, bem como armazenar os estados nos quais os mesmos encerraram a execução, tais como completou, falhou e terminou. A proveniência dos

dados é capturada durante a execução dos *workflows* e armazenada no banco de dados para que cientistas possam realizar consultas futuras referentes à execução de um *workflow* sem a necessidade de executá-lo novamente;

- View Results: apóia a visualização dos resultados da proveniência. Esta visualização é um fator importante para realizar o rastreamento dos valores de entrada e de saída de uma determinada atividade ou *workflow*, além de possibilitar uma validação posterior. Um ponto importante a destacar é que o ambiente *ClusterFlow* apenas oferece apoio à proveniência retrospectiva (FREIRE *et al.*, 2008), que consiste na coleta das informações em tempo de execução. Estes dados de proveniência devem ser capturados automaticamente pelo ambiente e, após a execução poderão ser excluídos pelo proprietário do *workflow*. Além disso, os dados de proveniência são compartilhados automaticamente no momento em que a definição do *workflow* é compartilhada;
- Show Tasks: visa gerenciar as requisições do possível proprietário da atividade humana, o `PotentialOwner`, com base no papel que foi atribuído a ele. Além disso, a mesma atividade humana pode ser atribuída a diversos pesquisadores, permitindo que qualquer um deste grupo possa realizá-la. Este caso de uso apresenta as informações das atividades a serem realizadas, tais como, o título, o estado em que a mesma se encontra, a prioridade de execução, a data e hora da criação, a data ou o tempo limite de expiração, os objetivos e a descrição de como esta deve ser realizada para atingir o resultado esperado;
- Claim Task: este caso de uso estende `Show Tasks` e é executado quando o pesquisador tiver a intenção de informar que irá realizar determinada atividade humana. Ele altera o estado desta bloqueando a mesma de forma que outros pesquisadores não tenham acesso à atividade. As atividades bloqueadas por meio deste caso de uso somente podem ser visualizadas, liberadas ou concluídas pelo seu proprietário;
- Release Task: visa liberar as atividades humanas que foram bloqueadas pelo caso de uso `Claim Task`. Assim, quando a atividade é liberada, os outros possíveis donos podem visualizá-la novamente, bem como um deles poderá declarar que irá executá-la;

- Complete Task: responsável por gerenciar a finalização da atividade humana e informar ao `TaskManagerProcess` que a atividade foi completada pelo seu proprietário e teve a sua execução encerrada; e
- Show Notifications: visa apresentar ao `PotentialOwner` todas as notificações que foram designadas a ele por meio do papel que o mesmo exerce no *workflow*. As notificações podem ser criadas tanto pelo `TaskManagerProcess`, bem como pelo *workflow* científico criado pelo cientista. Assim, as notificações criadas pelo `TaskManagerProcess` correspondem àquelas de finalização da atividade humana com o objetivo de informar o resultado da mesma aos possíveis proprietários, notificações de tempo limite para o início de uma atividade, bem como notificações do prazo para o encerramento da atividade. As notificações implementadas nas definições dos *workflows* podem informar a forma de realização de determinadas atividades humanas, bem como mostrar resultados de atividades de software aos pesquisadores.

Apêndice B

Este apêndice apresenta o diagrama de atividades que corresponde ao *workflow* de gerenciamento das instâncias das atividades humanas no *ClusterFlow*. As atividades do diagrama seguem a nomenclatura da especificação WS-BPEL para facilitar a geração do código XML deste *workflow*. A Figura 52 mostra este diagrama.

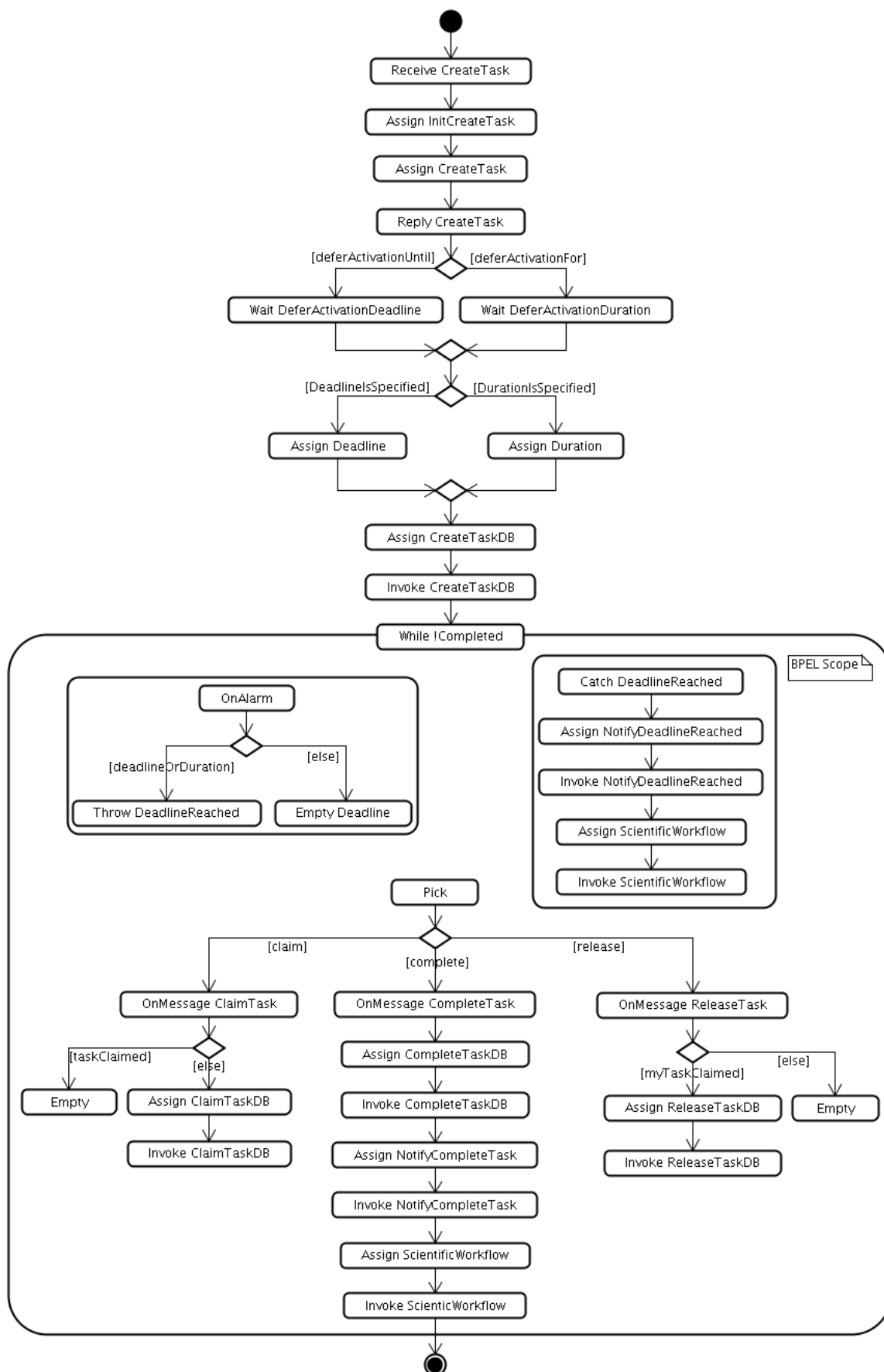


Figura 52. Diagrama de atividades completo do workflow TaskManagerProcess

Apêndice C

Este apêndice mostra o código completo do *workflow* utilizado como exemplo no Capítulo 5 para descrever os mapeamentos do modelo estático para as especificações XSD, WSDL e WS-BPEL.

```

1<bpeel:process exitOnStandardFault="no" name="CalcWorkflow"
2  suppressJoinFailure="yes"
3  targetNamespace="http://les.din.uem.br/clusterflow/process/CalcWorkflow"
4  xmlns:bpeel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
5  xmlns:implCalculadora="http://huff.calc.les.din.uem.br"
6  xmlns:provenanceServiceprovenance="http://provenance.clusterflow.din.uem.br"
7  xmlns:tns="http://les.din.uem.br/clusterflow/process/CalcWorkflow">
8  <bpeel:import importType="http://schemas.xmlsoap.org/wsdl/"
9    location="CalcWorkflow.wsdl" namespace="http://les.din.uem.br/clusterflow/process/CalcWorkflow" />
10 <bpeel:import importType="http://schemas.xmlsoap.org/wsdl/"
11   location="ProvenanceService.wsdl" namespace="http://provenance.clusterflow.din.uem.br" />
12 <bpeel:import importType="http://schemas.xmlsoap.org/wsdl/"
13   location="Calculadora.wsdl" namespace="http://huff.calc.les.din.uem.br" />
14 <bpeel:partnerLinks>
15   <bpeel:partnerLink myRole="CalcWorkflowProvider" name="client"
16     partnerLinkType="tns:CalcWorkflowPLT" />
17   <bpeel:partnerLink name="ProvenancePortTypePL"
18     partnerLinkType="tns:ProvenancePortTypePLT" partnerRole="ProvenancePortTypeProvider" />
19   <bpeel:partnerLink name="CalculadoraPL"
20     partnerLinkType="tns:CalculadoraPLT" partnerRole="CalculadoraProvider" />
21 </bpeel:partnerLinks>
22 <bpeel:variables>
23   <bpeel:variable messageType="tns:CalcWorkflowProcessRequestMessage"
24     name="input" />
25   <bpeel:variable
26     messageType="provenanceServiceprovenance:inputProvenanceResponseMessage"
27     name="workflowInstanceIdVariable" />
28   <bpeel:variable
29     messageType="provenanceServiceprovenance: faultProvenanceRequestMessage"
30     name="ProvenancePortTypefaultProvenancePLRequest" />
31   <bpeel:variable
32     messageType="provenanceServiceprovenance: faultProvenanceResponseMessage"
33     name="ProvenancePortTypefaultProvenancePLResponse" />
34   <bpeel:variable
35     messageType="provenanceServiceprovenance: startProvenanceRequestMessage"
36     name="ProvenancePortTypestartProvenancePLRequest" />
37   <bpeel:variable
38     messageType="provenanceServiceprovenance: startProvenanceResponseMessage"
39     name="ProvenancePortTypestartProvenancePLResponse" />
40   <bpeel:variable
41     messageType="provenanceServiceprovenance: inputProvenanceRequestMessage"
42     name="ProvenancePortTypeinputProvenancePLRequest" />
43   <bpeel:variable

```



```

44     messageType="provenanceServiceprovenance:inputProvenanceResponseMessage"
45     name="ProvenancePortTypeinputProvenancePLResponse" />
46 <φpel:variable messageType="implCalculadora:addRequest"
47     name="CalculadoraaddPLRequest" />
48 <φpel:variable messageType="implCalculadora:addResponse"
49     name="CalculadoraaddPLResponse" />
50 <φpel:variable
51     messageType="provenanceServiceprovenance:outputProvenanceRequestMessage"
52     name="ProvenancePortTypeoutputProvenancePLRequest" />
53 <φpel:variable
54     messageType="provenanceServiceprovenance:outputProvenanceResponseMessage"
55     name="ProvenancePortTypeoutputProvenancePLResponse" />
56 </bpel:variables>
57 <bpel:faultHandlers>
58   <φpel:catchAll>
59     <φpel:sequence>
60       <φpel:assign validate="no">
61         <φpel:copy>
62           <φpel:from>
63             <φpel:literal>
64               <provenanceService: faultProvenanceRequest
65                 xmlns:provenanceService="http://provenance.clusterflow.din.uem.br"
66                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
67                 <provenanceService:instanceId />
68               </provenanceService: faultProvenanceRequest>
69             </φpel:literal>
70           </φpel:from>
71           <φpel:to part="parameters" variable="ProvenancePortTypefaultProvenancePLRequest" />
72         </φpel:copy>
73         <φpel:copy>
74           <φpel:from part="parameters" variable="workflowInstanceIdVariable">
75             <φpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
76               <![CDATA[provenanceServiceprovenance:instanceId]]>
77             </φpel:query>
78           </φpel:from>
79           <φpel:to part="parameters" variable="ProvenancePortTypefaultProvenancePLRequest">
80             <φpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
81               <![CDATA[provenanceServiceprovenance:instanceId]]>
82             </φpel:query>
83           </φpel:to>
84         </φpel:copy>
85       </φpel:assign>
86       <φpel:invoke inputVariable="ProvenancePortTypefaultProvenancePLRequest"
87         name="FaultProvenanceInvoke" operation="faultProvenance"
88         outputVariable="ProvenancePortTypefaultProvenancePLResponse"
89         partnerLink="ProvenancePortTypePL" portType="provenanceServiceprovenance:ProvenancePortType" />
90     </φpel:sequence>
91   </φpel:catchAll>
92 </bpel:faultHandlers>
93 <bpel:sequence>
94   <ext:failureHandling xmlns:ext="http://ode.apache.org/activityRecovery">
95     <ext: faultOnFailure>true</ext: faultOnFailure>
96   </ext: failureHandling>
97   <φpel:receive createInstance="yes" name="CalcWorkflowReceive"
98     operation="process" partnerLink="client" portType="tns:CalcWorkflowPortType"
99     variable="input" />
100   <φpel:assign validate="no">
101     <φpel:copy>
102       <φpel:from>
103         <φpel:literal>
104           <provenanceService:startProvenanceRequest
105             xmlns:provenanceService="http://provenance.clusterflow.din.uem.br"
106             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
107             <provenanceService:workflowId />
108           </provenanceService:startProvenanceRequest>
109         </φpel:literal>
110       </φpel:from>
111       <φpel:to part="parameters" variable="ProvenancePortTypestartProvenancePLRequest" />
112     </φpel:copy>
113     <φpel:copy>
114       <φpel:from>
115         <φpel:literal>5</φpel:literal>
116       </φpel:from>
117       <φpel:to part="parameters" variable="ProvenancePortTypestartProvenancePLRequest">
118         <φpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
119           <![CDATA[provenanceServiceprovenance:workflowId]]>
120         </φpel:query>
121       </φpel:to>
122     </φpel:copy>
123   </φpel:assign>
124   <φpel:invoke inputVariable="ProvenancePortTypestartProvenancePLRequest"
125     name="StartProvenanceInvoke" operation="startProvenance"
126     outputVariable="ProvenancePortTypestartProvenancePLResponse"
127     partnerLink="ProvenancePortTypePL" portType="provenanceServiceprovenance:ProvenancePortType" />

```

```

128 <bpel:assign validate="no">
129   <bpel:copy>
130     <bpel:from>
131       <bpel:literal>
132         <provenanceService:inputProvenanceRequest
133           xmlns:provenanceService="http://provenance.clusterflow.din.uem.br"
134           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
135           <provenanceService:entity />
136           <provenanceService:entityId />
137           <provenanceService:enactmentId />
138           <provenanceService:inputProvenance>
139             <provenanceService:key />
140             <provenanceService:value />
141           </provenanceService:inputProvenance>
142           <provenanceService:inputProvenance>
143             <provenanceService:key />
144             <provenanceService:value />
145           </provenanceService:inputProvenance>
146         </provenanceService:inputProvenanceRequest>
147       </bpel:literal>
148     </bpel:from>
149     <bpel:to part="parameters" variable="ProvenancePortTypeinputProvenancePLRequest" />
150   </bpel:copy>
151   <bpel:copy>
152     <bpel:from part="parameters"
153       variable="ProvenancePortTypestartProvenancePLResponse">
154       <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:subLang:xpath1.0">
155         <![CDATA[provenanceServiceprovenance:enactmentId]]>
156       </bpel:query>
157     </bpel:from>
158     <bpel:to part="parameters" variable="ProvenancePortTypeinputProvenancePLRequest">
159       <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:subLang:xpath1.0">
160         <![CDATA[provenanceServiceprovenance:enactmentId]]>
161       </bpel:query>
162     </bpel:to>
163   </bpel:copy>
164   <bpel:copy>
165     <bpel:from>
166       <bpel:literal>WORKFLOW</bpel:literal>
167     </bpel:from>
168     <bpel:to part="parameters" variable="ProvenancePortTypeinputProvenancePLRequest">
169       <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:subLang:xpath1.0">
170         <![CDATA[provenanceServiceprovenance:entity]]>
171       </bpel:query>
172     </bpel:to>
173   </bpel:copy>
174   <bpel:copy>
175     <bpel:from>
176       <bpel:literal>5</bpel:literal>
177     </bpel:from>
178     <bpel:to part="parameters" variable="ProvenancePortTypeinputProvenancePLRequest">
179       <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:subLang:xpath1.0">
180         <![CDATA[provenanceServiceprovenance:entityId]]>,
181       </bpel:query>
182     </bpel:to>
183   </bpel:copy>
184   <bpel:copy>
185     <bpel:from>
186       <bpel:literal>81</bpel:literal>
187     </bpel:from>
188     <bpel:to part="parameters" variable="ProvenancePortTypeinputProvenancePLRequest">
189       <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:subLang:xpath1.0">
190         <![CDATA[provenanceServiceprovenance:inputProvenance[1]/provenanceServiceprovenance:key]]>
191       </bpel:query>
192     </bpel:to>
193   </bpel:copy>
194   <bpel:copy>
195     <bpel:from part="payload" variable="input">
196       <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:subLang:xpath1.0">
197         <![CDATA[tns:artefact2]]>
198       </bpel:query>
199     </bpel:from>
200     <bpel:to part="parameters" variable="ProvenancePortTypeinputProvenancePLRequest">
201       <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:subLang:xpath1.0">
202         <![CDATA[provenanceServiceprovenance:inputProvenance[1]/provenanceServiceprovenance:value]]>
203       </bpel:query>
204     </bpel:to>
205   </bpel:copy>
206   <bpel:copy>
207     <bpel:from>
208       <bpel:literal>82</bpel:literal>
209     </bpel:from>
210     <bpel:to part="parameters" variable="ProvenancePortTypeinputProvenancePLRequest">

```

```

211         <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
212             <![CDATA[provenanceServiceprovenance:inputProvenance[2]/provenanceServiceprovenance:key]]>
213         </bpel:query>
214     </bpel:to>
215 </bpel:copy>
216 <bpel:copy>
217     <bpel:from part="payload" variable="input">
218         <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
219             <![CDATA[tns:artefact1]]>
220         </bpel:query>
221     </bpel:from>
222     <bpel:to part="parameters" variable="ProvenancePortTypeinputProvenancePLRequest">
223         <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
224             <![CDATA[provenanceServiceprovenance:inputProvenance[2]/provenanceServiceprovenance:value]]>
225         </bpel:query>
226     </bpel:to>
227 </bpel:copy>
228 </bpel:assign>
229 <bpel:invoke inputVariable="ProvenancePortTypeinputProvenancePLRequest"
230     name="InputProvenanceInvoke" operation="inputProvenance"
231     outputVariable="workflowInstanceIdVariable" partnerLink="ProvenancePortTypePL"
232     portType="provenanceServiceprovenance:ProvenancePortType" />
233 <bpel:flow>
234     <bpel:sequence>
235         <bpel:assign validate="no">
236             <bpel:copy>
237                 <bpel:from>
238                     <bpel:literal>
239                         <impl:add xmlns:impl="http://huff.calc.les.din.uem.br"
240                             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
241                             <impl:value1 />
242                             <impl:value2 />
243                         </impl:add>
244                     </bpel:literal>
245                 </bpel:from>
246                 <bpel:to part="parameters" variable="CalculadoraaddPLRequest" />
247             </bpel:copy>
248             <bpel:copy>
249                 <bpel:from part="payload" variable="input">
250                     <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
251                         <![CDATA[tns:artefact1]]>
252                     </bpel:query>
253                 </bpel:from>
254                 <bpel:to part="parameters" variable="CalculadoraaddPLRequest">
255                     <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
256                         <![CDATA[implCalculadora:value1]]>
257                     </bpel:query>
258                 </bpel:to>
259             </bpel:copy>
260             <bpel:copy>
261                 <bpel:from part="payload" variable="input">
262                     <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
263                         <![CDATA[tns:artefact2]]>
264                     </bpel:query>
265                 </bpel:from>
266                 <bpel:to part="parameters" variable="CalculadoraaddPLRequest">
267                     <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
268                         <![CDATA[implCalculadora:value2]]>
269                     </bpel:query>
270                 </bpel:to>
271             </bpel:copy>
272         </bpel:assign>
273         <bpel:invoke inputVariable="CalculadoraaddPLRequest"
274             name="AddSoftwareActivityInvoke" operation="add"
275             outputVariable="CalculadoraaddPLResponse" partnerLink="CalculadoraPL"
276             portType="implCalculadora:Calculadora" />
277     </bpel:sequence>
278     <bpel:sequence>
279         <bpel:assign validate="no">
280             <bpel:copy>
281                 <bpel:from>
282                     <bpel:literal>
283                         <provenanceService:inputProvenanceRequest
284                             xmlns:provenanceService="http://provenance.clusterflow.din.uem.br"
285                             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
286                             <provenanceService:entity />
287                             <provenanceService:entityId />
288                             <provenanceService:enactmentId />
289                             <provenanceService:inputProvenance>
290                                 <provenanceService:key />
291                                 <provenanceService:value />
292                             </provenanceService:inputProvenance>
293                             <provenanceService:inputProvenance>

```

```

294         <provenanceService:key />
295         <provenanceService:value />
296     </provenanceService:inputProvenance>
297 </provenanceService:inputProvenanceRequest>
298 </bpel:literal>
299 </bpel:from>
300 <bpel:to part="parameters" variable="ProvenancePortTypeinputProvenancePLRequest" />
301 </bpel:copy>
302 <bpel:copy>
303     <bpel:from part="parameters"
304         variable="ProvenancePortTypestartProvenancePLResponse">
305         <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
306             <![CDATA[provenanceServiceprovenance:enactmentId]]>
307         </bpel:query>
308     </bpel:from>
309     <bpel:to part="parameters" variable="ProvenancePortTypeinputProvenancePLRequest">
310         <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
311             <![CDATA[provenanceServiceprovenance:enactmentId]]>
312         </bpel:query>
313     </bpel:to>
314 </bpel:copy>
315 <bpel:copy>
316     <bpel:from>
317         <bpel:literal>ACTIVITY</bpel:literal>
318     </bpel:from>
319     <bpel:to part="parameters" variable="ProvenancePortTypeinputProvenancePLRequest">
320         <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
321             <![CDATA[provenanceServiceprovenance:entity]]>
322         </bpel:query>
323     </bpel:to>
324 </bpel:copy>
325 <bpel:copy>
326     <bpel:from>
327         <bpel:literal>10</bpel:literal>
328     </bpel:from>
329     <bpel:to part="parameters" variable="ProvenancePortTypeinputProvenancePLRequest">
330         <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
331             <![CDATA[provenanceServiceprovenance:entityId]]>
332         </bpel:query>
333     </bpel:to>
334 </bpel:copy>
335 <bpel:copy>
336     <bpel:from>
337         <bpel:literal>86</bpel:literal>
338     </bpel:from>
339     <bpel:to part="parameters" variable="ProvenancePortTypeinputProvenancePLRequest">
340         <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
341             <![CDATA[provenanceServiceprovenance:inputProvenance[1]/provenanceServiceprovenance:key]]>
342         </bpel:query>
343     </bpel:to>
344 </bpel:copy>
345 <bpel:copy>
346     <bpel:from part="payload" variable="input">
347         <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
348             <![CDATA[tns:artefact1]]>
349         </bpel:query>
350     </bpel:from>
351     <bpel:to part="parameters" variable="ProvenancePortTypeinputProvenancePLRequest">
352         <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
353             <![CDATA[provenanceServiceprovenance:inputProvenance[1]/provenanceServiceprovenance:value]]>
354         </bpel:query>
355     </bpel:to>
356 </bpel:copy>
357 <bpel:copy>
358     <bpel:from>
359         <bpel:literal>88</bpel:literal>
360     </bpel:from>
361     <bpel:to part="parameters" variable="ProvenancePortTypeinputProvenancePLRequest">
362         <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
363             <![CDATA[provenanceServiceprovenance:inputProvenance[2]/provenanceServiceprovenance:key]]>
364         </bpel:query>
365     </bpel:to>
366 </bpel:copy>
367 <bpel:copy>
368     <bpel:from part="payload" variable="input">
369         <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
370             <![CDATA[tns:artefact2]]>
371         </bpel:query>
372     </bpel:from>
373     <bpel:to part="parameters" variable="ProvenancePortTypeinputProvenancePLRequest">
374         <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
375             <![CDATA[provenanceServiceprovenance:inputProvenance[2]/provenanceServiceprovenance:value]]>
376         </bpel:query>
377     </bpel:to>
378 </bpel:copy>

```

```

379         </bpel:assign>
380         <bpel:invoke inputVariable="ProvenancePortTypeinputProvenancePLRequest"
381             name="InputProvenanceInvoke" operation="inputProvenance"
382             outputVariable="ProvenancePortTypeinputProvenancePLResponse"
383             partnerLink="ProvenancePortTypePL" portType="provenanceServiceprovenance:ProvenancePortType" />
384     </bpel:sequence>
385 </bpel:flow>
386 <bpel:assign validate="no">
387     <bpel:copy>
388         <bpel:from>
389             <bpel:literal>
390                 <provenanceService:outputProvenanceRequest
391                     xmlns:provenanceService="http://provenance.clusterflow.din.uem.br"
392                     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
393                     <provenanceService:instanceId />
394                     <provenanceService:outputProvenance>
395                         <provenanceService:key />
396                         <provenanceService:value />
397                     </provenanceService:outputProvenance>
398                 </provenanceService:outputProvenanceRequest>
399             </bpel:literal>
400         </bpel:from>
401         <bpel:to part="parameters" variable="ProvenancePortTypeoutputProvenancePLRequest" />
402     </bpel:copy>
403     <bpel:copy>
404         <bpel:from part="parameters"
405             variable="ProvenancePortTypeinputProvenancePLResponse">
406             <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
407                 <![CDATA[provenanceServiceprovenance:instanceId]]>
408             </bpel:query>
409         </bpel:from>
410         <bpel:to part="parameters" variable="ProvenancePortTypeoutputProvenancePLRequest">
411             <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
412                 <![CDATA[provenanceServiceprovenance:instanceId]]>
413             </bpel:query>
414         </bpel:to>
415     </bpel:copy>
416     <bpel:copy>
417         <bpel:from>
418             <bpel:literal>85</bpel:literal>
419         </bpel:from>
420         <bpel:to part="parameters" variable="ProvenancePortTypeoutputProvenancePLRequest">
421             <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
422                 <![CDATA[provenanceServiceprovenance:outputProvenance[1]/provenanceServiceprovenance:key]]>
423             </bpel:query>
424         </bpel:to>
425     </bpel:copy>
426     <bpel:copy>
427         <bpel:from part="parameters" variable="CalculadoraaddPLResponse">
428             <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
429                 <![CDATA[implCalculadora:addReturn]]>
430             </bpel:query>
431         </bpel:from>
432         <bpel:to part="parameters" variable="ProvenancePortTypeoutputProvenancePLRequest">
433             <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
434                 <![CDATA[provenanceServiceprovenance:outputProvenance[1]/provenanceServiceprovenance:value]]>
435             </bpel:query>
436         </bpel:to>
437     </bpel:copy>
438 </bpel:assign>
439 <bpel:invoke inputVariable="ProvenancePortTypeoutputProvenancePLRequest"
440     name="OutputProvenanceInvoke" operation="outputProvenance"
441     outputVariable="ProvenancePortTypeoutputProvenancePLResponse"
442     partnerLink="ProvenancePortTypePL" portType="provenanceServiceprovenance:ProvenancePortType" />
443 <bpel:assign validate="no">
444     <bpel:copy>
445         <bpel:from>
446             <bpel:literal>
447                 <provenanceService:outputProvenanceRequest
448                     xmlns:provenanceService="http://provenance.clusterflow.din.uem.br"
449                     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
450                     <provenanceService:instanceId />
451                     <provenanceService:outputProvenance>
452                         <provenanceService:key />
453                         <provenanceService:value />
454                     </provenanceService:outputProvenance>
455                 </provenanceService:outputProvenanceRequest>
456             </bpel:literal>
457         </bpel:from>
458         <bpel:to part="parameters" variable="ProvenancePortTypeoutputProvenancePLRequest" />
459     </bpel:copy>

```

```

460 <bpel:copy>
461   <bpel:from part="parameters" variable="workflowInstanceIdVariable">
462     <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
463       <![CDATA[provenanceServiceprovenance:instanceId]]>
464     </bpel:query>
465   </bpel:from>
466   <bpel:to part="parameters" variable="ProvenancePortTypeoutputProvenancePLRequest">
467     <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
468       <![CDATA[provenanceServiceprovenance:instanceId]]>
469     </bpel:query>
470   </bpel:to>
471 </bpel:copy>
472 <bpel:copy>
473   <bpel:from>
474     <bpel:literal>90</bpel:literal>
475   </bpel:from>
476   <bpel:to part="parameters" variable="ProvenancePortTypeoutputProvenancePLRequest">
477     <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
478       <![CDATA[provenanceServiceprovenance:outputProvenance[1]/provenanceServiceprovenance:key]]>
479     </bpel:query>
480   </bpel:to>
481 </bpel:copy>
482 <bpel:copy>
483   <bpel:from part="parameters" variable="CalculadoraaddPLResponse">
484     <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
485       <![CDATA[implCalculadora:addReturn]]>
486     </bpel:query>
487   </bpel:from>
488   <bpel:to part="parameters" variable="ProvenancePortTypeoutputProvenancePLRequest">
489     <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
490       <![CDATA[provenanceServiceprovenance:outputProvenance[1]/provenanceServiceprovenance:value]]>
491     </bpel:query>
492   </bpel:to>
493 </bpel:copy>
494 </bpel:assign>
495 <bpel:invoke inputVariable="ProvenancePortTypeoutputProvenancePLRequest"
496   name="OutputProvenanceInvoke" operation="outputProvenance"
497   outputVariable="ProvenancePortTypeoutputProvenancePLResponse"
498   partnerLink="ProvenancePortTypePL" portType="provenanceServiceprovenance:ProvenancePortType" />
499 </bpel:sequence>
500</bpel:process>

```

Listagem 14. Código WS-BPEL completo do workflow de exemplo CalcWorkflow

Apêndice D

Este apêndice tem por objetivo apresentar e descrever as principais atividades presentes na especificação WS-BPEL. Estas atividades foram utilizadas pelo mecanismo de exportação do ambiente *ClusterFlow*. Desta forma, a descrição das mesmas pode auxiliar o entendimento dos códigos XML da especificação WS-BPEL.

D.1 Principais Atividades Presentes na Linguagem WS-BPEL

As atividades básicas e estruturadas da especificação WS-BPEL podem ser classificadas de acordo com a funcionalidade que cada uma oferece (ANDREWS *et al.*, 2003). Desta forma, as atividades básicas da especificação são as seguintes:

- invoke: submete uma mensagem de requisição a um serviço web e pode receber uma mensagem de resposta. A chamada ao serviço pode ocorrer de modo assíncrono ou síncrono. As invocações assíncronas exigem apenas a variável de entrada, pois elas não esperam resposta como parte da operação. Por outro lado as invocações síncronas exigem que as variáveis de entrada e saída estejam devidamente declaradas;
- receive: oferece o serviço de inicialização do *workflow* e recebe requisições de serviços web. Esta atividade tem característica bloqueante, pois permanece aguardando uma mensagem de um serviço web participante para iniciar a execução;
- reply: usada para enviar uma mensagem de resposta. As mensagens enviadas por esta atividade compreendem a execução de *workflow* síncrono. Para *workflows* assíncronos a atividade *invoke* é a responsável por responder as requisições;
- assign: atividade responsável por manipular os valores das variáveis do *workflow*;
- throw: é utilizada quando *workflows* necessitam sinalizar explicitamente uma falha interna de execução;

- wait: permite que o *workflow* especifique um atraso na execução por um certo período ou até que uma certa data final seja encontrada; e
- empty: é usada quando existe a necessidade de uma atividade que não execute alguma operação, por exemplo, quando uma falha precisa ser capturada e suprimida.

As principais atividades estruturadas e utilizadas neste trabalho são sumarizadas a seguir:

- sequence: define um conjunto de atividades que devem respeitar uma determinada ordem de execução;
- while: usada para executar uma ou mais atividades repetidas vezes até que uma condição booleana seja satisfeita;
- pick: aguarda a ocorrência de um evento dentre um conjunto de eventos e executa a atividade associada àquele evento ocorrido. Pode ser comparada à atividade *receive*, porém não pode instanciar um *workflow*;
- flow: oferece a concorrência e sincronização de atividades. Esta atividade encerra sua execução somente quando todas as suas atividades foram concluídas;
- scope: usada para definir que suas atividades internas pertençam a um contexto específico. Estas atividades internas não podem ser manipuladas por atividades externas pertencentes ao mesmo *workflow*. Ela permite que um conjunto de atividades possa ser agrupado em uma mesma transação; e
- if: executa suas atividades internas se uma condição booleana for verdadeira.

Apêndice E

Este apêndice visa mostrar as telas do ambiente *ClusterFlow* que não foram ilustradas no corpo da dissertação. As figuras são apresentadas sem as respectivas descrições textuais.

Figura 53. Tela de login do ClusterFlow

All Workflow List			
Workflow	Goals	Owner	Options
Ant Experiment	The workflow's goal is to abstract the commands to run the Ant Application	A. Huff	<input type="button" value="Run"/>
Knowledge Database	Discovery This workflow examines some algorithms to solve the classification task in data mining	Barros	

Figura 54. Tela para visualizar todos os workflows compartilhados

ClusterFlow - Scientific Experiments Environment User: huff

Home

Workflow Management

- All Workflow List
- My Workflow List
- Create Workflow
- Running Workflows
- All Workflow Results
- My Workflow Results

People Activities

- Inbox
- Notifications

More Actions

- User Profile
- Logout

Welcome to ClusterFlow

ClusterFlow is an environment which permits scientists to manage and perform scientific experiments in a cluster of computers through web services.

UEM - State University of Maringa
Colombo Avenue, 5.790 University Garden - Maringa - Parana - Brazil
Zip Code: 87020-900 Phone: +55.44.3011-4040

Figura 55. Página inicial e menu de operações do ambiente

Workflow Reuse

Workflow: --- Select a Workflow to Reuse ---

Reuse the whole workflow

Reuse Cancel

Figura 56. Tela para reutilizar workflows

New Input Artefact

Artefact Name: problemsToSolve

Type: Text

Add Cancel

Figura 57. Tela para definir os artefatos de entrada dos workflows

Activity Reuse

Workflow: --- Select a Worflow ---

Workflow's Activity: --- Select an Activity to Reuse ---

Reuse Cancel

Figura 58. Tela para o reuso de atividades de workflows compartilhados

Workflow Results		
Workflow	Goals	Owner
Calc Workflow	The Calc Workflow's goal is to sum two integer numbers	A. Huff
Ant Experiment	The workflow's goal is to abstract the commands to run the Ant Application	A. Huff

Figura 59. Tela para selecionar as instâncias de execução de workflows

Livros Grátis

(<http://www.livrosgratis.com.br>)

Milhares de Livros para Download:

[Baixar livros de Administração](#)

[Baixar livros de Agronomia](#)

[Baixar livros de Arquitetura](#)

[Baixar livros de Artes](#)

[Baixar livros de Astronomia](#)

[Baixar livros de Biologia Geral](#)

[Baixar livros de Ciência da Computação](#)

[Baixar livros de Ciência da Informação](#)

[Baixar livros de Ciência Política](#)

[Baixar livros de Ciências da Saúde](#)

[Baixar livros de Comunicação](#)

[Baixar livros do Conselho Nacional de Educação - CNE](#)

[Baixar livros de Defesa civil](#)

[Baixar livros de Direito](#)

[Baixar livros de Direitos humanos](#)

[Baixar livros de Economia](#)

[Baixar livros de Economia Doméstica](#)

[Baixar livros de Educação](#)

[Baixar livros de Educação - Trânsito](#)

[Baixar livros de Educação Física](#)

[Baixar livros de Engenharia Aeroespacial](#)

[Baixar livros de Farmácia](#)

[Baixar livros de Filosofia](#)

[Baixar livros de Física](#)

[Baixar livros de Geociências](#)

[Baixar livros de Geografia](#)

[Baixar livros de História](#)

[Baixar livros de Línguas](#)

[Baixar livros de Literatura](#)
[Baixar livros de Literatura de Cordel](#)
[Baixar livros de Literatura Infantil](#)
[Baixar livros de Matemática](#)
[Baixar livros de Medicina](#)
[Baixar livros de Medicina Veterinária](#)
[Baixar livros de Meio Ambiente](#)
[Baixar livros de Meteorologia](#)
[Baixar Monografias e TCC](#)
[Baixar livros Multidisciplinar](#)
[Baixar livros de Música](#)
[Baixar livros de Psicologia](#)
[Baixar livros de Química](#)
[Baixar livros de Saúde Coletiva](#)
[Baixar livros de Serviço Social](#)
[Baixar livros de Sociologia](#)
[Baixar livros de Teologia](#)
[Baixar livros de Trabalho](#)
[Baixar livros de Turismo](#)