Distribuição de Tarefas em Sistemas de Workflow com Base na Aptidão dos Recursos

Renê Rodrigues Veloso

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Ciência da Computação da Universidade Federal de Uberlândia

Programa de Pós–graduação em Ciência da Computação
Faculdade de Computação
Universidade Federal de Uberlândia
Minas Gerais – Brasil

Fevereiro de 2006

Renê Rodrigues Veloso

DISTRIBUIÇÃO DE TAREFAS EM SISTEMAS DE WORKFLOW COM BASE NA APTIDÃO DOS RECURSOS

Dissertação apresentada ao programa de Pós-graduação em Ciência da Computação da Universidade Federal de Uberlândia, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Engenharia de Software.

Orientador: Prof. Dr. Autran Macêdo

UBERLÂNDIA – MINAS GERAIS – BRASIL
Universidade Federal de Uberlândia - UFU
Fevereiro 2006

FICHA CATALOGRÁFICA

Elaborada pelo Sistema de Bibliotecas da UFU / Setor de Catalogação e Classificação

V443d Veloso, Renê Rodrigues, 1979-

Distribuição de tarefas em sistemas de workflow com base na aptidão dos recursos / Renê Rodrigues Veloso. – Uberlândia, 2006.

93f.:il.

Orientador: Autran Macêdo.

Dissertação (mestrado) – Universidade Federal de Uberlândia, Programa de Pós–Graduação em Ciência da Computação.

Inclui Bibliografia.

1. Engenharia de software – Teses. 2. Fluxo de trabalho – Teses. I. Macêdo, Autran. II. Universidade Federal de Uberlândia. Programa de Pós–Graduação em Ciência da Computação. III. Título.

CDU:681.3.06

Renê Rodrigues Veloso

DISTRIBUIÇÃO DE TAREFAS EM SISTEMAS DE WORKFLOW COM BASE NA APTIDÃO DOS RECURSOS

Dissertação apresentada ao Programa de Pós-graduação em Ciência da Computação da Universidade Federal de Uberlândia, para obtenção do título de Mestre.

Área de concentração: Engenharia de software.

Banca Examinadora:

Uberlândia, 16 de Fevereiro de 2006.

Prof. Dr. Autran Macêdo – UFU (orientador)

Prof. Dr. José Valdeni de Lima – UFRGS

Prof. Dr. Carlos Roberto Lopes – UFU

Dedicado à

Minha família.

Distribuição de Tarefas em Sistemas de Workflow com Base na Aptidão dos Recursos

Renê Rodrigues Veloso

Resumo

Distribuir tarefas em sistemas de *workflow* não é um prolema trivial. Há duas soluções para realizar essa distribuição: *Push* e *Pull*. Essas soluções, em geral, não consideram a aptidão dos recursos. No entanto, recursos que compartilham o mesmo papel no workflow podem possuir diferentes graus de aptidão a uma tarefa. A literatura de *worklfow* aponta a solução baseada em *Pull*, com aptidão dos recursos, como ideal para distribuir tarefas, resultando em execuções mais ágeis e com melhor qualidade.

Este trabalho mostra que, se utilizada a aptidão dos recursos, a solução baseada em *Push* apresenta melhor eficiência do que a *Pull* e permite um balanceamento mais refinado nos quesitos tempo e qualidade de execução dos processos de negócio. Os experimentos realizados mostram a efetividade da abordagem utilizada. Uma comparação com soluções tradicionais comprova ganhos significativos em termos de tempo e qualidade.

Palavras-chave: *workflow*, distribuição de tarefas, qualidade.

Tasks Distribution in Workflow Systems Based on Resources Aptitude

Renê Rodrigues Veloso

Abstract

The distribution of tasks in workflow systems is not a trivial problem. There are two solutions to make this distribution: Push and Pull. These solutions, in general, do not consider the resources aptitude. However, resources that share the same role in the workflow can have different degrees of aptitude to a task. The literature in workflow points the solution based on Pull mechanisms with resources aptitude as ideal to tasks, resulting in a more accurate executions and with more quality.

This work shows that, if the resources aptitute is used, the solution based on Push presents a better performance and allows a more tuned balancing, regarding to time and quality of the execution of the business processes. The experiments demonstrate the effectiveness of the used approach. A comparison with traditional solutions shows gains in terms of time and quality of tasks execution.

Keywords: workflow, distribution of tasks, quality.

Copyright

Agradecimentos

Muitos amigos e colegas, assim como membros da família, ajudaram a conceber este trabalho. Não é possível agradecer a todos pelo nome, mas quero mencionar alguns.

Obrigado à minha mãe, Cleusa, pelas orações e encorajamento; ao meu pai, Carlos Reinan, pela força e compreensão; e aos meus irmãos – Ramon, Renata e Emanuel – por serem quem são.

Obrigado aos vários amigos da querida "Moc–city" e a todos da comunidade N. S. do Perpétuo Socorro, pelo apoio e orações; em especial, obrigado a Lincoln, ao compadre Leandro de Jesus e à comadre Sandra, pelo apoio e pela amizade sincera.

Obrigado ao professor orientador, Dr. Autran Macêdo, pelo voto de confiança, pelo apoio constante, pela paciência, pelo encorajamento, pela amizade e pelo excelente humor.

Obrigado a muitos amigos e colegas da Faculdade de Computação (FACOM-UFU), como Gustavo Carmo, Grings, Rogério, Rodolfo, Alexandre Fieno, Ivan, Juliana, Joslaine, Vinícius, Daniel, Mylene, Caruline e Fernanda Francielle, pelo apoio, conselhos, suporte e encorajamento.

Obrigado ao Programa de Pós–Graduação em Ciência da Computação da Universidade Federal de Uberlândia, aos professores e funcionários, pela oportunidade.

Obrigado aos "caras do pensionato", pelo acolhimento, pela amizade e por tornarem estes dois anos de mestrado mais alegres e descontraídos. Obrigado também ao grande amigo Lupeinstein "Lupi" Einstein-Heimer, por existir.

Obrigado ao "Dina–bomba", por me manter acordado durante os estudos.

Obrigado ao vinho e ao *blues*, pela companhia nas noites solitárias.

Obrigado à Coordenação de Aperfeiçoamento de Pessoal de Estudo Superior (CAPES), pela concessão da bolsa de estudo.

Obrigado a todos que não impediram a realização deste trabalho.

Obrigado a Deus, pela vida, força e providência diárias.

"*Observei o conjunto da criação de Deus e percebi que o homem não consegue entender toda a obra que se faz debaixo do Sol. Por mais que se afadigue em pesquisar, não chega a compreendê-la, e mesmo que o sábio diga que a conhece, nem por isso é capaz de entendê-la*" Eclesiastes 8:17

Sumário

Lista de Abreviaturas

Abreviatura	**Detalhes**
Abs_alloc	*Absolute allocation*
DTD	*Data Type Definition*
EDD	*Earliest Due Date*
FIFO	*First-In, First-out*
RBAC	*Role-Based Access Control*
Sel-push	*Selective-push*
Sel-push-ft	*Selective-push-flowtime*
SGBD	Sistema de Gerência de Banco de Dados
SGWf	Sistema de Gerência de Workflow
SPT	*Shortest Processing Time*
UT	Unidades de tempo
XPDL	*XML Process Definition Language*
WAPI	*Workflow Application Programming Interface*
WES	*Workflow Enactment Service*
WfMC	*Workflow Management Coalition*
WIDE	*Workflow Interactive Development Environment*
WPDL	*Workflow Process Definition Language*

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Atualmente, a maioria das organizações utilizam sistemas baseados em computador para a realização de seus negócios. Esses sistemas, em maior ou menor grau, procuram implementar o processo de negócio, desde a execução de uma atividade no nível de produção até uma tomada de decisão no nível estratégico.

Os sistemas de *workflow* fornecem a tecnologia-chave que possibilita a gerência de processos de negócio nas organizações [36, 37]. No entanto, para que essa tecnologia seja aplicada com eficiência, é necessário que o sistema de gerência de *workflow* (SGWf) tenha informações sobre as proficiências dos recursos, que compõem o corpo organizacional. Essas informações permitem que o SGWf distribua tarefas para os recursos mais adequados à execução das mesmas.

À medida que um processo é executado, a função de distribuição do sistema de *workflow* transforma as tarefas em itens de trabalho. Esta transformação se caracteriza pela distribuição (ou entrega) de uma tarefa a um recurso disponível no sistema. Neste sentido, os sistemas de *workflow* atuais são regidos por dois mecanismos de distribuição:

- mecanismo *Push*: a tarefa é atribuída a um único recurso;
- mecanismo *Pull*: a tarefa é apresentada a vários recursos e um destes a seleciona para execução.

O mecanismo *Push* é visto pela literatura de *workflow* como o mais *seguro*, ou seja, mais confiável quanto à qualidade do serviço desempenhado, pois é o sistema quem escolhe o recurso. Essa escolha geralmente é feita levando-se em consideração

características conhecidas do recurso. Já o mecanismo *Pull* é tido como mais liberal e flexível, pois permite que uma quantidade maior de recursos tomem conhecimento das tarefas que devem ser executadas.

A distribuição de tarefas é uma função crítica dos sistemas de *workflow* [28, 40] e que está diretamente relacionada com o tempo de conclusão e com a qualidade da execução dos processos [15, 19, 21]. O problema é que a maioria dos sistemas de *workflow* atuais focam a dimensão dos processos (isto é, roteamento de tarefas e controle de fluxo) e simplificam a dimensão organizacional (isto é, os recursos) [19]. Esse tipo de enfoque faz com que o sistema deixe de considerar atributos dos recursos, tais como disponibilidade e aptidão, gerando uma distribuição de tarefas em geral ineficiente. Isto é, uma tarefa pode ser atribuída para muitos, poucos ou simplesmente para recursos inadequados. Esta falta de sensibilidade por parte do sistema de *workflow* é que se traduz em ineficiência (perda de desempenho e baixa qualidade dos processos executados).

O principal exemplo de mau desempenho de um processo é quando muitas tarefas são atribuídas a poucos recursos, proporcionando o surgimento de "filas de espera" de execução. Essas filas de espera são consideradas "gargalos" no fluxo de trabalho, pois acarretam atraso na conclusão de tarefas e esgotamento de recursos [12]. Este quadro pode piorar quando os recursos não possuem adequação necessária para a realização das tarefas, pois realizarão o trabalho de forma menos eficiente que os recursos mais aptos, diminuindo portanto a qualidade do processo final.

Este trabalho apresenta uma proposta de distribuição de tarefas tomando como base a aptidão dos recursos. Foram implementados três mecanismos de distribuição baseados em *Push* (*Sel-push*, *Sel-push-10* e *Sel-push-flowtime*) e avaliado o compromisso entre desempenho e qualidade na execução dos processos. O impacto de variação de tempo de processamento individual das tarefas, o número de recursos envolvidos e a carga de sistema nessas medidas foram avaliadas em comparação com os mecanismos propostos por Kumar *et al.* [19]. Os experimentos foram realizados em uma ferramenta de simulação capaz de executar instâncias de processos sob diferentes cargas de trabalho.

Observou-se que os mecanismos *Sel-push*, *Sel-push-10* e *Sel-push-flowtime* apresentaram ganhos de qualidade e tempo de execução médios no cenário simulado.

Estes ganhos foram verificados graficamente e por meio de testes estatísticos, pelos quais se constatou a viabilidade da solução em um sistema real de *workflow*.

1.1 Trabalhos Relacionados

A atual pesquisa em distribuição de tarefas pode ser dividida em duas ramificações [19]. A primeira é focada em considerações de desempenho e a segunda é dedicada à segurança na execução dos processos.

O estudo conduzido por Zapf e Heinzl [39] é um exemplo típico relacionado à primeira ramificação. Esse estudo investiga as diferenças de desempenho entre trabalhadores generalistas (i.e., que possuem um papel genérico) e trabalhadores especialistas (i.e., que possuem um papel específico) no contexto de processos de negócio em um *call center*. Shen *et al.* [25] também propuseram melhorias de desempenho por meio de um mecanismo de distribuição baseado em lógica fuzzy. Tal mecanismo considera a experiência dos recursos, os seus relacionamentos e a complexidade das tarefas; embora os autores não deixem claro como essas características podem ser capturadas, explicam a eficiência da solução, que é fortemente baseada no mecanismo *push*. Outras implementações [2, 17, 29] buscam atribuir tarefas a recursos por meio de mecanismos baseados em leilão. Nesses tipos de mecanismos, os recursos ou pagam uma determinada quantidade de moeda virtual, ou recebem créditos pela execução de tarefas, alternando entre os mecanismos *push* e *pull*. É comum encontrar soluções implementadas entre esses dois mecanismos. Zeng e Zhao [40], por exemplo, propõem mecanismos híbridos com a vantagem de *push* e *pull*. Nesta proposta os recursos possuem filas individuais de tarefas que são preenchidas dinamicamente por meio de uma heurística. Tal heurística movimenta as tarefas de uma fila compartilhada para as filas individuais. Essa solução entrega tarefas em lotes (*batching-based online optimization*), com o objetivo de minimizar o máximo *flowtime*, ou seja, o tempo entre a chegada de uma tarefa e seu término.

Um exemplo de trabalho pertencente à segunda ramificação é o modelo RBAC (*Role-Based Access Control*) [11, 13, 23, 24]. A principal característica do RBAC é que as permissões são associadas a papéis e usuários são membros desses papéis. Seguindo essa mesma linha, Castano e Fugini, em [9], propuseram um modelo baseado nas chamadas: *active-rules*, implementadas no projeto WIDE (*Workflow Interactive*

Development Environment) [8]. Essa solução toma como base conceitos de papel, agente, em padrões de autorização e regras. Atluri e Huang [3, 4] introduzem a *ativação e revocação dinâmica de privilégios*. Essa idéia propõe que usuários devem ganhar privilégios somente para acessar objetos no contexto de tarefas autorizadas. Por fim, todos esses trabalhos dizem respeito à segurança e são tipicamente de natureza qualitativa, ou seja, aspectos quantitativos como custos, taxa de utilização e tempo de execução não são levados em consideração.

Os trabalhos relacionados anteriormente estão localizados em uma das duas ramificações: com foco em segurança (e.g., papéis, separação de responsabilidades, controle de acesso) ou com foco em desempenho (e.g., taxa de utilização, tempo de execução). Kumar *et al.* [19] visualizam a distribuição de tarefas de ambas perspectivas de segurança e desempenho, tentando criar um balanço entre elas. Neste caso, esses autores relacionam segurança com a qualidade das atribuições, considerando que as tarefas serão atribuídas somente a recursos altamente capacitados e que pertençam aos papéis que as executam. Assim, eles propõem mecanismos de distribuição fortemente baseados em uma solução *Pull*, incluindo fatores que não são considerados nos trabalhos anteriormente citados, como a aptidão dos recursos às tarefas, a taxa de ocupação, a qualidade das execuções dos processos e a carga de trabalho do sistema. A aptidão indica o grau de adequação deles às tarefas, ou seja, o grau de adequação mede o nível de competência para a realização de uma tarefa específica.

Este trabalho dá seguimento à linha proposta por Kumar *et al.* [19]. Contudo, os resultados obtidos mostram que a abordagem *Push* apresenta ganhos de eficiência quando comparado à abordagem *Pull*. Esses ganhos foram preliminarmente apresentados no Workshop Brasileiro de Tecnologias para Colaboração (WCSCW'05) [34] e na Semana da Computação da FESURV (SECOMF'05) [33] que tratam dos passos iniciais desta pesquisa, em que foi proposto o mecanismo *Sel-push* e primeiros resultados. Os resultados alcançados neste trabalho comprovam a efetividade do uso de uma solução baseada em *Push* em comparação aos mecanismos de Kumar *et al.*, baseados em *Pull*.

1.2 Organização do Trabalho

Este trabalho está organizado da seguinte forma:

- o Capítulo 2 discute os conceitos básicos da tecnologia de *workflow*;
- o Capítulo 3 define os principais conceitos da área de distribuição de tarefas em workflow;
- o Capítulo 4 apresenta os mecanismos de distribuição baseados em aptidão dos recursos, bem como a implementação da solução proposta;
- o Capítulo 5 descreve o experimento computacional realizado neste estudo, o ambiente de experimentação e as considerações quanto à geração da carga de trabalho, em seguida os resultados obtidos são comentados;
- o Capítulo 6 apresenta de maneira superficial a ferramenta de simulação construída para dar suporte ao estudo realizado e;
- o Capítulo 7 conclui o trabalho, analisando seu impacto potencial e os problemas envolvendo o uso de tais mecanismos, visando à melhoria de desempenho e qualidade dos sistemas de *workflow* futuros.

Capítulo 2

Workflow — Conceitos Básicos

Workflow é uma disciplina, uma prática e um conceito [1]. É também uma tecnologia valiosa que, se corretamente implementada, tem um impacto significante nas operações de uma organização, para a qual pode trazer grandes benefícios.

Este capítulo descreve os conceitos básicos que envolvem essa área de pesquisa e que serão referência para um melhor entendimento dos capítulos seguintes.

2.1 O que é *workflow*?

Em 1999, a WfMC[1] publicou um glossário com todos os termos relacionados a *workflow* [37], com a seguinte definição:

> "Workflow é a automação, total ou parcial, de processos de negócio, durante a qual documentos, informações ou tarefas são passadas de um participante a outro, de acordo com um conjunto de regras".

Esse conceito reflete o que existe em várias outras definições da literatura de *workflow*, e todas elas convergem a um ponto principal: processos de negócio.

2.1.1 Processos de Negócio

Ao conjunto de atividades que são executadas para a realização de um trabalho, chama-se *processo*, que, em uma organização, pode ser entendido, também, como

[1] *Workflow Management Coallition*: instituição fundada para padronizar os conceitos e a tecnologia de *workflow*

sendo um *procedimento* em que documentos, informações e tarefas são passadas entre os participantes para a realização de um *objetivo de negócio* qualquer. Tem-se, então, o que se denomina *processo de negócio* [20, 31]

Os processos de negócio são representados por *fluxos de trabalho*, ou seja, modelos que especificam: as tarefas que compõem o processo, a ordem e as condições que as tarefas devem ser executadas, os executores de cada tarefa, as ferramentas a serem utilizadas e as informações manipuladas durante sua execução. Os fluxos de trabalho, que representam os processos de negócio, podem ser interpretados/automatizados por sistemas de *workflow* como mostrado na Figura 2.1.

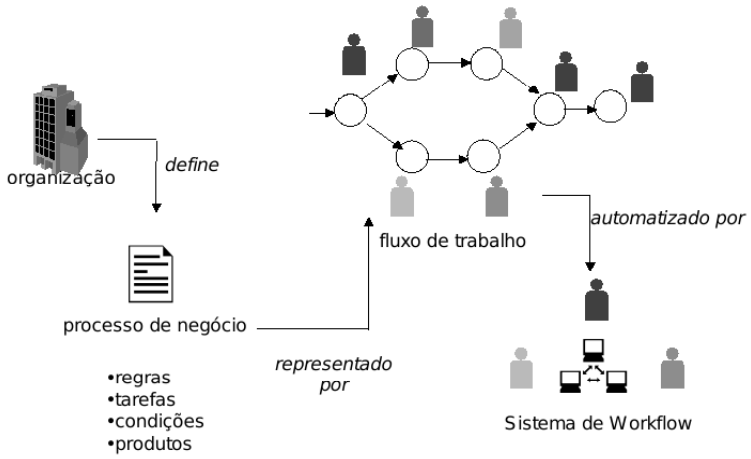

Figura 2.1: Processos de negócio automatizados por sistemas de *workflow*

2.1.2 Sistema de Workflow

Os fluxos de trabalho, em situações de negócio específicas, são contidos em sistemas de *workflow*, que têm por objetivo auxiliar as organizações na especificação, execução, monitoração e coordenação desses fluxos em um ambiente organizacional [36].

Sistemas *workflow* possuem grande relevância para o dia-a-dia das organizações de hoje, uma vez que podem interagir de modo pró-ativo com seus usuários, controlando o processo de negócio, emitindo avisos sobre os mais diferentes temas e auxiliando a execução de uma tarefa.

Sistema de Gerência de Workflow

Assim como os sistemas de bancos de dados são desenvolvidos e usados com o auxílio de um sistema de gerência de banco de dados (SGBD), um sistema de gerência de *workflow* (SGWf) é utilizado para implementar sistemas de *workflow*.

O sistema de gerência de *workflow* atua como um *sistema operacional*, controlando os processos entre os vários recursos (pessoas ou aplicações) [31]. A ele portanto, é destinada a logística dos processos. De acordo com Rob Allen, em [1], um sistema de gerência de *workflow* compreende:

> "Um sistema que define, cria e gerencia a execução de *workflow*s através do uso de software, executando em um ou mais *motores de workflow*, que é capaz de interpretar a definição de processo, interagir com os participantes do *workflow* e, quando necessário, invocar o uso de ferramentas de TI e aplicações"

Deve-se notar que existe uma leve diferença entre sistema de gerência de *workflow* (SGWf) e sistema de *workflow*. A palavra "sistema" no contexto de *workflow*, segundo Van Der Aalst e Van Hee [31], significa todas as pessoas, máquinas e sistemas de informação computadorizados (ou não) que executam processos *particulares*. Entre esses processos particulares pode estar, por exemplo, um processo para análise de currículos de candidatos à vaga de emprego em uma organização (Figura 2.2). Assim sendo, a automação desse processo é um *workflow* que, em conjunto com outras ferramentas, inclusive um SGWf, compõem o sistema de *workflow* para seleção dos candidatos.

Apesar de haver no mercado vários produtos diferentes que se colocam como ferramentas de gerência de *workflow*, eles possuem características comuns entre si no que diz respeito às suas funções principais [30]. A WfMC identifica que, em um nível mais alto, todos os SGWfs têm as seguintes funcionalidades:

- funções de "tempo de construção", que lidam com a definição e modelagem dos processos de *workflow* e suas tarefas constituintes. O resultado final dessa função é a definição acabada de processo;
- funções de "tempo de execução", que gerenciam e executam os processos de *workflow* em um ambiente operacional, além de seqüenciar e distribuir as tare-

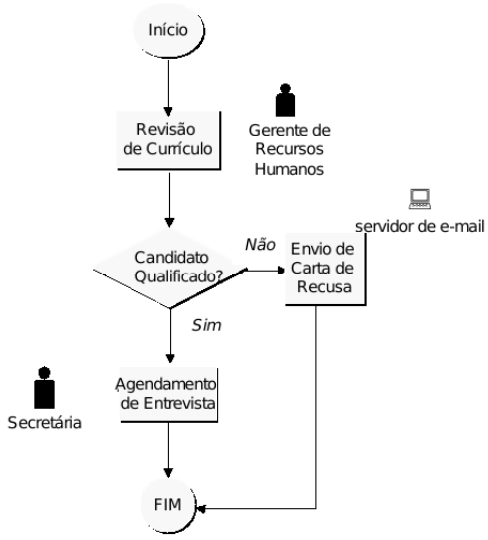

Figura 2.2: Processo de negócio para análise de currículos

fas necessárias para a execução desses processos. São essas funções, que cuidam de criar uma instância de processo para lidar com algum objetivo específico, distribuir suas tarefas para os usuários participantes e acompanhá-lo até o término de sua execução;

- funções de "interação" (em tempo de execução) com os usuários participantes e com ferramentas computacionais para o processamento dos vários passos de cada tarefa.

A relação entre esses itens pode ser vista na Figura 2.3. As setas indicam o sentido das interações.

2.1.3 Workflow Enactment Service

O componente mais importante que compõe um sistema de gerenciamento de *workflow*, de acordo com o modelo de referência da WfMC (seção 2.1.4), é chamado *workflow enactment service* (WES). Composto por uma ou mais *motores de workflow*, o WES é responsável por gerenciá-las e integrá-las, provendo o suporte necessário para a execução dos processos.

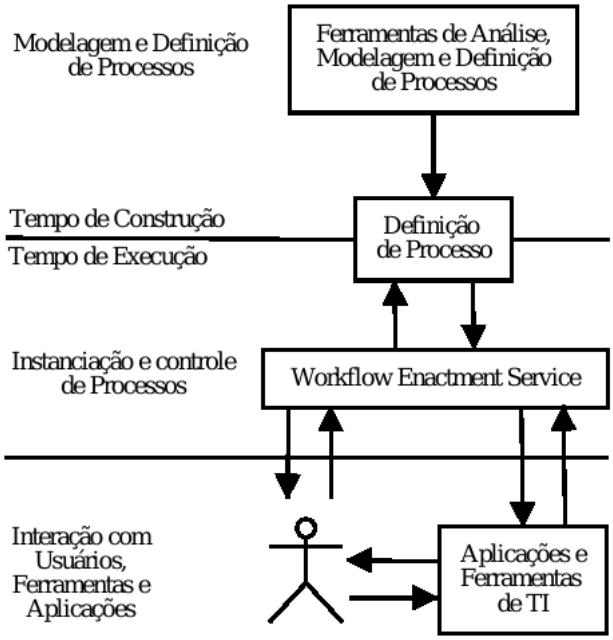

Figura 2.3: Relação entre as principais funções de um SGWf

Motor de *Workflow*

O motor de *workflow* é o ponto central de um SGWf [31]. Ela provê o ambiente de execução necessário para os processos e é onde se concentram funcionalidades tais como a criação de instâncias de processos, coordenação e roteamento de atividades, geração de itens de trabalho e controle de recursos.

2.1.4 O Modelo de Referência da WfMC

De forma a estabelecer padrões para a área de *workflow*, a WfMC desenvolveu um modelo de referência [36]. O objetivo principal desse modelo é identificar os muitos componentes que se integram para formar um ambiente de gerência e execução de *workflows*. O modelo é descrito em termos de suas respectivas interfaces e formatos de intercâmbio necessários para promover a interoperabilidade necessária entre as várias ferramentas de *workflow* de desenvolvedores diferentes.

O modelo de referência da WfMC, apresentado na Figura 2.4, define cinco interfaces entre componentes, além de uma sobre o serviço de execução de *workflow*, denominada WAPI (Workflow Application Programming Interface). A interface WAPI constitui-se da parte principal do modelo. Nela se encontram os motores de

workflow, que, por sua vez, ficam dentro do *workflow enactment service* (WES). A WAPI possui também cinco interfaces que são o meio por onde os serviços de *workflow* podem ser implementados, contanto que os métodos que as acessam obedeçam aos padrões da WfMC.

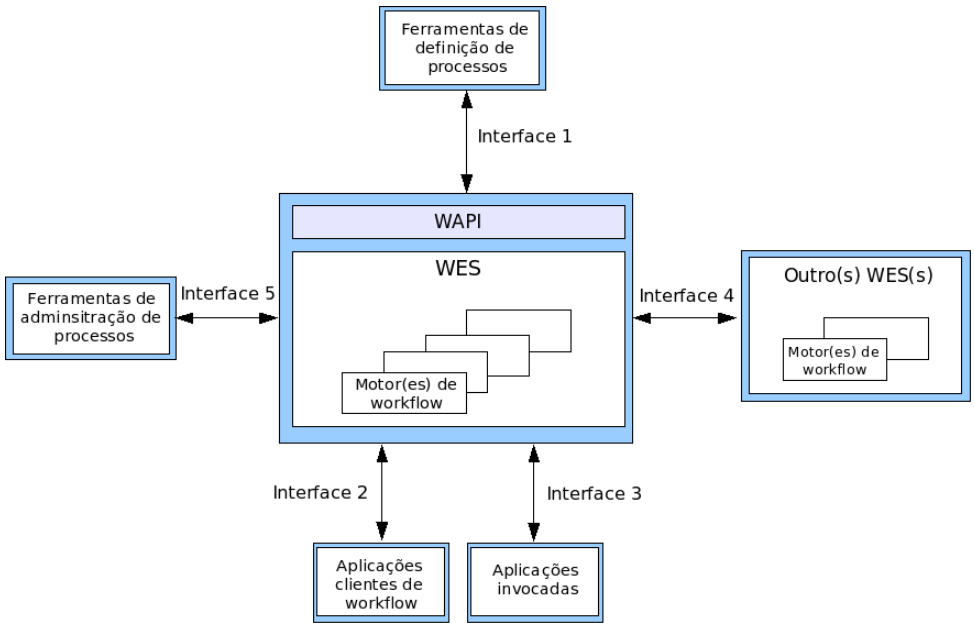

Figura 2.4: Modelo de referência da WfMC

Cada interface é responsável por uma parte do sistema. A interface 1 envolve as ferramentas de definição de processos que usam uma linguagem padrão (a WPDL - *Workflow Process Definition Language*), entendida por ferramentas de modelagem de *workflow* e por serviços de execução de *workflow*.

A interface 2 dedica-se aos aplicativos-cliente de *workflow*, que comunicam-se com o motor de *workflow* por meio do gerenciador da lista de trabalho. Esta interface padroniza as chamadas do gerenciador da lista, como, por exemplo, a recuperação de itens, e as declarações de início e término de atividades. Assim, um aplicativo construído por um terceiro pode utilizar os serviços de gerência de *workflow* de qualquer produto que suporte este padrão.

A interface 3 acessa as aplicações externas, invocadas pelo serviço de execução de *workflow* para a realização de atividades específicas. Com a interface 4 existe a possibilidade de que o WES se conecte a outros externos, visando ao intercâmbio ou a execução conjunta de processos. Com esta interface, torna-se possível a execução de um processo através de vários SGWf diferentes. Por exemplo, várias organizações

poderiam fazer parte de um único processo, aumentando ainda mais o grau de coordenação entre si. Por último, a interface 5 envolve a comunicação com ferramentas de administração e monitoramento dos processos executados. Assim, diversos aplicativos de outros fabricantes que suportem este padrão podem facilmente utilizar os serviços de gerência de *workflow*.

A WAPI e todas as suas interfaces são bem conhecidas pelos desenvolvedores de produtos de *workflow*, porém a WfMC ainda está trabalhando no processo de padronização do modelo.

2.2 Conceitos

2.2.1 Instância de Processo ou Caso

Sistemas de *workflow* auxiliam na gerência de tarefas que, quando encadeadas e distribuídas aos executores, visam a cumprir um certo objetivo. Esse objetivo caracteriza um serviço ou trabalho (organizacional ou não).

Existem muitos e diferentes tipos de trabalho no mundo real, assim como escrever um relatório, fazer um bolo, projetar uma casa ou selecionar candidatos a uma vaga de emprego. Para todos esses exemplos, há processos e trabalhos que modificam ou criam objetos: um relatório, um bolo, uma casa, um empregado; ou seja,"coisas" genéricas que podem ser nomeadas ou particularizadas como, por exemplo: escrever o relatório de vendas do ano anterior, fazer um bolo de chocolate, projetar uma casa de três quartos, uma sala e uma cozinha, ou selecionar um candidato à vaga de analista de sistemas senior.

Um *workflow* constitui-se de um processo genérico com objetivos globais, como é o caso do processo *fazer um bolo* em que o bolo pode ser de chocolate, côco, etc. Quando o processo genérico é utilizado para um fim específico e contextualizado, tem-se o que se chama de *caso* ou *instância* de processo.

2.2.2 Tarefa

A tarefa é uma unidade lógica de trabalho, indivisível[2], que é executada por um recurso [31]. Dessa forma, são as tarefas que, coletivamente, completam o objetivo do *workflow* [7]. Exemplos de tarefas incluem: escrever uma carta, avaliar um relatório, imprimir um documento, fechar uma venda, etc.

Uma tarefa refere-se a uma especificação genérica de algo a ser feito de maneira manual, automática ou semi-automática. Van Der Aalst e Van Hee identificam a tarefa em dois tipos que, dependendo de seu estado em uma instância de processo de *workflow*, podem ser um item de trabalho (*work item*, em inglês) ou uma atividade.

O item de trabalho é a combinação de um caso (instância de processo) e uma tarefa que aguarda para ser executada, conforme a representação da Figura 2.5. Dessa forma, um item de trabalho é uma tarefa que ainda não está em execução, mas pode ser executada a qualquer momento. Em geral, itens de trabalho são contidos em listas de trabalho (ou *work list*, em inglês) até que sejam selecionados para a execução. Uma atividade, ao contrário do item de trabalho, refere-se à tarefa em estado de realização dentro da instância de processo. Portanto, a atividade constitui-se do item de trabalho que foi selecionado da lista de trabalho e está sendo executado. É importante notar que, diferente da tarefa, o item de trabalho e a atividade estão ligados a uma instância específica.

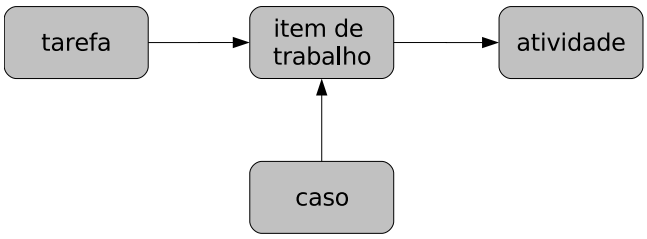

Figura 2.5: Relacionamento entre os termos tarefa, caso (instância de processo), item de trabalho e atividade (Van Der Aalst e Van Hee, em [31])

[2]O termo indivisível, retirado de [31], tem sido amplamente discutido em pesquisas, como, por exemplo, as que envolvem a antecipação de tarefas [27]. Em tais pesquisas, procura-se aumentar o paralelismo entre as tarefas quebrando-as, em tempo de execução, de maneira a flexibilizar a liberação de resultados intermediários.

2.2.3 Recurso

Em processos de negócio, o trabalho é direcionado aos participantes, também chamados de recursos [10, 15, 19, 21, 31, 37, 40, 42]. Assim como existe um modelo de processos (i.e., a ordem lógica e temporal das atividades que são necessárias para processar um objetivo de negócio), os recursos são descritos em um modelo de recursos [42]. Dessa forma, o modelo de recursos contém a definição dos recursos humanos e tecnológicos que estão envolvidos na execução do sistema de *workflow* (modelo de processos) como participantes.

De acordo com a WfMC [37], a partir do modelo de recursos da organização, os participantes do *workflow* podem ser divididos em dois tipos: os participantes concretos e os abstratos.

Os participantes concretos caracterizam o aspecto estático do modelo de recursos conhecidos em tempo de construção do modelo de processos. Portanto, são as pessoas, máquinas e software que compõem o corpo da organização e que executarão as tarefas a eles atribuídas. Já os participantes abstratos refletem os aspectos dinâmicos do sistema de *workflow* e são conhecidos em tempo de execução de um processo. Eles são determinados e atribuídos aos participantes concretos durante a execução do *workflow*. Em geral, esses participantes incluem: (a) recursos; (b) conjunto de recursos; (c) unidades organizacionais e (d) papéis.

Recurso é o nome genérico para uma pessoa, máquina ou grupo de pessoas e/ou máquinas que podem realizar tarefas específicas [31]. No entanto, as tarefas também podem ser atribuídas a um conjunto de recursos ou aos recursos que estão inseridos em uma unidade organizacional ou departamentos, como por exemplo, o departamento de vendas, secretaria, tesouraria ou a gerência.

Em geral, os recursos trabalham de acordo com o seu perfil organizacional [40]. Neste caso, somente as tarefas que são específicas daquele perfil organizacional podem ser executadas. Van Der Aalst e Van Hee [31] fazem a divisão de recursos em grupos; os recursos agrupados compartilham os mesmos cargos, qualificações ou funções operacionais dentro da organização e geram a noção de classes de recursos. A essas classes de recursos com o mesmo perfil, executando tarefas específicas, dá-se o nome de papel (ou *role*, em inglês) [31, 42].

2.2.4 Roteamento

Uma das funções básicas de um sistema de *workflow* é coordenar fluxos de trabalho. Todas as vezes em que um recurso ocupado por uma atividade termina a sua execução, o sistema direciona o fluxo de trabalho a outro recurso, até que o objetivo seja alcançado. Dessa forma, cada instância de processo segue um caminho que é especificado na definição de processos por meio do roteamento das tarefas que o compõem.

Neste sentido, a WfMC identificou seis tipos primitivos de roteamento (Figura 2.6) que podem ser mapeados utilizando o formalismo de Redes de Petri [22,31], em que:

- tarefas são mapeadas em transições e relações causais são representadas no modelo por lugares;
- a transição t1 representa a sincronização de dois subfluxos (*AND-join*);
- transições t21 e t22 representam um *OR-join*: dois subfluxos unidos em um subfluxo;
- a transição t3 representa um *AND-split*: um subfluxo dividido em dois subfluxos paralelos;
- transições t41 e t42 representam um *OR-split*: uma seleção apresentada entre as duas alternativas;
- a iteração pode ser representada pela adição de uma transição de retorno (t52);
- a conexão de duas transições (t61 e t62) por meio de um lugar intermediário resulta em duas tarefas seqüenciais.

A modelagem de *workflow* por meio das Redes de Petri tem sido bastante utilizada por diversos especialistas e pesquisadores [14, 35, 38, 41]. Isso ocorre devido às semelhanças existentes entre os diversos elementos de modelagem de *workflow* e os componentes envolvidos no formalismo das Redes de Petri. Nesse sentido, cada tarefa é representada por uma transição correspondente. Os lugares são componentes que indicam as pré e pós-condições requeridas para a execução dessas tarefas. E

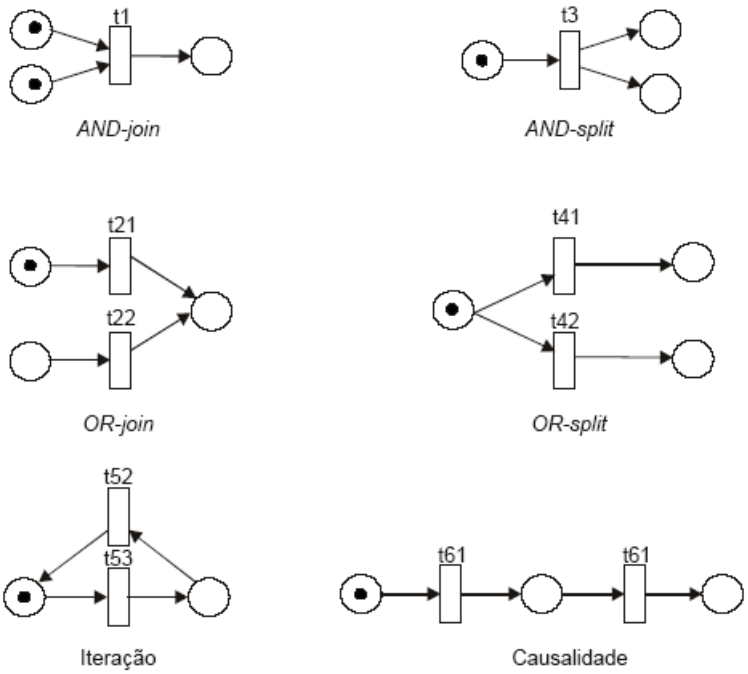

Figura 2.6: Tipos básicos de roteamento usando *AND/OR* e *split/join* por meio de Redes de Petri

os arcos expressam as relações lógicas entre elas, demonstrando, assim, o fluxo de trabalho exigido [22].

Em geral, a modelagem de processos complexos exige muito tempo e uma análise detalhada, devido ao grande número de estados. Essa situação pode agravar-se ainda mais com a detecção paralelismos e/ou sincronismo entre as atividades envolvidas. As Redes de Petri, com todos os seus recursos matemáticos e gráficos, permitem a modelagem de fluxos de trabalhos com elevados índices de complexidade. Tal complexidade é encontrada nas representações de paralelismo, concorrência e simultaneidade que revelam informações importantes sobre os comportamentos dinâmicos dos processos.

Com as Redes de Petri, é possível representar um fluxo de trabalho, bem como os seus vários recursos que são compartilhados e consumidos pelas atividades envolvidas. A composição gráfica das Redes de Petri auxilia na visualização das diversas atividades envolvidas no processo e como é feita a comunicação entre elas. As marcações dos lugares, também chamados de *tokens*, podem simular a informação (documentos, dados, formulários, etc) que são passadas e consumidas ao longo do

fluxo.

Muito embora as Redes de Petri sejam vistas como uma ótima ferramenta de modelagem de processos, esse formalismo é comentado aqui somente como forma de visualizar os tipos primitivos de roteamento de tarefas e não serão mais consideradas neste trabalho, pois a função de distribuição de tarefas em *workflow* independe do tipo de ferramenta utilizada para modelar os processos.

Capítulo 3

Distribuição de Tarefas

O procedimento de distribuição de tarefas a recursos é uma das funções críticas de um sistema de gerência de *workflow* (SGWf) [15, 21, 28, 40]. Isso se deve ao fato de que, apesar de ser especificado no momento da modelagem do processo, o casamento entre tarefa e recurso é ativado em tempo de execução pelo SGWf. Isto envolve uma série de considerações que o sistema deve fazer acerca dos recursos, entre elas, está o tempo que cada recurso leva para realizar uma tarefa, a sua disponibilidade e possíveis violações de restrições que aparecem durante a execução do processo de negócio.

Quando o sistema de gerência *workflow* atribui uma tarefa a um recurso, o que ocorre é uma modificação da especificação do processo em tempo de execução. Dessa forma, os processos de negócio sofrem modificações que ocorrem dinamicamente a medida em que o processo é executado. Este conceito de modificação dinâmica de processos e os princípios relacionados à distribuição de tarefas em *workflow*, são apresentados nesta seção.

3.1 Modificações Dinâmicas

A capacidade de adaptação dos processos às mudanças que ocorrem durante a sua execução, é uma característica desejável em *workflow* [21]. A reorganização de um departamento, a ausência de participantes e a indisponibilidade de uma impressora são exemplos de mudanças que podem ocorrer em tempo de execução. Às mudanças que ocorrem no sistema de *workflow* em tempo de execução, dá-se o nome de modi-

ficações dinâmicas. Dependendo de sua durabilidade, elas podem ser dos seguintes tipos:

- modificações de instância (ou de execução) de *workflow* e;
- modificações de definição de *workflow*.

As modificações de instância preocupam-se com as mudanças que ocorrem em tempo de execução dos processos de *workflow* a curto prazo, como exceções e *distribuição de tarefas* a recursos. Já as modificações de definição concentram-se em modificações na especificação dos processos a longo prazo. Essas mudanças incluem: aumento do número de tarefas, mudança de contexto de processo, reestruturação organizacional, etc.

Atualmente, alguns SGWfs suportam modificações dinâmicas em *workflow*, incluindo ambas modificações de instância e de definição. Porém, a atual pesquisa e desenvolvimento em *workflow* é focada em modificações de definição de processo (e.g. controle de fluxo), e, geralmente, omite os problemas envolvidos nas modificações de instância como o problema de distribuição de tarefas [21]. Portanto, a seção seguinte busca esclarecer alguns pontos importantes, assim como, os problemas envolvidos nessa área de pesquisa em *workflow*.

3.2 Princípios de Distribuição

Enviar tarefas aos recursos de maneira desorganizada pode elevar o tempo de finalização dos processos [31]. É importante que o sistema de *workflow* gerencie a distribuição das tarefas de forma que os processos terminem o mais rápido possível.

Distribuir tarefas, no contexto deste trabalho, é o ato de transformar tarefas prontas para serem executadas em itens de trabalho. Os itens de trabalho, por sua vez, quando acessados pelos recursos, são postos em execução como atividades. Para transformar tarefas em itens de trabalho e depois em atividades, algumas decisões devem ser tomadas [31]:

1. *Em qual ordem as tarefas são transformadas em itens de trabalho?* Se existe um excesso de tarefas prontas em um determinado momento, não é uma boa prática transformar cada tarefa em item de trabalho imediatamente. Pode ser

que haja mais tarefas que recursos disponíveis. Logo uma escolha deve ser tomada levando em conta a ordem pela qual as tarefas são enviadas.

2. *Por quais recursos os itens de trabalho serão executados?* Os recursos possuem características e habilidades diferentes uns dos outros, é importante saber para qual recurso será atribuída uma dada tarefa (i.e., qual recurso terá um item em sua lista de trabalho). Um recurso especialista, por exemplo, pode executar certas tarefas mais rapidamente. Porém, se ele fizer parte de muitos papéis, as tarefas de outros processos poderão concorrer entre si pelo recurso, elevando o tempo de conclusão dos processos.

3. *Em qual ordem os itens de trabalho são transformados em atividades?* Depois que as tarefas foram atribuídas aos recursos na forma de itens de trabalho, estas poderão ser selecionadas nas listas de trabalho para a execução. Essa seleção pode influenciar no tempo de finalização dos processos, por exemplo, se existirem tarefas independentes umas das outras (como de instâncias diferentes executando em paralelo). Dependendo da ordem com que os itens de trabalho são escolhidos, podem ser privilegiados às instâncias menos urgentes ou os mais demoradas. Portanto, a ordem da seleção dos itens de trabalho é importante.

Essas três decisões são importantes em sistemas de *workflow*, pois influenciam diretamente na taxa de ocupação dos recursos, no aumento do tempo de execução e qualidade dos processos.

A ordem com que as tarefas são transformadas em itens de trabalho (item 1 da lista anterior) está diretamente relacionada com a ordem com que as instâncias de processos chegam ao sistema [30]. Como os processos de negócio quase sempre se diferem entre si, a ordem com que são instanciados é importante no contexto do sistema. Se existirem tarefas independentes umas das outras (e.g., de instâncias diferentes executando em paralelo), a ordem de execução pode influenciar no tempo total de serviço dos recursos ou na alocação de recursos estratégicos.

Diversas heurísticas podem ser aplicadas para selecionar uma ordem específica de tarefas. Elas foram propostas para solucionar problemas de *job shop* da área de *scheduling* e são chamadas de *dispatching rules* [6, 26, 30]. Segundo Van Der Aalst e Van Hee, a rota de um processo através de vários recursos exibe similaridades com

a rota de um produto entre as máquinas de um departamento de produção, sendo a FIFO (*First-In, First-Out*) a mais utilizada atualmente [30, 31]. Algumas dessas heurísticas, aplicadas à ordem de execução das instâncias de processos, são descritas a seguir:

- FIFO (*First-In, First-Out*): Os processos são executados na ordem em que foram instanciados. O que chegou primeiro é executado em primeiro lugar.
- SPT (*Shortest Processing Time*): Seleciona as instâncias na ordem de tempo de execução. As instâncias que exigirem menos tempo são selecionadas primeiro. Essa heurística é conhecida pela eficácia em reduzir o tempo de execução total do sistema.
- EDD (*Earliest Due Date*): Se uma instância foi iniciada em certo tempo (ou data), ela deve preferencialmente ser completada em um tempo previsto (*due date*). Essa heurística determina a ordem de execução das instâncias baseado nas suas *due dates*. Então, uma instância que deve ser finalizada primeiro tem prioridade sobre as outras.

Percebe-se, então, que antes que sejam de fato alocadas aos recursos, as tarefas percorrem um caminho composto por três passos (Figura 3.1): (i) ordenação geral das instâncias; (ii) distribuição das tarefas e; (iii) ordenação local. A ordenação geral é feita antes que um mecanismo de distribuição seja executado, acontece no momento em que as tarefas estão prontas para se tornarem itens de trabalho. Já a ordenação local, geralmente realizada pelos recursos, é onde os itens de trabalho são escolhidos para tornarem-se atividades. Essas duas formas de ordenação podem ser guiadas pelas *dispatching rules* descritas anteriormente.

A ordem em que as tarefas entram em execução[1] está fortemente relacionada com a seleção dos recursos. Neste sentido, se as tarefas (que são transformadas em itens de trabalho) podem ser executados por mais de um recurso, então o sistema de gerência deve poder selecioná-las e distribuí-las entre os recursos mais adequados. A escolha dos recursos e a distribuição das tarefas deve acontecer de tal maneira que

[1]De maneira a evitar a confusão gerada pelo uso dos termos *tarefa*, *item de trabalho* e *atividade*, no restante deste trabalho, assume-se o termo *tarefa* designando os estados "em execução" e "em espera", para *atividade* e *item de trabalho*, respectivamente

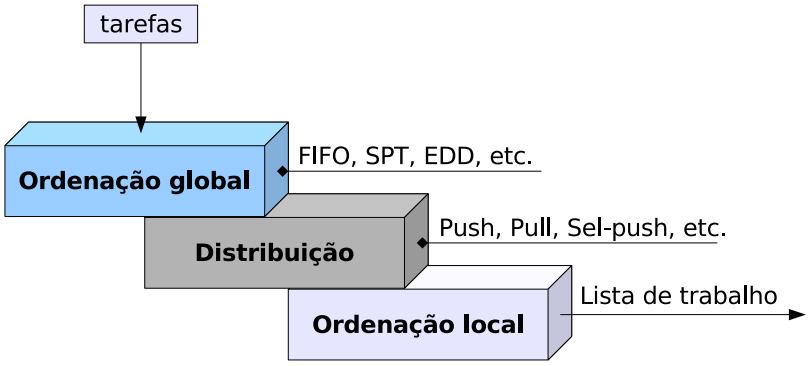

Figura 3.1: Passos que envolvem o processo de alocação de tarefas

não ocorra sobrecarga aos recursos e não prejudique o tempo médio de execução dos processos.

Apesar da importância da ordem em que as tarefas são selecionadas, o foco principal deste trabalho é a distribuição das tarefas, independentemente de como foi feita a ordenação. Nesse sentido, este trabalho procura mostrar através de experimentos, que uma distribuição de tarefas bem realizada pode ocasionar em ganhos efetivos de desempenho na execução dos processos. Por esse motivo, os passos de ordenação e sequenciamento que podem acontecer antes e após a distribuição de tarefas não são considerados. Contudo, reconhece-se a importância do estudo desses passos dentro da área de atribuição dinâmica e execução de processos de *workflow*.

Tipos de Distribuição

A característica de distribuição de trabalho em sistemas de *workflow*, deve assegurar que cada tarefa de um processo seja realizada por um recurso adequado. Governatori *et al.* [15] acreditam que a entrega de tarefas deve ser guiada pelos atributos (habilidades, capacidades e oportunidades) dos recursos. Já Muehlen em [42], considera dois diferentes conceitos que podem ser usados para reportar tarefas a recursos: designação direta e atribuição por papel.

O processo de designação direta, busca atribuir tarefas para um ou mais recursos diretamente. Em tempo de execução, o motor de *workflow* pode encontrar os recursos responsáveis e adicionar itens de trabalho em suas respectivas listas. Esse tipo de atribuição é fácil de gerenciar, porque é concentrado em único tipo de entidade participante: o executor de *workflow*. Dessa forma, se uma tarefa deve ser

feita por um grupo de executores, todos os membros do grupo são atribuídos à tarefa, um por um. O grande problema do conceito de designação direta é a total dependência existente entre o modelo de recursos e o modelo de processos. Assim, toda e qualquer mudança que ocorrer na população organizacional é refletida no modelo de processo, que, conseqüentemente, tem que ser mudado também. No entanto, segundo Muehlen, esse conceito é raramente utilizado na prática.

Muitos sistemas de gerência de *workflow* atuais fornecem ferramentas de modelagem de processos com suporte ao uso de papéis. Ao tentar distribuir tarefas, o sistema deve levar em consideração as classes de recursos especificadas na definição de processo. O principal propósito do uso de papéis é a separação do modelo de processos e o modelo de recursos, cujas mudanças na população organizacional não afetam o modelo de processos diretamente.

O uso de papéis ao invés de designação direta, provê, indiretamente, meios de balanceamento de carga, pois todos os membros de um papel qualificado são notificados sobre algum item de trabalho pendente, mas somente um membro desse grupo precisa realizar a tarefa. Alguns trabalhos [10,40] definem esse conceito de atribuição de tarefas a recursos de acordo com as suas qualificações de papéis, como sendo uma *resolução de papel* (ou *Role Resolution*, em inglês). Dessa forma, o sistema de gerenciamento de *workflow* deve realizar um processo de resolução para determinar os membros do papel antes de notificá-los. Essa denominação também diz respeito às decisões que afetam diretamente a produtividade e a eficiência dos trabalhadores em uma organização. Desse modo, faz-se necessário o desenvolvimento de mecanismos eficazes que administrem essas decisões.

3.3 Mecanismos de Distribuição de Tarefas

O suporte à distribuição de tarefas disponível nos sistemas de gerência de *workflow*, pode ser categorizado em dois tipos [19, 31, 40]:

- *Push*: Existe um relacionamento automático entre itens de trabalho e recursos. Dentro de certas condições, o motor de *workflow* escolhe quais recursos devem realizar cada item de trabalho. O recurso é incapaz de fazer escolhas. Assim que ele termina de executar uma atividade, outro item de trabalho é dado a

ele. Dessa forma, o motor "empurra" (*pushes*) itens de trabalho aos recursos (Figura 3.2).

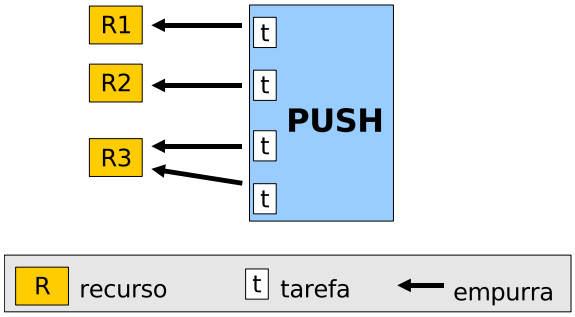

Figura 3.2: Mecanismo *Push*

- *Pull*: Os próprios recursos escolhem os itens de trabalho que querem executar. Cada recurso verifica o(s) item(s) que é capaz de executar, então seleciona-o de uma fila compartilhada. Essa fila pode ser visível para todos os recursos ou somente para papéis pré-determinados, em que somente recursos pertencentes a esses papéis podem acessá-la e retirar (*pull out*) itens (Figura 3.3).

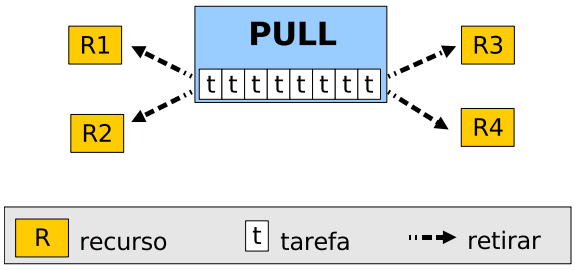

Figura 3.3: Mecanismo *Pull*

A modificação dinâmica de instância focada na função de distribuição de tarefas, constitui o objeto de estudo deste trabalho. Nesse contexto, os mecanismos de distribuição são diferentes implementações dos mecanismos *Push* e *Pull*. A solução proposta baseia-se na extensão aos mecanismos propostos por Kumar *et al.*, procurando atingir um melhor equilíbrio entre qualidade (segurança) e tempo de execução dos processos (desempenho).

3.3.1 Qualidade e Tempo de Execução de Processos

A qualidade da execução de um processo de *workflow* está diretamente relacionada ao grau de aptidão dos recursos alocados para cada tarefa [19]. Quanto mais aptos os recursos envolvidos, maior é a qualidade do trabalho realizado por eles e, conseqüentemente, maior é a qualidade do processo executado.

A aptidão dos recursos pode ser aferida por meio de métricas que indicam o seu grau de adequação à realização das tarefas[2]. O grau de adequação mede o nível de competência dos recursos para a realização das tarefas do *workflow*. No contexto deste trabalho, recursos com alto grau de adequação a uma determinada tarefa possuem maior aptidão para a executá-la. Por conseguinte, é de se esperar que esses recursos realizem essa tarefa de modo mais eficiente do que aqueles menos adequados. Eficiência está relacionada ao índice de re-trabalho e tempo de execução das tarefas. Quanto menores esses índices, maior a eficiência. Assim, é possível que melhorias efetivas de desempenho, na execução dos processos, sejam conseguidas com a implementação de mecanismos que distribuam melhor as tarefas e que essa distribuição seja feita de maneira a priorizar recursos mais adequados.

As métricas de alocação e qualidade são definidas como uma solução sistemática para criar dinamicamente um balanço entre qualidade e tempo de execução. Este balanço é conseguido por meio da construção de um modelo mais aprimorado de distribuição de tarefas em sistemas de *workflow*. Dessa forma, os sistemas devem saber selecionar os recursos mais aptos para realizarem as tarefas de um processo qualquer que se deseja automatizar.

3.4 Aptidão dos Recursos

3.4.1 Métrica de Alocação

A métrica de alocação é definida por Kumar *et al.* [19] como uma solução sistemática para criar dinamicamente um balanço entre qualidade e tempo de execução. Essa métrica é utilizada para estabelecer os recursos mais apropriados à execução de tarefas, sendo composta por parâmetros, tais como adequação, urgência, conformidade

[2]A aptidão é quantificada considerando-se recursos pertencentes a um mesmo papel.

e disponibilidade.

Adequação

Adequação diz respeito à aptidão de um recurso em relação a uma determinada tarefa. Esse parâmetro é dado pela função $adequacao(w, rs) \in [0, 1]$, onde w é uma tarefa do *workflow* e rs é um recurso do *workflow*. Quanto menor o valor de *adequação*, menos aptidão rs possui para executar w.

Urgência

Urgência representa a pressa para a realização de uma tarefa. Esse parâmetro é expresso pela função $urgencia(w) \in [0, 1]$. Quanto maior o valor, maior é a urgência de w. Intuitivamente, uma combinação de fatores afeta a urgência de uma tarefa. Esses fatores podem ser, por exemplo, a proximidade com a data de expiração da tarefa, a quantidade de dinheiro envolvido, o tamanho de uma compra ou a reclamação de um cliente. As tarefas terão níveis de urgência variados, e valores altos irão facilitar a sua atribuição. Essa é uma medida subjetiva determinada pelo gerente de *workflow*, que é responsável por especificá-las. No entanto, o autor propõe que o valor inicial seja ajustado em 0,8, indicando um nível relativamente alto de urgência para todas as tarefas e que é incrementado gradativamente. Este parâmetro influencia diretamente a ordem em que as tarefas serão distribuídas.

Conformidade

Conformidade mede o quanto um recurso respeita as restrições definidas para o *workflow*. Faz uso de dois outros parâmetros: *penalidade* e *violação*.

Dado um conjunto de restrições C, cada $c \in C$ possui uma penalidade que determina um valor de *multa* pela violação da restrição ($penalidade(c) \in [0, 1]$). Se a penalidade é 1, significa que esta restrição nunca deve ser violada. Tais restrições são chamadas inflexíveis (ou *hard*). Por outro lado, se a penalidade for estritamente menor que 1, então a restrição é flexível (ou *soft*). Por exemplo, uma restrição assim como "o mesmo vendedor que fez o pedido de compra deve manusear as reclamações relacionadas a ela" pode ter uma penalidade de 0,1 refletindo uma restrição flexível e de menos importância. Mas, de forma contrária, uma restrição como "três vice-

presidentes devem aprovar a nomeação de um diretor departamental" pode ter uma penalidade de 0,95 para sugerir que esta restrição é mais inflexível e é raramente violada. Algumas restrições podem depender também dos atributos de uma instância de processo . Por exemplo, uma restrição pode especificar que, se a quantidade de dinheiro envolvida em uma tarefa for superior a um milhão de reais, então somente um gerente pode aprová-la, e a violação deve significar uma penalidade de 0,99. Assim, em geral, essas penalidades são associadas com base na percepção de importância de cada restrição e combinadas usando a respectiva fórmula.

Violação ($violacao(c, w, rs) \in \{true, false\}$), indica se a restrição c foi violada (ou não) pelo recurso rs na execução da tarefa w. Assim sendo, *conformidade* pode ser medida pela função:

$$conformidade(w, rs) = \prod_{c \in C, violacao(c,w,rs)=true} (1 - penalidade(c))$$

Como um outro exemplo, considere-se que, em um conjunto de restrições, duas restrições flexíveis com penalidades de 0,3 e 0,4 sejam violadas. Então, a fórmula anterior retorna uma *conformidade* de 0,42. Similarmente, se ambas as restrições forem de 0,4, então a conformidade será de 0,36. O fator *conformidade* é outro parâmetro incluído na métrica de alocação e é uma medida do alcance de aceitação com as restrições. Um valor 0 de conformidade significa que uma restrição inflexível (*hard*) está sendo violada. Assim, grandes valores de conformidade implicam menores violações, enquanto valores pequenos refletem maiores violações (ou baixa conformidade com as restrições).

Disponibilidade

Disponibilidade indica o quanto um recurso está disponível para a realização de uma tarefa. Esse parâmetro é expresso pela função $disponibilidade(rs) \in [0, 1]$. Valores abaixo de 1 indicam que o recurso está alocado a alguma tarefa. Valor igual a zero indica que o recurso está indisponível.

O parâmetro *disponibilidade* leva em consideração o tempo disponível para um trabalhador durante um período de tempo, sua carga de trabalho e ausência. Para o autor, esse parâmetro é muito subjetivo e não pode ser determinado de uma maneira exata. Para facilitar isso, ele define que, de um modo geral, um recurso possui três

níveis de disponibilidade: (i) Baixa disponibilidade: 0,8; (ii) Média disponibilidade: 0,9 e; (iii) Alta disponibilidade: 1,0.

Além disso, uma disponibilidade igual a zero indica a ausência de um recurso ao qual não deve ser oferecida tarefa alguma. Inicialmente, um recurso assumirá ter alta disponibilidade por padrão. No entanto, à medida que o recurso for se ocupando, ela poderá ser reduzida em decrementos até 0,8. No contexto deste trabalho, a disponibilidade é calculada por meio de uma regra simples comentada com mais detalhes na seção 4.1.

Fator de Alocação Absoluta

Todos os parâmetros são combinados em uma métrica que determina a alocação de uma tarefa a um recurso. Esta métrica, chamada de *fator de alocação absoluta* ou, simplesmente, *abs_alloc*, é definida como o produto dos quatro parâmetros explicados anteriormente.

$$\begin{aligned} abs\_alloc(w, rs) = \; & adequacao(w, rs) \times \\ & urgencia(w) \times \\ & conformidade(w, rs) \times \\ & disponibilidade(rs) \end{aligned} \tag{3.1}$$

A métrica *abs_alloc* depende somente de uma tarefa e de um recurso, isoladamente. Essa métrica assume um valor entre 0 e 1, que é uma medida absoluta de adequação de um recurso para realizar uma tarefa num instante de tempo. Quanto maior o valor de *abs_alloc* de um recurso, mais adequado (ou apropriado) ele é para executar a tarefa.

É importante notar a diferença entre *adequacao(w,rs)* e *abs_alloc(w,rs)*. A primeira é independente de contexto e não leva em consideração quaisquer restrições. A segunda depende da urgência de uma tarefa, de possíveis violações de restrições e da disponibilidade do recurso.

A métrica da equação 3.1 pode, ainda, ser adaptada caso o sistema de *workflow* (ou a definição de processos) não suporte certos parâmetros. Por exemplo, se na modelagem dos processos não existirem restrições especificadas, basta considerar o parâmetro *conformidade* igual a 1. Deve-se lembrar que o valor 1 para conformidade

significa que o recurso não violou quaisquer restrições, o que é verdadeiro, já que elas não existem ou não foram especificadas. Da mesma forma, é possível que todas as tarefas mantenham sempre o mesmo valor de urgência, como uma constante, esse valor permaneceria inalterado não importando as condições em que se encontre a tarefa.

Exemplo 1: Suponha a execução de um processo qualquer em que exista uma tarefa *A* pronta para ser executada e cuja urgência de alocação é 0,8. Considera-se também que o processo não possui restrições modeladas. Para esse processo, três recursos estão disponíveis, a saber, o *Recurso1*, o *Recurso2* e o *Recurso3*, valores respectivos de adequação e disponibilidade expressos na Tabela 3.1.

Tabela 3.1: Parâmetros de adequação para a tarefa A

Recursos	**Disponibilidade**	**Adequação**
Recurso1	0,8	1,0
Recurso2	1,0	0,9
Recurso3	0,9	0,8

Utilizando a equação 3.1, o sistema calcula o valor de alocação absoluta para cada recurso disponível, podendo identificar o mais apropriado para a tarefa em questão. Os valores de alocação absoluta calculados são os da Tabela 3.2. Neste caso, o melhor recurso, isto é, o mais apropriado é o recurso *Recurso2*, por seus altos valores de adequação e disponibilidade.

Tabela 3.2: Valores de alocação absoluta para a tarefa *A*

Recursos	***abs_alloc***
Recurso1	0,64
Recurso2	0,72
Recurso3	0,58

3.4.2 Métrica de Qualidade

A métrica de qualidade do trabalho realizado é uma importante medida que deve ser considerada em sistemas *workflow*. Ela informa se os recursos alocados para a instância de processo recém-executada foram os mais adequados ou não. Essa

métrica é a razão da qualidade do trabalho realizado (qualidade atual) e a máxima qualidade possível depois da atribuição de todas as tarefas que são parte da instância de *workflow*. Nesse sentido, definimos a qualidade atual como sendo:

$$Q_{atual}(wf) = \sum_{w \in wf} adequacao(w, exec(w)) \times conformidade(w, exec(w)) \quad (3.2)$$

A equação 3.2 avalia a qualidade atual da realização das tarefas w em um *workflow wf*. A função *exec* mapeia uma tarefa em um recurso que atualmente executou w. Essa equação define a qualidade atual como o produto entre adequação e conformidade. Os valores resultantes dessa métrica estão entre 0 e 1. Os valores mais próximos a 1 indicam uma boa qualidade do trabalho. Um valor zero nunca irá ocorrer. Para a qualidade ser zero, ou o recurso não é adequado ou não está em conformidade com as restrições, e se isso acontecesse, ele não seria alocado para realizar a tarefa. Portanto, a qualidade nunca assumirá o valor zero.

A qualidade máxima possível do *workflow*, ou qualidade ideal, é definida como:

$$Q_{ideal}(wf) = \sum_{w \in wf} \max_{rs \in RS} [adequacao(w, rs) \times conformidade(w, rs)] \quad (3.3)$$

A métrica da equação 3.3 considera o valor máximo do produto entre adequação e conformidade de um recurso rs dentre todos os recursos (conjunto RS) disponíveis. São avaliados todos os recursos que participam do *workflow wf* e que podem realizar a tarefa w.

A qualidade atual e a ideal são utilizadas na definição da qualidade total. Essa qualidade total é a razão entre as qualidades atual e ideal:

$$Q_{total}(wf) = \frac{Q_{atual}(wf)}{Q_{ideal}(wf)}$$

A qualidade total é tida como um valor no intervalo entre 0 e 1. Valores próximos a zero (nunca em zero) indicam baixa qualidade, ou seja, os recursos selecionados não foram os melhores. Valores próximos a 1 mostram que os recursos mais indicados foram escolhidos e realizaram o trabalho.

Exemplo 2: Suponha que, no exemplo 1, a tarefa A seja alocada para o recurso *Recurso2*. Logo, aplicando as equações 3.2 e 3.3, pode-se obter a qualidade resultante dessa alocação. Por exemplo:

$$Q_{atual} = 0,9 \times 1,0 = 0,9$$

$$Q_{ideal} = \max_{recursos}[(1,0 \times 1,0), (0,9 \times 1,0), (0,8 \times 1,0)] = 1,0$$

$$Q_{total} = \frac{0,9}{1,0} = 0,9$$

Verifica-se, então, que a qualidade da alocação realizada possui qualidade igual a 0,9, ou seja, o grau de recursos adequados que executaram tarefas no sistema foi próximo ao ideal.

Capítulo 4

Mecanismos de Distribuição Baseados em Aptidão

A aptidão dos recursos é uma importante característica a ser considerada pelos mecanismos de distribuição, pois, com ela, é possível inferir a capacidade e a eficiência dos recursos para realizar tarefas específicas. Nesse sentido, Kumar *et al.* [19] definem três mecanismos baseados em *pull*, em que recursos aptos podem selecionar tarefas de uma fila compartilhada. Esses mecanismos possuem uma política que permite que as tarefas estejam visíveis aos recursos. A política baseia-se, praticamente, na definição de limiares de adequação que aumentam ou diminuem a quantidade e a qualidade dos recursos envolvidos (Figura 4.1).

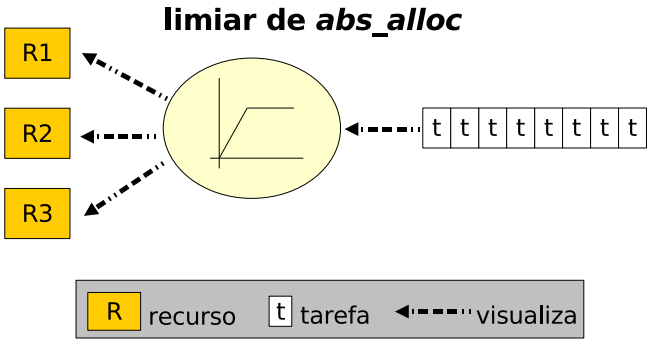

Figura 4.1: Mecanismos *Pull-driven* baseados em aptidão

Os limiares são os seguintes:

- *Typical*: uma tarefa w é visível ao recurso rs se, e somente se, *abs_alloc(w, rs)* excede um valor de 0,5. Este representa um recurso típico.

- *Receptive*: uma tarefa w é visível ao recurso rs se, e somente se, *abs_alloc(w, rs)* excede um valor de limiar menor que o típico, e.g., 0,4. Este indica um recurso mais receptivo.

- *Selective*: uma tarefa w é visível ao recurso rs se, e somente se, *abs_alloc(w, rs)* excede uma limiar mais alto que o típico, e.g., 0,6. Este indica um recurso mais seletivo que o recurso típico.

Esses mecanismos, segundo os autores, combinam características dos mecanismos *Push* e *Pull*, melhorando o tempo de execução e qualidade dos processos. Os três mecanismos (*Typical*, *Receptive* e *Selective*) foram simulados no trabalho de Kumar *et al.*, e os resultados obtidos em função do tempo de execução e qualidade dos processos. Esses resultados são mostrados graficamente nas figuras 4.2 e 4.3. Para as simulações, o autor considerou que o mecanismo *Push* entrega as tarefas somente aos recursos mais adequados, segundo o parâmetro $adequacao(w, rs)$. O mecanismo *Pull* entrega para qualquer recurso disponível, simulado de maneira aleatória.

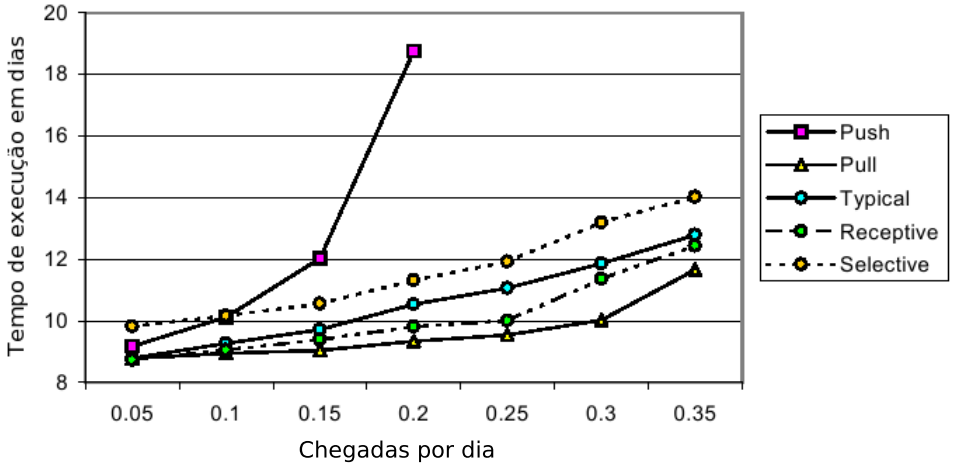

Figura 4.2: Resultados de Kumar *et al.*. Média de tempos de execução.

A Figura 4.2 mostra a média de tempos de execução dos processos para cinco mecanismos diferentes (*Push*, *Pull*, *Typical*, *Receptive* e *Selective*) resultante dos experimentos de Kumar *et al.*. Nesse gráfico, o mecanismo *Push* é o pior mecanismo, ou seja, eleva muito os tempos de execução dos processos, pois facilita a formação de pilhas de tarefas para os recursos. O mecanismo *Pull*, ao contrário, obtém os melhores tempos, pois é muito liberal e muitos recursos são candidatos às tarefas, resultando em pouca carga de trabalho para cada recurso.

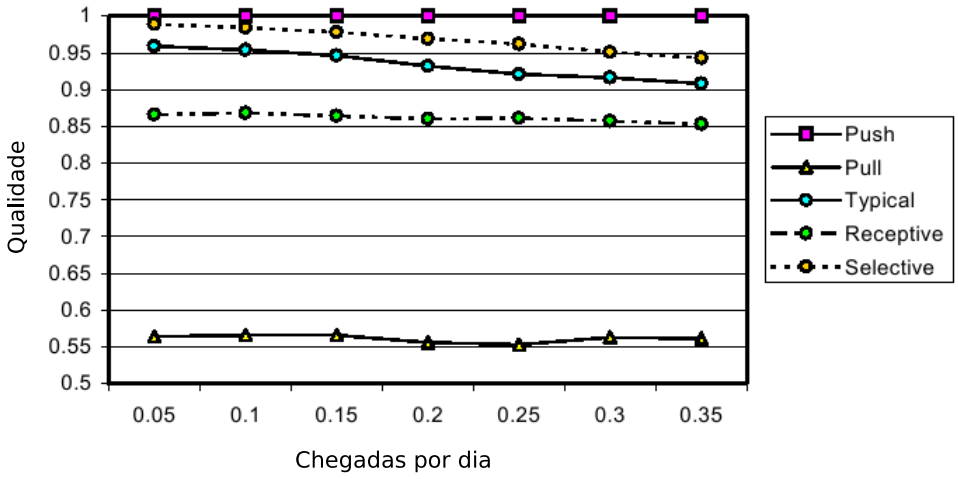

Figura 4.3: Resultados de Kumar *et al.*. Média de métricas de qualidade.

Na Figura 4.3, a média de qualidade para cada um dos mecanismos é apresentada. Nesse caso, o mecanismo *Pull* realiza um trabalho mais pobre em comparação com os outros quatro mecanismos. É interessante notar que, no gráfico da Figura 4.2, esse mecanismo foi o que obteve melhor desempenho de tempo de execução. Houve uma inversão de valores entre o tempo e a qualidade, o que acontece também com o *Push*, que possui maior qualidade e pior tempo.

Segundo Kumar *et al.*, o mecanismo *Selective* resulta num maior balanço entre tempo e qualidade, seguido pelo *Typical* e *Receptive*, unindo, assim, o melhor dos mecanismos *Push* e *Pull*.

4.1 Mecanismo *Sel-Push* de Distribuição

A proposta apresentada nesta dissertação envolve uma extensão aos mecanismos de distribuição citados anteriormente. Tomando como base os valores de alocação absoluta, o diferencial dessa extensão está na forma com que as tarefas são entregues aos recursos. Antes que os recursos recebam as tarefas, verificam-se os seus valores de disponibilidade. Ao receberem tarefas, os recursos ficam menos disponíveis no contexto do sistema. Dessa forma, um recurso que esteja com uma carga de trabalho alta não deverá receber mais tarefas, sendo estas repassadas a outros que sejam adequados e que tenham disponibilidade para tal. Baseada em *push*, a extensão é implementada como o mecanismo *Sel-push* e suas duas modificações, *Sel-push-10*

e o *Sel-push-flowtime*; obtendo um balanço mais fino entre tempo e qualidade na execução dos processos.

O mecanismo *Sel-push* utiliza uma heurística em que cada recurso procura ocupar-se de tarefas enquanto estiver disponível. O algoritmo 1 mostra, em alto nível, como é implementada essa heurística.

A regra utilizada para decrementar a disponibilidade de um recurso é simples. Seja w uma tarefa e $FilaW$ o conjunto de todas as tarefas prontas para serem executadas, tal que $w \in FilaW$, tem-se que:

$$percent = tempo(w)/tempo(FilaW) \tag{4.1}$$

A variável *percent* da equação acima, contém a porcentagem de tempo que a tarefa w ocupa ($tempo(w)$) considerando a soma de tempo total das tarefas em $FilaW$ ($tempo(FilaW)$).

A equação 4.2 calcula o novo valor de disponibilidade do recurso rs (onde rs é um recurso de um dado conjunto RS de recursos que interagem com o sistema) em função do percentual exigido pela tarefa w.

$$disponibilidade(rs) = calcula\_Disponibilidade(rs, percent) \tag{4.2}$$

O algoritmo *sel-push* é mostrado a seguir:

Algoritmo 1 Algoritmo Sel-push

1: **para toda** tarefa w tal que $w \in FilaW$ **faça**
2: selecione um recurso $rs \in RS$ com maior *abs_alloc*
3: calcule o novo valor de *disponibilidade* para rs
4: **se** *disponibilidade* < 0 **então**
5: descarte rs
6: volte ao passo 2
7: **senão**
8: aloque w a rs
9: **fim se**
10: **fim para**

O algoritmo de distribuição (em geral) é invocado sempre que a fila de tarefas (prontas) não se encontra vazia e continua a execução até que todas as tarefas da

fila sejam entregues, ou até que os recursos não possuam disponibilidade para o trabalho. Nesse último caso, o mecanismo entra em estado de espera, aguardando que algum recurso se desocupe. A Figura 4.4 mostra o esquema de funcionamento do algoritmo, em que cada recurso de um conjunto pode ser selecionado para receber a tarefa. Na figura, a seta dupla indica que o algoritmo pode selecionar outro recurso, se o atual não tiver disponibilidade para a tarefa.

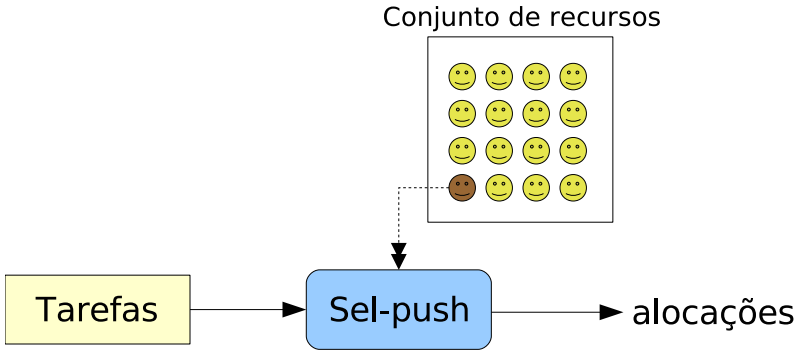

Figura 4.4: Esquema de funcionamento Sel-push

É característica importante do *sel-push* o passo 2 do algoritmo. O passo 2 é onde ocorre a seleção do recurso com maior *abs_alloc*, guiada por um *limiar dinâmico*, isto é, um recurso que, inicialmente, encontra-se dentro de uma faixa de limiar seletivo (maior ou igual a 0,6) e, à medida que recebe tarefas, sofre decréscimos em seu valor de *abs_alloc*, passando para o limiar típico (maior ou igual a 0,5) e depois para o receptivo (maior ou igual a 0,4). O *sel-push* verifica de antemão se os recursos mais aptos para a tarefa são seletivos; caso não existam, então procura-os em limiares mais baixos.

A idéia por trás desse mecanismo é a mesma utilizada pelo clássico "problema da mochila" [5]. Nele, deve-se poder colocar o máximo de itens úteis em uma mochila, de maneira a preencher toda a sua capacidade. Esse mesmo conceito é empregado à distribuição de tarefas, onde o máximo de tarefas serão atribuídas a um recurso apto, considerando a sua capacidade. É um procedimento chamado de "guloso" pela literatura de análise de algoritmos e baseia-se na busca de máximos globais a partir de uma visão local do problema.

Mesmo que se utilize heurística aproximada, o mecanismo *Sel-push* visa uma distribuição segura e confiável, no sentido de que os recursos mais aptos serão es-

colhidos, sem abusar da sua capacidade. Quando um recurso apto excede a sua capacidade, outro é tentado e assim por diante. No entanto, é importante frisar que, se houver somente um recurso disponível ao processo de *workflow*, esse abuso é inevitável, porque o algoritmo apresentará comportamento semelhante ao *Push*, elevando o tempo de conclusão da instância de processo em questão.

4.1.1 O Mecanismo *Sel-push-10*

Semelhante ao *sel-push* em funcionamento, o mecanismo *sel-push-10* procura atribuir tarefas a recursos adequados. No entanto, além de selecionar o recurso com maior *abs_alloc*, este mecanismo também considera a vizinhança do recurso. A vizinhança de um recurso são todos os outros recursos que estão a uma certa distância dele. A distância é medida em termos de valor de alocação absoluta. No caso do *sel-push-10* essa vizinhança é encontrada a um raio de 10% de *abs_alloc* do recurso mais adequado. Assim, o algoritmo 2 descreve o funcionamento do *sel-push-10*.

Algoritmo 2 Algoritmo Sel-push-10

1: **para toda** tarefa w tal que $w \in FilaW$ **faça**
2: selecione um recurso $rs \in RS$ com maior *abs_alloc*
3: selecione o recurso $rs' \in RS$, mais disponível, num raio de 10% de *abs_alloc* de rs
4: calcule o novo valor de *disponibilidade* para rs'
5: **se** *disponibilidade* < 0 **então**
6: descarte rs'
7: volte ao passo 2
8: **senão**
9: aloque w a rs'
10: **fim se**
11: **fim para**

O passo 3 do algoritmo 2 é o mais importante e envolve a escolha do recurso com menor taxa de ocupação dentro da vizinhança. Esse passo é o que diferencia este algoritmo do *Sel-push*. Com ele, é possível encontrar na vizinhança um outro recurso também adequado, mas que tenha recebido poucas tarefas ou nenhuma (Figura 4.5).

O objetivo desta versão do *Sel-push* é diminuir uma possível sobrecarga do recurso mais adequado. Isso pode ocorrer quando, mesmo tendo recebido algumas

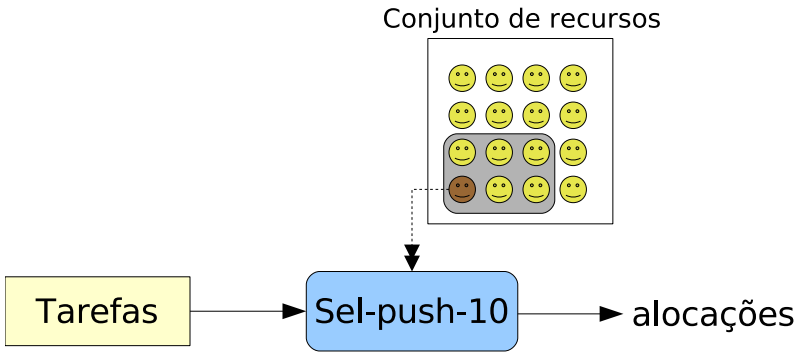

Figura 4.5: Esquema de funcionamento Sel-push-10

tarefas, o recurso mais adequado ainda possua um valor de *abs_alloc* maior que os demais. Se não for considerada a sua "vizinhança", esse recurso mais adequado "atrairá" muitas tarefas para si, causando contenção e filas de espera.

A solução aqui construída é mais "democrática" e delega trabalho a um número maior de recursos. Essa solução contribui para diminuir a sobrecarga de trabalho dos recursos, não prejudicando a qualidade do serviço, pois somente os recursos que estiverem a uma baixa distância do mais adequado serão considerados.

4.1.2 O Mecanismo *Sel-push-flowtime*

Baseando-se no *Sel-push*, um outro mecanismo denominado *Sel-push-flowtime* é proposto aqui. Esse mecanismo agrega ao *Sel-push* a característica de considerar, além da disponibilidade e a aptidão dos recursos, o *flowtime* das tarefas.

Define-se o *flowtime* de uma tarefa, como o tempo entre o seu início e sua finalização dentro de uma instância de processo [6]. Assim, o objetivo é basear as entregas das novas tarefas nos *flowtimes* daquelas já alocadas.

Quando o mecanismo detecta uma tarefa pronta para ser executada, procura pelo recurso com maior *abs_alloc*. No entanto, o recurso escolhido será aquele com menor *flowtime* para a sua última alocação.

Primeiramente, todos os recursos acima do limiar *Selective* serão considerados, mas, semelhante à *Sel-push*, à medida que as alocações forem sendo feitas, os outros limiares são utilizados (*limiar dinâmico*). O algoritmo *sel-push-flowtime* é implementado como se segue:

O passo 2 do algoritmo 3, que contém a instrução $abs\_alloc(limiar)$, deve retornar

Algoritmo 3 Algoritmo Sel-push-flowtime

1: **para toda** tarefa w tal que $w \in FilaW$ **faça**
2: selecione todos os recursos RS com maior $abs\_alloc(limiar)$
3: selecione o recurso rs, tal que $rs \in RS$, com menor flowtime
4: calcule o novo valor de *disponibilidade* para rs
5: **se** *disponibilidade* < 0 **então**
6: descarte rs
7: volte ao passo 2
8: **senão**
9: aloque w a rs
10: calcule o novo flowtime para rs
11: **fim se**
12: **fim para**

todos os recursos que estiverem acima do valor definido no parâmetro *limiar*. Já no passo 3, para que o recurso que tiver menor *flowtime* seja selecionado, a sua disponibilidade proveniente da futura alocação é calculada e, se não for excedida, ele recebe a tarefa para a execução. O esquema desse algoritmo é mostrado na Figura 4.6, onde cada recurso possui um atributo que mede o *flowtime* de cada tarefa (e.g., 5ut ou 5 unidades de tempo).

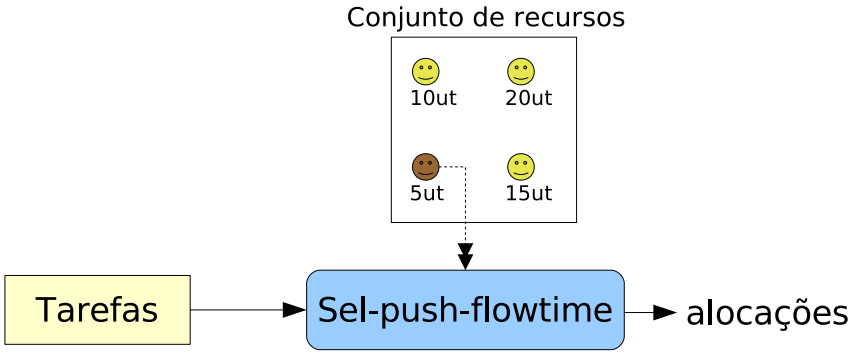

Figura 4.6: Esquema de funcionamento Sel-push-flowtime

O mecanismo *Sel-push-flowtime* é menos restritivo do que o *Sel-push* e o *Sel-push-10*. Isso ocorre porque um maior número de recursos é considerado. O *Sel-push* trabalha apenas com o recurso mais apto. O *Sel-push-10*, com o mais apto e uma vizinhança restrita. Já o *Sel-push-flowtime* percebe todos os recursos que estão

acima de um determinado limiar, e.g., o limiar seletivo (ou *Selective*).

Mesmo sendo menos restritivo, não significa que o processo em execução no momento levará menos tempo para finalizar. Como os tempos das tarefas variam muito, elas podem ser acumuladas pelos recursos. Esse acúmulo acontece quando a um mesmo recurso são atribuídas tarefas com pequenos *flowtimes*. Logo, pode acontecer que sempre o mesmo recurso seja possuidor da última alocação menos custosa. Dessa forma, as tarefas tendem a ser alocadas a um único recurso, elevando muito o tempo de execução do processo.

Capítulo 5

Experimentos

Foram realizados experimentos considerando um ambiente de *workflow* com baixa, média e alta carga de trabalho. Os resultados de *Sel-push*, *Sel-push-10* e *Sel-push-flowtime* foram comparados com os mecanismos tradicionais *Push* e *Pull*, juntamente com as variantes *Selective*, *Typical* e *Receptive*.

5.1 Considerações sobre os Experimentos

O processo-alvo das simulações contém três tarefas, como o representado na Figura 5.1, onde existem três tarefas (*tarefa 1*, *tarefa 2* e *tarefa 3*) associadas a três papéis (*papel 1*, *papel 2* e *papel 3*).

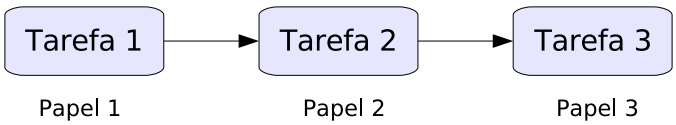

Figura 5.1: Processo de negócio utilizado para experimentos

Os mecanismos de distribuição não se preocupam com a rota e nem com a quantidade de tarefas envolvidas, o que importa é que as tarefas sejam entregues aos recursos de maneira confiável. Portanto, a simplicidade e o número de tarefas do processo utilizado (Figura 5.1) mostram-se suficientes, não prejudicando os experimentos.

Carga de trabalho (*workload*) é definida como o número de tarefas prontas para serem executadas [30] e pode ser um aspecto importante no desempenho de um sistema (*work in progress*). A carga de trabalho dos experimentos procura ambientar

o comportamento típico de uma organização e seus processos de negócio, por meio da variação da carga e do tempo de processamento das tarefas [40]. Os parâmetros que caracterizam esses ambientes e seus ajustes são apresentados na Tabela 5.1.

Tabela 5.1: Parâmetros de Simulação

Números de recursos	5
Carga média de sistema	0,2(baixa), 0,6(média), 1,0(alta)
Variabilidade de tarefas	0,2(baixa), 0,8(média), 1,4(alta)
Variabilidade de recursos	0,2(baixa), 0,8(média), 1,4(alta)

O número de recursos foi escolhido arbitrariamente como cinco. É comum em uma organização que um número pequeno de recursos seja atribuído a cada papel [40]. Desse modo, somente uma quantidade pequena de recursos foi utilizada, sendo suficiente para a simulação de um cenário de *workflow* organizacional.

A carga média de sistema mede a proporção de tarefas que ficam prontas para a execução. Essa proporção é relacionada à capacidade do recurso, portanto, controla a geração dinâmica de tarefas. O tempo de processamento de cada tarefa é fixado em dez unidades de tempo, assim, a capacidade média de um recurso por unidade de tempo é de 1/10 [12,40]. Assumindo que n recursos são apresentados no sistema, a capacidade total de recursos por unidade de tempo é $n/10$. Dessa forma, sendo l a carga média de sistema, o número de tarefas que chegam, a cada unidade de tempo, segue uma distribuição de Poisson [12] com λ:

$$\lambda = l \times (n/10) \tag{5.1}$$

A variabilidade de tarefas é responsável por gerar uma amostra populacional diversificada de tarefas. As tarefas terão tempos de processamento variando ao redor da média caracterizando assim tarefas diferentes entre si. Já a variabilidade de recursos procura diferenciar as tarefas de acordo com a influência que os recursos causam em seus tempos, significando que eles podem realizar a mesma tarefa em tempos diferentes. Estes dois parâmetros, variabilidade de tarefas e de recursos, controlam, então, o ajuste do tempo de processamento da tarefa p_{ij}. Aqui, esse tempo p_{ij} é gerado em dois passos:

- *Passo 1.* Para a tarefa i, a média de tempo de processamento $\overline{p_i}$ é gerada baseando-se numa distribuição normal com média 10 (tempo médio de toda tarefa) e desvio padrão *varTarefa* (equação 5.2).

$$\overline{p_i} \approx N(10, (10 \times varTarefa)^2) \tag{5.2}$$

  Por exemplo, considerando um processo com três tarefas e uma variabilidade de tarefas baixa (0,2), teremos um tempo de processamento variando entre 8 e 12 unidades (Figura 5.2).

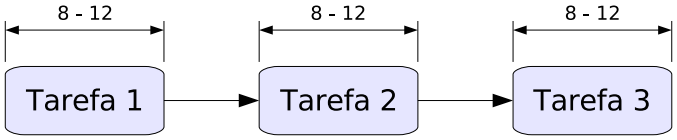

Figura 5.2: Exemplo de influência da variabilidade de tarefas na geração de tempos de processamento

- *Passo 2.* O tempo de processamento da tarefa i pelo recurso j expresso por p_{ij} e é obtido após a média $\overline{p_i}$ ter sido calculada. De maneira semelhante à *varTask*, a variabilidade de recurso *varRecurso* indica a porcentagem de desvio da média de processamento individual de uma tarefa. Assim, p_{ij} é gerado de uma distribuição normal como na equação 5.3:

$$p_{ij} \approx N(\overline{p_i}, (\overline{p_i} \times varRecurso)^2) \tag{5.3}$$

Todos os valores de médias de tempo, gerados pelo modelo, foram arredondados para números inteiros e valores negativos são assumidos como um.

A Tabela 5.1 representa um total de 27 condições experimentais ($3 \times 3 \times 3 = 27$), que é a combinação de todos os parâmetros da tabela. Para cada condição, foram geradas aleatóriamente 20 instâncias de problemas e aplicados os oito mecanismos sob estudo. No total, foram simuladas 4320 instâncias de problemas.

Implementou-se um simulador capaz de executar as instâncias de problemas sob quaisquer um dos mecanismos de distribuição apresentados [33]. Escrito em linguagem *Java*, o programa se baseia em análise de grafos e conceitos da área de *scheduling* (vide capítulo 6).

O tempo de execução total dos processos e o valor de qualidade resultante das atribuições, estão entre os dados que são retornados pelo simulador. Esses dados foram devidamente coletados e são analisados de modo a viabilizar a identificação do mecanismo de maior compromisso em qualidade e em tempo de execução. Essa análise é feita para os três tipos de carga de sistema considerados, i.e., carga baixa (20%), média (60%) e alta (100%). Cada ponto de dados nos gráficos de resultados é a média de 20 instâncias com a mesma carga de sistema e variação de tempo de processamento para cinco recursos.

Os valores de dois parâmetros da métrica de alocação absoluta, a saber, *adequação* e *disponibilidade* estão apresentados na Tabela 5.2. Essa tabela é responsável pelo mapeamento *papel/tarefa*, associando recursos (pertencentes aos papéis) às respectivas tarefas. Ela especifica a existência de três papéis, sendo um para cada tarefa. Cada papel contém cinco recursos. Portanto, tem-se dedicados cinco recursos para a cada uma das tarefas.

Tabela 5.2: Mapeamento papel/tarefa para 5 recursos

Papéis	**Recursos**	**disponibilidade**	**tarefa 1**	**tarefa 2**	**tarefa 3**
papel1	recurso1	1,0	1,0	0	0
	recurso2	1,0	0,9	0	0
	recurso3	1,0	0,8	0	0
	recurso4	1,0	0,8	0	0
	recurso5	1,0	0,8	0	0
papel2	recurso6	1,0	0	0,8	0
	recurso7	1,0	0	0,8	0
	recurso8	1,0	0	0,6	0
	recurso9	1,0	0	1,0	0
	recurso10	1,0	0	0,7	0
papel3	recurso11	1,0	0	0	0,4
	recurso12	1,0	0	0	1,0
	recurso13	1,0	0	0	0,9
	recurso14	1,0	0	0	0,8
	recurso15	1,0	0	0	0,5

A *disponibilidade* diz respeito a cada recurso individualmente. Inicialmente, todos os recursos possuem alta disponibilidade (valor 1). A *adequação* relaciona tarefas a recursos por um valor no intervalo $[0, 1]$ refletindo um ambiente contendo recursos nas três faixas de adequação. Esses valores são obtidos de maneira aleatória dentro do intervalo entre 0,4 e 1.

Considerou-se para os experimentos, que todas as tarefas teriam o mesmo nível de *urgência* (0,8) e que esse não seria incrementado. A urgência da tarefa influencia somente na ordem em que elas são enviadas aos recursos. Essa ordem, como comentado anteriormente, pode causar alterações significativas no tempo médio de execução das instâncias de processos de *workflow*. No entanto, como o intuito deste trabalho é analisar soluções de distribuição de tarefas e o seu impacto no desempenho das instâncias, essa ordem não é inerte. Outra consideração foi quanto às restrições modeladas nos processos de negócio e, conseqüentemente, o parâmetro *conformidade*. Optou-se por não especificar tais restrições, mantendo constante o grau de conformidade. Não havendo restrições, os recursos compartilham alto grau de conformidade (valor 1) para com as instâncias de processos.

5.2 Resultados

Existem dois parâmetros independentes: variabilidade de tarefa e variabilidade de recurso, cada um com três possíveis valores (baixo, médio e alto), resultando em nove combinações. Contudo, analisa-se somente cinco combinações representativas das nove possíveis, ou seja, tem-se $(varTarefa, varRecurso) \in \{$(Baixa, Baixa), (Baixa, Alta), (Media, Media), (Media, Alta), (Alta, Alta)$\}$. Essas combinações mostraram-se suficientes para o propósito dos experimentos, pois envolvem os resultados intermediários das outras combinações. Além dos mais, a inclusão de todas as combinações dificultariam a representação gráfica dos resultados e tornariam a análise mais difícil.

Os resultados são confirmados pela execução de um teste estatístico chamado *pairwise t-test* [16]. O *t-test* avalia o tamanho da diferença entre as médias de dois grupos que, segundo Grauziano e Raulin [16], é fácil de ser aplicado e usual quando existe a necessidade de testar a diferença entre dois grupos.

5.2.1 Análise sobre Tempo de Processamento

Estes experimentos procuram mostrar a variação no tempo de finalização das instâncias de processos em um sistema de *workflow*. As Figuras 5.3, 5.4 e 5.5 sumarizam o comportamento dos oito mecanismos simulados. Nota-se nas figuras que os tempos referentes ao mecanismo *Push* sobressaem-se aos demais em todas as variações de carga de sistema (baixa, média e alta).

De comportamento semelhante, também com tempos elevados, está o mecanismo *Selective*. Esses resultados (*Push* e *Selective*) ocorrem devido ao fato de serem mecanismos muito restritivos, entregando tarefas sempre a um único ou a um grupo pequeno e seleto de recursos. Desse modo, é inevitável a ocorrência de filas de espera nos recursos mais adequados, acumulando tarefas e aumentando o tempo de finalização dos processos.

Os tempos resultantes dos mecanismos *Pull* e *Typical* sofrem menos impacto, por serem muito liberais, o que significa que dificultam o acúmulo de tarefas pois atingem uma maior quantidade de recursos.

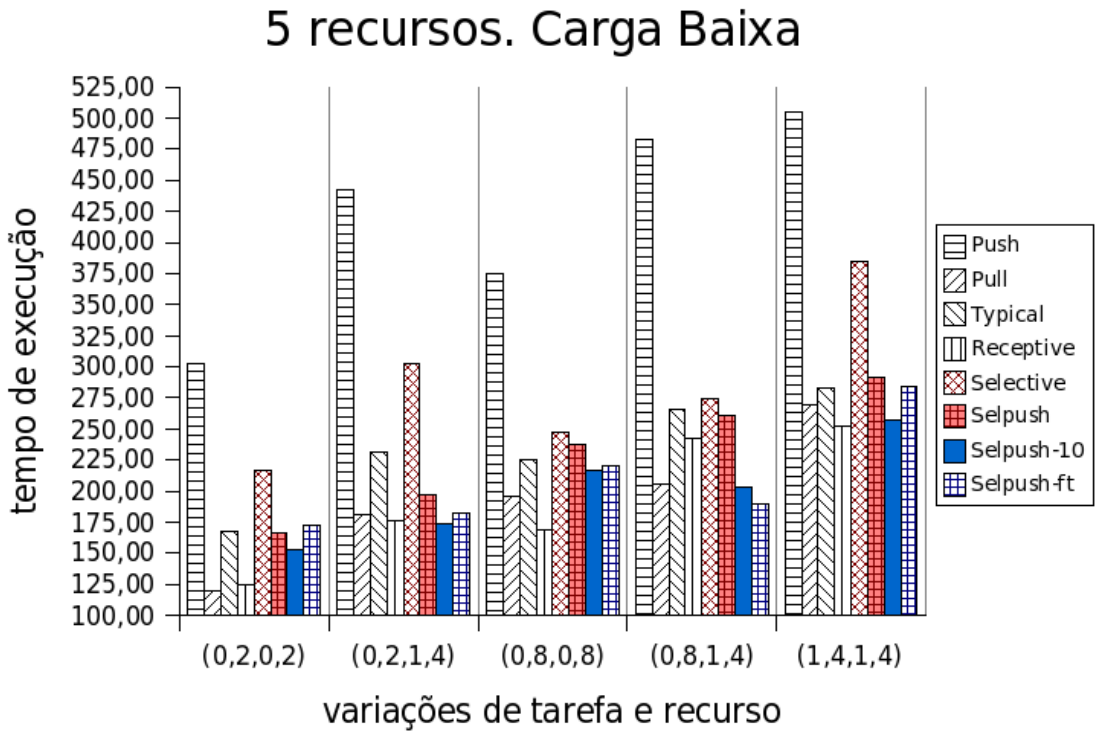

Figura 5.3: Variação de tempos com carga de sistema baixa no tempo de execução

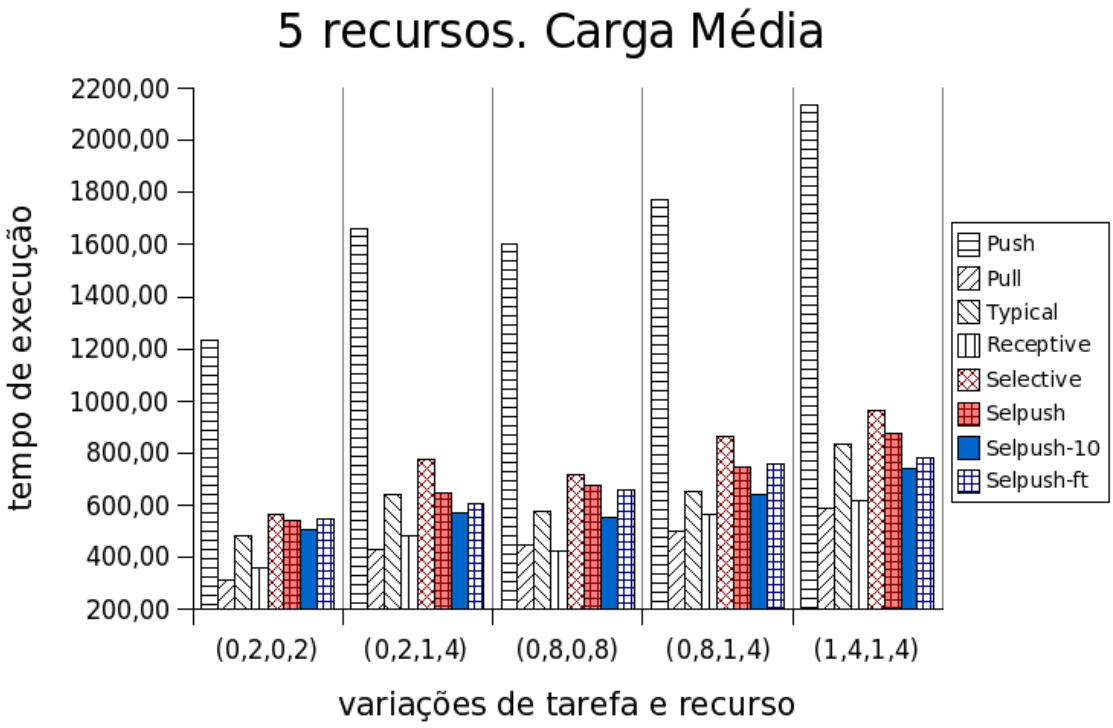

Figura 5.4: Variação de tempos com carga de sistema média no tempo de execução

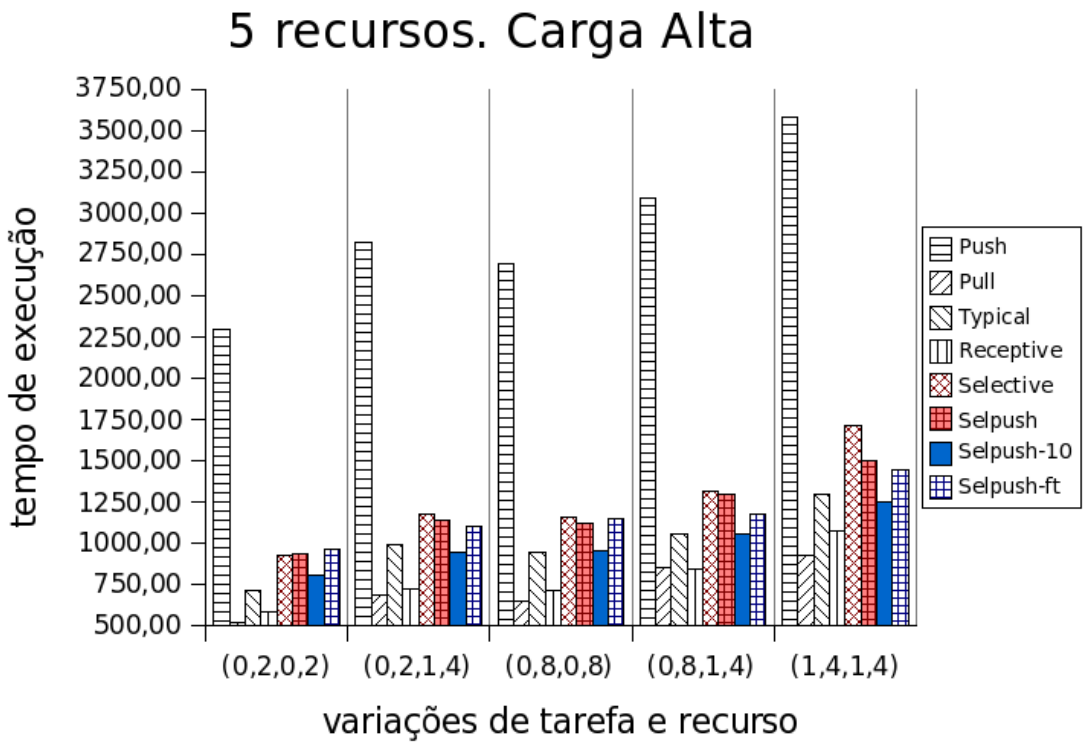

Figura 5.5: Variação de tempos com carga de sistema alta no tempo de execução

O mecanismo *Sel-push*, seguido de suas variações (*Sel-push-10* e *Sel-push-flowtime*), combinam as características de ambos mecanismos liberais e restritivos, distribuindo tarefas a um "bom custo". O bom custo proveniente dessas distribuições faz-se notório tanto em cargas leves quanto pesadas.

O gráfico da Figura 5.6 confirma que o *Sel-push* permaneceu constante entre os tempos dos mecanismos *Selective* e *Typical*, que são, segundo Kumar *et al.*, os mecanismos mais compromissados em distribuir com maior qualidade e menor tempo de execução.

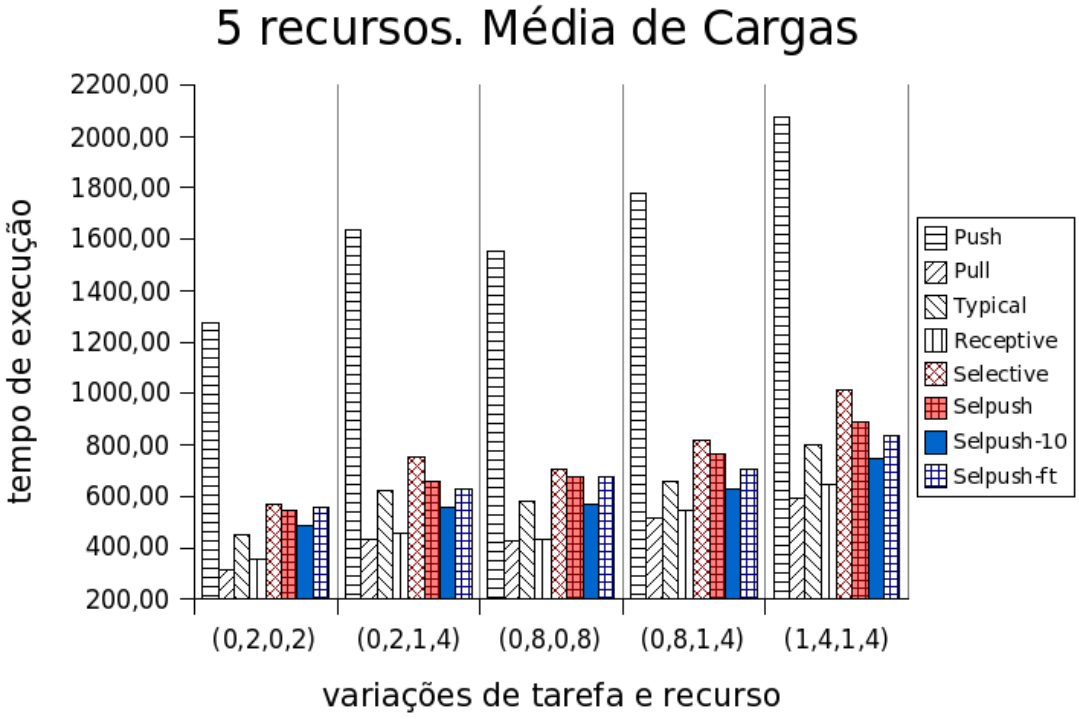

Figura 5.6: Impacto de variação de tarefa e recurso no tempo de execução com média de cargas de sistema

O *t-test* é utilizado aqui para verificar as diferenças de desempenho de cada mecanismo. Nas tabelas apresentadas a seguir, existem três símbolos: "+", "-" e "=". O símbolo "+" significa que o mecanismo que está na linha da tabela é estatisticamente melhor do que o mecanismo que está na respectiva coluna, com 95% de confiança. O símbolo "-", ao contrário, significa que o mecanismo que está na linha é estatisticamente pior que o da coluna. Já o símbolo "=" confirma, também com

95% de confiança, que não existe diferença significativa entre os correspondentes linha e coluna.

Tabela 5.3: Teste estatístico para resultados de tempo de execução

Mecanismos	Push	Pull	Typical	Receptive	Selective
Sel-push	+	-	-	-	+
Sel-push-10	+	-	=	-	+
Sel-push-flowtime	+	-	-	-	+

A Tabela 5.3 apresenta o resultado da aplicação do teste estatístico para os mecanismos "*Sel-pushes*" em comparação com o mecanismo *Push* e baseados em *Pull*. Em geral, os mecanismos propostos podem ser considerados piores que os mecanismos *Pull*, *Receptive* e *Typical*, com 95% de confiança. No entanto, são estatisticamente melhores que o *Push* e o *Selective*. Dos três mecanismos propostos, o *Sel-push-10* é o melhor colocado, pois sendo semelhante ao mecanismo *Typical* é superior ao *Sel-push* e ao *Sel-push-flowtime*, como pode ser visto na Tabela 5.4. Essa tabela afirma o que é percebido também na Tabela 5.3, que é a semelhança de comportamento entre o *Sel-push* e o *Sel-push-flowtime*. Portanto, comprovam-se os ganhos da solução proposta em termos de tempo de execução, pois está claro que os resultados ficaram entre os mecanismos mais restritivos e os mais liberais.

Tabela 5.4: Comparação estatística de tempo entre mecanismos propostos

Mecanismos	Sel-push-10	Sel-push-ft
Sel-push	-	=
Sel-push-10		+

5.2.2 Análise sobre Qualidade

A qualidade, como medida de sucesso da distribuição de tarefas em um sistema de *workflow*, é tão importante quanto o tempo de execução das instâncias de processos. Essa qualidade das instâncias simuladas é aqui verificada.

As Figuras 5.7, 5.8 e 5.9 mostram os gráficos provenientes do processo de simulação. Os gráficos são compostos tendo em vista a qualidade das execuções das instâncias de processos, submetidos a cargas baixa, média e alta de sistema.

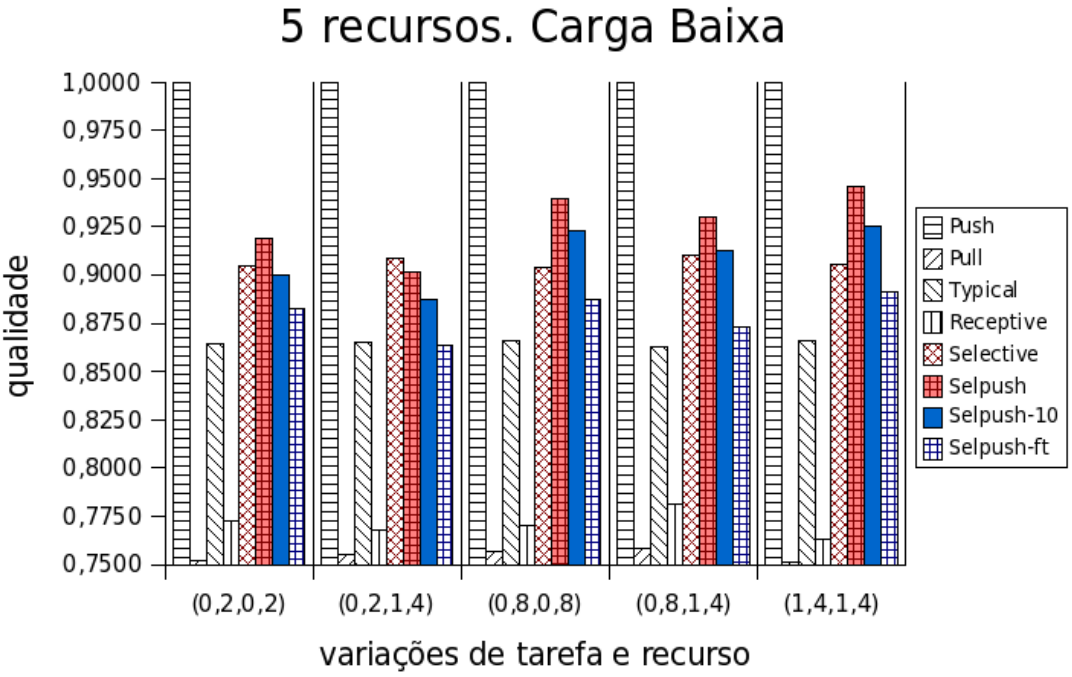

Figura 5.7: Variação de tempos com carga de sistema baixa na qualidade

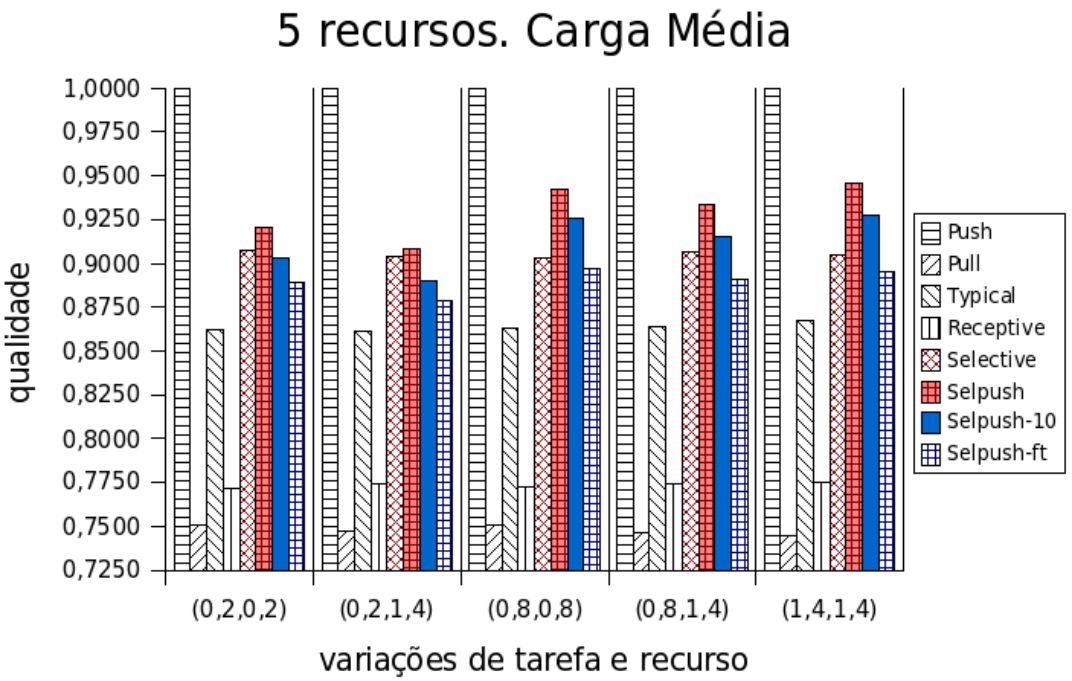

Figura 5.8: Variação de tempos com carga de sistema média na qualidade

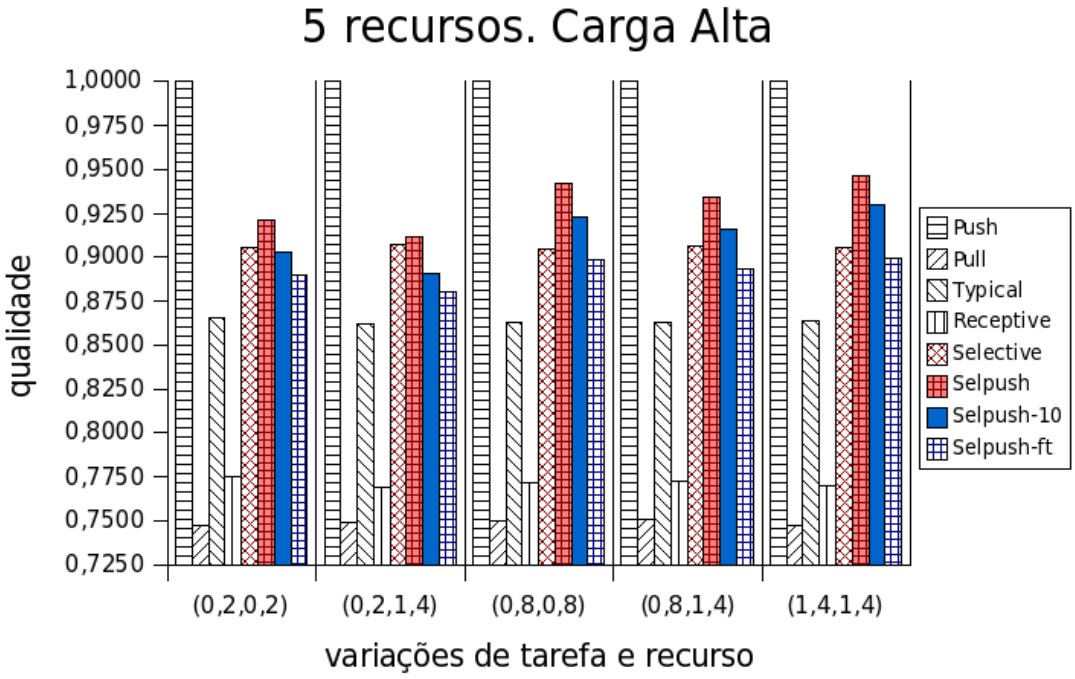

Figura 5.9: Variação de tempos com carga de sistema média na qualidade

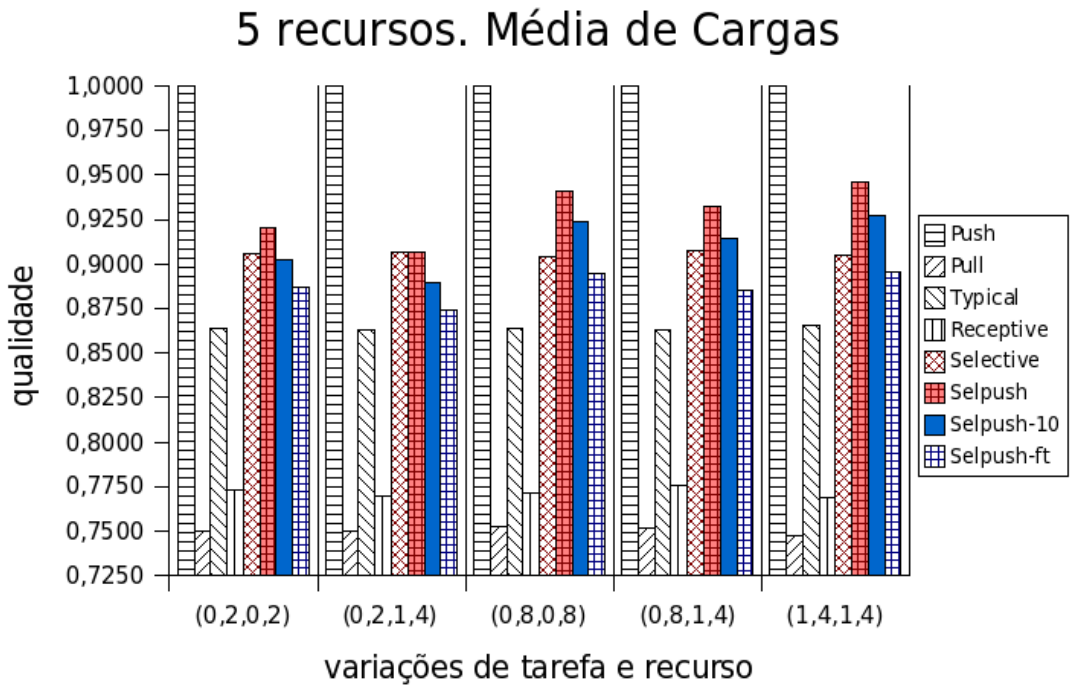

Figura 5.10: Impacto de variação de tarefa e recurso na qualidade com média de cargas de sistema

O mecanismo *Sel-push* permanece apresentando os bons resultados alcançados anteriormente em tempo de execução. Porém, supera o mecanismo *Selective*, sendo inferior somente ao *Push*, que mantém a qualidade constantemente alta. O *Sel-push-10* é inferior ao *Sel-push* em variações de tempo de processamento baixas e apresenta índices de qualidade de execução inferior ao *Selective*. No entanto, esses índices aumentam e ultrapassam o *Selective* à medida que as variações tempo de processamento das tarefas aumentam (de média para alta). Essas observações são melhor percebidas no gráfico da Figura 5.10 que contém a média das três variações de carga de sistema.

A Tabela 5.5 mostra o resultado da aplicação do teste estatístico para as médias de qualidade das instâncias simuladas. Nela, o mecanismo *Sel-push* é estatisticamente melhor que todos os outros mecanismos, exceto o *Push*. O *Sel-push-10* também obteve boas médias, sendo inferior ao mecanismo *Push*, similar ao *Selective* e superior ao restante. De comportamento mais liberal, o *Sel-push-flowtime* manteve-se abaixo do *Push* e *Selective*, porém, acima do *Pull*, *Typical* e *Receptive*.

Tabela 5.5: Teste estatístico para resultados de qualidade

Mecanismos	Push	Pull	Typical	Receptive	Selective
Sel-push	-	+	+	+	+
Sel-push-10	-	+	+	+	=
Sel-push-flowtime	-	+	+	+	-

A Tabela 5.6 mostra a comparação estatística feita entre os mecanismos propostos; o *Sel-push-10* obteve melhor qualidade que o *Sel-push-flowtime*, mas, estes dois mecanismos foram comprovadamente piores que o *Sel-push* em qualidade. Percebe-se, portanto, que o mecanismo *Sel-push* e derivados possuem um maior compromisso de qualidade e tempo na distribuição de tarefas. Esses mecanismos, além de manterem um tempo médio de execução próximo ao dos liberais, garantem um bom nível de qualidade, igualando-se aos mais restritivos.

Tabela 5.6: Comparação estatística de qualidade entre mecanismos propostos

Mecanismos	Sel-push-10	Sel-push-ft
Sel-push	+	+
Sel-push-10		+

5.3 Sumário de Resultados

As simulações dos problemas de instâncias serviram para mostrar a eficácia dos mecanismos propostos neste trabalho, no que diz respeito ao comprometimento de balanço entre qualidade e tempo de execução.

Entre os valores expressos nos gráficos da seção anterior, percebe-se um desequilíbrio entre os tempos de execução e os níveis de qualidade resultante dos mecanismos tradicionais. Por exemplo, o mecanismo *Push* que, atingindo os piores índices de tempo, é o mecanismo que obteve maior qualificação. De forma oposta, o mecanismo *Pull*, sendo o mais "rápido" é o pior em qualidade. Segundo Kumar *et al.* (em [19]) o balanceamento de desempenho e segurança é melhor apresentado pelo mecanismo *Selective*. No entanto, como visto na seção 5.2.1, o mecanismo *Sel-push* e derivados resultam em tempos mais amenos que ele. *Sel-push*, em média, também é melhor que o *Selective* em qualidade, conforme o gráfico da Figura 5.10.

Verfica-se com os gráficos, que os mecanismos se comportam de maneira mais ou menos constante quanto à variação de tempo de processamento das tarefas. Percebe-se que é próprio dos mecanismos mais restritivos elevarem o tempo de finalização dos processos, devido ao acúmulo de tarefas. Pelo mesmo motivo, aumentam a qualidade média dos processos executados. De maneira oposta, enquanto os mecanismos mais liberais resultam em menos tempo de execução, por terem mais recursos recebendo tarefas, na mesma medida favorecem a queda de qualidade.

Os gráficos das Figuras 5.11 e 5.12 ilustram o comportamento dos mecanismos propostos, sob uma média de cargas de sistema e variação de tempo de processamento das tarefas. A curva média, tanto no gráfico de tempo de execução quanto de qualidade, resume a média dos resultados do mecanismo *Push* e dos baseados em *Pull*. Conforme os gráficos, os mecanismos *Sel-push*, *Sel-push-10* e *Sel-push-flowtime* mantiveram os resultados abaixo da curva média de tempo de execução e acima da curva média de qualidade.

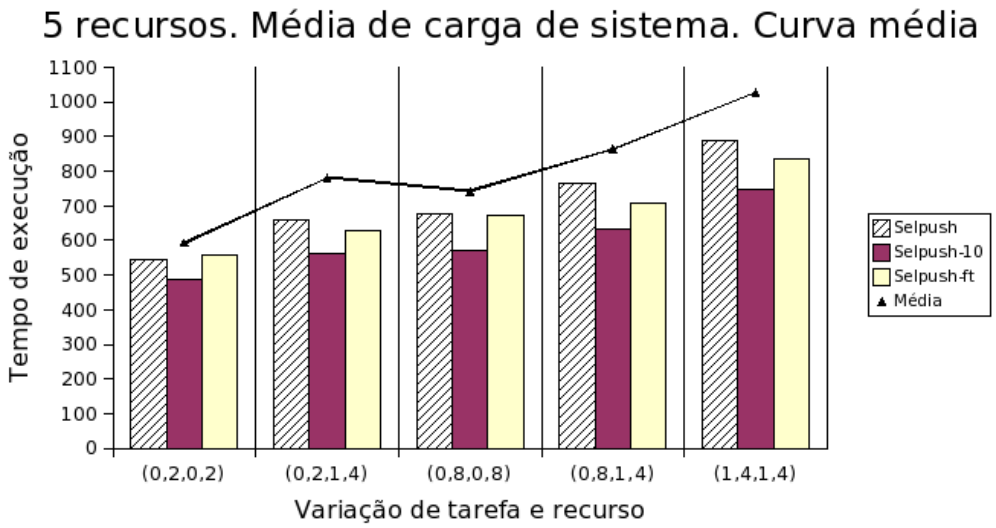

Figura 5.11: Comparação entre os mecanismos e a média de tempo

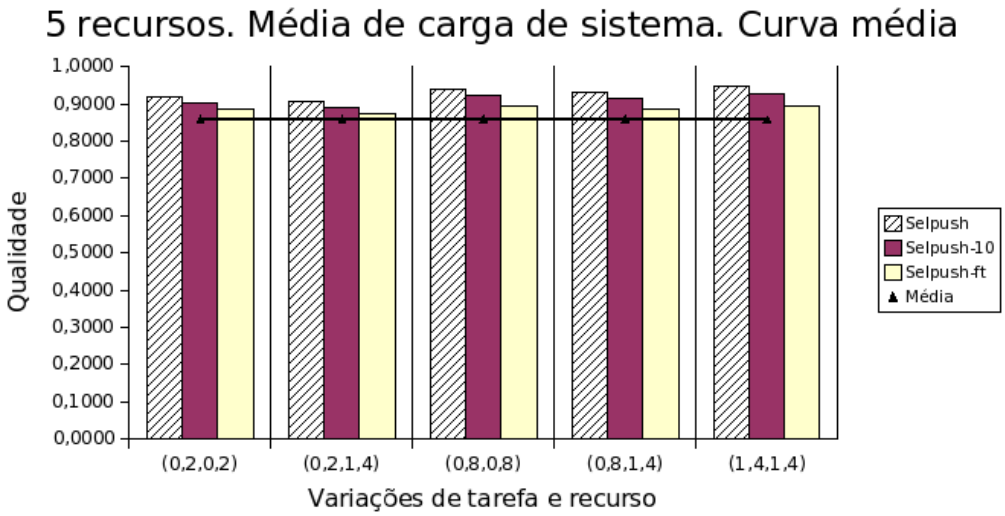

Figura 5.12: Comparação entre os mecanismos e a média de qualidade

Comparada com mecanismos mais comuns em sistemas de *workflow* como, por exemplo, o *Push* e o *Pull* e com os descritos em [19], a proposta apresenta uma maior garantia de tempo e qualidade. Essa garantia mostra-se satisfatória no caso de grandes quantidades de instâncias de processos que concorrem aos mesmos recursos. Essa concorrência por recursos é observada com a ajuda de cenários empíricos onde existem poucos recursos com diferentes valores de adequação para as tarefas.

Os resultados da execução dos cenários mostram que, além de combinar características dos mecanismos *Pull* e *Push*, o *Sel-push* e derivados apresentam valores médios de qualidade e tempo melhores que os resultados dos mecanismos *Recep-*

tive, *Typical* e *Selective*. Os resultados dos testes estatísticos, aplicados nas seções anteriores, são resumidos a seguir:

- Tempo de execução:
  1. O mecanismo *Push* resulta no pior tempo de execução;
  2. O mecanismo *Pull* resulta no melhor tempo de execução;
  3. O mecanismo *Selective* é o segundo pior, após o mecanismo *Push*;
  4. O mecanismo *Sel-push* resulta em tempos mais rápidos que o *Selective*;
  5. O mecanismo *Sel-push-flowtime* se assemelha ao *Sel-push*;
  6. Os mecanismos *Pull*, *Receptive* e *Typical* são os que resultam em tempos mais rápidos, nessa ordem;
  7. O mecanismo *Sel-push-10* é melhor que o *Sel-push* e iguala-se ao *Typical*.
- Qualidade:
  1. O mecanismo *Push* é o que possui os melhores índices de qualidade;
  2. O mecanismo *Pull* é o que possui os mais baixos índices;
  3. O mecanismo *Selective* é melhor que o *Pull*, *Receptive*, *Typical* e *Sel-push-flowtime*, e se assemelha ao *Sel-push-10*;
  4. O mecanismo *Sel-push-10* é melhor que o *Sel-push-flowtime*;
  5. O mecanismo *Sel-push* é o segundo melhor em qualidade após o *Push*.

A garantia de qualidade de execução de um processo, quando submetido a um mecanismo baseado em aptidão, indica que os recursos executores possivelmente são especialistas. Espera-se que recursos especialistas, por serem adequados ao trabalho, errem menos e acelerem o fluxo de tarefas. Logo, o mecanismo *Sel-push* e derivados podem contribuir para reduzir os índices de re-trabalho, filas de espera de tarefas e, conseqüentemente, aumentar o desempenho na execução das instâncias de processos.

De maneira mais específica, o *Sel-push* e o *Sel-push-10* são os mais indicados para a distribuição de tarefas. O *Sel-push-flowtime* sendo mais liberal que os outros, perde muito em qualidade. Isso acontece principalmente nos instantes iniciais do processo de distribuição onde, no momento em que os recursos ainda não receberam nenhuma tarefa, os atributos individuais que indicam o *flowtime* das tarefas alocadas estão

zerados. Nesse momento, todos os recursos podem vir a receber tarefas, pois o recurso mais "adequado" é aquele que está com *flowtime* igual a zero.

O *Sel-push* e o *Sel-push-10* são os mais indicados aqui resultando em maior qualidade e em menor tempo, respectivamente. Contudo, como a questão de tempo é muito dependente do contexto do problema, do ambiente e das condições dos recursos, podem ocorrer variações em seu valor para mais ou para menos. Dessa forma, o *Sel-push* é preferível, pois a qualidade, além de ser quantificada por métricas bem definidas, é o objetivo de todo processo organizacional.

5.4 Outros Cenários

Nesta seção são apresentados outros cenários experimentados com os mecanismos de distribuição. Estes experimentos visam dar condições de análise para esses mecanismos, com um número menor de recursos (neste caso, três) e com recursos altamente adequados. O número dessas simulações somaram um total de 8640 instâncias de problemas, sendo suficientes para verificar um comportamento regular dos mecanismos de distribuição.

5.4.1 Experimentos com 3 Recursos

Apresenta-se aqui a análise da simulação realizada com somente três recursos disponíveis ao sistema. Essa simulação foi especificada sob as mesmas condições de ambiente e cenário utilizados pelo experimento do início deste capítulo.

Os valores de adequação foram gerados de maneira aleatória semelhante a Tabela 5.2. Assume-se que os recursos envolvidos possuem alta disponibilidade (valor igual a 1). Esses valores foram assumidos conforme a Tabela 5.7. Nessa tabela, a *tarefa 1* deve ser executada pelo *papel1*, a *tarefa 2* pelo *papel2* e a *tarefa 3* pode ser executada pelo *papel1* ou pelo *papel3*.

As Tabelas 5.8 e 5.9 contém o teste estatístico em que cada coluna está no formato *símbolo/símbolo* representando o *tempo/qualidade* das execuções. Por exemplo, conforme a Tabela 5.8, sabe-se que o mecanismo *Sel-push* é melhor em tempo e pior em qualidade em relação ao mecanismo *Push*, pois na respectiva linha e coluna encontra-se o símbolo $+/-$.

Tabela 5.7: Mapeamento papel/tarefa para 3 recursos

Papéis	**Recursos**	**disponibilidade**	**tarefa 1**	**tarefa 2**	**tarefa 3**
papel1	recurso1	1,0	1,0	0	0,9
	recurso2	1,0	0,9	0	1,0
	recurso3	1,0	0,8	0	0,7
papel2	recurso4	1,0	0	0,6	0
	recurso5	1,0	0	0,7	0
	recurso6	1,0	0	1,0	0
papel3	recurso7	1,0	0	0	0,7
	recurso8	1,0	0	0	0,5
	recurso9	1,0	0	0	0,5

Tabela 5.8: Teste estatístico para resultados de tempo e qualidade com 3 recursos

Mecanismos	Push	Pull	Typical	Receptive	Selective
Sel-push	+/-	-/+	=/+	-/+	+/=
Sel-push-10	+/-	-/+	=/+	-/+	+/-
Sel-push-flowtime	+/-	-/+	=/=	-/+	+/-

Tabela 5.9: Comparação estatística de tempo e qualidade entre mecanismos propostos com 3 recursos

Mecanismos	Sel-push-10	Sel-push-ft
Sel-push	-/+	=/+
Sel-push-10		-/-

Verifica-se com o teste, que o mecanismo *Sel-push* também apresenta os melhores resultados considerando somente três recursos. Esse mecanismo, na média de variação de tempo de processamento e cargas de sistema, igualou-se ao *Typical* em tempo de execução e ao *Selective* em qualidade. O mesmo não ocorre com o *Sel-push-10* e o *Sel-push-flowtime* que, embora se assemelhem ao *Typical* em tempo de execução, são piores que o *Selective* em qualidade. O *Sel-push* apresenta índices de qualidade e tempos maiores que o *Sel-push-10* e consegue também maior qualidade em relação ao *Sel-push-flowtime*, porém com tempos semelhantes.

5.4.2 Experimentos com 5 Recursos com Máximo de Adequação

Simulou-se também e sob as mesmas condições de ambiente anteriores, um cenário contendo cinco recursos. No entanto, esses recursos são considerados aqui como altamente adequados. Os valores foram distribuídos entre os recursos de maneira uniforme acima do limiar seletivo (maior ou igual a 0,6). A Tabela 5.10 mostra essa configuração.

Tabela 5.10: Mapeamento papel/tarefa para 5 recursos com valores de adequação seletivos

Papéis	**Recursos**	**disponibilidade**	**tarefa 1**	**tarefa 2**	**tarefa 3**
papel1	recurso1	1,0	1,0	0	0
	recurso2	1,0	0,9	0	0
	recurso3	1,0	0,8	0	0
	recurso4	1,0	0,7	0	0
	recurso5	1,0	0,6	0	0
papel2	recurso6	1,0	0	1,0	0
	recurso7	1,0	0	0,9	0
	recurso8	1,0	0	0,8	0
	recurso9	1,0	0	0,7	0
	recurso10	1,0	0	0,6	0
papel3	recurso11	1,0	0	0	1,0
	recurso12	1,0	0	0	0,9
	recurso13	1,0	0	0	0,8
	recurso14	1,0	0	0	0,7
	recurso15	1,0	0	0	0,6

As Tabelas 5.11 e 5.12 sumarizam o resultado da aplicação do teste estatístico a este cenário simulado. O *Sel-push* é inferior em qualidade somente ao restritivo *Push*. Em tempo de execução, o *Sel-push* é equivalente ao *Selective*. *Sel-push-10* supera o *Selective* em tempo de execução e é semelhante em qualidade. Já o *Sel-push-flowtime* resulta em tempos iguais ao *Selective*, mas é superior em qualidade.

Tabela 5.11: Teste estatístico para resultados de tempo e qualidade com 5 recursos seletivos

Mecanismos	Push	Pull	Typical	Receptive	Selective
Sel-push	+/-	-/+	-/+	-/+	=/+
Sel-push-10	+/-	-/+	=/+	-/+	+/=
Sel-push-flowtime	+/-	-/+	-/+	-/+	=/-

Tabela 5.12: Comparação estatística de tempo e qualidade entre mecanismos propostos com 5 recursos seletivos

Mecanismos	Sel-push-10	Sel-push-ft
Sel-push	-/+	-/+
Sel-push-10		+/+

Capítulo 6

Ambiente de Experimentação

6.1 Ferramenta de Simulação

Simulações são geralmente utilizadas quando a complexidade dos sistemas sendo modelados vai além de modelos estáticos ou outras técnicas representativas [18].

Workflows são sistemas estocásticos onde as limitações provenientes das incertezas e as implicações dos resultados devem ser entendidos e avaliados. Nesse sentido, a simulação fornece um mecanismo flexível e usual para capturar incertezas relacionadas à sistemas complexos e entender a natureza dinâmica dos processos.

A ferramenta de simulação utilizada neste trabalho foi o **Lambari** [33]. O **Lambari** é um motor de *workflow* implementado em linguagem *Java*. Ele dá suporte a execução de vários processos simultâneos com recursos conectados remotamente e utilizando quaisquer regras de distribuição de tarefas descritas em [19]. Este ambiente também é capaz de simular estas execuções e alocar os recursos mais adequados às tarefas, baseando-se na disponibilidade de cada recurso para sua realização.

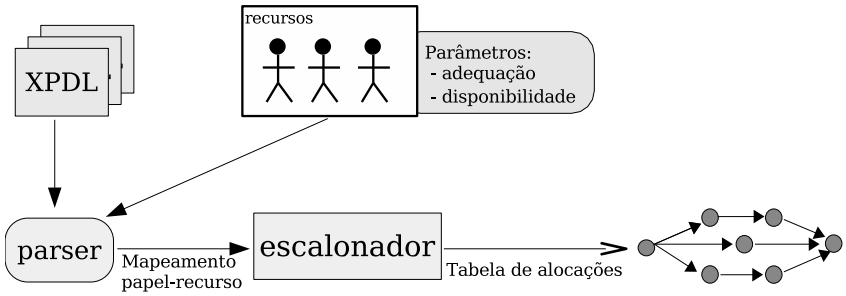

Figura 6.1: Arquitetura do simulador

A Figura 6.1 mostra a arquitetura do simulador **Lambari**. Nessa figura estão representados o *parser* [32] e o *escalonador* que são responsáveis, respectivamente, por analisar o arquivo XPDL [1] e os parâmetros provenientes dos recursos, e fazer a distribuição das tarefas. O XPDL utilizado pelo Lambari constitui-se de uma definição mais simplificada de um processo de negócio, contendo somente a especificação de tarefas e transições. Um exemplo desse arquivo XML (e da gramática DTD) utilizado nos experimentos é listado no Apêndice A.

Os parâmetros de adequação e disponibilidade dos recursos são recebidos pelo *parser* por meio de uma tabela que engloba todas as tarefas do processo contido no XPDL. A Figura 6.2 ilustra essa tabela no **Lambari**. O *parser* faz o mapeamento entre recursos e tarefas de acordo com os papéis associados a cada um deles.

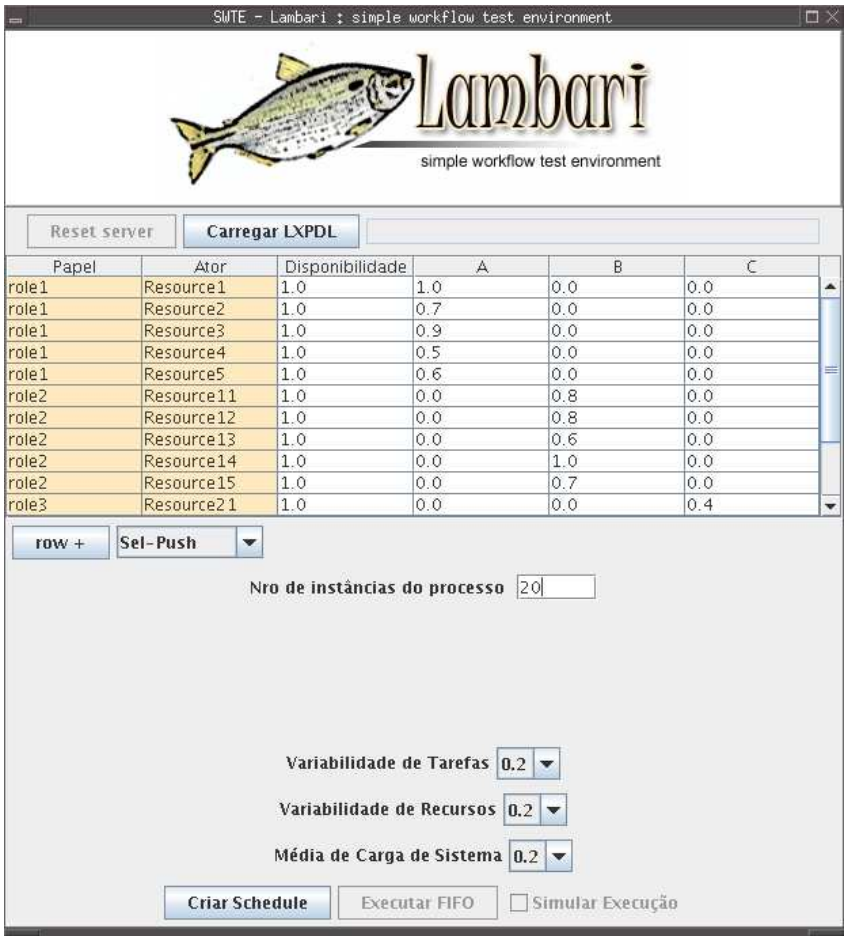

Figura 6.2: Tabela de adequação e disponibilidade do simulador Lambari

[1] XML Process Definition Language [37]

A Figura 6.3 apresenta com mais detalhes como é implementado o que se chama de "escalonador", mostrado na arquitetura do Lambari na Figura 6.1. Esse escalonador possui dois submódulos: o motor de *workflow* e o submódulo de distribuição de tarefas.

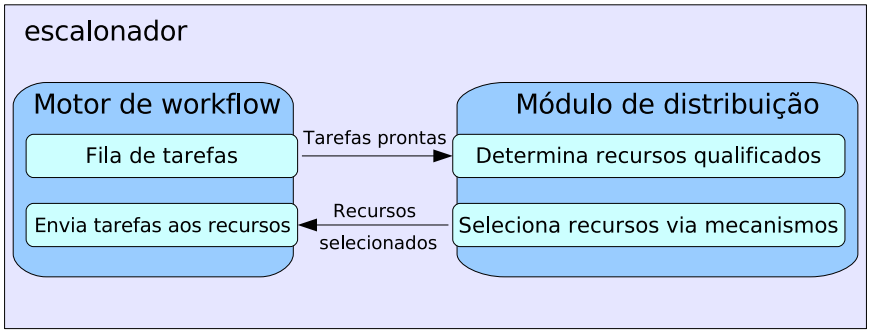

Figura 6.3: Detalhes do escalonador

O motor de *workflow*, do ponto de vista do simulador, é responsável por gerenciar a fila de tarefas e suas rotas especificadas na definição do processo. É o motor quem recebe a tabela com os parâmetros de adequação, disponibilidade, urgência das tarefas e possíveis restrições contidos na definição do processo. Assim que o motor identifica que existem tarefas prontas para serem executadas, o módulo de distribuição de tarefas é acionado.

O módulo de distribuição de tarefas, sugerido por Stohr e Zhao [28], foi desenvolvido como uma extensão à arquitetura-padrão de sistema de *workflow*. Sob essa arquitetura estendida, o motor de *workflow* pode invocar o módulo de distribuição quando há necessidade de distribuir tarefas. Dependendo da natureza do mecanismo (*Push* ou *Pull*), as tarefas se tornam visíveis aos recursos envolvidos no processo. Quando esse módulo é acionado pelo motor, um conjunto de recursos qualificados é reunido. A partir daí, os algoritmos que implementam os mecanismos de distribuição são executados e retornam para o motor os recursos selecionados. O motor, então, baseado nesses recursos, percorre a definição de processo casando recursos às tarefas e montando um grafo de alocações. Esse grafo resultante possui o formato de um *job shop* [30] como mostrado na Figura 6.4.

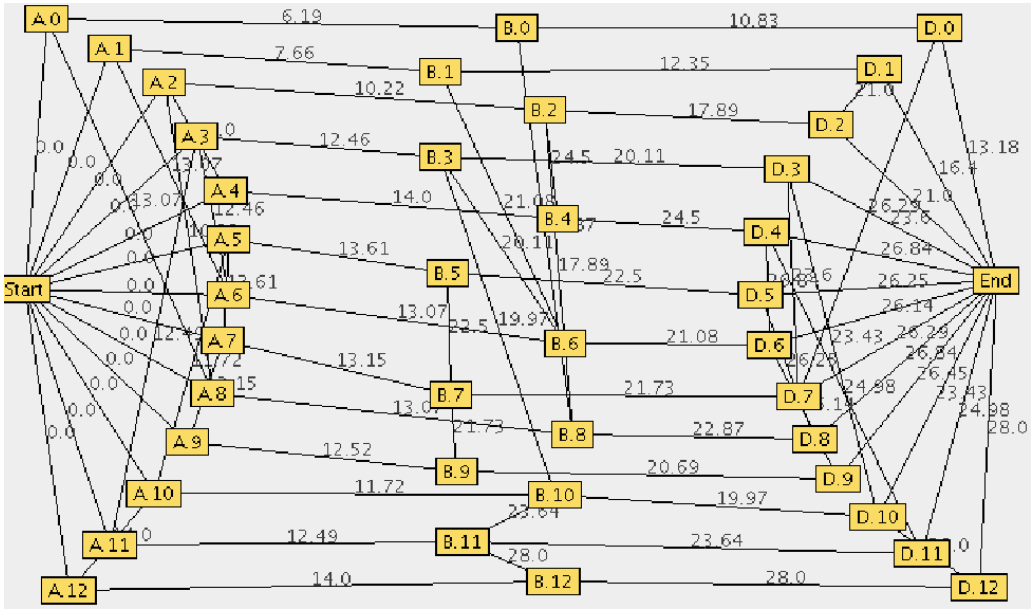

Figura 6.4: Grafo de escalonamento do Lambari

O grafo de escalonamento da Figura 6.4 mostra o resultado da simulação de treze instâncias de processo de *workflow* numerados de zero a doze. Cada instância possui três tarefas, a saber, a tarefa *A*, *B* e *C*. Assim, cada nó do grafo possui etiquetas como *A.0*, indicando, respectivamente, o nome da tarefa e o número da instância executada. Juntamente com o grafo de escalonamento, são retornados os valores de tempo de execução de cada instância e o grau de qualidade alcançado pelo mecanismo de distribuição.

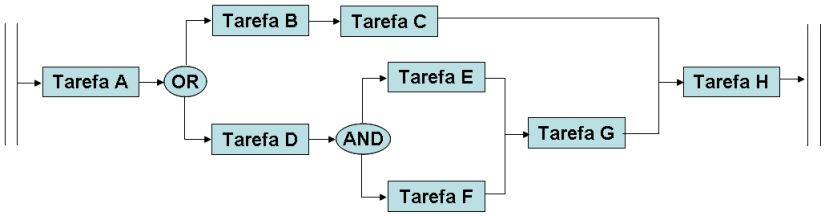

Figura 6.5: Cenário

Simular processos de *workflow*, como já mencionado, é uma tarefa complicada pois se trata de um ambiente estocástico. Mas como as atribuições de recursos à papéis são feitas de maneira a verificar o tempo de execução total, o simulador analisa e realiza as atribuições somente para as tarefas que se encontram no caminho de maior custo dos processos.

A Figura 6.6 mostra um exemplo da simulação utilizando o cenário da Figura

6.5 e os recursos da Tabela 6.1 para a execução de 3 processos simultâneos. As alocações provenientes dessa execução compõem a Tabela 6.2.

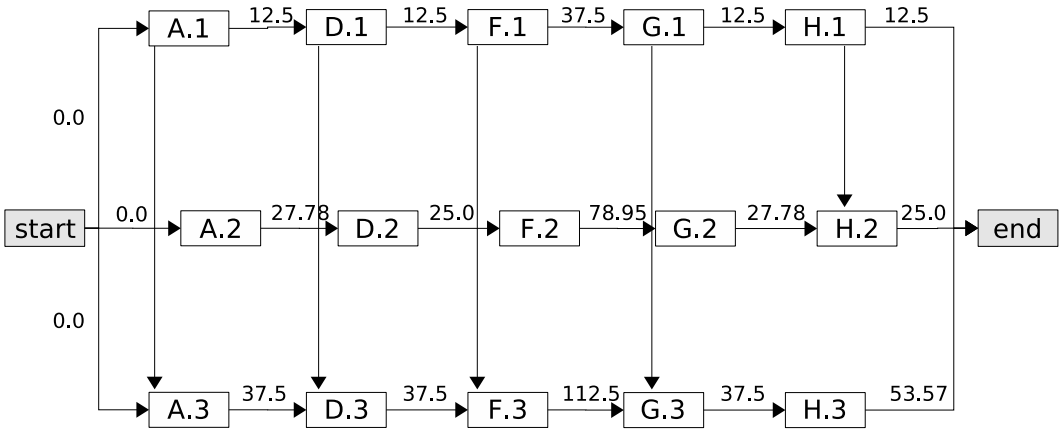

Figura 6.6: Representação das alocações

O escalonador do **Lambari** utiliza os dados da Tabela 6.1 para gerar a tabela de alocações (Tabela 6.2). Percebe-se que somente as tarefas *A*, *D*, *F*, *G* e *H* são consideradas. Essas tarefas compõem o caminho de maior custo das instâncias simuladas. Para essas alocações têm-se o gráfico de Gantt da Figura 6.7. Com essa simulação é possível, por exemplo, verificar que o tempo total exigido para o término das instâncias foi 291,07 (unidades de tempo).

Tabela 6.1: Valores de adequação de cada recurso com as tarefas

Papéis	**Recursos**	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**
Ro1	Rec1	1,0	0	0	0	1,0	0	0	0
	Rec2	0,9	0	0	0	0,95	0	0	0
Ro2	Rec3	0	1,0	0	0	0	0,95	0	0
	Rec4	0	0,95	0	0	0	1,0	0	0
Ro3	Rec5	0	0	0,9	1,0	0	0	0,9	0
	Rec6	0	0	1,0	1,0	0	0	1,0	0
Ro4	Rec7	0	0	0	0	0	0	0	1,0
	Rec8	0	0	0	0	0	0	0	0,7

Para agilizar a realização dos experimentos, foi construída uma versão do simulador que executasse todas as combinações de instâncias de problemas. Essa

versão, ao final de cada conjunto de instâncias, apresenta a média dos tempos e a qualidade de cada conjunto em uma tabela, possibilitando a confecção de gráficos demonstrativos.

Tabela 6.2: Tabela de alocações para o cenário simulado

Processos	**A**	**D**	**F**	**G**	**H**
1	Rec1	Rec5	Rec4	Rec6	Rec7
2	Rec2	Rec6	Rec3	Rec5	Rec7
3	Rec1	Rec5	Rec4	Rec6	Rec8

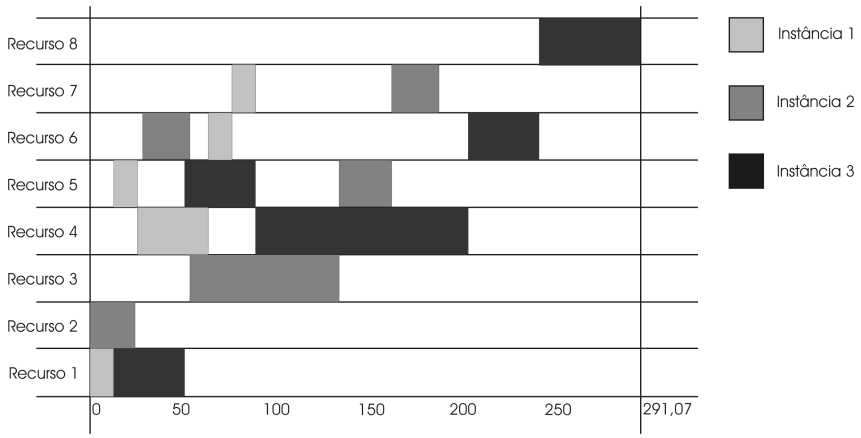

Figura 6.7: Gráfico de Gantt para as alocações da Tabela 6.2

A ferramenta de simulação, *a priori*, não leva em consideração a possível rejeição das tarefas pelos recursos. Na implementação avaliada, o sistema aloca o recurso baseando-se na sua adequação e disponibilidade. Nos casos de rejeição das tarefas, o sistema teria que reavaliar todos os recursos disponíveis, tentando oferecer outras tarefas a esses recursos. Para que isso fosse possível, o sistema deveria, antes de fazer as alocações, simplesmente oferecer as tarefas aos recursos e aguardar as respostas.

Capítulo 7

Conclusão e Trabalhos Futuros

A distribuição de tarefas é uma das funções críticas de um sistemas de workflow. Geralmente, essa distribuição acontece pelos mecanismos *Push* e *Pull* [31]. No entanto, o nível de competência dos recursos para a realização das tarefas de *workflow* é uma importante medida que não é considerada por esses mecanismos tradicionais.

Este trabalho mostrou que a abordagem *Push* associada à aptidão dos recursos é mais eficiente que a abordagem *Pull*. Três variantes *Push* foram implementados, o *Sel-push*, o *Sel-push-10* e o *Sel-push-flowtime*. Entre esses o *Sel-push* proporcionou um melhor balanço entre qualidade e tempo de execução médios.

Independentemente da carga de trabalho aplicada, soluções baseadas em *Push* e na aptidão dos recursos apresentam desempenho superior aos mecanismos tradicionais e aos baseados em *Pull* propostos por Kumar *et al.* [19]. Considerando o trabalho desses autores, *Sel-push* foi superior, em média, 13% em tempo de execução e 8% em qualidade.

Este trabalho também gerou a ferramenta de simulação e motor de *workflow* Lambari, que permite simular a execução de processos de *workflow*. Essa ferramenta proporciona mais agilidade, principalmente, na análise de comportamento de ambientes onde os recursos estão sujeitos grandes variações de cargas de trabalho.

7.1 Trabalhos Futuros

Ao longo do tempo, identificou-se vários pontos que de uma forma ou de outra estendem o presente trabalho:

- A atribuição de tarefas, é geralmente definida por um usuário do sistema que avalia o grau de adequação de cada recurso para as tarefas do *workflow* de acordo com sua experiência. Carece portanto, de formalismos para avaliar essa aptidão dos recursos de maneira mais confiável por meios automáticos ou mesmo semi-automáticos.

- Grandes organizações executam vários processos por dia e muitas vezes várias instâncias do mesmo processo, gerando uma sobrecarga administrativa no sistema [15, 42]. Seria desejável, então, que o sistema de *workflow* se "retro-alimentasse", baseando-se nas execuções passadas. Assim, os parâmetros na tabela *papel/tarefa* seriam refinados a cada instância de processo finalizada.

- Os resultados deste trabalho indicam que as soluções baseadas em *Push* são as mais adequadas. Entretanto, soluções baseadas em *Pull* mais "amigáveis" [40]. De fato, são mais atraentes as políticas de distribuição nas quais os próprios usuários escolhem as tarefas a serem realizadas. De maneira oposta, uma "super-especialização" pode ocorrer quando soluções *Push* são aplicadas rigidamente. Neste caso, sob os mecanismos baseados em aptidão, um recurso que é eficiente em um tipo de tarefa tende a receber tarefas do mesmo tipo. Devido a esse fato, talvez uma solução híbrida conciliasse o máximo desempenho e qualidade do *Push* à simpatia dos mecanismos *Pull*. Contudo, o impacto da implementação de uma solução "*Sel-push-pull*" e uma análise de seus benefícios fica como pesquisa futura.

- No Capítulo 3, foram expostos os conceitos e alguns problemas relacionados à distribuição de tarefas de *workflow*. Como mencionado, existe uma preocupação quanto à ordem com que as tarefas são selecionadas da fila de tarefas prontas. No entanto, essa ordem e seu impacto, não foram considerados neste trabalho. Nesse sentido, poderia ser verificada a influência do parâmetro *urgência*, dando prioridade para algumas tarefas no tempo de execução das instâncias de processo. Outra sugestão é o uso de alguma *dispatching rule* como FIFO, SPT ou EDD, encontrados na literatura de *scheduling* [6, 30]. Além da ordem de pré-distribuição das tarefas, um estudo da influência de uma ordenação pós-distribuição sobre itens em listas de trabalho também não

foi considerado nesta pesquisa.

- Os processos de negócio executados por grandes organizações podem muitas vezes sofrer alterações emergenciais. Tais alterações, como, por exemplo, curtos prazos de finalização ou entrega urgente de produtos, podem acarretar em violações das restrições (quando estas forem especificadas). Essa quebra de restrições, em casos extremos, pode ser um fator de impedimento no cumprimento do objetivo do processo em questão pois, para cada restrição violada, existe uma penalidade associada. De maneira a tornar a atribuição de tarefas mais realista e mais flexível, a formula de conformidade descrita na seção 3.4.1 pode ser adaptada. Para tanto, um fator de *compensação* deveria ser aplicado quando a violação de alguma restrição fosse realmente necessária. Dessa forma, não prejudicando o grau de conformidade dos recursos que devem executar as tarefas.

Referências Bibliográficas

[1] Allen, R. *Workflow handbook 2001.* Future Strategies Inc., 2001, ch. Workflow: An Introduction.

[2] Alt, R., Klein, S., and Kuhn, C. Service task allocation as an internal market. In *Proc. of the Second European Conf. on Information Systems* (Breuklen: Nijenrode University Press, 1994).

[3] Atluri, V., and Huang, W.-K. An extended petri net model for supporting workflow in a multilevel secure environment. In *Proceedings of the tenth annual IFIP TC11/WG11.3 international conference on Database security: volume X : status and prospects* (London, UK, UK, 1997), Chapman & Hall, Ltd., pp. 240–258.

[4] Atluri, V., kuang Huang, W., and Bertino, E. A semantic-based execution model for multilevel secure workflows. *Journal of Computer Security 8*, 1 (2000).

[5] Black, P. E. Knapsack problem, 2005.

[6] Brucker, P. *Scheduling Algorithms.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[7] Casati, F., Ceri, S., Pernici, B., and Pozzi, G. Conceptual modeling of workflows. In *9th International Conference* (Gold Cost, Australia, 1995).

[8] Casati, F., Grefen, P., Pernici, B., Pozzi, G., and Sanchez, G. Wide workflow model and architecture. Tech. Rep. CTIT 96–19, University of Twente, 1996.

[9] Castano, S., and Fugini, M. G. Rules and patterns for security in workflow systems. In *Proceedings of the IFIP TC11 WG 11.3 Twelfth International Working Conference on Database Security XII* (Deventer, The Netherlands, The Netherlands, 1999), Kluwer, B.V., pp. 59–74.

[10] Cheng, E. C. An object-oriented organizational model to support dynamic role-based access control in electronic commerce. *Decision Support Systems 29*, 4 (2000), 357–369.

[11] D.F. Ferraiolo, J. Cugini, D. K. Role-based access control: Features and motivations. In *11th Annual Computer Security Applications Conference* (1995), IEEE Computer Society Press, pp. 241–248.

[12] do Prado, D. S. *Teoria das Filas e da Simulação.* Editora de Desenvolvimento Social, 1999.

[13] Ferraiolo, D. F., and Kuhn, D. R. Role-based access control. In *15th National Computer Security Conference* (1992), pp. 554–563.

[14] Ferscha, A. Qualitative and quantitative analysis of business workflows using generalized stochastic petri nets, 1994.

[15] Governatori, G., Rotolo, A., and Sadiq, S. A model of dynamic resource allocation in workflow systems. In *XV Australasian Database Conference* (Dunedin, Nova Zelândia, 2004).

[16] Graziano, A. M., and Raulin, M. L. *Research Methods: a process of inquiry.* Allyn and Bacon, Inc., 1999.

[17] Harker, P. T., and Ungar, L. H. A market-based approach to workflow automation.

[18] Kellner, M. I., Madachy, R. J., and Raffo, D. M. Software process simulation modeling: Why? what? how? *Journal of Systems and Software* (1999).

[19] Kumar, A., van der Aalst, W. M., and Verbeek, E. M. Dynamic work distribution in workflow management systems: How to balance quality

and performance? *Journal of Management Information Systems 18*, 3 (2001), 157–193.

[20] MENDES DE ARAUJO, R., AND DA SILVA BORGES, M. R. Sistemas de workflow. XX Jornada de Atualização em Informática, 2001. Fortaleza, Ceará, Brasil.

[21] MOMOTKO, M., AND SUBIETA, K. Dynamic changes in workflow participant assignment. In *ADBIS Research Communications* (Bratislava, Slovakia, setembro 2002), Slovak University of Technology, Bratislava, pp. 175–184.

[22] PÁDUA, S. I. D., SILVA, A. R. Y., INAMASU, R. Y., AND PORTO, A. J. V. O potencial das redes de petri em modelagem e análise de processos de negócios. *Gestão e Produção* (2004).

[23] SANDHU, R., BHAMIDIPATI, V., COYNE, E., CANTA, S., AND YOUMAN, C. The ARBAC97 model for role-based administration of roles: Preliminary description and outline. pp. 41–54.

[24] SANDHU, R. S., COYNE, E. J., FEINSTEIN, H. L., AND YOUMAN, C. E. Role-based access control models. *IEEE Computer 29*, 2 (1996), 38–47.

[25] SHEN, M., TZEN, G. H., AND LIU, D. R. Multi-criteria task assignment in workflow management systems. In *Proc. Thristysixth Hawaii Internat. Conf. System Sci.* (Big Island, Hawaii, 2003).

[26] SINGH, A., AND MEERAN, S. A state-of-the-art review of job-shop scheduling techniques. Tech. rep., Depatment of Applied Physics, Electronic and Mechanical Engineering, University of Dundee, 1998.

[27] STEINMACHER, I., AND DE LIMA, J. V. Melhorando a cooperação através da antecipação de tarefas em workflow. In *2nd Latin American Web Congress and the 10th Brazilian Symposium on Multimedia and the Web - PHD and MSC Workshop* (Ribeirão Preto,São Paulo, 2004), vol. 2.

[28] STOHR, E. A., AND ZHAO, J. L. Workflow automation: Overview and research issues. *Inform. Systems Frontiers* (2001).

[29] Tan, J. C., and Harker, P. T. Designing workflow coordination: centralized versus market-based mechanisms. Tech. rep., Department of Systems Engineering, University of Pennsylvania, 1997.

[30] Tramontina, G. B. Análise de problemas de escalonamento de processos em workflow. Master's thesis, Instituto de Computação, Universidade de Campinas (UNICAMP), Campinas, SP, Brazil, 2004.

[31] van der Aalst, W. M. P., and van Hee, K. M. *Workflow Management: Models, Methods, and Systems*. MIT Press, 2002.

[32] Veloso, R. R. *Java e XML: Processamento de Documentos XML com Java*. Novatec Editora Ltda, 2003.

[33] Veloso, R. R., e Silva, R. S., and Macêdo, A. Uso de simulações para a análise de alocações e tempo de processamento de tarefas em sistemas de workflow. In *Anais da Semana da Computação da Fesurv 2005* (Rio Verde, Goiás, Brasil, novembro 2005), vol. 2. Mídia digital.

[34] Veloso, R. R., and Macêdo, A. Distribuição de tarefas em sistemas de workflow baseado na aptidão dos recursos. In *Anais do Workshop Brasileiro de Tecnologias de Colaboração (WCSCW)* (Juiz de Fora, Minas Gerais, Brasil, novembro 2005), vol. 2. Mídia digital.

[35] Verbeek, H. M. W., Basten, T., and van der Aalst, W. M. P. Diagnosing workflow processes using Woflan. *The Computer Journal 44*, 4 (2001), 246–279.

[36] WfMC. The workflow reference model. Tech. Rep. WfMC-TC00-1003, Workflow Management Coalition, Hampshire, Reino Unido, Janeiro 1995.

[37] WfMC. Terminology and glossary. Tech. Rep. WfMC-TC-1011, Workflow Management Coalition, Hampshire, Reino Unido, Fevereiro 1999.

[38] Wikarski, D. An introduction to modular process net. Tech. Rep. TR-96-019, International Computer Science Institute, Berkeley, 1996.

[39] Zapf, M., and Heinzl, A. Evaluation of generic process design patterns: An experimental study. In *Business Process Management, Models, Techniques, and Empirical Studies* (London, UK, 2000), Springer-Verlag, pp. 83–98.

[40] Zeng, D. D., and Zhao, J. L. Effective role resolution in workflow management. *INFORMS journal on computing 17*, 3 (2005), 374–387. ISSN: 1091-9856.

[41] Zisman, M. D. *Representation, specification and automation of office procedures*. PhD thesis, University of Pensylvania, Wharton School of Business, Pensylvania, 1997.

[42] zur Muehlen, M. Resource modeling in workflow applications. In *Proceedings of the 1999 Workflow Management Conference* (novembro 1999), pp. 137–153.

Apêndice A

Arquivo XPDL utilizado pelo Simulador Lambari

O simulador e motor de *workflow* Lambari utiliza um modelo de XPDL simplificado. Este modelo contém somente as definições de tarefas, pré e pós-condições, tempo das tarefas, papéis associados e transições do fluxo.

A.1 Exemplo de XPDL

Exemplo de arquivo XPDL utilizado nos experimentos deste trabalho. Este arquivo contém três tarefas (tarefas 1, 2 e 3) associadas a três papéis: papel 1, papel 2 e papel 3. A ordem que as tarefas encontram no fluxo está definido na *tag transitions*, sendo que a tarefa 1 é a primeira, seguida da tarefa 2 e tarefa 3 por ultimo.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE lxpdl SYSTEM "lxpdl.dtd">
<lxpdl>
    <tasks>
        <task name="1" precond="NULL" poscond="" time="10">
            <role name="role1"/>
        </task>
        <task name="2" precond="1" poscond="2" time="10">
            <role name="role2"/>
```

```
        </task>
        <task name="3" precond="2" poscond="" time="10">
            <role name="role3"/>
        </task>
        <start name="1"/>
        <end name="3"/>
    </tasks>
    <transitions>
        <transition source="1" destination="2"/>
        <transition source="2" destination="3"/>
    </transitions>
</lxpdl>
```

A.2 Gramática DTD para XPDL

Apresenta-se aqui a gramática DTD que define a sintaxe do XPDL utilizado pelo simulador. O código dessa gramática é listado a seguir.

```
<?xml version='1.0' encoding='UTF-8'?>
<!--
    TODO define vocabulary indentification
    PUBLIC ID: -//Facom, Federal University of Uberlandia //DTD //EN
    SYSTEM ID: lxpdl.dtd
-->
<!ELEMENT transition EMPTY>
<!ATTLIST transition
    destination CDATA #IMPLIED
    source CDATA #IMPLIED
  >
<!ELEMENT transitions (transition)+>
<!ELEMENT role EMPTY>
<!ATTLIST role
    name CDATA #IMPLIED
```

```
  >
<!ELEMENT task (role)>
<!ATTLIST task
    time CDATA #IMPLIED
    poscond CDATA #IMPLIED
    precond CDATA #IMPLIED
    name CDATA #IMPLIED
  >
<!ELEMENT start EMPTY>
<!ATTLIST start
    name CDATA #IMPLIED
  >
<!ELEMENT end EMPTY>
<!ATTLIST end
    name CDATA #IMPLIED
  >
<!ELEMENT tasks (end|start|task)+>
<!ELEMENT lxpdl (transitions|tasks)+>
```

Livros Grátis

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

Milhares de Livros para Download:

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