COPPE/UFRJ

ESPECIFICAÇÃO FORMAL E VERIFICAÇÃO AUTOMÁTICA DE *WORKFLOWS* CIENTÍFICOS

Edno Vicente da Silva

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Marta Lima de Queirós Mattoso
Mario Roberto Folhadela Benevides

Rio de Janeiro
Setembro de 2010

ESPECIFICAÇÃO FORMAL E VERIFICAÇÃO AUTOMÁTICA
DE *WORKFLOWS* CIENTÍFICOS

Edno Vicente da Silva

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

_____

Profª. Marta Lima de Queirós Mattoso, D.Sc.

_____

Prof. Mario Roberto Folhadela Benevides, PhD.

_____

Prof. Alexandre de Assis Bento Lima, D.Sc.

_____

Profª. Jonice de Oliveira Sampaio, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2010

Silva, Edno Vicente da

Especificação Formal e Verificação Automática de Workflows Científicos / Edno Vicente da Silva - Rio de Janeiro: UFRJ/COPPE, 2010.

IX, 40 p.: il.; 29,7 cm.

Orientadores: Marta Lima de Queirós Mattoso

Mario Roberto Folhadela Benevides.

Dissertação (mestrado) – UFRJ / COPPE / Programa de Engenharia de Sistemas e Computação, 2010.

Referências Bibliográficas: p. 36-40.

1. Verificação de Workflows. 2. Workflows Científicos. 3. Álgebras de Processo. 4. Verificação de Modelos. I. Mattoso, Marta Lima de Queirós et al. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

À minha esposa, minha mãe e minha avó.

Agradecimentos

Em primeiro lugar, agradeço a Professora Marta Lima de Queirós Mattoso, que me guiou durante os três últimos anos.

Agradeço também ao Mario Roberto Folhadela Benevides, que desde o projeto final da graduação vem me dando suporte e orientação.

Aos Professores Jonice de Oliveira Sampaio e Alexandre de Assis Bento Lima, que fazem parte desta banca.

Ao CNPq, que deu suporte financeiro a este trabalho, sem o qual eu não conseguiria desenvolvê-lo.

Ao Professor Jano Moreira de Souza, líder da linha de banco de dados, que me me deu a oportunidade de fazer o mestrado.

Aos Professores José Roberto Blaschek e Jano Moreira de Souza pela oportunidade na Fundação COPPETEC.

Aos Professores da graduação e do mestrado, Geraldo Zimbrão, Geraldo Xexéo, Márcia Cerioli, Pedro Manoel, Severino Collier, Guiherme Chagas, Claudson Bornstein, Miguel Jonathan, Adriano Cruz, Maria Luiza, Vanessa Braganholo, Ageu Pacheco e Milton Flores, que me tornaram o profissional que sou hoje.

A minha mãe, Valdete, que por muito tempo se sacrificou para dar aos filhos uma boa educação.

A minha avó, Hercília, que ajudou na minha criação e formação.

A Cristiane Furtado da Silva, minha esposa, que me apoiou em todos os momentos difíceis que passei neste ano.

Não posso me esquecer de agradecer aos alunos do PESC que contrinuiram para que esta dissertação fosse concretizada, em especial a Sérgio Serra, Daniel Cardoso, Eduardo Ogasawara, Frederico Tosta, Fernando Morgado, Clarissa Vilela, Vanessa Batista e Cimar José.

Agradeço também aos amigos que fiz na Fundação COPPETEC e na DATAPREV, especialmente a Gustavo Pinto, Amanda Mattos, Rafael Brand, Rodrigo Hungria, Fernando Teijeira, Fernando Diniz, Diogo Pisaneschi, João Fernando, Carlos Leão e Raphael Martins.

Por fim, agradeço a Deus, que me dá saúde e paz.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ESPECIFICAÇÃO FORMAL E VERIFICAÇÃO AUTOMÁTICA DE WORKFLOWS CIENTÍFICOS

Edno Vicente da Silva

Setembro / 2010

Orientadores: Marta Lima de Queirós Mattoso
Mario Roberto Folhadela Benevides.

Programa: Engenharia de Sistemas e Computação

Workflows são utilizados em diversos domínios com propósitos científicos. Nos últimos anos estes workflows tornaram-se mais complexos e os cientistas necessitam de métodos para verificar a sua correção. A maioria dos sistemas disponíveis pressupõe que um workflow está correto se respeita os controles e as dependências definidos pelo cientista. Além disso, muitos workflows científicos devem ser completamente confiáveis, e por isso devem estar especificados corretamente. Este trabalho propõe uma abordagem para verificação de workflows baseada em uma álgebra de processo (CCS) e ferramentas de verificação de modelos. Essa abordagem foi implementada como prova de conceito junto à ferramenta de especificação de workflows GExpLine. Experimentos realizados sob a ferramenta mostram as vantagens da verificação automática.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

FORMAL SPECIFICATION AND AUTOMATIC VERIFICATION
OF SCIENTIFIC WORKFLOWS

Edno Vicente da Silva

September / 2010

Advisors: Marta Lima de Queirós Mattoso
Mario Roberto Folhadela Benevides.

Department: Computer and Systems Engineering

Workflows are used in several domains for scientific purposes. In the last years these workflows are becoming more complex and scientists need methods to verify its correctness. Most of the available systems assume that a workflow is correct if it respects control and dependencies specified by the scientist. In addition, many scientific workflows must be completely reliable, that is why they must be correctly specified. This work proposes an approach that supports workflow verification based on process algebra specifications (CCS) and model checking tools. This approach was implemented as proof of concept with the GExpLine workflow specification tool. Experiments performed under the tool show the advantages of automatic verification.

Índice:

1.1 – Motivação

Workflow científico pode ser definido como a especificação formal de um processo científico, que representa os passos a serem executados em uma experiência científica **(DEELMAN ET AL., 2009)**. Estes passos são normalmente associados à seleção de dados, integração de programas, análise e visualização. Workflows científicos são construídos em sistemas de execução complexos chamados Sistemas Gerenciadores de Workflows Científicos (SGWfC) que suportam a especificação de workflows em termos de artefatos executáveis (programas ou serviços). Eles executam uma especificação de workflow científico fornecendo mecanismos para encadeamento de atividades e monitoramento.

Existem várias iniciativas para padronizar a especificação de linguagens de workflow **(BARGA ET GANNON, 2007, SLOMINSKI, 2007)**, mas isto ainda é um problema em aberto. Cada SGWfC tem sua própria linguagem, o que significa que um mesmo experimento científico pode ser especificado de diferentes formas em sistemas distintos.

Como a tarefa de especificação tem a característica de ser trabalhosa, pode haver algumas falhas na especificação dos workflows. Muitos experimentos são complexos e sua especificação pode levar a problemas como *deadlocks*, falhas de sincronização e execuções por tempo indeterminado **(CHEN ET YANG, 2008)**. A execução de workflows incorretos desperdiça recursos computacionais valiosos, além de que é necessária alguma interferência dos cientistas para impedir as execuções problemáticas. Dependendo do ambiente computacional utilizado para executar o experimento, é longe de ser trivial interrompê-lo. Mesmo quando é possível cancelar a execução, a re-execução pode ser complexa demais.

A construção de workflows corretos (estrutural e semanticamente) é colocada como uma tarefa fundamental para o processo de experimentação. Ao garantir a correção dos workflows científicos, é possível poupar quantidade significativa de trabalho dos cientistas e poupar valiosos recursos computacionais. No entanto, isto está longe de ser tarefa trivial. Técnicas especiais devem ser aplicadas para verificar os workflows gerados. Uma vez que existem muitos SGWfC, e os cientistas podem usar qualquer um deles para especificar seus experimentos, deveríamos ser capazes de verificar os workflows em todos estes sistemas, o que pode não ser viável.

A verificação de workflows é especialmente importante para o processo de experimentação científica. Existem várias propostas nesta área. No entanto, permanece um problema em aberto. Algumas obras usam *parsers* para verificar a sintaxe do workflow **(CHEN ET YANG, 2008)**. Embora isso seja útil, não garante a correção da especificação em relação a *deadlocks*, por exemplo. Outras abordagens usam álgebras de processo para verificar a especificação de workflows **(WOODMAN ET AL., 2007)**. Estas abordagens apresentam vantagens, como fornecer meios para procurar padrões em especificações de workflows e encontrar erros com base nesses padrões.

Todas essas abordagens estão centradas na especificação de workflows científicos prontos para execução, porém eles são suscetíveis ao problema de linguagens de workflow não padronizadas. Cada uma das técnicas propostas deve ser aplicada a todos os SGWfC para garantir a correção da especificação executável. No entanto, as técnicas de aplicação a todos os sistemas não é recomendado, pois cada SGWfC deve ser remodelado e re-implementado com a inclusão de mecanismos de verificação.

Uma solução possível é utilizar workflows abstratos para ajudar na verificação de experimentos científicos. Como os workflows se tornam cada vez mais complexos, há um aumento na necessidade de apoiar a abstração de workflows em experimentos científicos **(LUDÄSCHER, 2009, MATTOSO ET AL., 2010, OGASAWARA ET AL., 2009, SHOSHANI, 2009)**. Especificações abstratas de experimentos científicos permitem que os cientistas se concentrem na própria ciência, em vez de se preocuparem com recursos computacionais específicos. No entanto, o aspecto mais importante é que workflows abstratos podem ser derivados em especificações de workflows executáveis

em SGWfC diferentes. As especificações executáveis, prontas para execução em SGWfC, são conhecidas como workflows concretos **(DEELMAN ET AL., 2009)**. É importante ressaltar que a proposta de verificação de um workflow não se dá no nível abstrato e sim no momento da derivação de uma especificação abstrata para uma equivalente concreta. Executar a verificação do workflow no nível abstrato não é o foco deste trabalho. Uma primeira verificação pode ser realizada no nível abstrato, mas os benefícios não são tangíveis.

1.2 – Objetivo

Esta dissertação propõe uma abordagem baseada em padrões de workflow **(AALST ET AL., 2003, RUSSELL ET AL., 2006)** e álgebras de processos **(MILNER ET AL., 1992a, 1992b)** para apoiar a especificação e verificação dos workflows científicos. A verificação é feita durante o processo de derivação de uma especificação de um workflow abstrato em um executável. Avaliamos a nossa abordagem usando GExpLine como a ferramenta de modelagem de workflow **(OGASAWARA ET AL., 2009)**. Assim sendo, é possível verificar workflows no momento da derivação para SGWfC diferentes, como a Taverna **(OINN ET AL., 2004)**, VisTrails **(CALLAHAN ET AL., 2006)**, Kepler **(ALTINTAS ET AL., 2004)**.

Atualmente, a maioria dos mecanismos para verificação de workflows existentes foram construídas para tratar workflows para processos de negócio. Neste contexto, destaca-se o uso dois formalismos: redes de Petri **(PETRI, 1962)** e álgebras de processo, tais como CCS **(MILNER, 1989)** e Pi-cálculo **(MILNER ET AL., 1992a, 1992b)**. Alguns destes mecanismos também utilizam-se de padrões de workflow **(AALST ET AL., 2003, RUSSELL ET AL., 2006)** como formalismo complementar. Assim sendo, esta dissertação aborada justamente o uso de padrões de workflow e álgebras de processo (notadamente CCS) e como formalismo para verificação de workflows científicos.

Nesta dissertação veremos um modelo formal de representação de workflows científicos, uma abordagem de verificação, e ainda os detalhes de implementação destes conceitos na ferramenta de especificação de workflows GExpLine. Os resultados obtidos mostram que é possível detectar erros na especificação de workflows

científicos, e assim podemos evitar que erros deste tipo possam aparecer em ambientes de execução real.

1.3 – Organização dos Capítulos

Esta dissertação está dividiva em 5 capítulos.

No capítulo 2 são levantados os principais conceitos e tecnologias empregados no seu desenvolvimento, dentre eles: padrões de workflow, álgebrtas de processo, sistemas gerenciadores de workflows científicos e verificação de modelos.

O capítulo 3 apresenta a proposta da dissertação, os formalismos adotados e a arquitetura de implementação dos mecanismos de verificação incluídos na ferramenta GExpLine.

O capítulo 4 apresenta aplicações da verificação de workflows que demonstram o uso e a avaliação da proposta.

O capítulo 5 apresenta a conclusão do trabalho, assim como aponta futuros trabalhos relacionados.

Netse capítulo, estaremos levantando as bases que são pertinentes à dissertação. Iremos contemplar definções sofre workflows, um sistema para gerência de workflows (SGWfC), formalismos para representar sistemas concorrentes (álgebras de processos), técnicas e ferramentas de verificação de modelos. Verificar a correção da estrutura de workflows é uma tarefa fundamentail na área de workflows científicos, pois a execução de tais processos costuma consumir meses na execução de atividades complexas.

2.1 – Workflows

Nesta seção discutiremos a especificidade dos workflows científicos, e apresentaremos ainda um Sistema Gerenciador de Workflows Científicos, uma linguagem de representação de workflows, e a ferramenta de especificação de workflows GExpLine.

2.1.1 – Workflow e Workflow Científico

Segundo a Workflow Management Coalition (WFMC, 1999) workflow é definido como "a automação de um processo de negócio, inteiro ou por partes, durante o qual documentos, informações e atividades são passadas de um participante para outro para que estes desenvolvam suas ações, respeitando um conjunto de regras procedimentais". Esta definição é atribuída aos workflows para processos de negócio, mas sua generalização serve para descrever os workflows especificados por cientistas para realizar seus experimentos. Em geral, os workflows científicos muito se assemelham a eles no que tange a sua forma e aos padrões de workflow ao quais ambos aderem. Entretanto, os workflows científicos caracterizam-se por serem centrados em dados, enquanto os comerciais utilizam maior número de estruturas de controle. Deste modo, sistemas de workflows para processos de negócio são construídos utlizando fluxo de controle e sistemas de workflows cientficos são construídos baseados em fluxo de dados. Workflows científicos processam grandes quantidades de dados que posteriormente são examinados pelos cientistas, e caracterizam-se muitas vezes pela execução de atividades concorrentes. Além disso, características como tolerância a

falhas e capacidade de adaptação forçam a utilização de estruturas de fluxo de controle em sistemas essencialmente orientados a fluxo de dados, o que aumenta consideravelmente a complexidade no trabalho com workflows científicos (BOWERS ET AL., 2006), já que o uso simultâneo de fluxos de dados e de controle comprometem o entendimento, a configuração e manutenção dos workflows.

2.1.2 – Kepler

O Kepler **(ALTINTAS ET AL., 2004, LUDÄSCHER ET AL., 2006)** é um sistema de código aberto de gerência de workflows científicos que permite aos cientistas projetar e executar eficientemente workflows científicos. Dentre suas características, destacam-se: prototipagem de workflows, Execução distribuída, e acesso e consulta a bases de dados.

São cinco os componentes principais do Kepler: Diretores, Atores, Portas, Relações e Parâmetros. O Kepler utiliza a metáfora do diretor/ator para representar visualmente os componentes de um workflow. Diretores controlam a execução do workflow e determinam quando o processamento ocorre. Todo workflow criado no Kepler é monitorado por pelo menos um Diretor. Existem vários Diretores disponíveis, a cada um deles representa um modelo de computação. Dentre os Diretores existentes estão o SDF Director, DDFDirector, DEDirector e PNDirector. Atores são unidades de execução do workflow e determinam qual processamento ocorre. Eles podem ser atômicos ou compostos. Atores compostos são coleções de Atores agrupados para executar operações mais complexas. São utilizados como essencialmente como workflows aninhados (sub-workflows). Cada Ator pode ter uma ou mais portas utilizadas para consumir ou produzir dados e se comunicar com outros atores do workflow. As portas podem ser de entrada (consomem dados), de saída (produzem dados), ou de entrada e saída (consomem e produzem dados). Uma conexão que representa um fluxo de dados entre dois atores do workflow é chamada canal. Uma porta é ainda classificada em única (pode ser conectada a apenas um canal) ou múltipla (pode ser conectada a vários canais). Relações permitem que os dados sejam ramificados em um workflow. Dados ramificados podem ser enviados para diferentes lugares do workflow. Parâmetros são valores configuráveis que podem ser anexados a

um workflow, a diretores, ou a atores individuais. Parâmetros de diretores controlam o número de iterações de um workflow e também o critério relevante para cada iteração.

2.1.3 – XPDL

XML Process Definition Language (XPDL) (WFMC, 2002) é a linguagem proposta pela Workflow Management Coalition para intercâmbio de definições de processos entre produtos de workflow diferentes. É uma linguagem baseada em XML, e é formalizada por um XML Schema. Seus elementos principais são: Package (Pacote), Application, WorkflowProccess, Activity, Transition, Participant, DataField, e DataType.

Package é o elemento que contém os outros elementos. O elemento Application é usado para especificar as aplicações ou ferramentas invocadas pelos processos de workflow definidos em um package. O elemento WorkflowProcess é usado para definir processos de workflow ou partes deles. Um WorkflowProcess é composto de elementos dos tipos Activity e Transition.

O elemento Activity é o bloco básico de construção de uma definição de um processo de workflow. Elementos do tipo Activity são ligados por meio de elementos do tipo Transition. Há três tipos de atividades: Route, Implementation e BlockActivity. Atividades do tipo Route são pseudo-atividades usadas somente para propósitos de "roteamento". Atividades do tipo BlockActivity são usadas para executar conjuntos de atividades menores. O elemento ActivitySet se refere a um conjunto auto-contido de atividades e transições. Um BlockActivity executa tal qual um ActivitySet. Atividades do tipo Implementation são passos no processos que são implementadas por procedimentos manuais, uma ou mais aplicações, ou por outro processo de workflow.

O elemento Participant é utilizado para especificar os participantes no workflow, isto é, as entidades que podem executar o trabalho. Há seis tipos de participantes: ResourceSet, Resource, Role, OrganizationalUnit, Human, e System. Elementos dos tipos DataField e DataType são usados para especificar dados relevantes do workflow, que são usados para tomada de decisão ou para se referir a dados que estão fora do workflow, ou ainda dados passados entre atividades e sub-processos.

A XPDL é um passo importante como uma linguagem universal para representação de workflows. A ferramenta GExpLine (descrita a seguir) utiliza-se de XPDL como uma desuas possíveis representações de workflows.

2.1.4 – GExpLine

GExpLine (OGASAWARA ET AL., 2009) é um sistema de especificação de workflows. Ela suporta workflows abstratos e permite a derivação de workflows concretos para os SGWf Kepler, Taverna (HULL ET AL., 2006, OINN ET AL., 2004) e VisTrails (CALLAHAN ET AL., 2006). A GExpLine representa workflows abstratos em XPDL.

A seguir, o modelo de classes de domínio utilizado pela GExpLine:

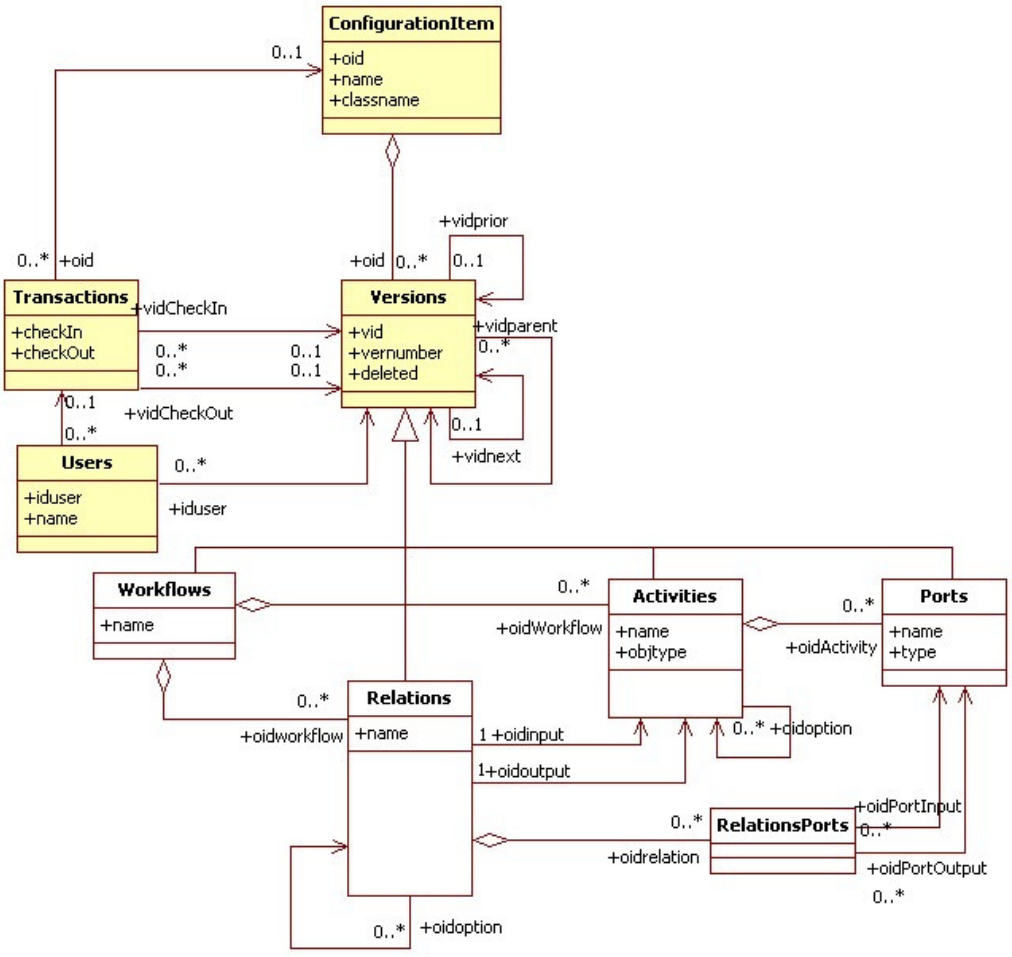


Figura 1 Classes de domínio da GExpLine

As classes em amarelo são destinadas ao tratamento de gerência de configuração existente na GExpLine. Estas classes não são utilizadas diretamente nesta dissertação. As classes em branco especificam os conceitos mais básicos de workflows científicos: workflows, atividades, portas, relações (ou conexões).

2.2 – Padrões de *Workflow*

A iniciativa Workflow Patterns (**AALST ET AL., 2003, RUSSELL ET AL., 2006**), baseada na análise dos diversos sistemas existentes, se propôs a identificar os padrões de construções encontrados com mais freqüência, e de formalizar seus comportamentos de maneira precisa e sem ambigüidades. Tais padrões existem para demonstrar que os sistemas de workflow possuem diferentes níveis de portabilidade e expressividade, além de representar requisitos para linguagens de workflows. Um importante uso dos padrões se dá na comparação entre os diversos sistemas.

Os padrões de workflow são classificados em muitas categorias. Os vinte padrões mais conhecidos estão divididos em cinco categorias: Padrões básicos de controle; Padrões avançados de ramificação e sincronização; Padrões estruturais; Padrões envolvendo múltiplas instâncias; Padrões baseados em estados; e Padrões de cancelamento.

2.2.1 – Padrões básicos de fluxo de controle

A seguir temos uma descrição dos padrões de workflow mais importantes para o desenvolvimento deste trabalho.

Padrão 1 – Sequência

Uma atividade em um processo de workflow é habilitada após o término de uma atividade precedente no mesmo processo.

Padrão 2 - Divisão paralela

A separação de um ramo em dois ou mais ramos paralelos, cada um dos quais executa concorrentemente.

Padrão 3 – Sincronização

A junção de dois ou mais ramos em um único ramo subsequente. O fluxo de controle é passado para o ramo subsequente quando todos os ramos de entrada forem habilitados.

Padrão 4 - Escolha Exclusiva

A separação de um ramo em dois ou mais ramos. Quando o ramo que chega à atividade é habilitado, o fluxo de controle é imediatamente passado para exatamente um dos ramos que saem da atividade, baseado no resultado de uma expressão lógica associada aos ramos que saem da atividade.

Padrão 5 – Junção Simples

A junção de dois ou mais ramos em um único ramo subsequente. Cada habilitação de um ramo que chega na atividade resulta na passagem do fluxo de controle para ao ramo subsequente.

2.2.2 – Padrões avançados de ramificação e sincronização

Padrão 6 – Escolha Múltipla

A separação de um ramo em dois ou mais ramos. Quando o ramo que chega à atividade é habilitado, o fluxo de controle é passado para um ou mais ramos que saem da atividade, baseado no resultado de expressões lógicas distintas associadas a cada um dos ramos que saem da atividade.

Padrão 7 – Junção Sincronizada Estruturada

A junção de dois ou mais ramos (que se separaram anteriormente no processo em um único ponto identificável) em um único ramo subsequente. O fluxo de controle é passado para ramo subsequente quando cada ramo ativo que chega à atividade for habilitado.

Padrão 8 – Junção Múltipla

A junção de dois ou mais ramos em um único ramo subsequente. Cada habilitação de um ramo que chega à atividade resulta na passagem do fluxo de controle para o ramo subseqüente

Padrão 9 – Discriminador Estruturado

A junção de dois ou mais ramos em um único ramo subsequente seguindo uma separação anterior correspondente no processo. O fluxo de controle é passado para o ramo subsequente quando o primeiro ramo que chega é habilitado. Habilitações subsequentes de ramos que chegam não resultam na passagem do fluxo de controle. O discriminador se reinicia quando todos os ramos que chegam forem habilitados.

2.2.3 – Padrões estruturais

Padrão 10 – Ciclos Arbitrários

Habilidade de representar ciclos em um modelo de processo que tem mais de um ponto de entrada ou saída.

Padrão 11 – Término Implícito

Uma instância de um processo (ou de um sub-processo) deve terminar quando não há itens de trabalho restantes que estão aptos a serem terminados neste momento ou em qualquer momento no futuro.

2.3 – Formalismos para representação de *Workflows*

Nesta seção temos a definição de alguns formalismos utilizados com frequência na especificação de workflows: álgebras de processos, como CCS e Pi-cálculo, e redes de Petri.

2.3.1 – CCS

CCS é uma álgebra de processos para representação de sistemas concorrentes **(MILNER, 1989)**. Um sistema é composto de partes que interagem concorrentemente e que se comunicam, chamados agentes. A álgebra se baseia em ações que modelam uma comunicação indivisível entre dois agentes ou uma comunicação interna a um agente. Uma comunicação é feita através de portas de entrada e saída (portas complementares) dos agentes envolvidos. A álgebra usa um conjunto de nomes para representar tais ações, dado por Act = $A \cup \overline{A} \cup \tau$. A representa nomes em letras minúsculas, como a, b, c, ..., que denotam ações em portas de entrada. $\overline{A}$ representa nomes complementares em letras minúsculas com barras superiores, como $\overline{a}$, $\overline{b}$, $\overline{c}$, ..., que denotam ações em portas de saída. A letra grega τ representa uma ação silenciosa. Nomes em letras maiúsculas são utilizados para identificar agentes.

Os agentes de CCS são definidos em termos da seguinte gramática:

P ::= p.P | P+P | P|P | P\L | P[S] | 0

A seguir, temos informalmente a definição dos agentes e operadores de CCS.

p.*P* (Prefixo): o agente executa a ação *p* e depois se comporta como o agente *P*; a ação *p* pode ser uma leitura em uma porta de entrada ($\alpha \in A$), uma escrita em uma porta de saída ($\overline{\alpha} \in \overline{A}$), ou uma ação silenciosa (τ);

P + *Q* (Soma): o agente se comporta como o agente *P* ou como outro agente *Q*, de maneira exclusiva;

P | *Q* (Composição): um agente que é a execução concorrente dos agentes *P* e *Q*;

P \ *L* (Restrição): o conjunto de ações $L \in A \cup \overline{A}$ tem sua execução restrita ao escopo do agente *P*;

P[*S*] (Substituição): os nomes das ações do agente *P* são trocados simultaneamente por meio da função de substituição *S*.

0 (Agente nulo): um agente especial incapaz de realizar qualquer ação.

Também é possível definir um agente como uma Constante. Por exemplo, um agente *A* poderia ser definido do seguinte modo: $A \overset{def}{=} P$, onde *P* é um agente qualquer.

A semântica de CCS é definida por um sistema de transições rotuladas: os estados são dados pelas definições dos agentes *P*, e *Act* representa o conjunto de rótulos de transição. Uma transição em CCS é do tipo $A \xrightarrow{a} A'$. Esta notação quer dizer que o agente *A* executa a ação *a* e torna-se o agente *A*'. A seguir temos a lista de regras de transição de CCS:

$\dfrac{}{P \xrightarrow{a} P'}$ (Ação)	$\dfrac{F \xrightarrow{a} F'}{E \mid F \xrightarrow{a} E \mid F'}$ (Paralelismo 2)
$\dfrac{P \xrightarrow{a} P'}{A \xrightarrow{a} P'}$ $(A \overset{def}{=} P)$ (Constante)	$\dfrac{E \xrightarrow{a} E', F \xrightarrow{\bar{a}} F'}{E \mid F \xrightarrow{\tau} E' \mid F'}$ (Paralelismo 3)

$\dfrac{E_j \xrightarrow{a} E'_j}{\sum_{i \in I} E_j \xrightarrow{a} E'_j}\ (j \in I)$ (Escolha)	$\dfrac{E \xrightarrow{a} E'}{E \setminus L \xrightarrow{a} E' \setminus L}\ (\alpha, \bar{\alpha} \notin L)$ (Restrição)
$\dfrac{E \xrightarrow{a} E'}{E \mid F \xrightarrow{a} E' \mid F}$ (Paralelismo 1)	$\dfrac{E \xrightarrow{a} E'}{E[f] \xrightarrow{f(a)} E'[f]}$ (Substituição)

Por exemplo, a regra de ação denota que um processo *a.E* pode evoluir para um processo *E* executando a ação *a*; a regra de escolha denota que apenas uma das alternativas pode ser evoluída. As regras de paralelismo indicam que todas as alternativas podem evoluir simultaneamente. Note-se que a ação silenciosa τ pode ocorrer quando dois agentes se comunicam (pela regra Paralelismo 3). Isto reflete o fato de que ambos os agentes mudarão de estado, mas a ação que eles executam não é observada por agentes externos.

Um conceito muito importante em CCS é o de bissimulação. Em linhas gerais, uma bissimulação é uma relação de equivalência entre dois processos, onde cada uma das ações que ambos podem executar tem uma correspondência no outro, isto é, se um processo pode executar uma ação, então há uma ação correspondente no outro processo e vice-versa. Duas noções de equivalência são possíveis:

- Equivalência forte (*strong equivalence*): a ação silenciosa τ é tratada como se fosse uma outra ação qualquer;

- Equivalência fraca (ou de observação) (*observational equivalence* ou *weak equivalence*): a existência da ação silenciosa τ é ignorada de modo que, por exemplo, o agente $a.\tau.0$ seja considerado equivalente ao agente $a.0$.

2.3.2 – Pi-cálculo

Pi-cálculo é uma álgebra para sistemas comunicantes onde processos têm estruturas mutáveis (MILNER ET AL., 1992a, 1992b). É uma expansão de CCS que inclui mobilidade enquanto preserva suas propriedades algébricas. A álgebra não

distingue variáveis e constantes. Conexões de comunicação entre processos são identificadas por nomes; a computação é representada pela comunicação de nomes através de conexões, ou seja, uma comunicação transmite (ou recebe) um nome que admite comunicações futuras. Deste modo podemos expressar a transmissão de conexões entre dois processos. Assim, em Pi-cálculo só há duas categorias essenciais de entidades: processos (chamados de agentes em CCS) e nomes.

Existem também algumas diferenças menores em relação ao CCS: o operador de replicação, onde !P = P | !P, e o operador condicional, onde [x = y]P age como P se x = y, e como 0 caso contrário. Há ainda uma nova representação do operador de restrição: νzP equivale a P\{z}.

Assim, em pi-cálculo os processos P são definidos por:

P ::= M | P|P | νzP | !P

M ::= 0 | p.P | M + M

p ::= $\bar{x}\langle y \rangle$ | x(z) | p | [x = y]p

Maiores detalhes sobre a linguagem e sua semântica podem ser encontrados em (MILNER ET AL., 1992a, 1992b), (MILNER, 1999), e (SANGIORGI ET WALKER, 2001).

2.3.3 – Outros formalismos

Existem outras álgebras de processo que podem ser utilizadas como formalismo de representação de workflows. Dentre elas, podemos citar CSP **(HOARE, 1978)**, ACP **(BERGSTRA ET KLOP, 1984)** e LOTOS **(BOLOGNESI ET BRINKSMA, 1987)**.

Outro formalismo frequentemente utilizado são redes de Petri **(PETRI, 1962)**, que têm sido adaptadas por muitos SGWf para descrever processos de negócio. Redes de Petri possuem um profundo fundamento matemático e também uma poderosa representação visual. Redes de Petri utilizam o conceito de representação explícita de estados para sistemas paralelos. Cada rede de Petri está sempre em um estado precisamente definido, denotado por uma distribuição de *tokens* sobre os lugares contidos na rede. O estado de um sistema pode ser alterado por meio de transmissões que mudam a distribuição posição dos *tokens* sobre os lugares.

2.3.4 – Linguagens para workflows derivadas de álgebras de processos

Algumas pesquisas recentes se propuseram a criar novas linguagens baseadas em álgebras de processos de modo a facilitar o processo de geração de especificação formal de workflows. Tais linguagens podem servir como linguagens intermediárias entre as álgebras de processo e o modelo dos workflows.

NPDL **(BRAGHETTO ET AL., 2007)** é uma linguagem para representar processos de negócio baseada em ACP, e foi implementada como uma extensão da linguagem SQL.

SMAWL **(STEFANSEN, 2005)** é uma linguagem para representar processos de negócio baseada em CCS. Sua definição inclui um algoritmo de tradução para CCS.

Todas as linguagens acima foram concebidas para representar todos os 20 primeiros padrões de workflow propostos por Aalst **(AALST ET AL., 2003)**, mas nestes trabalhos não são explorados mecanismos de verificação.

2.3.5 – Geração de especificação de workflows em álgebras de processo

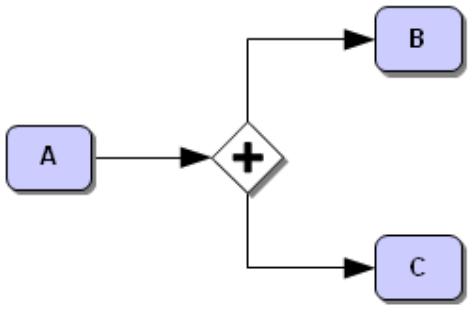
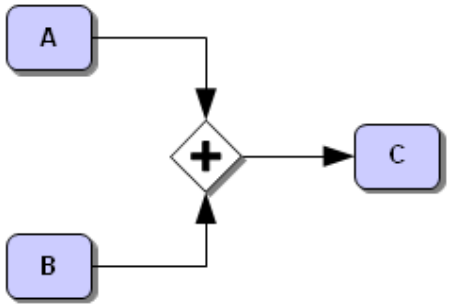
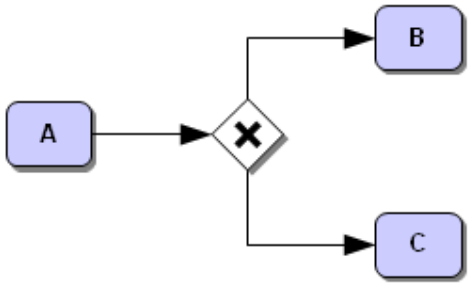
Uma relevante abordagem de representação de atividades de workflows em álgebras de processo é aquela onde cada atividade é mapeada conceitualmente em um processo da álgebra **(PUHLMANN ET WESKE, 2005)**. Estes processos utilizam eventos em forma de comunicação para coordenar o comportamento de um workflow. Vários processos juntos formam um padrão de comportamento, que representa um padrão de workflow.

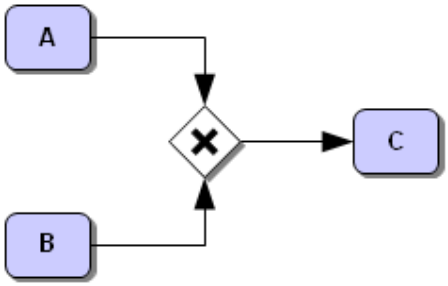
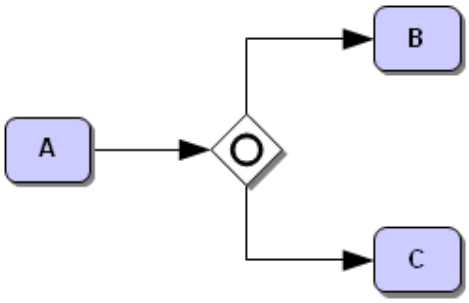
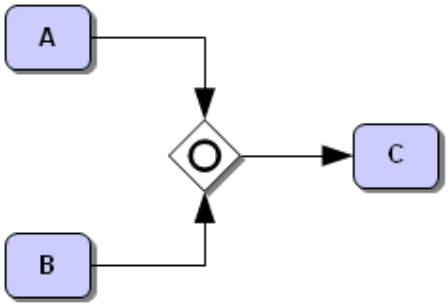
Uma atividade simples, que chamaremos de *A*, que obtém valores de suas portas de entrada *x* e *y*, faz um processamento interno e depois transmite valores em suas portas de saída *z* e *w* poderia ser descrito da seguinte forma em CCS:

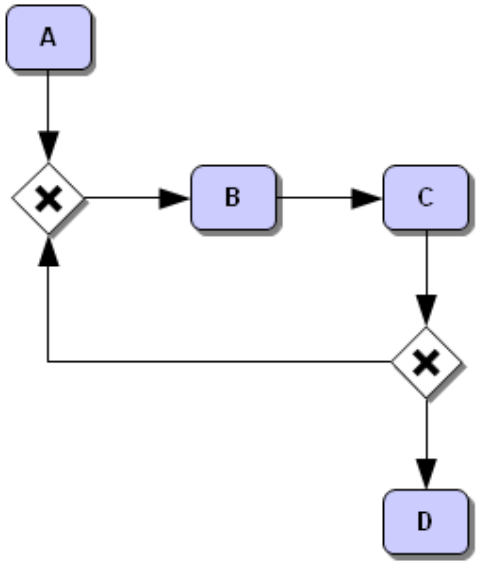
$$A = x \,.\, y \,.\, \tau \,.\, \overline{z} \,.\, \overline{w} \,.\, 0$$

Onde a ação silenciosa τ representa o processamento interno da atividade A.

Os padrões mais básicos definidos em **(AALST ET AL., 2003, RUSSELL ET AL., 2006)** podem ter a seguinte representação, sendo P o processo que define cada padrão, e *A*, *B* e *C* processos que representam atividades do workflow:

	1. Seqüência: $P = A \mid B \setminus \{ab\}$ $A = \tau . \overline{ab} . 0$ $B = ab . \tau . 0$
	2. Divisão paralela: $P = A \mid B \mid C \setminus \{ab, ac\}$ $A = \tau.(\overline{ab} . 0 \mid \overline{ac} . 0)$ $B = ab . \tau . 0$ $C = ac . \tau . 0$
	3. Sincronização: $P = A \mid B \mid C \setminus \{ab, ac\}$ $A = \tau . \overline{ac} . 0$ $B = \tau . \overline{bc} . 0$ $C = ac . bc . \tau . 0$
	4. Escolha exclusiva: $P = A \mid B \mid C \setminus \{ab, ac\}$ $A = \tau . (\overline{ab} . 0 + \overline{ac} . 0)$ $B = ab . \tau . 0$ $C = ac . \tau . 0$

	5. Junção Simples: $P = A \mid B \mid C \setminus \{ab, ac\}$ $A = \tau . \overline{ac} . 0$ $B = \tau . \overline{bc} . 0$ $C = ac . \tau . 0 + bc . \tau . 0$
	6. Escolha múltipla: $P = A \mid B \mid C \setminus \{ab, ac\}$ $A = \tau.((\overline{ab} . 0 + d.0) \mid (\overline{ac} . 0 + d.0) \mid \overline{d} .0) \setminus \{d\}$ $B = ab . \tau . 0$ $C = AC . \tau . 0$
	7. Junção Sincronizada: $P = A \mid B \mid C \setminus \{ab, ac\}$ A = $\tau . \overline{ac} . 0$ B = $\tau . \overline{bc} . 0$ C =



4. Ciclos Arbitrários:

$P = A \mid B \mid C \mid D \setminus \{ab, bc, cd, cb\}$

$A = \tau . \overline{ab} . 0$

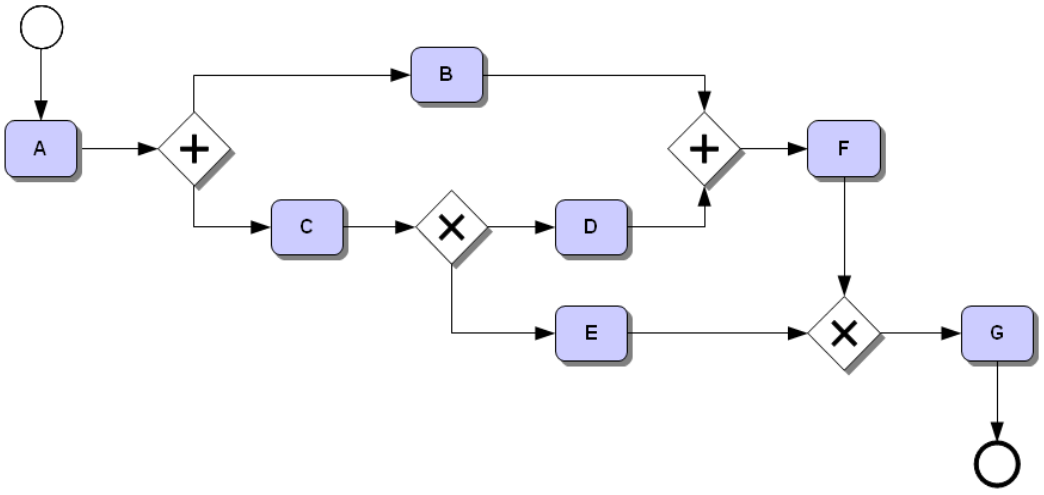
$B = ab . \tau . (\overline{bc} . 0 \mid B) + cb . \tau . (\overline{bc} . 0 \mid B)$

$C = bc . \tau . ((\overline{cb} . 0 + \overline{cd} . 0) \mid C)$

$D = cd . \tau . 0$

Figura 2 Classes de domínio da GExpLine

A seguir um exemplo de um pequeno workflow em notação BPMN com sua respectiva representação em CCS.



(a)

$A = \tau . (\overline{ab} . 0 \mid \overline{ac} . 0)$

$B = ab . \tau . \overline{bf} . 0$

$C = ac . \tau . (\overline{cd} . 0 + \overline{ce} . 0)$

$D = cd . \tau . \overline{df} . 0$

$E = ce . \tau . \overline{eg} . 0$

$F = bf . df . \tau . \overline{fg} . 0$

$G = eg . \tau . 0 + fg . \tau . 0$

(b)

Figura 3 Um workflow simples representado em BPMN (a) e em CCS (b)

2.4 – Verificação de modelos

Ferramentas de Verificação de Modelos **(BERARD ET AL., 2001)** permitem que possam ser detectados erros e outros problemas de especificação em sistemas computacionais de maneira automática. Como os SGWfC têm sido utilizados em cada vez mais aplicações, e podem ser usados para modelar experimentos que são de suma importância para uma determinada instituição (nesses casos qualquer tipo de falha não é aceitável), devemos ter confiança que os workflows funcionam corretamente A grande vantagem ao se usar uma álgebra de processos para representar workflows é justamente a possibilidade de verificação automática, além de ser uma representação genérica que pode ser usada como um padrão.

O processo de verificação de modelos se divide em duas etapas. A primeira, a etapa de modelagem, é aquela onde é gerado um modelo a partir do projeto original do sistema. Tal modelo deve obedecer a um formalismo, como álgebras de processo. A segunda etapa é a de especificação, no qual se determina quais propriedades serão verificadas no sistema.

2.4.1 – Lógicas Temporais

Dentre os formalismos mais utilizados para especificação e avaliação de propriedades, destacam-se as lógicas temporais. Na verificação de modelos, ao invés de especificarmos um "estado ruim" a ser procurado dentre os estados possíveis do sistema, uma fórmula lógica define um comportamento que o sistema deve ter quando executa. As lógicas utilizadas em verificações de modelos contemplam operadores temporais que permitem descrever a forma como um sistema se comporta com o passar do tempo e não simplesmente uma característica do sistema em um determinado momento do tempo. Utilizando lógicas temporais é possível verificar propriedades como: em algum momento futuro um dado evento irá ocorrer; sempre que um dado evento ocorre, um outro evento irá ocorrer em um momento futuro.

Lógicas temporias são utilizadas com sistemas reativos. Sistemas reativos são compostos de estados e transições. Um estado é uma descrição do sistema em um instante de tempo, isto é, os valores associados as suas variáveis naquele instante. Transições são relações entre estados que expressam as mudanças de estados. Uma

computação é uma sequência infinita de estados, onde cada estado é obtido a partir do anterior usando uma relação de transição. Sistemas reativos são representados como autômatos. Todas as possssíveis execuções do autômato geram uma árvore chamada Estrutura de Kripke. Computações são representadas como caminhos numa Estrutura de Kripke. Um estrutura de Kripke é composta de um conjunto de estados, uma relação de transições entre estados e uma função que rotula cada estado com o conjunto de propriedades que nele são verdadeiras.

O uso de Lógicas Temporais nesta dissertação se dá pela definição de propriedades que são desejáveis de serem atestadas, ou ainda pela definição de propriedades que desejamos nunca serem satisfeitas.

2.4.2 – CWB-NC

CWB-NC (The Concurrency Workbench of the New Century) **(CLEAVELAND ET AL., 2000)** é uma ferramenta de verificação de modelos que suporta várias formulações para tratar verificação. O tipo mais simples de verificação suportado pela ferramenta é a análise de alcançabilidade. Aqui, como em qualquer tipo de verificação, o primeiro passo para o uso da ferramenta é descrever o modelo do sistema projetado em uma das diversas linguagens suportadas pelo CWB-NC, como CCS, CSP e Basic Lotos. O modelo é então analisado pela ferramenta, que verifica sua correção sintática. O usuário fornece em seguida uma fórmula lógica que descreve uma propriedade que o sistema sempre deve satisfazer. Dada tal fórmula e o modelo do sistema, o CWB-NC explora cada possível estado que sistema pode alcançar durante a sua execução e verifica se em algum deles a propriedade não é satisfeita. Se um estado incorreto é detectado, uma descrição da sequência de execução que leva o sistema ao estado é comunicada ao usuário. Muitos erros, tais como *deadlocks* e violações de região crítica podem ser encontrados usando esta abordagem. A análise de alcançabilidade é na verdade um caso especial de verificação de modelos.

A sintaxe de CCS para o CWB-NC é semelhante à apresentada na Seção 2.3.1. Dentre diferenças existentes, podemos destacar:

- A existência do termo "proc", que identifica um agente da álgebra;

- A substituição da barra superior nos nomes complementares pelo caractere apóstrofo;

- A representação da ação silenciosa se dá com a letra *t* ao invés de τ;

- A representação do agente nulo se dá com a palabvra *nil* ao invés de 0;

Assim sendo, o workflow da Figura 3 seria espeficicado em CCS no CWB-NC da seguinte forma:

```
proc A = t . ('ab . nil | 'ac . nil )
proc B = ab . t . 'bf . nil
proc C = ac . t . ('cd . nil + 'ce . nil )
proc D = cd . t . 'df . nil
proc E = ce . t . 'eg . nil
proc F = bf . df . t . 'fg . nil
proc G = eg . t . nil + fg . t . nil
```

Dentre os commandos disponíveis no CWB-NC, podemos destacar aqueles que conferem mecanismos de verificação:

- *chk*: executa o verificador de modelos que determina se um agente satisfaz uma fórmula;

- *search*: executa a rotina de análise de alcançabilidade. Dados um agente e uma fórmula, a rotina busca um estado satisfazendo a fórmula no conjunto de estados alcançáveis a partir do agente. Se algum estado é encontrado, o caminho de execução do agente até chegar ao estado é mostrado.

- *eq*: verifica se dois agents dados possuem uma Relação de Equivalência (como Equivalência Forte ou Equivalência Fraca).

- *fd*: verifica se algum estado alcançável a partir de um dado agente tem um *deadlock*.

O CWB-NC é utlizado nesta dissertação como a ferramenta que efetivamente fará as verificações dos modelos de workflows gerados.

2.5 – Verificação de workflows

Uma maneira de fazer verificação de workflows é utilizando o processo de verificação de modelos. Assim sendo, temos que representar o workflow em um modelo matemático formal (como CCS), representar as propriedades que queremos atestar em um outro formalismo (como Lógias Temporais), e submeter estas duas representações à uma ferramenta de verificação de modelos. Este é exatamente o processo que é utilizado na abordagem desta dissertação.

2.6 – Trabalhos relacionados na área de verificação de worklows

Existem algumas abordagens que propõem verificar workflows científicos. No entanto, muitos deles são focados em fornecer mecanismos de verificação para sistemas específicos.

Zhao et al **(ZHAO ET AL., 2009)** propõem um método de modelagem baseado em CCS para descrever o comportamento das atividades de workflows. Elas adicionam restrições de dependência e parâmetros nas expressões das atividades. Eles também fornecem uma linguagem de especificação para estabelecer um modelo formal. Sua abordagem é avaliada em um estudo de caso utilizando um protótipo desenvolvido.

Liang e Zhao **(LIANG ET ZHAO, 2008)** propõem um método de verificação para workflows científicos que é baseado em lógica proposicional. Esta abordagem apresenta algumas vantagens como um formalismo lógico e sua habilidade em manipular modelos de processos baseados em atividades genéricos. Mostraram que este abordagem é capaz de detectar anomalias de processos em modelos de workflow.

Kim et al **(KIM ET AL., 2009)** propõem uma abordagem que usa descrições sobre componentes de workflow e suas restrições para validar a sua composição. Também propõem o uso de técnicas de planejamento de inteligência artificial, de modo a verificar sistematicamente propriedades de workflows. Esta abordagem analisa workflows e determina se estes são consistentes. Se a abordagem encontra erros, ela notifica os cientistas sobre problemas a serem resolvidos e sugere o que ele pode fazer para corrigí-los.

Aalst e Hofsted **(AALST ET HOFSTEDE, 2000)** e Aalst **(AALST, 2000)** propõem o uso efetivo de redes de Petri para a verificação de workflows. De acordo com os autores, a especificação de controle de processos pode se beneficiar das técnicas de análise baseadas em redes de Petri. Tais técnicas exploram a estrutura das redes de Petri para encontrar erros na especificação de workflows.

Capítulo 3 – Abordagem para Especificação Formal e Verificação Automática de *Workflows* Científicos

Apresentamos neste capítulo uma abordagem baseada nos padrões de workflow e em álgebras de processos que viabilize a especificação formal e a verificação automática de workflows. A abordagem permite a definição de workflows científicos com base nos padrões, inclui a capacidade de gerar automaticamente uma especificação em uma álgebra de processo (CCS), além fornecer integração com uma ferramenta de verificação de modelos (CWB-NC), onde se pode verificar e garantir propriedades desejadas na especificação gerada. Independentemente da linguagem em que o workflow esteja especificado, ele segue alguns padrões de workflows que podem ser identificados e utilizados para auxiliar a geração de uma especificação em álgebra de processos. Assim sendo, as expressões geradas em CCS devem estar de acordo com os padrões de workflow que as atividades seguem.

Especificações baseadas em padrões de workflow são ainda importantes no que tange a portabilidade, pois é possível portar um workflow para qualquer SGWf existente que seja capaz de representar os padrões utilizados. Especificações formais são importantes na medida em que não há ambiguidades no comportamento das atividades. A verificação automática se torna uma ferramenta interessante, já que vários erros de especificação podem ser detectados em um momento anterior à execução do workflow.

Dentre os problemas de especificação que podem ocorrer, estamos especialmente interessados em detectar deadlocks, falhas de sincronização e execuções por tempo indeterminado. Também é possível verificar se algumas outras propriedades desejadas são válidas: cada atividade é executada pelo menos uma vez; sempre que uma atividade A é executada, houve anteriormente a execução de outra atividade B. Após o armazenamento de um workflow na GExpLine, podemos gerar a especificação formal

em CCS. Assim, podemos submeter esta especificação à ferramenta de verificação de modelos CWB-NC, que é responsável por responder as questões de verificação.

3.1 – Modelo formal de workflow e representação em CCS

Adaptando Puhlmann e Weske **(PUHLMANN, 2007, PUHLMANN ET WESKE, 2006a, 2006b)**, podemos definir a estrutura de um workflow como uma 4-tupla composta de Atividades, Conexões, Tipos de Junção e Tipos de Divisão. Formalmente, temos Wf = (A, C, TJ, TD), onde:

- A é um conjunto finito e não vazio de atividades;
- $C \subseteq (A \times A)$ é um conjunto de conexões direcionadas entre atividades;
- TJ: A → TIPOJUNCAO é uma função de atividades para seus tipos de junção;
- TD: A → TIPODIVISAO é uma função de atividades para seus tipos de divisão;

Os tipos de junção e ou tipos de divisão são representados por um subconjunto dos padrões de workflows. Os tipos de junção são: AND (padrão de workflow Sincronização), XOR (padrão de workflow Junção Simples) e OR (padrão de workflow Junção Sincronizada). Os tipos de divisão são: AND (padrão de workflow Divisão Paralela), XOR (padrão de workflow Escolha Exclusiva) e OR (padrão de workflow Escolha Múltipla).

Adaptando Pulhmann **(PUHLMANN, 2007, PUHLMANN ET WESKE, 2006a, 2006b)**, a transformação de uma definição formal de um workflow para a sua representação em CCS se dá como segue:

- Cada atividade do workflow é mapeada em um único agente;
- O processo completo do workflow é também mapeado em um agente, que é a composição de todos os agentes que mapeiam as atividades;

- Cada conexão do workflow é mapeada em um único nome em CCS.

Algumas alterações foram feitas em relação ao modelo de Puhlmann **(PUHLMANN, 2007, PUHLMANN ET WESKE, 2006a, 2006b)**.

Como exemplo, temos um pequeno workflow construído no Kepler que utiliza alguns dos padõres de workflows mais comuns. Na figura Y abaixo, temos a representação visual do workflow no Kepler.

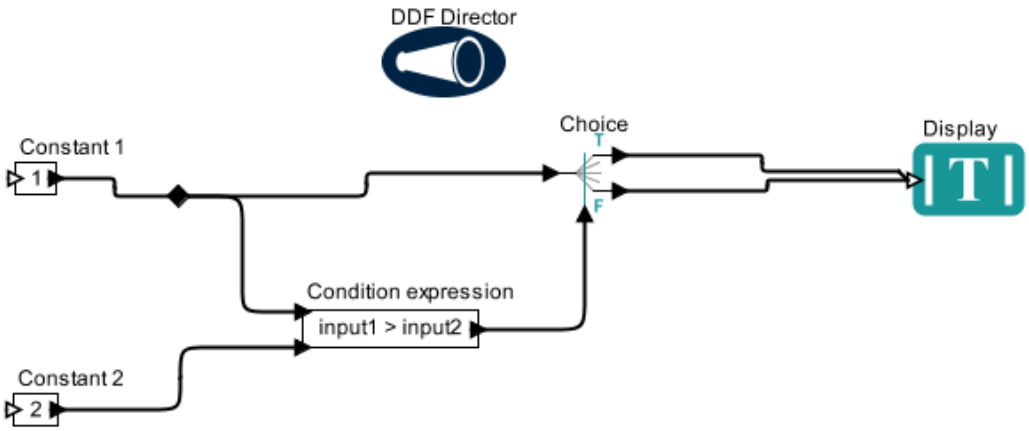


Figura 4 Um workflow simples no SGWf C Kepler

Este workflow contempla os seguintes padrões de workflows: Sequência, Divisão Paralela, Sincronização e Escolha Exclusiva.

Por questão de simplicidade, vamos abreviar os nomes das atividades. Assim sendo, temos as seguintes trocas de nomes: Constant 1 por C1, Constant 2 por C2, Condition expression por Ce, Choice por C e Display por D.

Levando em conta os nomes abreviados, o modelo formal deste workflow é como segue:

Wf = (A, C, TJ, TD)

A = {C1, C2, Ce, C, D}

C = { (C1, C), (C1, Ce), (C2, Ce), (Ce, C), (C, D), (C, D) }

TJ = { (C1, AND), (C2, AND), (Ce, AND), (C, AND), (D, AND) }

TD = { (C1, AND), (C2, AND), (Ce, AND), (C, XOR), (D, AND) }

A partir deste workflow é gerada a seguinte especificação em CCS, já em conformidade com o sistema CWB-NC:

```
proc C = choicecontrol.choiceinput.t.( 'displayinput.nil + 'displayinput.nil )

proc D = displayinput . displayinput.t.nil

proc C1 = t.( 'choiceinput.nil | 'conditionexpressioninput1.nil )

proc C2 = t.( 'conditionexpressioninput2.nil )

proc Ce = conditionexpressioninput1.conditionexpressioninput2.t.(
'choicecontrol.nil )

proc Wf = (C | D | C1 | C2 | Ce) \
{displayinput,choiceinput,conditionexpressioninput1,conditionexpressioninput2,choicec
ontrol}
```

3.2 – Implementação da abordagem

A ferramenta utilizada como base deste trabalho é a GExpLine. Assim sendo, com a incorporação de nossa abordagem, a GExpLine permite:

- Especificação baseada nos padrões de workflow: as diversas categorias de interação entre as atividades do workflow (ou o seu comportamento) passam a ser representadas de forma padronizada e sem ambiguidades, o que contribui para melhorar as possibilidades de portabilidade de um dado workflow especificado na ferramenta;

- Geração de especificação de workflows em álgebras de processos: as especificações têm uma representação equivalente em uma álgebra de processos. A implementação permite a geração de especificação em CCS;

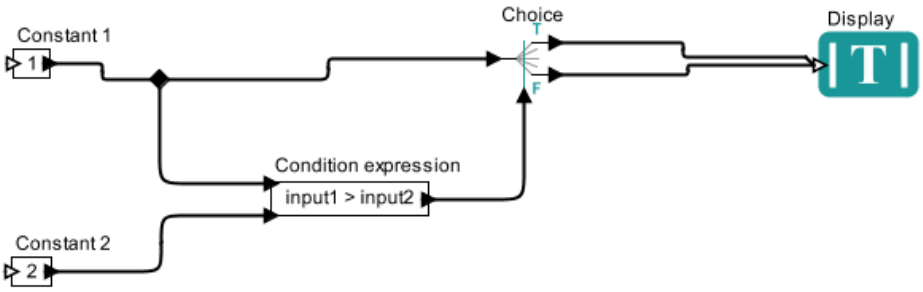
- Interação com uma ferramenta de verificação de modelos: habilita a verificação automática de algumas propriedades utilizando a especificação gerada em CCS. A ferramenta de verificação integrada foi o CWB-NC.

Também poderíamos portar uma especificação já descrita em uma linguagem de um SGWf para álgebra de processos e verificar se ela se comporta corretamente, e se necessário, modificá-la e reconstituí-la na linguagem original.

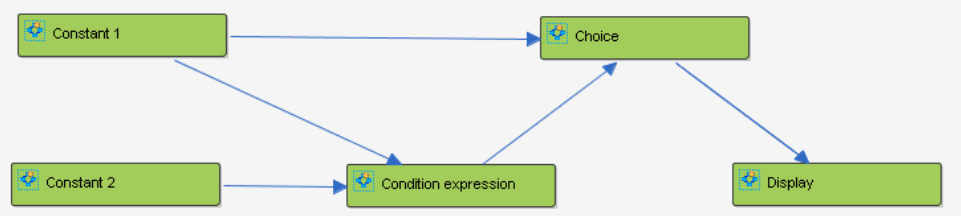
Neste capítulo será descrita a avaliação da abordagem da dissertação utilizando um workflow como alvo de verificações.

4.1 – Verificação para um workflow simples

O workflow nesta verificação é aquele descrito como exemplo na seção 3.1. Sua representação gráfica no SGWfC Kepler pode ser vista na Figura 4 e na Figura 5 (a), e sua representação correspondente na GExpLine pode ser vista na Figura 5 (b).



(a)



(b)

Figura 5 Um workflow simples no Kepler (a) e na GExpLine (b)

A representação do workflow na ferramenta GExpLine é um pouco diferente da representação no Kepler. Por exemplo, não existe uma representação correspondente na GExpLine para o conceito de Relação existente no Kepler (que neste workflow é o

losango que permite a conexão entre as atividades Constant 1, Condition Expression e Choice). Uma outra diferença notável é que na GExpLine parece só haver uma conexão entre as atividades Choice e Display, ao contrário da representação no Kepler, que tem duas conexões. Na verdade, as duas conexões também existem na GExpLine, mas a representação visual na ferramenta desenhou as duas conexões de maneira sobreposta, dando a falsa impressão de que só há uma conexão.

Ao executar o workflow do SGWfC Kepler, notamos que sua execução não foi bem sucedida. A atividade Display não foi executada. No Kepler é possível rastrear a execução do workflow pela opção do menu "Listen To Diretor". Ao abrirmos esta janela e executando o workflow mais uma vez, temos a confirmação de que a atividade Display não foi executada, e mais, mostra que ocorreu um *deadlock*. Verificamos a capacidade de nossa abordagem de detectar o *deadlock* produzido na execução do workflow. A Figura 6 mostra a interface da GExpLine para verificação de workflows.

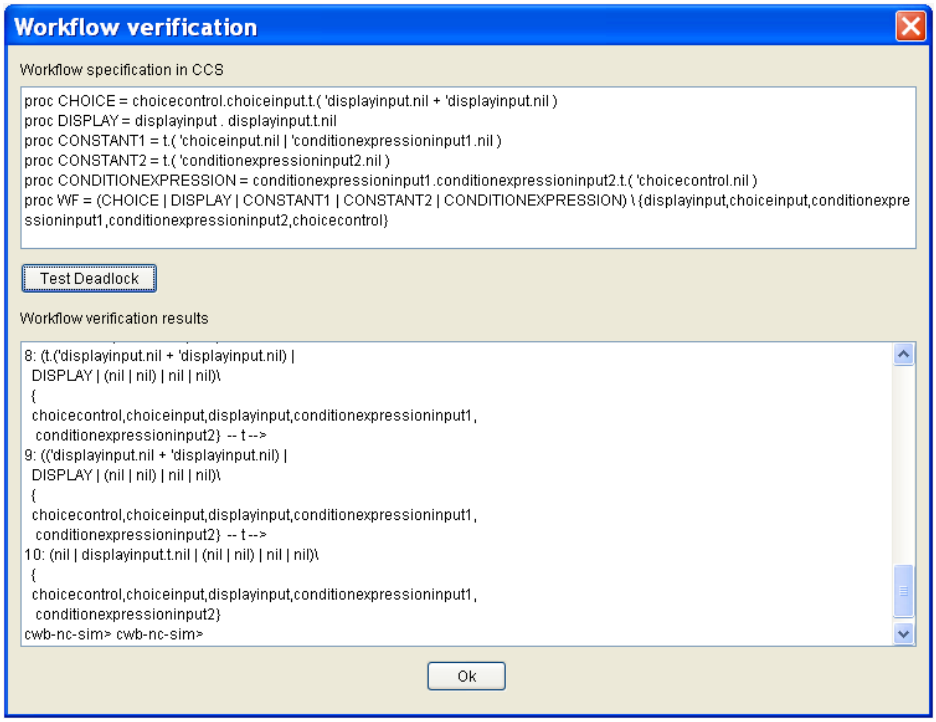


Figura 6 Interface de verificação da GExpLine

Na parte superior da Figura 6 temos a especificação do workflow em CCS. A Figura 6 também nos traz, em sua parte inferior, o resultado da verificação na ferramenta CWB-NC quando executado para buscar *deadlocks*. A verificação consiste em buscar, dentre todas as execuções possíveis da especificação, alguma execução que contenha um estado onde há um *deadlock*. O resultado completo gerado pelo CWB-NC não é apresentado na Figura 6 por questões de espaço; por isso, o resultado completo é mostrado a seguir:

cwb-nc>

States explored: 10

Deadlock found, invoking simulator.

1: WF -- t -->

2: (CHOICE | DISPLAY | CONSTANT1 | 'conditionexpressioninput2.nil | CONDITIONEXPRESSION)\ { choicecontrol,choiceinput,displayinput,conditionexpressioninput1, conditionexpressioninput2} -- t -->

3: (CHOICE | DISPLAY | ('choiceinput.nil | 'conditionexpressioninput1.nil) | 'conditionexpressioninput2.nil | CONDITIONEXPRESSION)\ { choicecontrol,choiceinput,displayinput,conditionexpressioninput1,conditionexpressioninput2} -- t -->

4: (CHOICE | DISPLAY | ('choiceinput.nil | nil) | 'conditionexpressioninput2.nil | conditionexpressioninput2.t.'choicecontrol.nil)\ { choicecontrol,choiceinput,displayinput,conditionexpressioninput1, conditionexpressioninput2} -- t -->

5: (CHOICE | DISPLAY | ('choiceinput.nil | nil) | nil | t.'choicecontrol.nil)\ {
choicecontrol,choiceinput,displayinput,conditionexpressioninput1,
conditionexpressioninput2} -- t -->

6: (CHOICE | DISPLAY | ('choiceinput.nil | nil) | nil | 'choicecontrol.nil)\ {
choicecontrol,choiceinput,displayinput,conditionexpressioninput1,
conditionexpressioninput2} -- t -->

7: (choiceinput.t.('displayinput.nil + 'displayinput.nil) | DISPLAY |
('choiceinput.nil | nil) | nil | nil)\ {
choicecontrol,choiceinput,displayinput,conditionexpressioninput1,
conditionexpressioninput2} -- t -->

8: (t.('displayinput.nil + 'displayinput.nil) | DISPLAY | (nil | nil) | nil | nil)\ {
choicecontrol,choiceinput,displayinput,conditionexpressioninput1,
conditionexpressioninput2} -- t -->

9: (('displayinput.nil + 'displayinput.nil) | DISPLAY | (nil | nil) | nil | nil)\ {
choicecontrol,choiceinput,displayinput,conditionexpressioninput1,
conditionexpressioninput2} -- t -->

10: (nil | displayinput.t.nil | (nil | nil) | nil | nil)\ {
choicecontrol,choiceinput,displayinput,conditionexpressioninput1,
conditionexpressioninput2}

cwb-nc-sim> cwb-nc-sim>

O resultado diz que a procura por *deadlocks* explorou 10 dos estados possíveis do workflow, e isto foi suficiente para detectar o deadlock. Observando novamente a Figura 3, é possível ver o fim da verificação, que mostra o último estado do workflow, na qual a atividade Display está à espera de algum dado em sua porta de entrada, e não existem outras atividades habilitadas para fornecer este dado para a atividade Display. Na verdade, a única atividade que não está finalizada neste estado é a própria Display.

Assim, o workflow não tem opções para continuar a execução, o *deadlock* está confirmado.

Mais detalhes sobre a execução desse experimento podem ser acompanhados com o vídeo:

http://gexp.nacad.ufrj.br/documents/gexp-videos/verification-mechanism.avi/view

Capítulo 5 – Conclusões

Esta dissertação propõe a verificação de workflows científicos baseada em álgebras de processo. Utilizando a abordagem proposta é possível detectar problemas de especificação de workflows antes de sua execução real em Sistemas Gerenciadores de Workflows Científicos. Durante o processo de especificação de workflows, a abordagem traz as vantagens da utilização de padrões de workflow, CCS e de ferramentas de verificação de modelos. O processo de verificação é capaz de detectar problemas de especificação em workflows científicos, o que pode evitar uma execução real com erros.

A maioria dos mecanismos para verificação de workflows existentes foram construídas para tratar workflows para processos de negócio; a maioria delas utilizando-se de redes de Petri ou álgebras de processo, além de padrões de workflow como formalismo complementar. Esta dissertação trata justamente da utlização padrões de workflow e álgebras de processo junto à verificação de workflows científicos, já que na área científica este tipo de abordaem ainda é insipiente.

Para avaliar a proposta desta dissertação, foi implementada uma prova de conceito como um componente junto à ferramenta de especificação de workflows GExpLine, acoplando assim um mecanismo de verificação dos workflows concretos que são gerados através do processo de derivação. Assim, é possível detectar problemas de especificação, mesmo em workflows simples, para todos estes SGWfC sem que os mecanismos de verificação tenham que ser implementados e acoplados a cada um deles. O experimento realizado com um workflow definido para o SGWfC Kepler e sua verificação com o componente resultante dessa dissertação mostrou a detecção de um deadlock e a compatibilidade das técnicas de verificação desenvolvidas junto à GExpLine.

A principal vantagem do uso de workflows armazenados na ferramenta GExpLine é que podemos aplicar essas técnicas independentemente de SGWfC, pois cada um deles apresenta uma linguagem de especificação de workflow diferente. Todos os mecanismos de verificação são incluídos em um único sistema. A ferramenta GExpLine em sua essência já é capaz de derivar, a partir de uma representação abstrata, workflows concretos para diversos SGWfC como o Taverna, Kepler e VisTrails.

AALST, W. M. P. V. D.; HOFSTEDE, A. H. M. T.; KIEPUSZEWSKI, B.; ET AL., 2003, "Workflow Patterns", *Distrib. Parallel Databases*, v. 14, n. 1, pp. 5-51.

AALST, W. M. P. V. D., 1997, "Verification of Workflow Nets". In: *Proceedings of the 18th International Conference on Application and Theory of Petri Nets*, pp. 407-426

AALST, W. M. P. V. D., 2000, "Workflow Verification: Finding Control-Flow Errors Using Petri-Net-Based Techniques". In: *Business Process Management, Models, Techniques, and Empirical Studies*, pp. 161-183

AALST, W. M. V. D.; HOFSTEDE, A. H. M. T., 2000, "Verification of workflow task structures: A petri-net-based approach", *Inf. Syst.*, v. 25, n. 1, pp. 43-69.

ALTINTAS, I.; BERKLEY, C.; JAEGER, E.; ET AL., 2004, "Kepler: an extensible system for design and execution of scientific workflows". In: *16th SSDBM*, pp. 423-424, Santorini, Greece.

BARGA, R.; GANNON, D., 2007, "Scientific versus Business Workflows", *Workflows for e-Science*, Springer, pp. 9-16.

BERARD, B.; BIDOIT, M.; FINKEL, A.; ET AL., 2001, *Systems and Software Verification: Model-Checking Techniques and Tools*. Springer.

BERGSTRA, J.; KLOP, J.ERRO. Process algebra for synchronous communication.

BOLOGNESI, T.; BRINKSMA, E., 1987, "Introduction to the ISO specification language LOTOS", *Comput. Netw. ISDN Syst.*, v. 14, n. 1, pp. 25-59.

BOWERS, S.; LUDASCHER, B.; NGU, A. H. H.; ET AL., 2006, "Enabling ScientificWorkflow Reuse through Structured Composition of Dataflow and Control-Flow". In: *Proceedings of the 22nd International Conference on Data Engineering Workshops*, p. 70

BRAGHETTO, K. R.; FERREIRA, J. E.; PU, C., 2007, "Using control-flow patterns for specifying business processes in cooperative environments". In: *Proceedings of the 2007 ACM symposium on Applied computing*, pp. 1234-1241, Seoul, Korea.

CALLAHAN, S. P.; FREIRE, J.; SANTOS, E.; ET AL., 2006, "VisTrails: visualization meets data management". In: *Proceedings of the 2006 ACM SIGMOD*, pp. 745-747, Chicago, IL, USA.

CHEN, J.; YANG, Y., 2008, "A taxonomy of grid workflow verification and validation", *Concurr. Comput. : Pract. Exper.*, v. 20, n. 4, pp. 347-360.

CLEAVELAND, R.; LI, T.; SIMS, S., 2000, "The concurrency workbench of the new century", *User's manual, SUNY at Stony Brook, Stony Brooke, NY, USA*

DEELMAN, E.; GANNON, D.; SHIELDS, M.; ET AL., 2009, "Workflows and e-Science: An overview of workflow system features and capabilities", *Future Generation Computer Systems*, v. 25, n. 5, pp. 528-540.

HOARE, C. A. R., 1978, "Communicating sequential processes", *Commun. ACM*, v. 21, n. 8, pp. 666-677.

HULL, D.; WOLSTENCROFT, K.; STEVENS, R.; ET AL., 2006, "Taverna: a tool for building and running workflows of services", *Nucleic Acids Research*, v. 34, n. Web Server issue, pp. 729-732.

KIM, J.; GIL, Y.; SPRARAGEN, M., 2009, "Principles for interactive acquisition and validation of workflows", *Journal of Experimental & Theoretical Artificial Intelligence*

LIANG, Q.; ZHAO, J., 2008, "Verification of Unstructured Workflows via Propositional Logic". In: *Computer and Information Science, 2008. ICIS 08. Seventh IEEE/ACIS International Conference onComputer and Information Science, 2008. ICIS 08. Seventh IEEE/ACIS International Conference on*, pp. 247-252

LUDÄSCHER, B., 2009, "What Makes Scientific Workflows Scientific?". In: *Proceedings of the 21st International Conference on Scientific and Statistical Database Management*, pp. 217-217, New Orleans, LA, USA.

LUDÄSCHER, B.; ALTINTAS, I.; BERKLEY, C.; ET AL., 2006, "Scientific workflow management and the Kepler system: Research Articles", *Concurrency and Computation: Practice and Experience*, v. 18, n. 10, pp. 1039-1065.

MATTOSO, M.; WERNER, C.; TRAVASSOS, G. H.; ET AL., 2010, "Towards Supporting the Life Cycle of Large Scale Scientific Experiments ", *To be published in Int. J. Business Process Integration and Management*, n. Special Issue on Scientific Workflows

MILNER, R., 1989, *Communication and concurrency*. Prentice-Hall, Inc.

MILNER, R., 1999, *Communicating and Mobile Systems: the Pi-Calculus*. Cambridge University Press.

MILNER, R.; PARROW, J.; WALKER, D., 1992a, "A calculus of mobile processes, I", *Inf. Comput.*, v. 100, n. 1, pp. 1-40.

MILNER, R.; PARROW, J.; WALKER, D., 1992b, "A calculus of mobile processes, II", *Inf. Comput.*, v. 100, n. 1, pp. 41-77.

OGASAWARA, E.; PAULINO, C.; MURTA, L.; ET AL., 2009, "Experiment Line: Software Reuse in Scientific Workflows". In: *21th SSDBM*, pp. 264–272, New Orleans, LA.

OINN, T.; ADDIS, M.; FERRIS, J.; ET AL., 2004, *Taverna: a tool for the composition and enactment of bioinformatics workflows*. Oxford Univ Press.

PETRI, C., 1962, *Kommunikation mit Automaten*, Institut für instrumentelle Mathematik

PUHLMANN, F., 2007, "Soundness Verification of Business Processes Specified in the Pi-Calculus"

PUHLMANN, F.; WESKE, M., 2005, "Using the π-Calculus for Formalizing Workflow Patterns", *Business Process Management*, , pp. 153-168.

PUHLMANN, F.; WESKE, M., 2006a, "Investigations on Soundness Regarding Lazy Activities", *Business Process Management*, , pp. 145-160.

PUHLMANN, F.; WESKE, M., 2006b, "Interaction Soundness for Service Orchestrations", *SERVICE-ORIENTED COMPUTING – ICSOC 2006, VOLUME 4294 OF LNCS*, pp. 302-313.

RUSSELL, N.; HOFSTEDE, A. H. M. T.; MULYAR, N., 2006, "Workflow ControlFlow Patterns: A Revised View"

SANGIORGI, D.; WALKER, D., 2001, *PI-Calculus: A Theory of Mobile Processes*. Cambridge University Press.

SHOSHANI, A., 2009, "The Scientific Data Management Center: Providing Technologies for Large Scale Scientific Exploration", *Scientific and Statistical Database Management*, , pp. 1-2.

SLOMINSKI, A., 2007, "Adapting BPEL to Scientific Workflows", *Workflows for e-Science*, Springer, pp. 208-226.

STEFANSEN, C., 2005, "SMAWL: A small workflow language based on CCS", *HARVARD UNIVERSITY*

WFMC, 1999, *Workflow Management Coalition Terminology & Glossary (WFMC-TC-1011)*. Workflow Management Coalition Specification.

WFMC, 2002, *Workflow Management Coalition Workflow Standard: Workflow Process Definition Interface – XML Process Definition Language (XPDL) (WFMC-TC-1025)*. Workflow Management Coalition Specification.

WOODMAN, S.; PARASTATIDIS, S.; WEBBER, J., 2007, "Protocol-Based Integration Using SSDL and π-Calculus", *Workflows for e-Science*, Springer, pp. 227-243.

ZHAO, L.; LI, Q.; LIU, X.; ET AL., 2009, "A modeling method based on CCS for workflow". In: *Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication*, pp. 376-384, Suwon, Korea.