Especificação Formal e Verificação de *Workflows* Científicos

Edno V. Silva[1], Eduardo Ogasawara[1], Daniel de Oliveira[1], Marta Mattoso[1], Mario R. F. Benevides[2]

[1]PESC - COPPE – Universidade Federal do Rio de Janeiro (UFRJ)
Caixa Postal 68511 – 21.941-972 – Rio de Janeiro – RJ – Brasil

[2]DCC - Instituto de Matemática – Universidade Federal do Rio de Janeiro (UFRJ)
Caixa Postal 68511 – 21.941-972 – Rio de Janeiro – RJ – Brasil

{edno, ogasawara, danielc, marta, mario}@cos.ufrj.br

Abstract. *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 are critical and must be completely reliable, that is why they must be correctly specified. This paper proposes an approach that supports workflow verification based on process algebra specifications (CCS) and model checking tools. We have evaluated our approach using the GExpLine workflow tool.*

Resumo. *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 assume que um workflow está correto se respeita os controles e as dependências definidos pelo cientista. Além disso, muitos workflows científicos são críticos e 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. Avaliamos a abordagem utilizando a ferramenta de workflows GExpLine.*

1. Introdução

Um *workflow* científico pode ser definido como a especificação formal de um processo científico, que representa os passos a serem executados em um experimento científico [Deelman et al. 2009]. Estes passos são normalmente associados à seleção de dados, análise e visualização. *Workflows* científicos são gerenciados por sistemas complexos chamados Sistemas Gerenciadores de *Workflows* Científicos (SGWfC) que apóiam a especificação de *workflows* em termos de artefatos executáveis.

Existem várias iniciativas para padronizar a especificação de linguagens de *workflow* [Barga e 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.

Muitos experimentos são complexos e suas especificações podem conter problemas. A execução de *workflows* incorretos desperdiça recursos computacionais valiosos. A construção de *workflows* corretos é uma tarefa fundamental para o processo de experimentação. Ao garantir a correção dos *workflows* científicos, é possível poupar quantidade significativa de trabalho dos cientistas. 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. Algumas abordagens utilizam 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. As abordagens existentes 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. 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 [Ogasawara et al. 2009]. 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 SGWfCs, como o Kepler [Altintas et al. 2004], são conhecidas como *workflows* concretos [Deelman et al. 2009].

Esta trabalho 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 fluxo de trabalho abstrato em um executável. Avaliamos a nossa abordagem utilizando GExpLine como a ferramenta de modelagem de *workflow* [Ogasawara et al. 2009].

Este artigo está organizado da seguinte forma. A seção 2 apresenta padrões de *workflow*, álgebras de processos e o processo de verificação de *workflows*. A seção 3 contempla a abordagem proposta. A seção 4 apresenta a aplicação da abordagem como uma prova de conceito. A seção 5 traz trabalhos relacionados, e finalmente, a seção 6 contém as conclusões.

2. Verificação de *workflows*

Esta seção apresenta alguns conceitos fundamentais relacionados ao processo de verificação de *workflows*. Uma forma comum de validar *workflows* é identificar padrões para subsequente transformação em uma especificação em álgebra de processo, e em seguida submetendo a especificação gerada a uma ferramenta de verificação de modelos. Padrões de *workflow* [Aalst et al. 2003, Russell et al. 2006], são padrões de construção encontrados com mais freqüência na área de *workflows*. Sua proposta é formalizar o comportamento de *workflows* de maneira precisa e sem ambiguidades. 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.

2.2 Álgebra CCS

O 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 como a, b, c,..., que denotam ações em portas de entrada. $\overline{A}$ representa nomes complementares 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.

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, definimos informalmente a definição dos agentes de CCS: (i) 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); (ii) P + Q (Soma): o agente se comporta como o agente P ou como outro agente Q, de maneira exclusiva; (iii) P | Q (Composição): um agente que é a execução concorrente dos agentes P e Q; (iv) P \ L (Restrição): o conjunto de ações L $\in A \cup \overline{A}$ tem sua execução restrita ao escopo do agente P; (v) P[S] (Substituição): os nomes das ações do agente P são trocados simultaneamente por meio da função de substituição S; (vi) 0 (Agente nulo): um agente especial incapaz de realizar qualquer ação.

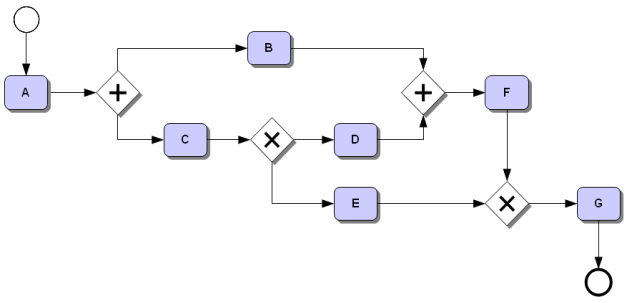
A semântica de CCS é definida por um sistema de transições do tipo $A \xrightarrow{a} A'$. Esta notação quer dizer que o agente *A* executa a ação *a* e torna-se o agente *A*'.

CCS possui uma extensão de nome Pi-cálculo [Milner 1989]. Nesta álgebra os processos (chamados agentes em CCS) têm estruturas mutáveis. Pi-cálculo inclui mobilidade enquanto preserva as propriedades algébricas de CCS. Também tem sido utilizada como formalismo para verificação de *workflows*, embora não tenha sido demonstrado que suas características de mobilidade possam ser úteis na área de *workflows*. Por isso, muitos trabalhos limitam-se a utilizar CCS como formalismo.

2.3 Representação de *workflows* em CCS e Verificação de Modelos

Uma abordagem possível para a representação de atividades de *workflows* em álgebras de processo é aquela onde cada atividade é mapeada conceitualmente em um processo da álgebra [Puhlmann e 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*. Por

exemplo, a apresenta um *workflow* representado em BPMN [BPMN 2009] com tradução para CCS, adaptado de [Puhlmann e Weske 2005].



$$A = \tau.(\overline{ab}.0 \mid \overline{ac}.0),$$
$$B = ab.\tau.\overline{bf}.0,$$
$$C = ac.\tau.(\overline{cd}.0 + \overline{ce}.0),$$
$$D = cd.\tau.\overline{df}.0,$$
$$E = ce.\tau.\overline{eg}.0,$$
$$F = bf.df.\tau.\overline{fg}.0,$$
$$G = eg.\tau.0 + fg.\tau.0.$$

Figura 1 – Um *workflow* simples em BPMN traduzido para CCS

Os padrões de *workflow* na Figura 1 são: (i) divisão paralela (entre as atividades A, B e C); (ii) escolha exclusiva (entre C, D e E); (iii) sincronização (entre B, D e F) e; (iv) junção simples (entre E, F e G).

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 muitas vezes controlando processos críticos (onde falhas não são aceitáveis), 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. Uma das ferramentas que podem ser utilizadas é o CWB-NC (The Concurrency Workbench of the New Century) [Cleaveland et al. 2000]**.**

3. Abordagem para especificação formal e verificação de *workflows*

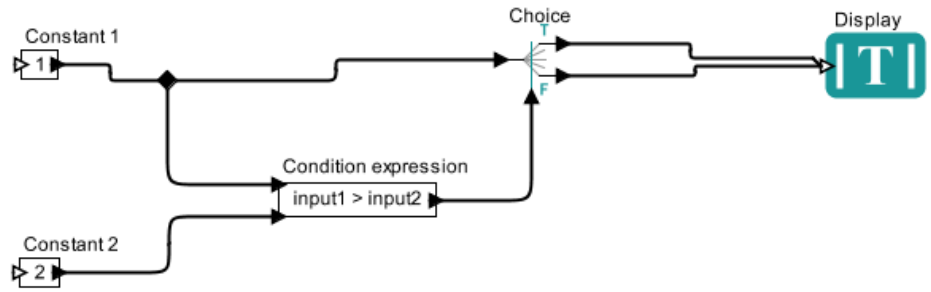
Apresentamos agora uma abordagem baseada nos padrões de *workflow* e em álgebras de processos que viabilize a especificação formal e a verificação automática de *workflows*. A abordagem permite a definição de *workflows* científicos com base nos padrões, inclui a capacidade de gerar automaticamente uma 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. Especificações baseadas em padrões de *workflow* são 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*.

Nossa abordagem é feita incluindo mecanismos de verificação na ferramenta GExpLine [Ogasawara et al. 2009].

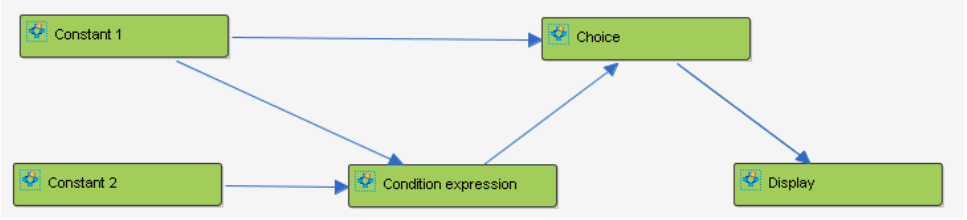
Assim sendo, com a incorporação de nossa abordagem, a GExpLine permite: (i) Especificação baseada nos padrões de *workflow*: as diversas categorias de interação entre as atividades do *workflow* (ou o seu comportamento) passam a ser representadas de forma padronizada e sem ambigüidades; (ii) Geração de especificação de *workflows* em álgebra de processos: as especificações têm uma representação equivalente em uma

álgebra de processos. A abordagem permite a geração de especificação em CCS; (iii) Interação com ferramentas de verificação de modelos: habilita a verificação automática de algumas propriedades utilizando a especificação gerada em CCS.

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. Avaliamos a nossa abordagem usando um *workflow* muito simples que mostra os seguintes padrões de *workflows*: sequência, divisão paralela, sincronização e escolha exclusiva. A Figura 2 mostra esse *workflow* no Kepler (2a) e na ferramenta GExpLine (2b).



(a)



(b)

Figura 2 Um *workflow* de exemplo no Kepler (a) e na GexpLine (b)

Ao executar o *workflow* de 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 3 mostra a interface da GExpLine para verificação de *workflows*.

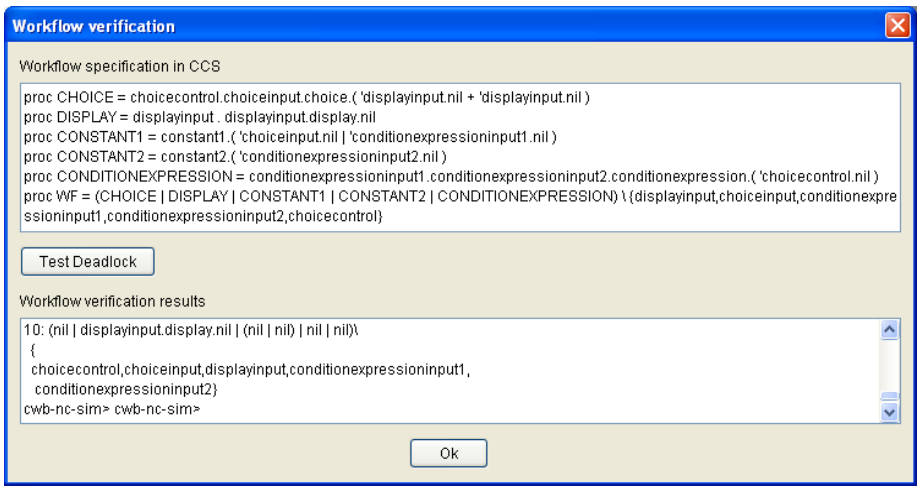


Figura 3 Interface de verificação da GexpLine

A Figura 3 mostra a especificação do *workflow* em CCS. A interface também mostra o resultado da verificação na ferramenta CWB-NC quando executado para buscar *deadlocks*. A sintaxe de CCS para o CWB-NC é quase igual à apresentada na seção 2.2; a principal diferença é a existência do termo "proc", que identifica um agente CCS. O início do resultado gerado pelo CWB-NC não é apresentado na Figura 3 por questões de espaço; por isso, é mostrado a seguir:

```
cwb-nc>

States explored: 10

Deadlock found, invoking simulator.
```

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*, onde a atividade *Display* está a espera de algum dado em sua porta de entrada, e não há 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. Maiores detalhes sobre a abordagem de verificação proposta podem ser encontrados na página http://gexp.nacad.ufrj.br/news/new-verification-mechanism-in-gexpline, onde um vídeo apresenta a utilização do mecanismo de verificação na ferramenta GExpLine.

4. Trabalhos relacionados

Existem algumas abordagens que propõem especificar formalmente e verificar *workflows*. No entanto, muitos deles são focados em fornecer mecanismos de verificação para sistemas específicos. 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. Zhao et al [Zhao et al. 2009] propõem um método de modelagem baseado em CCS para descrever o comportamento das atividades de *workflows*.

Fornecem uma linguagem de especificação para estabelecer um modelo formal. Liang e Zhao [Liang e Zhao 2008] propõem um método de verificação para *workflows* científicos que é baseado em lógica proposicional. 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 inteligência artificial para verificar sistematicamente propriedades de *workflows*. Esta abordagem analisa *workflows* e determina se estes são consistentes. Aalst e Hofsted [Aalst e Hofstede 2000] e Aalst [Aalst 2000] propõem o uso de redes de Petri e suas técnicas para a verificação de *workflows*.. As técnicas exploram a estrutura das redes de Petri para encontrar erros na especificação de *workflows*.

5. Conclusões e Observações Finais

Este trabalho propõe a verificação de *workflows* científicos baseada em álgebras de processo. Utilizando nossa abordagem é possível detectar problemas de especificação de *workflows* antes de sua execução real em SGWfCs. 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 evitar a execução de *workflows* com problemas de especificação, o que poderia gerar cálculos desnecessários e perda de tempo. 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. Para avaliar nossa proposta, implementamos uma prova de conceito para a ferramenta GExpLine, que pode derivar especificações executáveis para os SGWfC Kepler, Taverna e VisTrails. Assim, somos capazes de detectar falhas, mesmo em *workflows* simples, para todos estes SGWfC.

6. Referências

Aalst, W. M. P. V. D., Hofstede, A. H. M. T., Kiepuszewski, B., Barros, A. P., 2003, "Workflow Patterns", *Distrib. Parallel Databases*, v. 14, n. 1, p. 5-51.

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*, p. 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, p. 43-69.

Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludascher, B., Mock, S., 2004, "Kepler: an extensible system for design and execution of scientific workflows". In: *16th SSDBM*, p. 423-424, Santorini, Greece.

Barga, R., Gannon, D., 2007, "Scientific versus Business Workflows", *Workflows for e-Science*, Springer, p. 9-16.

Berard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L., Schnoebelen, P., 2001, *Systems and Software Verification: Model-Checking Techniques and Tools*. 1 ed. Springer.

BPMN, 2009, *Business Process Modeling Notation (BPMN), Version 1.0*, http://www.petrobras.com.br.

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*, p. 1234-1241, Seoul, Korea.

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., Taylor, I., 2009, "Workflows and e-Science: An overview of workflow system features and capabilities", *Future Generation Computer Systems*, v. 25, n. 5, p. 528-540.

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*, p. 247-252

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

Milner, R., Parrow, J., Walker, D., 1992a, "A calculus of mobile processes, I", *Inf. Comput.*, v. 100, n. 1, p. 1-40.

Milner, R., Parrow, J., Walker, D., 1992b, "A calculus of mobile processes, II", *Inf. Comput.*, v. 100, n. 1, p. 41-77.

Ogasawara, E., Paulino, C., Murta, L., Werner, C., Mattoso, M., 2009, "Experiment Line: Software Reuse in Scientific Workflows". In: *21th SSDBM*, p. 264–272, New Orleans, LA.

Puhlmann, F., Weske, M., 2005, "Using the π-Calculus for Formalizing Workflow Patterns", *Business Process Management*, , p. 153-168.

Russell, N., Hofstede, A. H. M. T., Mulyar, N., 2006, "Workflow ControlFlow Patterns: A Revised View"

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

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

Woodman, S., Parastatidis, S., Webber, J., 2007, "Protocol-Based Integration Using SSDL and π-Calculus", *Workflows for e-Science*, Springer, p. 227-243.

Zhao, L., Li, Q., Liu, X., Du, N., 2009, "A modeling method based on CCS for workflow". In: *Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication*, p. 376-384, Suwon, Korea.