Pós-Graduação em Ciência da Computação

Milton Secundino de Souza Júnior

LONGWISE4CLOUD: UM AMBIENTE ADAPTATIVO PARA EXECUÇÃO DE *WORKFLOWS* DE LONGA DURAÇÃO COMO SERVIÇO

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
<www.cin.ufpe.br/~posgraduacao>

RECIFE
2018

Milton Secundino de Souza Júnior

LONGWISE4CLOUD: UM AMBIENTE ADAPTATIVO PARA EXECUÇÃO DE *WORKFLOWS* DE LONGA DURAÇÃO COMO SERVIÇO

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Doutor em Ciência da Computação.

Orientador: Nelson Souto Rosa
Coorientador: Fernando Antônio Aires Lins

RECIFE
2018

Catalogação na fonte
Bibliotecário Jefferson Luiz Alves Nazareno CRB4-1758

S729l Souza Júnior, Milton Secundino de.
LONGisE4cloud: um ambiente adaptativo para execução de workflows de longa duração como serviço / Milton Secundino de Souza Júnior. – 2018.
132 f.: fig. tab.

Orientador: Nelson Souto Rosa.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn. Ciência da Computação, Recife, 2018.
Inclui referências e apêndices.

1. Computação em nuvem. 2. Sistemas adaptativos . 3. Automação de processos de negócio. I. Rosa, Nelson Souto. (Orientador) II. Titulo.

004.6782 CDD (23. ed.) UFPE-MEI 2018- 65

Milton Secundino de Souza Júnior

LONGWisE4cloud: Um Ambiente Adaptativo para Execução de Workflows de Longa Duração como Serviço

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação

Aprovado em: 12/03/2018.

_____

Orientador: Prof. Dr. Nelson Souto Rosa

BANCA EXAMINADORA

_____

Prof. Dr. Paulo Romero Martins Maciel
Centro de Informática /UFPE

_____

Prof. Dr. Ricardo Massa Ferreira Lima
Centro de Informática /UFPE

_____

Prof. Dr. Vinicius Cardoso Garcia
Centro de Informática /UFPE

_____

Prof. Dr. Daniel Cardoso Moraes de Oliveira
Instituto de Computação/UFF

_____

Prof. Dr. Robson Wagner Albuquerque de Medeiros
Departamento de Estatística e Informática/UFRPE

Eu dedico esta tese a toda minha família, amigos e professores que me deram o suporte necessário para que eu conseguisse chegar aqui.

Agradecimentos

Se fosse para lançar nesta Seção todas as palavras que realmente seriam dignas de serem dedicadas a todos os que tornaram este trabalho possível, ela certamente ficaria maior que a própria proposta contida na tese. Entretanto, seguem meus singelos, mas sinceros, agradecimentos aos que tanto ajudaram na construção deste projeto:

- A Deus por toda a força, coragem, discernimento, saúde, paciência e sabedoria para trilhar este caminho tão longo, mas tão enriquecedor;
- Também agradeço demais ao meu falecido pai, Sr. Milton, que se esforçou bastante para ensinar aos filhos o valor dos estudos para a vida de uma pessoa;
- Aos meus orientadores, os professores Dr. Nelson Rosa e Dr. Fernando Lins, que me deram um apoio absurdamente grande em todos os passos da construção deste projeto. Tiveram enorme paciência para entender percalços e pausas que comumente ocorrem em esforços que levam um tempo razoável para serem concluídos, como é o caso da construção de uma tese de doutorado, e sempre estiveram à disposição para sanar dúvidas, discutir ideias, avaliar resultados e prestar orientações sem as quais este trabalho jamais seria concluído;
- Aos colegas de trabalho da Unimed Norte/Nordeste, onde eu trabalhava no início dos esforços deste trabalho de pós-graduação. Eles tiveram não apenas a camaradagem de permitir que eu me ausentasse frequentemente para poder me dedicar às aulas e reuniões de orientação, como deram muita força e torceram pelo meu sucesso após minha saída da empresa;
- Aos queridos colegas do Campus Igarassu do Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco - IFPE, que também me ajudaram bastante sendo compreensivos com minhas ausências e também com a minha impossibilidade de assumir cargos de coordenação e supervisão para ter mais tempo disponível para construir esta tese. Além disso, a permissão e o apoio para a preparação de ambientes voltados aos experimentos no laboratório do Campus também merece um grande agradecimento da minha parte;
- Aos amigos que sempre dedicaram palavras de força e incentivo ao longo dos anos dedicados na construção deste projeto de pesquisa;
- Aos meus amados familiares que também sempre deram muita força, torceram, oraram e me incentivaram para dar continuidade aos esforços de construção desta pesquisa;

- Aos meus filhos, hoje razões principais de tudo que eu faço, que nem haviam nascido no início da construção deste projeto, mas que se tornaram grandes motivações para que o mesmo fosse finalizado;

- Finalmente, um agradecimento todo especial a minha amada esposa, Roberta Serrano, profissional e líder extremamente competente, companheira indiscutivelmente exemplar além de mãe absurdamente dedicada. Não faço ideia de como ela consegue desempenhar tão bem tantos papéis de forma simultânea, mas, sem ela, este trabalho também não teria sido concluído. Em alguns momentos, sua força e seu incentivo demonstraram que ela acreditava mais em mim do que eu mesmo. Esta postura evitou que eu abandonasse os esforços mesmo quando as situações pessoais e profissionais me empurravam para longe do projeto. Eu me sinto verdadeiramente abençoado por ter uma pessoa assim na minha vida e ela, certamente, merece bem mais que estas poucas linhas de homenagem.

Tu, Senhor, guardarás em perfeita paz aquele cujo propósito está firme,
porque em Ti confia.
—ISAÍAS 26:3

Resumo

Workflow de Longa Duração (WLD) é um tipo especial de *workflow* cuja característica principal é o considerável consumo de tempo para a geração de resultados. Nos dias atuais, ele está cada vez mais presente em cenários como ambientes corporativos, onde a adoção de *workflows* na automação de processos de negócio é crescente. Uma vez que a complexidade destes processos também vem aumentando, há, comumente, um aumento na demanda de processamento computacional e no tempo para executá-los. Nestes contextos, aumentam as probabilidades de ocorrências de eventos como mudanças em regras de negócio, interrupções de serviços responsáveis por atividades que compõem os WLDs ou uma sobrecarga no ambiente responsável pelo gerenciamento da execução destas atividades. Estes eventos podem levar a problemas como a perda de resultados intermediários gerados ao longo da execução dos *workflows*, a necessidade de reprocessar completamente atividades que já haviam sido concluídas ou ainda o esgotamento de recursos computacionais utilizados pelo próprio sistema de gerenciamento. Para evitar ou minimizar estes problemas, os WLDs devem ser executados em ambientes projetados para atender as suas demandas particulares, como a necessidade de estratégias para acomodar mudanças em regras de negócio, suporte à adaptação, soluções para o provisionamento de recursos, gerenciamento de dados e estados, escalabilidade e assim por diante. Soluções existentes se restringem a propostas que focam em aspectos como robustez, tolerância a falhas, adaptações associadas à substituição e reprocessamento de serviços em tempo de execução ou à migração de objetos dentro de uma infraestrutura para garantir a continuidade do processamento. Nenhum destes trabalhos atende, de forma integrada, a todos os aspectos relevantes à execução dos WLDs mencionados anteriormente. Além disso, ainda há uma ausência de soluções que ofereçam suporte à gerência da execução de WLDs em uma abordagem *como serviço* (*aaS* do inglês *as a Service*) em ambientes de nuvem. Considerando este contexto, esta tese apresenta o *LONGWisE4cloud* (do inglês *LONg runninG Workflows Execution environment for cloud*), um ambiente adaptativo para gerenciamento da execução de WLDs como serviço que atende às particularidades relativas a estes tipos especiais de *workflows*. Experimentos envolvendo WLDs associados a processos de negócio reais de uma operadora brasileira de planos de saúde foram realizados para avaliar funcionalidades e aspectos do *LONGWisE4cloud* e demonstraram ganhos de desempenho na execução destes *workflows* submetidos ao ambiente proposto.

Palavras-chave: *Workflows* de Longa Duração. Automação de Processos de Negócio. Sistemas Adaptativos. Computação Orientada a Serviços. Computação em Nuvem.

Abstract

Long-Running Workflow (*LRW*) is a particular kind of workflow whose most relevant characteristic is a considerable amount of time to produce results. Nowadays, it is increasingly present in scenarios such as corporative environments, where the adoption of workflows in the automation of business processes is growing. Since the complexity of these processes is also increasing, there is commonly more demand for computational processing and time to execute them. In these contexts, there is an augment in the probabilities of occurrence of events such as changes in business rules, interruption of services responsible for activities that compose the *LRWs* or an overload of the management environment responsible for performing these activities. Those events can lead to problems such as the loss of intermediary results generated during workflows execution, the need to completely reprocess activities that have already been finished or the exhaustion of computational resources used by the management system itself. To avoid or minimize these troubles, the *LRWs* should be deployed for execution in properly designed environments to support their particular demands such as the need for strategies to accommodate changes in business rules, adaptive support, solutions for resource provisioning, data and state management, scalability and so on. Existing solutions are restricted to proposals that focus on aspects such as robustness, fault tolerance, adaptations associated with the replacement and reprocessing of services at runtime or the migration of objects within an infrastructure to ensure continuity of processing. None of these works addresses, in an integrated way, all aforementioned relevant aspects to the execution of the *LRWs*. In addition, there is still an absence of solutions that support the management of *LRWs* execution in an *as a Service* approach atop cloud environments. Considering this context, this thesis presents the *LONGWisE4cloud* (LONg runninG Workflows Execution environment for cloud), an adaptive environment turned to the execution management of *LRWs* as a service which addresses the particularities of these special types of workflows. Experiments involving *LRWs* associated with actual business processes of a brazilian health insurance company were performed to evaluate the functionalities and aspects of *LONGWisE4cloud* and demonstrated performance gains in the execution of these *workflows* submitted to the proposed environment.

Keywords: Long-Running Workflow. Business Process Automation. Adaptive Systems. Service-Oriented Computing. Cloud Computing.

Lista de Figuras

1.1 Processo de Pesquisa. 23

2.1 Processo para Cobrança dos Clientes de um Cartão de Crédito 26
2.2 Modelos de Serviços em Nuvens 36

3.1 *Circuito Reo* para Transações de Saque em Caixa Eletrônico 40
3.2 Estrutura de Distribuição da Execução de *Workflows* 42
3.3 Modelo para a Execução de *Workflows* Pervasivos 44
3.4 Replicação de Recursos Computacionais para Suporte à Execução de WLDs . . 47
3.5 Arquitetura para Suporte a Transações Longas Envolvendo *GTM-middleware* . 50

4.1 Contexto de Uso do *LONGWisE4cloud* 60
4.2 Arquitetura Geral do *LONGWisE4cloud* 61
4.3 Interações Entre Componentes do *LONGWisE4cloud* 64
4.4 Sequência para Início da Execução de um WLD no *LONGWisE4cloud* 65
4.5 Estrutura de Componentes do *Execution Core* 67
4.6 Detalhes Dinâmicos de uma *ECI* 71
4.7 Sequência de Início da Execução de um WLD em uma *ECI* 72
4.8 Cenários de Reconfiguração Dinâmica 73
4.9 Sequência de Reconfiguração Dinâmica no *LONGWisE4cloud* 77

5.1 *Bundles* do *LONGWisE4cloud* 81
5.2 *Bundles* do *Execution Core* 82
5.3 Classes no *LONGWisE4cloud* 83

6.1 *Workflow* para Cobrança de Clientes 92
6.2 Tempos de Processamento na Reconfiguração Dinâmica 93
6.3 Variação nos Tempos de Processamento na Reconfiguração Dinâmica 95
6.4 Média de Tempos de Processamento em Reconfigurações 97
6.5 *Overhead* em Reconfigurações Dinâmicas 98
6.6 *Workflow* para Pagamento de Prestadores de Serviços Médicos 100
6.7 Resultados do Experimento Relativo ao Provisionamento de Recursos 103
6.8 *Overhead* em Provisionamento de Recursos 104
6.9 Resultados do Experimento Relativo à Escalabilidade 107
6.10 Resultados da Escalabilidade por *Workflow* 109
6.11 Resultados da Escalabilidade por *Workflow* 110

B.1 Arquitetura em Camadas do Padrão OSGi 131

C.1 Definição de Esquema XML para WLD 132

Lista de Tabelas

3.1 Quadro Comparativo - Trabalhos Associados à Execução de WLDs 52

5.1 Propriedades para Execução de WLDs 84
5.2 *LRWs General Information* 86
5.3 *Execution Log* 87
5.4 *States Repository* 87

6.1 Fatores e Níveis - Avaliação da Reconfiguração Dinâmica 90
6.2 Configurações de *Hardware* para Implantação do *LONGWisE4cloud* 92
6.3 Fatores e Níveis - Avaliação do Provisionameto de Recursos 100
6.4 Fatores e Níveis - Avaliação da Escalabilidade 105
6.5 Configurações de *Hardware* do Ambiente de Virtualização 105

Lista de Acrônimos

BPM	*Business Process Management*	26
BPMN	*Business Process Model and Notation*	26
BPMS	*Business Process Management Systems*	30
LRW	*Long-Running Workflow*	9
OSGi	*Open Services Gateway initiative*	80
SOA	*Service-Oriented Architecture*	31
SOC	*Service-Oriented Computing*	30
UML	*Unified Modeling Language*	45
WfMS	*Workflow Management System*	30
WLD	*Workflow* de Longa Duração	17
WS-BPEL	*Web Services Business Process Execution Language*	26

Sumário

1 INTRODUÇÃO 16
1.1 Contexto e Motivação 16
1.2 Problema 18
1.3 Considerações sobre o Estado da Arte 19
1.4 Proposta 21
1.4.1 Objetivos da Tese 21
1.5 Metodologia 22
1.6 Estrutura da Tese 23
2 CONCEITOS BÁSICOS 25
2.1 Processos de Negócio 25
2.2 *Workflows* 26
2.1 *Workflows* de Longa Duração 27
2.2 Sistemas de Automação de Processos de Negócio 29
2.5 Computação Orientada a Serviços 30
2.5.1 Arquitetura Orientada a Serviços 31
2.6 Sistemas Adaptativos 32
2.6.1 Quiescência 33
2.7 Computação em Nuvem 34
2.8 Considerações Finais 37
3 TRABALHOS RELACIONADOS 38
3.1 Controle de Fluxo de Execução em *Workflows* de Longa Duração 38
3.2 Adaptação em WLDs 43
3.2.1 Composições e Reconfigurações Dinâmicas 43
3.2.2 Migrações em *Workflows* de Longa Duração 46
3.3 Outras Abordagens sobre *Workflows* de Longa Duração 49
3.4 Sumário dos Trabalhos Relacionados 51
3.5 Considerações Finais 54
4 PRINCIPAIS CONCEITOS E ARQUITETURA DO *LONGWISE4CLOUD* . . . 55
4.1 Princípios Básicos 55
4.2 Visão Geral 60
4.3 Aspectos Arquiteturais 61

4.3.1 Estrutura 61
4.3.2 Dinâmica 64
4.3.3 Aspectos Estruturais da *ECI* 66
4.3.4 Aspectos Dinâmicos de uma *ECI* 70
4.3.4.1 Zonas de Execução e Mudança 72
4.3.4.1 Reconfigurações Dinâmicas 76
4.4 Considerações Finais 79
5 IMPLEMENTAÇÃO DO *LONGWISE4CLOUD* 80
5.1 *OSGi* na Implementação do *LONGWisE4cloud* 80
5.2 Classes e Objetos 82
5.3 Repositórios 86
5.4 Considerações Finais 88
6 AVALIAÇÃO EXPERIMENTAL 89
6.1 Reconfiguração Dinâmica 89
6.2 Provisionamento de Recursos 99
6.3 Escalabilidade 104
6.4 Considerações Finais 110
7 CONCLUSÕES E TRABALHOS FUTUROS 112
7.1 Contribuições 112
7.2 Limitações 114
7.3 Trabalhos Futuros 117
REFERÊNCIAS 120
APÊNDICE A - PUBLICAÇÕES DO AUTOR 129
APÊNDICE B - OSGI 130
APÊNDICE C - ESQUEMAS XML UTILIZADOS 132

1

INTRODUÇÃO

Este capítulo apresenta uma visão geral desta tese de doutorado. A Seção 1.1 detalha o contexto no qual este trabalho está inserido assim como as motivações que o levaram a ser desenvolvido. A Seção 1.2 apresenta detalhes a respeito do problema tratado. Considerações sobre deficiências encontradas no estado da arte são discutidas na Seção 1.3. A Seção 1.4 apresenta uma visão resumida da solução proposta para o problema juntamente com os objetivos deste projeto de pesquisa. A metodologia utilizada ao longo da construção da tese é explicada na Seção 1.5. Finalmente, detalhes a respeito da organização deste documento são mostrados na Seção 1.6.

1.1 Contexto e Motivação

Empresas e organizações estão cada vez mais estruturando, formalizando, automatizando e aperfeiçoando seus processos de negócio visando mais eficiência na geração de resultados (ZENG et al., 2011; ANJU et al., 2013; SIADAT; SHAMASBI, 2015; NURHAYATI; FITRISARI, 2016; JAYAKODY et al., 2016). Estes processos de negócio podem ser entendidos como uma série de regras e passos para a solução de um problema corporativo (VERGIDIS; TIWARI; MAJEED, 2008). Eles são uma vantagem competitiva em termos de mercado, já que permitem um melhor atendimento das necessidades dos clientes além de possibilitarem que recursos humanos e tecnológicos sejam mais bem utilizados (GONG et al., 2004). Em órgãos governamentais, estes avanços no uso de processos podem levar à prestação de serviços à população com mais qualidade e com mais racionalidade no uso de recursos públicos (OROSZ; OROSZ, 2012; SILVIA; SUHARDI; YUSTIANTO, 2016).

Ao mesmo tempo em que ocorre um aumento na formalização dos processos associados à criação de produtos ou prestação de serviço, é possível perceber também um incremento na complexidade destes processos. Aspectos como a globalização e a especialização de atividades assumidas por empresas em diversos setores geram cenários onde, por exemplo, bens materiais podem ser confeccionados através de cadeias complexas de tarefas produtivas (QI; CHU, 2009). Observando esta realidade, é possível perceber atividades executadas por organizações geograficamente dispersas, envolvimento de atores com papéis bem específicos, mas com auto-

nomia dentro da cadeia produtiva, clientes capazes de interferir, através de meios sofisticados de comunicação, em etapas intermediárias de um processo produtivo e assim por diante.

Neste contexto de aprimoramento, automação e formalização de processos de negócio, o uso intensivo de sistemas de software está cada vez mais presente. Um dos reflexos deste avanço é a adoção cada vez maior nas organizações dos chamados *workflows*: abstrações usadas na representação dos processos de negócio, permitindo sua modelagem, implantação e distribuição em um esforço computacional com diversos passos, também conhecidos como tarefas ou atividades, que podem viabilizar a automação destes processos (DECKER et al., 2010). Através do uso destas abstrações, as organizações podem usar um suporte de software que seja mais alinhado com as estratégias observadas atualmente na construção de produtos, prestação de serviços e onde seus processos podem ser executados por serviços computacionais especializados (NITTO et al., 2008).

Workflows vêm, assim, atraindo cada vez mais o interesse das organizações, que podem aumentar a eficiência em seus processos de forma padronizada e com uso intensivo de novas tecnologias (PATHIRAGE et al., 2011). Além de permitirem o desdobramento e a distribuição física de um processamento computacional complexo em diversos passos, os *workflows* podem ser associados a infraestruturas como as nuvens computacionais para gerar soluções corporativas que melhoram a eficiência da execução de processos de negócio em aspectos como custo, desempenho e qualidade (SAURABH; RANJAN, 2012; ROBINSON; ZUVIRIA; VINITA, 2012; KIKUCHI, 2014; SUPING et al., 2016). Benefícios como elasticidade, alta disponibilidade, escalabilidade e robustez, comumente encontrados em infraestruturas de nuvem, favorecem a obtenção dos resultados observados nestes trabalhos.

Considerando este cenário de processos de negócio cada vez mais complexos, há situações onde os resultados do processamento podem levar horas ou dias para serem gerados. *Workflows* de Longa Duração (WLDs) são tipos especiais de *workflows* associados a processos que envolvem as chamadas transações de longa duração (HU et al., 2010). Este período considerável de tempo gasto no processamento tem origem em vários fatores, tais como a existência de dependências envolvendo tarefas onde uma ou algumas delas levam um longo tempo para serem concluídas (ASSUNCAO; CUNHA, 2013) ou ainda falhas em pontos específicos do processo sendo executado, como a indisponibilidade momentânea de um rotina computacional responsável por uma atividade deste processo. Como exemplos de processos de negócio de longa duração podem ser citados os passos para uma operadora de planos de saúde permitir que um cliente se submeta a um procedimento médico especializado, a burocracia para se autorizar a aposentadoria de um contribuinte da Previdência Social ou obtenção dos valores de ressarcimento relativos a um seguro automotivo.

Assim como *workflows* comuns, amplamente adotados em organizações, os WLDs permitem a execução de processos de negócio através de um conjunto de serviços computacionais especializados, trazendo eficiência e melhoria contínua para tarefas corporativas. Apesar disso, a longa duração aumenta a probabilidade de ocorrência de eventos como mudanças em regras

de negócio, instabilidade ou falha inesperada de atividades que compõem o WLD, falta de recursos de hardware (e.g., memória) ou de software (e.g., máquinas virtuais, serviços de banco de dados) para completar um processamento, falhas de infraestrutura ou problemas de segurança. Estes eventos podem causar transtornos como a perda de resultados já gerados pelo *workflow*, a necessidade de reprocessar uma atividade já concluída ou o esgotamento de recursos computacionais utilizados pelo ambiente que gerencia a execução dos *workflows*. O desperdício de trabalho já realizado na execução de um processo de negócio ou a aplicação de novos esforços para refazer um processamento pode levar a um aumento indesejável, ou até inaceitável, no custo de execução dos *workflows* como pagamentos adicionais a provedores de nuvem.

Os ambientes que gerenciam a execução dos WLDs (daqui por diante chamados apenas de *ambientes de execução*, *sistemas de execução* ou simplesmente *ambientes*) possuem desafios específicos, apresentados na Seção 1.2, relacionados a seus requisitos, arquitetura e, consequentemente, a sua implementação. Sendo assim, a construção de um ambiente que atenda às particularidades que precisam ser tratadas quando se lida com o gerenciamento da execução de *workflows* de longa duração é um esforço que traz avanços para este contexto e constitui o foco principal deste trabalho.

1.2 Problema

No contexto dos *workflows* de longa duração, os principais desafios estão frequentemente relacionados ao elevado consumo de tempo necessário para a execução das atividades do *workflow*. Conforme já discutido na Seção 1.1, este aspecto aumenta as probabilidades de ocorrência de eventos como mudanças em regras de negócio ou a interrupção de atividades que podem levar a problemas indesejáveis nos contextos de uso destas abstrações, tais como desperdício de recursos computacionais e prejuízos financeiros.

Quando uma regra de negócio é modificada por uma organização, por exemplo, WLDs que estejam em execução e associados àquela regra podem gerar (ou já terem gerado) resultados que não são mais compatíveis com a nova regra. Reprocessar completamente o *workflow* pode ser um transtorno considerável dependendo do tempo de execução do mesmo e da urgência com que a empresa vai precisar dos resultados por ele produzidos. Nestes casos, é desejável que o WLD possa se submeter a mudanças ainda em execução, como a substituição dos serviços responsáveis por uma ou mais atividades que o compõem. Ainda assim, um serviço responsável por uma atividade pode ser substituído, mas resultados intermediários já gerados por ele, dependendo do tipo de abordagem utilizada nesta mudança, podem precisar ser descartados. Entretanto, dependendo do tempo que foi gasto na geração destes dados, o reuso dos mesmos pode ser indispensável para evitar grandes esforços para regerá-los.

A continuidade do processamento em casos de falhas na execução das atividades através de ações como a troca de serviços que fazem parte da composição do *workflow* também é um desafio no contexto dos WLDs. Assim como na mudança de regras de negócio, uma falha em uma

parte do WLD deve poder ser sanada sem necessariamente ter que se reprocessar todo o WLD. Quando trocas de serviços associados a atividades são necessárias nestes casos, a possibilidade de reuso de resultados intermediários também é importante para evitar os desperdícios de recursos computacionais já mencionados anteriormente.

Tanto em mudanças de regras de negócio como em situações de falhas em serviços o reprocessamento do *workflow* (ou de parte dele) pode ser mais eficiente do que reuso de dados. Isto é observado, por exemplo, quando o esforço para este reuso é maior que o esforço para se reprocessar as atividades que precisam ser interrompidas. Um gerenciamento de execução de WLDs que seja restrito a uma única estratégia não atende a este cenário onde as ações mais adequadas dependem da situação na qual se encontra a execução do WLD.

Outro fator importante para este contexto dos WLDs é que eles são comumente associados a atividades que cruzam as fronteiras organizacionais. Eles podem estar sendo executados sob a responsabilidade de diferentes empresas, através de agentes sendo executados em infraestruturas distintas, tais como nuvens computacionais, ou implementados em diferentes linguagens de programação. Estes cenários requerem que o ambiente responsável por gerenciar a execução destes *workflows* seja capaz de lidar com esta heterogeneidade.

Por estarem associados a um alto consumo de tempo para finalizar seus processamentos, os *workflows* de longa duração acabam também ficando mais tempo expostos a ataques contra a segurança e a integridade dos dados gerados e processados pelos mesmos.

Finalmente, o longo tempo de processamento dos WLDs em geral implica em considerável consumo de recursos computacionais tais como processadores, memória ou serviços de armazenamento de dados. O acúmulo de *workflows* desta natureza sendo processados simultaneamente exige não apenas que o ambiente de execução dos WLDs tenha à sua disposição uma infraestrutura adequada em termos de recursos disponíveis, com alta disponibilidade e elasticidade, mas que o mecanismo de gerenciamento de execução de WLDs possua elementos que permitam o uso apropriado dos recursos presentes na infraestrutura que dá suporte a estas execuções. Ter à disposição dos *workflows* um número considerável de máquinas virtuais, por exemplo, mas com um sistema de gerenciamento incapaz de distribuir os processamentos ao longo das mesmas pode implicar em desperdícios de elementos desta infraestrutura.

Sendo assim, o problema a ser tratado nesta tese é como gerenciar a execução de WLDs considerando todas estas particularidades anteriormente apresentadas e que são inerentes ao uso destes tipos especiais de *workflows*.

1.3 Considerações sobre o Estado da Arte

A análise de soluções relacionadas ao contexto de WLDs mostra que existem algumas propostas voltadas mais especificamente ao fluxo de execução das transações associadas aos WLDs, focando na identificação de alternativas e no redirecionamento deste fluxo para obter uma maior robustez no processamento dos *workflows* (HU et al., 2010; MENG; ARBAB, 2010;

KOKASH; ARBAB, 2013). Em todos eles, os autores não consideram situações comuns no contexto dos WLDs como, por exemplo, a necessidade de mudanças na composição dos serviços por conta de alterações nas regras de negócio.

Outras soluções focam na adaptação através de composição e reconfiguração dinâmica dos *workflows* (MONTAGUT; MOLVA; GOLEGA, 2008; JUHNKE; DORNEMANN; FREISLEBEN, 2009; ASSUNCAO; CUNHA, 2013; OLMSTED, 2015), ou migração de objetos (PEREZ-SORROSAL et al., 2006; FRANK; FONG; LAM, 2010) como parte do ambiente de execução. Entretanto, no caso dos trabalhos envolvendo mecanismos dinâmicos de configuração e reconfiguração, a solução normalmente consiste em substituir (ou adicionar) um componente e executar (ou reiniciar) atividades dos WLDs. Por sua vez, as soluções baseadas em migração requerem recursos adicionais como uma infraestrutura computacional redundante, algo que pode onerar consideravelmente, inclusive em termos financeiros, a execução dos *workflows*. Em todos estes trabalhos, os mecanismos propostos reforçam a robustez da execução dos *workflows* quando falhas em uma atividade são observadas. Entretanto, há uma ausência de abordagens que tenham múltiplos mecanismos de adaptação associados à execução de WLDs. Conforme mencionado na Seção 1.2, situações similares podem requerer diferentes estratégias de adaptação para evitar problemas como o desperdício de esforços e recursos computacionais para gerar resultados executando estes *workflows*. Entre as soluções existentes, há, por exemplo, uma falta de propostas que usem gerenciamento de dados para reforçar o reuso de resultados intermediários dos *workflows*. Este aspecto também é relevante para evitar os desperdícios mencionados.

Ainda neste contexto, existem soluções voltadas à identificação da propagação de falhas ao longo da execução do WLD (ALI; REIFF-MARGANIEC, 2012), aquelas focadas na integridade de informações processadas pelo WLD (CHIANESE et al., 2008; AMATO et al., 2014), e soluções de modelagem e simulação de WLDs (STANKEVICIUS; VASILECAS, 2016). Todos estes trabalhos, no entanto, possuem limitações à cobertura de falhas tratadas, ausência de adaptação e de suporte à execução, respectivamente.

As soluções propostas por Stein, Payne e Jennings (2011), Yang et al. (2014), FernÁndez, Tedeschi e Priol (2016), Schäfer et al. (2016) apresentam importantes contribuições para o processamento de *workflows*, trazendo contribuições voltadas à robustez e alta disponibilidade na execução de processos de negócio. Entretanto, nenhum destes trabalhos foca nas particularidades dos WLDs, sendo todos voltados ao contexto dos *workflows* comuns.

No que diz respeito à elasticidade e alta disponibilidade do ambiente de execução de WLDs, há soluções (WANG et al., 2014; ESTEVES; VEIGA, 2016) que suportam a execução de *workflows* utilizando uma estratégia *como serviço* (*as a Service*). Mas elas não atendem totalmente às particularidades relativas ao consumo de tempo observadas nos *workflows* de longa duração, sendo também restritas aos *workflows* comuns. Não há, por exemplo, uma integração dos aspectos destas soluções com adaptações tais como uma reconfiguração dinâmica de serviços com reuso de dados. Além disso, estes trabalhos focam principalmente nos *workflows* propriamente ditos, alocando recursos, como máquinas virtuais, e distribuindo atividades para

otimizar o desempenho da execução destes *workflows*. Recursos usados pelo ambiente de execução não recebem a atenção necessária.

Todos os trabalhos citados nesta seção serão detalhados e discutidos no Capítulo 3.

1.4 Proposta

Conforme as discussões apresentadas nas seções anteriores deste capítulo, a construção de um ambiente para o gerenciamento da execução de *workflows* de longa duração que atenda, de forma integrada, aos principais aspectos requeridos por estes tipos específicos de *workflows* é um desafio relevante para o contexto da automação de processos de negócio. Sendo assim, a proposta de trabalho apresentada nesta tese envolve a análise do domínio de WLDs, a caracterização deste tipo particular de *workflow*, a composição de uma lista de aspectos importantes para a execução de WLDs e, com base nos resultados destes esforços, definir uma arquitetura de software e a implementação de um ambiente com as características identificadas e desejadas para gerenciar a execução de WLDs.

O *LONGWisE4cloud* (*LONg runninG Workflows Execution environment for cloud*) pode ser utilizado como referência para outras implementações que se proponham a dar suporte à execução dos *workflows* de longa duração. A existência de uma referência ao desenvolvimento destes ambientes favorece a disseminação de soluções mais robustas voltadas a atender cenários que, conforme discutidos na Seção 1.1, já são comuns na realidade dos contextos corporativos. WLDs são encontrados nas cadeias produtivas de diversos setores e, quanto mais eficiente for o gerenciamento da execução dos mesmos, melhores serão os resultados obtidos por empresas e organizações em suas operações do dia a dia.

Para o desenvolvimento deste projeto de pesquisa, considera-se que a longa duração e as suas consequências são os fatores mais desafiadores dos WLDs. Sendo assim, a hipótese desta tese é que a adoção de mecanismos de adaptação integrados a diversos aspectos particulares necessários à execução deste tipo de *workflow* (como o uso de serviços, o gerenciamento de dados e estados) e consolidados em uma solução voltada para ambientes de nuvem são essenciais para melhoria do desempenho e para maior racionalização no uso dos recursos necessários à execução dos WLDs.

1.4.1 Objetivos da Tese

Esta tese possui os seguintes objetivos:

- *Geral*: Conceber uma solução para o gerenciamento da execução de *workflows* de longa duração que atenda as suas particularidades específicas e que possa ser utilizada como referência ao desenvolvimento de ambientes de execução para estes tipos especiais de *workflows*.
- *Específicos*:

- Caracterizar os *workflows* de longa duração;
- Estabelecer o conjunto de requisitos necessários à execução de WLDs;
- Definir uma arquitetura *genérica* (independente de elementos de implementação tais como linguagens de programação) para ambientes de execução de WLDs;
- Implementar um ambiente de execução de acordo com a arquitetura definida; e
- Avaliar a execução de WLDs no ambiente desenvolvido.

1.5 Metodologia

O desenvolvimento desta tese foi realizado seguindo uma abordagem metodológica inspirada nas recomendações propostas por Wieringa (2014). O projeto de pesquisa foi construído através de um processo cíclico que abrange investigações a respeito do problema, projetos de soluções, implementações dos projetos propostos e avaliações dos artefatos construídos. Os resultados destas avaliações obtidas em um ciclo são utilizados em ciclos subsequentes para que o projeto de pesquisa como um todo possa ser incrementado com novas soluções voltadas aos problemas identificados e tratados ao longo dos esforços desprendidos. A Figura 1.1 apresenta o processo seguido nesta tese para a construção do *LONGWisE4cloud*.

As etapas do processo compreendem as seguintes ações:

1. Investigação do Problema: Neste primeiro passo, fontes de informação (tais como pessoas, sistemas de software e publicações) devem ser identificadas, selecionadas e analisadas. Através deste esforço, é possível obter informações a respeito do domínio sobre o qual está sendo desenvolvida a pesquisa além de identificar desafios e problemas ainda não solucionados existentes tanto na literatura como em soluções já existentes voltadas a este domínio.

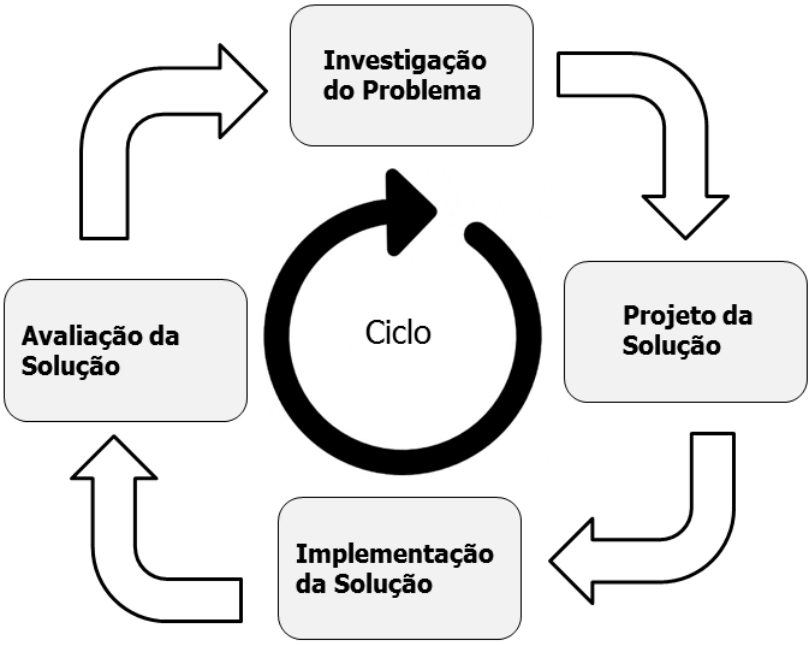
2. Projeto da Solução: Inicialmente, são identificados requisitos e restrições voltados à solução a ser construída. Após isso, a arquitetura da aplicação é definida com seus aspectos estruturais e dinâmicos. Diagramas e modelos devem ser construídos para especificar o que será implementado.

3. Implementação da Solução: Programação e implantação efetiva da solução a ser entregue ao final do ciclo.

4. Avaliação da Solução: O foco deste último passo é avaliar o que foi implementado. Aspectos relativos à qualidade da solução, atendimento aos requisitos previamente estabelecidos e vantagens no uso da mesma devem ser identificados para uso posterior em melhorias e inovações a serem desenvolvidos em ciclos posteriores do desenrolar do processo.

O conjunto de requisitos, a arquitetura, a especificação e implementação de um ambiente de gerenciamento de WLDs são artefatos a serem construídos ao longo processo.

Figura 1.1: Processo de Pesquisa.



Fonte: O próprio autor.

O uso do processo descrito nesta seção permitiu que o *LONGWisE4cloud* passasse por uma evolução sistemática, algo que se refletiu nos resultados obtidos e que serão discutidos ao longo dos capítulos deste documento.

1.6 Estrutura da Tese

Este documento está organizado de acordo com o processo aplicado no projeto de pesquisa e apresentado na Seção 1.5. Sendo assim, os capítulos foram agrupados em partes obedecendo a seguinte estrutura:

Parte II: Investigação do Problema

No Capítulo 2 são introduzidos conceitos básicos relacionados à proposta do trabalho. Definições de elementos importantes associados ao domínio dos *workflows* de longa duração são discutidas assim como ferramentas específicas utilizadas neste trabalho são descritas.

O Capítulo 3 apresenta um conjunto de trabalhos relacionados ao tema proposto para a tese. Esta revisão da literatura é importante para se identificar soluções já existentes relativas ao domínio dos WLDs. Além disso, diversos destes trabalhos foram tomados como base para a definição dos aspectos do ambiente a ser construído. Alguns são importantes para a escolha de definições que devem ser refletidas em sua arquitetura. Outros auxiliam no estabelecimento de requisitos e restrições já considerados como fundamentais para a execução de WLDs.

Parte III: Projeto da Solução

No Capítulo 4 são apresentados os requisitos, restrições e aspectos cobertos pelo ambiente proposto neste trabalho. A arquitetura da solução, com seus detalhes estruturais e dinâmicos são detalhados. Diagramas que auxiliam no entendimento da proposta também são disponibilizados neste capítulo.

Parte IV: Implementação da Solução

No Capítulo 5 são descritos detalhes da implementação do *LONGWisE4cloud*. Diagramas, referências a tecnologias, linguagens de programação e modelos de dados são apresentados de forma a esclarecer como a construção do ambiente foi realizada.

Parte V: Avaliação da Solução

No Capítulo 6 alguns experimentos utilizando o *LONGWisE4cloud* são detalhadamente descritos e contribuem para apresentar o desempenho da execução dos WLDs no ambiente desenvolvido, validar a hipótese de pesquisa e identificar objetivos atingidos.

Parte VI: Considerações Finais da Tese

Finalmente, o Capítulo 7 discute conclusões a respeito dos resultados obtidos no trabalho de pesquisa e expõe aperfeiçoamentos e inovações que podem ser desenvolvidos a partir desta tese.

2

CONCEITOS BÁSICOS

Definições importantes ligadas ao tema desta tese são apresentadas ao longo deste capítulo. Uma discussão sobre processos de negócio é apresentada na Seção 2.1. Os conceitos de *workflows* e *workflows* de longa duração são apresentados, respectivamente, nas Seções 2.2 e 2.3. Sistemas associados à automação de processos de negócio são comentados na Seção 2.4. A computação orientada a serviços é discutida na Seção 2.5. Características gerais associadas aos sistemas adaptativos são encontradas na Seção 2.6. Finalmente, a computação em nuvem é apresentada na Seção 2.7.

2.1 Processos de Negócio

Um negócio pode ser entendido como um conjunto de pessoas que interagem entre si para executarem uma série de atividades voltadas a entregar valor para os clientes assim como gerar retorno às partes interessadas (ABPMP-BR, 2013). O termo *negócio* abrange, desta forma, organizações de todas as naturezas, com ou sem fins lucrativos, públicas ou privadas, de portes diferentes e que estão inseridas em qualquer segmento público ou de mercado existente. Neste contexto, um processo de negócio pode ser definido como sendo uma série de regras e passos para a solução de um problema corporativo (VERGIDIS; TIWARI; MAJEED, 2008). Os processos são compostos por atividades inter-relacionadas, governadas por regras de negócio e que também são vistas no contexto de suas relações com outras atividades para a montagem de uma visão de sequência e fluxo.

Atualmente, há um aumento dos investimentos, por parte das organizações, na definição, modelagem e automação de seus processos de negócio (ZENG et al., 2011; ANJU et al., 2013; SIADAT; SHAMASBI, 2015; NURHAYATI; FITRISARI, 2016; JAYAKODY et al., 2016). Estes esforços voltados aos processos de negócio tornam estas organizações mais eficientes, já que as atividades a serem assumidas por pessoas ou sistemas podem ser mais bem esclarecidas, direcionadas, publicadas, avaliadas e otimizadas. Estes processos são uma vantagem competitiva em termos de mercado, já que permitem um melhor atendimento das necessidades dos clientes além de possibilitarem que recursos humanos e tecnológicos sejam mais bem utilizados (GONG et al., 2004). Além disso, no contexto da administração pública, o uso efetivo de processos de

negócio é capaz de otimizar a aplicação de recursos públicos bem como melhorar a prestação de serviços à população (OROSZ; OROSZ, 2012; SILVIA; SUHARDI; YUSTIANTO, 2016). A modelagem, e consequente automação, de processos de negócio pode ser feita através de notações já bastante difundidas no mercado tais como *Business Process Model and Notation* (*BPMN*) (OMG, 2011) e/ou *Web Services Business Process Execution Language* (*WS-BPEL*) (OASIS, 2007).

A Figura 2.1 apresenta um processo de negócio simplificado, contendo apenas três atividades, relativo à geração e envio por *e-mail* da cobrança direcionada aos clientes de uma administradora de cartões de crédito. A modelagem foi construída com o uso da notação *BPMN*. As setas indicam o ordenamento temporal das atividades, ou seja, cada uma só inicia após o término da anterior. Além disso, as engrenagens observadas no canto superior esquerdo de cada atividade indicam que ela é executada por elementos computacionais (e.g., serviços).

Figura 2.1: Processo para Cobrança dos Clientes de um Cartão de Crédito

Fonte: O próprio autor.

No primeiro passo (*Selecionar Transações Realizadas*), as transações relativas às compras realizadas pelos clientes em um determinado período são identificadas e selecionadas tendo em vista a composição da cobrança. No segundo (*Gerar Faturas*), os valores relativos às compras são consolidados em um título a ser pago. Tributos, descontos, além de outros lançamentos financeiros, tais como juros e multas referentes a atrasos de pagamento em faturas anteriores, são incorporados à cobrança neste passo. Finalmente os títulos de cobrança (faturas) são enviados aos clientes no último passo (*Enviar Faturas por e-mail*).

Finalmente, um conceito relevante associado ao domínio dos processos de negócio é o *Business Process Management* (*BPM*), uma estratégia de gerenciamento que envolve métodos, técnicas e ferramentas voltadas a oferecer suporte à análise, projeto, implantação e acompanhamento dos processos de negócio de uma organização (AALST; HOFSTEDE; WESKE, 2003). A adoção das práticas relativas ao *BPM* vem se tornando cada vez mais comum em ambientes corporativos, uma vez que elas auxiliam as organizações a lidarem melhor com todo os desafios relativos aos processos de negócio e, assim, conseguirem obter resultados positivos, tais como ganhos financeiros, satisfação de clientes ou uma melhoria nas rotinas operacionais internas.

2.2 *Workflows*

À medida que as organizações investem na formalização, automação e otimização de seus processos de negócio, o uso intensivo de sistemas de software começa a crescer proporcio-

nalmente. Neste contexto, ganham destaque os chamados *workflows*, abstrações utilizadas para representar processos de negócio, permitindo que eles sejam modelados, implantados e distribuídos ao longo de um processamento computacional composto de diversos passos, conhecidos como tarefas ou atividades, que podem viabilizar a automação de tais processos (DECKER et al., 2010).

Sendo assim, a disseminação do uso de *workflows* vem atraindo cada vez mais o interesse das organizações, que podem automatizar seus processos de forma controlada e padronizada, dispondo de um recurso capaz de modelar, dividir e distribuir fisicamente o processamento em diversas tarefas, também de forma controlada, organizada e automatizada (PATHIRAGE et al., 2011). Com o uso destas abstrações, as organizações podem, assim, obter um suporte de software que seja mais próximo das formas observadas atualmente na construção de produtos, prestação de serviços e onde seus processos podem ser executados por serviços computacionais especializados (NITTO et al., 2008).

Finalmente, modelagens como a observada na Figura 2.1 são voltadas à definição formal de processos de negócio, conforme mencionado na Seção 2.1. Entretanto, elas são comumente aplicadas na própria automação destes processos através de *workflows* executados em ferramentas como as descritas na Seção 2.4. A execução destes *workflows* pode ser realizada, por exemplo, com base na invocação de serviços computacionais associados as suas atividades.

2.3 *Workflows* de Longa Duração

Conforme apresentado no Capítulo 1.1, *workflows* de longa duração são tipos especiais de *workflows* cuja principal característica é o considerável consumo de tempo para gerarem resultados e serem concluídos. As razões para a longa duração podem ter diversas origens:

- A natureza de algumas regras de negócio;
- Falhas em pontos específicos do processo;
- A necessidade de grande quantidade de recursos computacionais para se processar atividades especializadas;
- Exigência de intervenções humanas para a conclusão de alguma etapa do processo; e/ou
- A interdependência entre atividades onde uma ou algumas levam um tempo longo para serem concluídas.

Hoje em dia, há um aumento, nas organizações, da complexidade de seus processos de negócio, tanto nas suas estruturas (e.g., participantes, responsabilidades, conexões) como em suas dinâmicas (e.g., regras de execução, protocolos de comunicação, fluxo de informações). Isto favorece o surgimento de *workflows* de longa duração sendo executados em ambientes

corporativos. Processos mais complexos, em geral, demandam computações mais intensas, o que frequentemente aumenta o tempo necessário para a conclusão destes processamentos. Como exemplos de processos que podem ser associados a WLDs estão:

- Os passos para uma operadora de planos de saúde permitir que um cliente se submeta a um procedimento médico especializado: podem ser necessárias análises contratuais relativas ao plano do paciente, levantamento de valores envolvidos com os custos para a realização do procedimento ou até mesmo uma auditoria médica para avaliar a necessidade da intervenção;

- A burocracia para se autorizar a aposentadoria de um contribuinte da Previdência Social: levantamentos sobre tempo de contribuição, deduções de valores a serem pagos, além de auditorias médicas em casos específicos são etapas que podem levar meses para serem concluídas;

- A obtenção dos valores de ressarcimento relativos a um seguro automotivo: evidências sobre a ocorrência do sinistro precisam ser coletadas, análise das condições sob as quais um roubo ou incêndio aconteceram, deduções de valores a serem pagos entre outros pormenores que podem levar dias, semanas ou até meses para serem concluídos também;

- A geração da cobrança destinada aos clientes de uma administradora de cartões de crédito: o levantamento de todas as despesas realizadas, valores de anuidade, cálculo de juros, multas, descontos e a montagem final das faturas a serem pagas podem levar um tempo considerável para serem concluídos se o volume de informações for consideravelmente grande.

Um aspecto importante a se observar neste contexto dos WLDs é que a noção de *longa duração* é algo relativo. Em certas situações, alguns minutos já podem ser considerados um tempo longo. Um exemplo deste cenário é um processo de negócio que envolve a compra com um cartão de crédito. Consumidores normalmente não querem esperar minutos até que uma transação eletrônica seja completada. A companhia responsável pelo cartão de crédito também pode ter problemas, como consecutivas falhas em responder a transações remotas, que podem levar um consumidor a desistir de uma compra ou, em condições mais extremas, abandonar o uso daquele cartão em particular, caso estas experiências inconvenientes se tornem frequentes. Em outros cenários, alguns dos quais mencionados anteriormente, o tempo de execução de um processo de negócio pode durar semanas.

Independente desta relatividade quanto à percepção da longa duração, os problemas mencionados na Seção 1.1, tais como a interrupção de um serviço responsável por uma atividade, podem ocorrer tanto no contexto dos WLDs assim como em *workflows* comuns. Entretanto, quando WLDs estão sendo executados, o tratamento destes problemas precisa de uma diferenciação. O elevado consumo de tempo normalmente implica em mais recursos computacionais sendo

utilizados para executar estes *workflows*. O desperdício de uma quantidade maior de recursos pode envolver custos maiores, como o preço para utilizar uma infraestrutura de nuvem computacional. Além disso, esperar para que um *workflow* seja reprocessado completamente pode ser inviável em cenários onde o tempo envolvido neste processamento seja consideravelmente alto. Por exemplo, um *website* de viagens pode utilizar um *workflow* regular para procurar por valores de diárias de hotéis, bilhetes de avião ou aluguéis de automóveis para um viajante. Se uma interrupção ocorre durante a busca, o *workflow* pode ser reiniciado. A velocidade com que *websites* existentes atualmente podem gerar suas respostas e considerando que esta não é uma atividade crítica para o usuário, este reprocessamento não necessariamente implica em um problema relevante. Por outro lado, o *workflow* associado com a geração de cobrança para milhares (ou milhões) de clientes de uma grande companhia não pode ser visto com a mesma simplicidade do *website* de viagem. Se o processo consome um tempo considerável e está quase finalizado quando há uma interrupção, por exemplo, pode atrasar o envio da cobrança aos clientes, o que pode levar a uma perda financeira para a organização.

Sendo assim, um ambiente que suporta a execução de *workflows* de longa duração deve apresentar mecanismos apropriados para atender a suas particularidades, tais como uma maior probabilidade de mudanças em regras de negócio ou o consumo de recursos computacionais, que, em geral, também tende a ser elevado em virtude do tempo ao longo do qual o processo permanece sendo executado até a sua conclusão. Aplicar soluções desenvolvidas para *workflows* tradicionais não é uma abordagem apropriada, uma vez que estas particularidades dos WLDs devem possuir soluções especificamente projetadas para elas. Por exemplo, o descarte de dados já gerados por um processamento longo que foi interrompido pode ser algo inaceitável se o esforço para regerá-los for grande, uma vez que isto pode gerar prejuízos razoáveis, inclusive financeiros, para uma organização.

Uma vez que WLDs são tipos especiais de abstrações para formalização e automação de processos de negócio, é possível afirmar que uma proposta que associe estes processos a atividades e rotinas computacionais também é válida para eles. Um WLD pode ser formalmente definido e modelado utilizando notações que já são populares, tais como *BPMN* e *WS-BPEL*. Depois de formalizados, eles podem ser submetidos a ambientes que suportam sua execução. Nestes cenários de processamento de WLDs, um conceito importante é o de *instância de um workflow* (FRANK; FONG; LAM, 2010). Uma vez que um *workflow* é definido e submetido a um ambiente para ser executado, muitos processamentos daquele *workflow* podem ser realizados em paralelos. Cada uma destas execuções é chamada de uma *instância ativa* do *workflow* (ou simplesmente *instância*).

2.4 Sistemas de Automação de Processos de Negócio

Um tipo de software importante e relacionado ao domínio dos *workflows* juntamente com seu papel na automação de processos de negócio são os chamados sistemas de gerenciamento de

workflows, ou, no inglês, *Workflow Management System* (WfMSs) (FERME et al., 2017). Estes ambientes constituem soluções de software que viabilizam a automação dos processos através da definição e gerenciamento da execução de *workflows* normalmente com o apoio do uso de notações como as previamente citadas *BPMN* e *WS-BPEL*.

Já os *Business Process Management Systems* (*BPMS*) ou sistemas de gerenciamento de processos de negócio (AALST; HOFSTEDE; WESKE, 2003) são softwares capazes de dar suporte a todas as responsabilidades associadas ao *BPM* listadas na Seção 2.1.

Há, conforme explicado na Seção 2.2, uma relação direta entre *workflows* e processos de negócio. Pode-se dizer que o *BPM engloba* o gerenciamento de *workflows* (AALST; HOFSTEDE; WESKE, 2003). Sendo assim, é comum encontrar nos *BPMS* ferramentas para modelar e executar processos de negócio que, na verdade, estão realizando também a definição e a execução de *workflows* associados a estes processos. Como exemplos destes *BPMS* podem ser citados o *IBM Business Process Manager* (IBM, 2018) , o *Red Hat JBoss BPM Suite* (Red Hat, 2018) e o *Oracle Business Process Management* (Oracle, 2018).

2.5 Computação Orientada a Serviços

De acordo com Papazoglou e Heuvel (2007), serviços são módulos computacionais bem definidos e autocontidos, ou seja, capazes de manter seu próprio estado independente da aplicação que os utiliza ou de contextos e estados de outros serviços. Eles são descritos a partir de linguagens de definição padronizadas, possuem interfaces públicas a partir das quais se pode estabelecer interações com os mesmos e se comunicam entre si a partir da invocação de operações presentes nestas interfaces. Sendo assim, eles são capazes de, coletivamente, darem suporte a atividades relativas a processos de negócio, por exemplo. Neste contexto, a *Computação Orientada a Serviços*, do inglês *Service-Oriented Computing* (*SOC*), tem como objetivo construir sistemas baseados nestes módulos computacionais. *SOC* associa uma maior robustez às soluções, confere adaptabilidade às mesmas e incrementa a possibilidade de construir softwares que atendam melhor à dinâmica das organizações (NITTO et al., 2008).

O desenvolvimento de soluções orientadas a serviços pode usar uma abordagem envolvendo a composição de módulos entregues por fornecedores diferentes (KHALAF; KELLER; LEYMANN, 2006). Algumas das vantagens do uso de *SOC* incluem:

- O esforço para desenvolver uma aplicação passa a ser acelerado, uma vez que o foco é mantido mais na definição de requisitos e regras relativas ao negócio, iniciando a implementação da solução propriamente dita através da composição de serviços já construídos por fornecedores (LU; CAI; LI, 2011);
- Os clientes podem escolher entre diferentes provedores de módulos computacionais aqueles que melhor atendem seus requisitos específicos, tais como segurança, desempenho, garantia de qualidade, preço e assim por diante (NITTO et al., 2008);

- Falhas em uma parte da solução causada por um serviço específico podem ser eliminadas através de tratamentos localizados apenas sobre o componente que está causando o problema (JUHNKE; DORNEMANN; FREISLEBEN, 2009); e

- Reconfigurações específicas (mudanças de parâmetros gerais, retomada da execução de rotinas) para alguns destes serviços podem ser aplicadas de forma dinâmica e isolada (JUHNKE; DORNEMANN; FREISLEBEN, 2009).

Conforme já mencionado anteriormente, *workflows* podem ser entendidos como um conjunto de atividades que executam processos de negócio definidos para cenários de uma organização. Neste contexto de *SOC*, estas soluções podem ser construídas sobre uma infraestrutura onde cada atividade é executada por um ou vários serviços. Desta forma, estes *workflows* (assim como os WLDs) podem ser implementados como um agrupamento de módulos computacionais independentes capazes de executar um processo particular e formalmente definido (VERGIDIS; TIWARI; MAJEED, 2008).

2.5.1 Arquitetura Orientada a Serviços

A *Arquitetura Orientada a Serviços*, do inglês *Service-Oriented Architecture (SOA)* (PAPAZOGLOU; HEUVEL, 2007), é uma abordagem bastante utilizada e baseada nos conceitos de *SOC*. Quando se trabalha com *SOA*, a unidade de construção fundamental de uma solução é o serviço. Pode-se afirmar que *SOA* corresponde a um estilo arquitetural capaz de trazer benefícios a aplicações, tais como a capacidade de se adaptar e a facilidade de estabelecer uma interoperabilidade envolvendo sistemas que possam fazer parte de negócios diferentes (MACLENNAN; BELLE, 2014).

Os princípios associados à elaboração de uma arquitetura orientada a serviços, tais como a comunicação baseada na invocação de operações presentes nas interfaces destes módulos computacionais, são independentes de tecnologias, tais como Java ou *web services* (GEORGAKOPOULOS; PAPAZOGLOU, 2008). Além disso, considerando que é nos serviços que estão concentradas todas as funcionalidades relativas à execução dos processos de negócio, a construção de uma abordagem baseada em *SOA* requer que estes serviços atendam a um conjunto básico de requisitos (GEORGAKOPOULOS; PAPAZOGLOU, 2008):

- Autocontido: o serviço deve ser capaz de manter seu próprio estado, independente da aplicação que o utiliza ou de outros serviços;

- Independente de plataforma: o uso de um serviço independe de seus detalhes internos de implementação tais como a linguagem de programação utilizada para criá-lo; e

- Passível de ser encontrado, invocado e incluído em composições de forma dinâmica: um serviço pode ser armazenado em algum tipo de repositório onde clientes podem encontrá-lo através de buscas baseadas, por exemplo, em suas propriedades. Uma vez

encontrado, ele pode ser invocado a partir de operações presentes em sua interface e, assim, entrar na composição de aplicações construídas sobre os princípios da orientação a serviços.

Ainda dentro do escopo da arquitetura orientada a serviços, os seguintes conceitos são importantes para este estilo arquitetural (PAPAZOGLOU; HEUVEL, 2007):

- Provedores: são os componentes da arquitetura que provêm os serviços computacionais especializados capazes de realizar ações como processamentos ou armazenamentos de informações;
- Consumidores: componentes que consomem serviços disponibilizados pelos provedores; e
- Registro de serviços: mecanismo onde os consumidores são capazes de localizar serviços colocados à disposição pelos provedores.

Finalmente, trabalhos como os de Anjum e Budgen (2012), Machado, Areias e Cunha (2016) e Yimin, Ping e Qi (2017) são exemplos de como *SOA* é utilizada na criação de soluções computacionais encontradas atualmente, inclusive quando o domínio das aplicações diz respeito aos *workflows* de longa duração, como pode ser observado nas soluções propostas por Montagut, Molva e Golega (2008) e Juhnke, Dornemann e Freisleben (2009).

2.6 Sistemas Adaptativos

O crescente aumento da complexidade dos sistemas computacionais tem contribuído para que o conjunto de requisitos a serem exigidos dos mesmos também passe por uma considerável ampliação. Versatilidade, resiliência, robustez, facilidade de customização são apenas alguns aspectos que cada vez mais passam a ser indispensáveis nas aplicações desenvolvidas atualmente. Neste contexto, popularizam-se os chamados sistemas adaptativos, soluções capazes de modificar seu comportamento em resposta à percepção do ambiente e do próprio sistema (CHENG et al., 2009). Comportamentos adaptativos estão hoje presentes em diversas áreas relacionadas à computação (CHENG et al., 2009): robótica, computação autônoma, sistemas embarcados, arquiteturas orientadas a serviços, redes móveis e assim por diante. A adaptação a mudanças que ocorrem à medida que o sistema executa suas ações previamente programadas passa, então, a ser algo desejável ou até mesmo indispensável dependendo do contexto de uso da solução. Por exemplo, um *website* que consegue manter um usuário conectado mesmo após uma falha em seu servidor através de um redirecionamento automático da geração de respostas a solicitações deste usuário para outra infraestrutura pode ser enxergado como uma característica desejável neste tipo de serviço. Por outro lado, um carro autônomo desviar automaticamente de uma colisão com um obstáculo pode ser entendido como um requisito indispensável para a viabilização deste tipo de produto.

Existem ainda muitas discussões na literatura a respeito das características que realmente definem um sistema adaptativo ou até mesmo sobre termos como *autoadaptativo* e simplesmente *adaptativo*. Em Salehie e Tahvildari (2009) há a proposta de que o termo *autoadaptativo* só deve ser utilizado se o sistema possuir um conjunto de propriedades também identificadas pelo prefixo *auto*, tais como *autoconfiguração*, *autoreparo* e *autoproteção*. Em Bruni et al. (2012) defende-se a ideia de que o termo *adaptativo* é aplicável a componentes que são passíveis de adaptações e os chamados dados de controle, responsáveis por seu comportamento, são modificados em tempo de execução. Neste mesmo trabalho, *Autoadaptativo* seria uma classificação aplicável àqueles componentes capazes de modificar seus próprios dados de controle.

Um *framework* para classificar e ajudar a entender melhor as soluções adaptativas foi proposto por Cheng et al. (2009). Uma proposta voltada a classificar sistemas autônomos por *níveis* de acordo com o grau de automação e intervenção externa, tal como a ação humana, nas adaptações que ocorrem nestes sistemas é encontrada em (NAMI; SHARIFI, 2007). Até mesmo práticas e desafios relativos à engenharia de software associada ao desenvolvimento deste tipo de aplicação tem recebido especial atenção (LEMOS et al., 2013), dada a crescente importância de sistemas de software adaptativos hoje em dia. Ações como estabelecimento de requisitos, projeto arquitetural, modelagem e manutenção são associados a desafios e propostas para melhor atenderem ao ciclo de desenvolvimento destes sistemas.

Para o contexto dos *workflows* de longa duração, as adaptações capazes de serem aplicadas nestas abstrações pelos sistemas que gerenciam suas execuções são de grande importância. O tempo consumido até a conclusão das atividades dos WLDs é consideravelmente alto, o que aumenta, conforme discutido no Capítulo 1, as chances de ocorrerem mudanças ao longo do processamento que afetem diretamente a execução destas atividades. Sendo assim, as adaptações para que estes *workflows* possam ter suas ações concluídas com o mínimo de problemas, tais como perdas de dados ou reprocessamentos, passam a ser elementos fundamentais para os cenários que dependem dos resultados gerados por estes WLDs, como é o caso de ambientes corporativos.

2.6.1 Quiescência

No domínio de aplicações onde processos de negócio, *workflows* e composição de serviços são encontrados, os componentes especializados responsáveis pela execução das atividades que constituem os processos de negócio são entendidos como nós que estão envolvidos em transações destinadas a gerar os resultados dos processamentos.

Neste contexto, a quiescência (KRAMER; MAGEE, 1990) é uma propriedade entendida como o estado de um nó que se encontra em uma situação de passividade e onde não há transações presentes no cenário que este nó precise aceitar requisições ou servir. A passividade de um nó é observada quando (KRAMER; MAGEE, 1990):

1. Ele não está envolvido em uma transação que ele tenha iniciado; e

2. Ele não iniciará novas transações.

Sendo assim, um nó é considerado quiescente se ele se encontra passivo (atendendo às duas condições enumeradas anteriormente) e também:

1. Não está servindo a nenhuma transação; e

2. Nenhuma transação foi iniciada ou será iniciada por outros nós e que requeiram o envolvimento deste nó.

Logo, para atingir um estado de quiescência, um nó não depende apenas dele. As conexões com outros nós de uma composição onde o mesmo se encontra inserido também é relevante para a quiescência ser observada.

A relevância da quiescência em mecanismos adaptativos pode ser observada quando reconfigurações dinâmicas de componentes precisam ser feitas, por exemplo. Nestes casos, medidas para que as mesmas sejam conduzidas de forma segura e estável precisam ser tomadas. Para conseguir isso, a quiescência é um aspecto importante a ser tratado. A passividade dos nós envolvidos e a ausência do engajamento deles em transações ativas auxilia a obtenção de mudanças sem perdas de informações significantes ou interrupções inesperadas de instruções que estejam sendo executadas.

2.7 Computação em Nuvem

As nuvens computacionais, elementos fundamentais da chamada computação em nuvem, são grandes reservatórios de recursos computacionais virtualizados, tais como hardware, plataformas de desenvolvimento ou serviços, que são facilmente acessíveis e utilizáveis (VAQUERO et al., 2009). Sendo assim, o uso destes recursos pode ser feito de forma mais eficiente, uma vez que a alocação dos mesmos para um usuário ocorre de acordo com a demanda existente em um determinado momento (ou, no inglês, *on demand*). Com isto, evitam-se alocações desnecessárias, situação onde eles poderiam ficar ociosos ao invés de estarem sendo aplicados onde há uma real necessidade. Quando há cobranças pelo uso destes recursos, como é o caso de infraestruturas fornecidas por empresas de mercado, o modelo para a geração dos valores a serem pagos pelos clientes é baseado em cálculos que levam em consideração o uso efetivo dos recursos (ou, no inglês, *pay-per-use*) e os contratos entre cliente e fornecedor da infraestrutura normalmente envolvem garantias de disponibilidade e qualidade do que está sendo ofertado para uso.

Sendo assim, podem ser elencados alguns princípios básicos que caracterizam a computação em nuvem (ROSENBERG; MATEOS, 2010):

1. Os recursos computacionais se encontram disponíveis para qualquer usuário que esteja inscrito para utilizá-los;

2. Há a virtualização dos recursos computacionais visando a maximização da utilização do hardware;

3. Elasticidade: a alocação e a liberação de recursos ocorrem de acordo com a demanda;

4. Máquinas virtuais são criadas e removidas do ambiente automaticamente;

5. O pagamento ocorre apenas pelos recursos efetivamente utilizados.

Com a computação em nuvem, também se popularizaram os chamados modelos de serviços (ou, no inglês, *as a Service*) (LIU et al., 2012), que são caracterizados por serem oferecidos seguindo a mesma ideia da oferta e da cobrança de acordo com a demanda e que, dependendo do serviço ofertado, podem receber diferentes variações em seus nomes, como, por exemplo (MOTAHARI-NEZHAD; STEPHENSON; SINGHA, 2009):

- Infraestrutura como Serviço (ou, no inglês, *IaaS - Infrastructure as a Service*): recursos de hardware são oferecidos aos usuários finais;

- Plataforma como Serviço (ou, no inglês, *PaaS - Platform as a Service*): recursos necessários à construção de aplicações, cobrindo desde a definição, passando pela codificação, testes e/ou manutenção; e

- Software como Serviço (ou, no inglês, *SaaS - Software as a Service*): soluções especializadas que também podem ser consumidas pelos usuários através da Internet, por exemplo, sem a necessidade de instalações locais (modelo tradicional de aquisição e uso de aplicações).

A Figura 2.2 apresenta os modelos anteriormente descritos dispostos em camadas. Como exemplo de *SaaS* podem ser citados o *Gmail* (GMAIL, 2017), *DropBox* (DROPBOX, 2017) e o *Microsoft Office 365* (MSOFFICE365, 2017). *AWS Elastic Beanstalk* (AWS-EB, 2017) e o *Heroku* (HEROKU, 2017) são exemplos de plataformas oferecidas como serviço (*PaaS*). *Amazon EC2* (AMAZON-EC2, 2017), *Amazon S3* (AMAZON-S3, 2017) e Microsoft Windows Azure (AZURE, 2017) são soluções classificadas como *IaaS*.

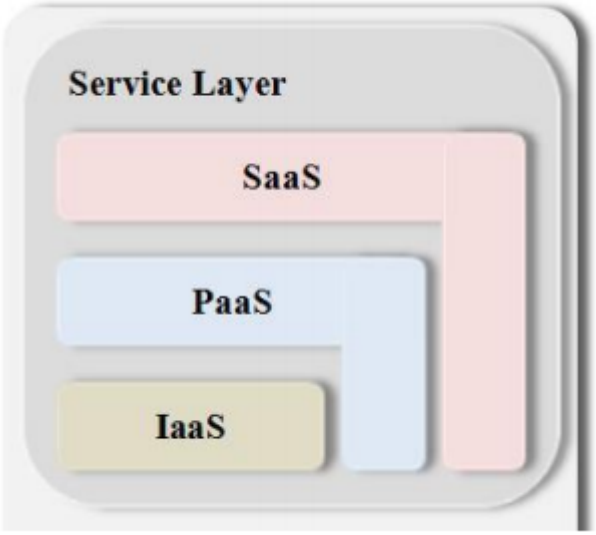
Ainda segundo Liu et al., uma infraestrutura de nuvem pode ser disponibilizada de três formas distintas, dependendo do nível de exclusividade com que os recursos da infraestrutura são colocados à disposição de sues clientes:

- Nuvens públicas: os recursos do ambiente são disponibilizados ao público geral através de uma infraestrutura de rede pública;

- Nuvens privadas: o acesso aos recursos da infraestrutura computacional é colocado à disposição de apenas um cliente;

- Nuvens híbridas: composição de duas ou mais nuvens computacionais que pode reunir nuvens públicas e privadas interligadas a partir de tecnologias que permitam a portabilidade de dados e aplicações implantados em nuvens distintas.

Figura 2.2: Modelos de Serviços em Nuvens



Fonte: (LIU et al., 2012)

Neste contexto relativo à nuvem, também são bem comuns as chamadas aplicações para múltiplos clientes ou *inquilinos* (do inglês, *multi-tenancy applications*) (PATHIRAGE et al., 2011). Estes softwares podem ser comercializados seguindo o modelo *SaaS* e servirem a vários clientes simultaneamente. Neste caso, o fornecedor da aplicação garante a disponibilidade da solução além da garantia da integridade dos dados de cada um destes clientes e a segurança necessária para não haver um acesso indevido aos dados de um cliente por parte de outro que também utiliza o software.

Em geral, estes recursos das nuvens computacionais são facilmente acessíveis através da Internet. Sendo assim, o advento da computação em nuvem atraiu a atenção não apenas de empresas como também de pessoas, uma vez que a virtualização e o modelo utilizado para disponibilizar recursos computacionais sob demanda sugerem uma forma simples de se acessar e adquirir componentes de software e hardware. Há algum tempo, estes acessos e aquisições só seriam possíveis através da compra de equipamentos ou instalação de softwares localmente em estações de trabalho do usuário final. Além do mais, a ideia de se pagar apenas pela utilização também se mostrou bastante interessante para clientes e fornecedores, uma vez que, para os primeiros, o custo não envolve nada além do que foi consumido e, para os últimos, recursos que poderiam ficar ociosos são oferecidos para quem está precisando e também disposto a pagar pelo uso.

Finalmente, todos estes aspectos da computação em nuvem são capazes de aumentar a eficiência na execução de *workflows*, uma vez que estas características permitem ganhos como o aumento do desempenho, maior escalabilidade e disponibilidade dos sistemas responsáveis por estas execuções. Desta forma, as organizações são capazes de otimizar suas rotinas operacionais apoiadas por processos automatizados com o uso dos *workflows*, favorecendo as ações que visam atingir suas metas e objetivos corporativos.

2.8 Considerações Finais

Ao longo deste capítulo foram apresentados conceitos importantes para a construção do projeto de pesquisa descrito nesta tese. Os processos de negócio, assim como os *workflows* e os *workflows* de longa duração foram detalhados juntamente com sua importância para cenários como os contextos corporativos que procuram automatizar cada vez mais seus ambientes operacionais. Além disso, sistemas que podem auxiliar a viabilização destas automações e que podem ser encontrados não apenas na literatura, mas também no mercado de soluções de software, foram descritos. Finalmente, a computação orientada a serviços, os sistemas adaptativos e as nuvens computacionais, áreas da Ciência da Computação que tiveram influência considerável na construção deste projeto de pesquisa, também foram apresentadas. Alguns destes conceitos, tais como *SOC*, *SOA* ou a computação em nuvem estarão presentes em diversas propostas discutidas no próximo capítulo, que apresenta trabalhos relacionados ao contexto dos WLDs, o que reforça a importância deles para o domínio relativo a estas entidades, foco principal desta tese.

3

TRABALHOS RELACIONADOS

Trabalhos relacionados ao domínio dos WLDs serão apresentados neste capítulo. Parte destes trabalhos tem um foco maior na definição de mecanismos relacionados ao controle do fluxo de execução dos *workflows*, tendo sido agrupadas na Seção 3.1. Outro conjunto de trabalhos associado ao contexto dos WLDs pode ser identificado a partir de suas contribuições ao suporte adaptativo à execução destes *workflows*. Estas propostas são discutidas na Seção 3.2. A Seção 3.3 apresenta trabalhos relacionados a diversos aspectos da modelagem e execução de WLDs, mas que não têm seu foco voltado ao controle de fluxo ou à adaptação de WLDs. Um quadro comparativo dos trabalhos apresentados juntamente com as considerações gerais a respeito destas propostas são encontrados na Seção 3.4.

3.1 Controle do Fluxo de Execução em *Workflows* de Longa Duração

Hu et al. (2010) propõem um novo modelo associado a transações de longa duração denominado LTMBWF (*Long Running Transaction Model Based on WorkFlow*). Neste modelo, as transações consistem de subtransações que podem ser confirmadas ou desfeitas isoladamente. Regras para execução de transações e recuperação de falhas baseadas em análises e seleção de estratégias voltadas para a execução das subtransações são estabelecidas. Além disso, uma das características principais do trabalho é o uso das compensações, propostas por Garcia-Molina e Salem (1987) para executar transações de longa duração em ambientes envolvendo aplicações e sistemas de bancos de dados.

Hu et al. identificam as subtransações que compõem uma transação, analisam e classificam as mesmas para que as chamadas *transações críticas* se submetam a regras de execução diferenciadas. Uma subtransação é dita crítica quando envolve operações manuais e possuem, assim, um alto custo para serem desfeitas quando ocorre, por exemplo, o cancelamento de parte de um processamento. Também é apresentado na proposta um algoritmo (STCBCT - *Sub-Transaction Committing Based on Critical Transaction*) que posterga a confirmação (*commit*) destas subtransações, conduzindo as demais subtransações que compõem a transação longa através de um escalonamento de ações que não interfira nos resultados finais produzidos, mas que permitam a geração destes resultados de forma mais eficiente. Postergando as confirmações

destas ações críticas, há um melhor uso de recursos computacionais e um aumento na robustez da execução das transações longas. Isto ocorre em virtude da otimização no desempenho das transações, uma vez que menos ações como as relativas ao desfazimento de operações serão realizadas. Além disso, falhas podem ser tratadas de forma localizada, sem envolver necessariamente toda a transação.

Apesar das vantagens apresentadas pela proposta, não são tratados aspectos tais como a reconfiguração dinâmica envolvendo trocas de componentes envolvidos nas transações longas ou reuso de resultados intermediários gerados pelas transações.

O uso das compensações consiste em um relaxamento das propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade), muito utilizadas para a execução de transações nos sistemas de bancos de dados. A proposta de Garcia-Molina e Salem exige que uma transação seja enxergada como composta por subtransações e, para cada uma delas, um mecanismo de compensação deve ser associado. Ao contrário de um desfazimento (*rollback*) convencional, cujas ações se limitam a desfazer operações, a compensação atua em um nível semântico, executando ações que devem levar dados e estados envolvidos com uma transação (ou subtransação) já confirmada a uma situação de consistência e integridade. Logo, a compensação exige mais esforço do que uma ação comum de desfazimento, já que se trata de um mecanismo mais complexo.

O uso de mecanismos envolvendo compensações traz vantagens como a liberação de recursos, tais como dados ou processadores, de forma que outras transações possam iniciar ou continuar sua execução. Em transações realizadas em um contexto que segue rigidamente as propriedades ACID, recursos utilizados em uma transação podem ficar bloqueados por ela até sua conclusão ou seu desfazimento, evitando efeitos colaterais por conta de ações como o acesso concorrente a estes recursos. Para ambientes onde as transações são mais complexas e longas, este tipo de bloqueio pode causar transtornos como um incremento ainda maior nos tempos de processamento.

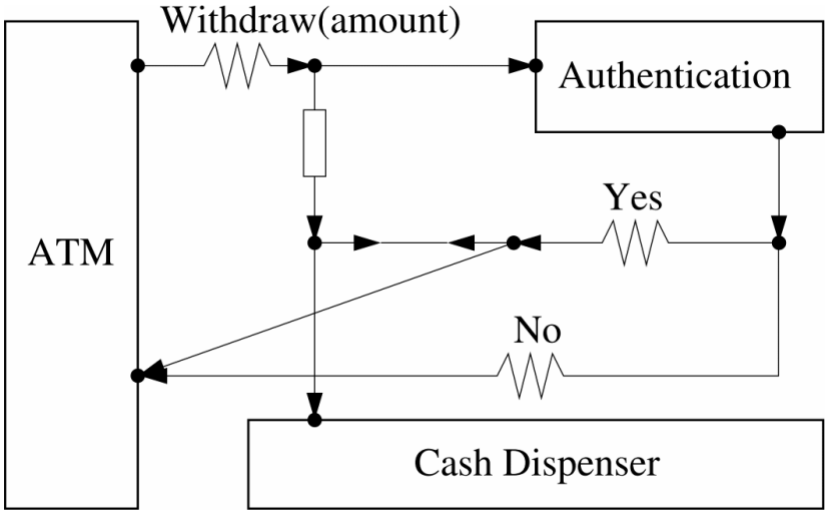
Neste cenário de uso de compensações em transações de longa duração, há esforços na formalização (BOCCHI; LANEVE; ZAVATTARO, 2003; YIN et al., 2005; VAZ; FERREIRA; RAVARA, 2009; CAIRES; FERREIRA; VIEIRA, 2009; XIAOLAN; HONG, 2010) e no desenvolvimento de ferramentas, como a linguagem *StAC* (*Structured Activity Compensation*) (BUTLER; FERREIRA, 2000) e suas extensões (HAO; ZHONG, 2011). Estas ferramentas viabilizam o uso prático de transações de longa duração com mecanismos de compensação.

Apesar das compensações estarem presentes em diversos trabalhos associados ao contexto do WLDs, elas exigem grande esforço de implementação, uma vez que não se tratam, como mencionado anteriormente, de simples *rollback*. Além disso, há cenários onde ocorre a interrupção da transação e o uso de compensações não garante a consistência dos dados e dos estados dos objetos (GREENFIELD et al., 2003). Um exemplo pode ser observado quando *workflows* executam atividades complexas que se estendem por ambientes corporativos distintos e onde um resultado intermediário, uma vez cancelado, pode implicar em ter que se cancelar o

processo por completo, já que um elemento chave para toda a transação foi comprometido. Para evitar estes problemas, alguns trabalhos utilizam alternativas diferentes das compensações para garantir ou melhorar a robustez, a integridade e a consistência em transações longas. Para isto são usadas replicações de recursos computacionais, estratégia de identificação de falhas e até reconfigurações dinâmicas.

O trabalho proposto por Meng e Arbab (2010) também utiliza as compensações em sua proposta. Os autores definem regras para o gerenciamento de composições de *web services* especificadas em uma linguagem de alto nível voltada à definição de aplicações orientadas a serviços denominada TNCA (*TraNsactional Constraint Automata*). Além disso, o trabalho se baseia na linguagem de coordenação Reo, que identifica visualmente os componentes que fazem parte de um sistema e a interação entre os mesmos, incluindo fluxos de dados, definindo os chamados *circuitos Reo* (ARBAB, 2004). A Figura 3.1 apresenta um exemplo de *circuito Reo* para uma operação de saque em um caixa eletrônico ou *Automatic Teller Machine (ATM*). Quando um saque (*withdraw*) é solicitado à máquina, a mesma invoca o serviço de autenticação que, quando responde positivamente, faz com que um sinal seja enviado ao depósito de dinheiro (*cash dispenser*) da máquina que libera as notas para o cliente. Em caso de resposta negativa por parte do serviço de autenticação, o dinheiro não é liberado.

Figura 3.1: *Circuito Reo* para Transações de Saque em Caixa Eletrônico



Fonte: (MENG; ARBAB, 2010)

A interação entre os elementos que compõem um *workflow* é modelada formalmente, permitindo a antecipação de fluxos alternativos que possam ser percorridos nas transações quando estiverem sendo executadas.

Embora seja uma abordagem direcionada especificamente aos *web services*, ela traz vantagens interessantes para o contexto dos WLDs, como a possibilidade de se especificar, em um único modelo, tanto aspectos internos relativos aos serviços independentes que fazem parte de uma composição como também a coordenação entre estes componentes. A antecipação de eventuais mudanças no fluxo de uma transação e o uso de compensações contribui para a

minimização ou a eliminação de perdas de esforços computacionais.

As compensações, a linguagem e os circuitos *Reo* também são tratadas em outras abordagens, como a proposta de modelagem visual apresentada por Kokash e Arbab (2013), que utiliza estes elementos na formalização de transações longas cobrindo aspectos estruturais e semânticos. Neste trabalho, a coordenação do fluxo de execução de transações envolvendo componentes e serviços suportada pela linguagem *Reo* não se restringe às arquiteturas orientadas a serviços, mas pode servir como uma camada intermediária entre linguagens de descrição (e.g., *BPMN*) e execução/implementação de processos de negócio (e.g., *WS-BPEL* e Java). Desta forma, esta proposta funciona como um *framework* onde projetistas modelam transações e associam a execução das mesmas a conjuntos elaborados de combinações de ações relacionadas a falhas e cancelamento de transações do processo de negócio.

A proposta acrescenta novos elementos à linguagem *Reo*, como tipos adicionais de *canais*, que são elementos de modelagem por onde informações trafegam de acordo com regras previamente estabelecidas pela linguagem (canais pré-definidos) ou pelos projetistas que modelam os *circuitos*. Através destas extensões permite-se o tratamento de requisitos não-funcionais em transações de longa duração (e.g., robustez).

O trabalho propõe ainda verificação formal do comportamento das transações. Entretanto, não há uma explícita preocupação com outros aspectos relevantes para o contexto dos WLDs como, por exemplo, medidas especificamente focadas na escalabilidade dos mecanismos que se dedicam à execução dos mesmos.

Ainda considerando contribuições relacionadas ao fluxo de execução de *workflows*, outro trabalho importante é o de Yang et al. (2014), que apresenta uma arquitetura onde são considerados os chamados *humanware services* ou serviços cuja execução depende das habilidades inerentes a um ser humano, como sua capacidade de decisão, julgamento, análise e assim por diante.

Nesta proposta, os *humanware services (HS)* são associados aos *serviços convencionais* (realizados por rotinas computacionais) através de mecanismos de invocação e agendamento dos *HSs* que passam, naturalmente, a fazer parte do fluxo de execução de um *workflow*. A abordagem propõe um *framework* para o uso de *WS-BPEL* incluindo a possibilidade de uso de parâmetros de *QoS* para selecionar entre um *HS* e um serviço computacional *convencional*.

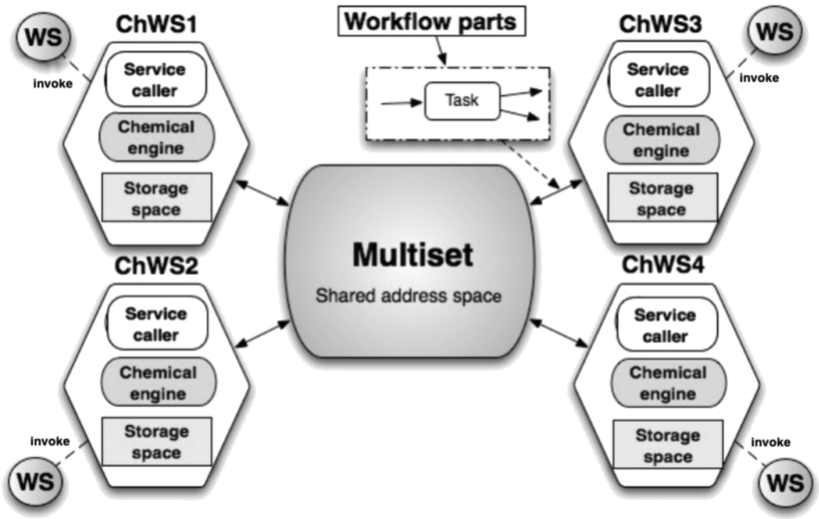
Embora seja uma interessante contribuição para um fluxo de execução de *workflows* contendo atividades heterogêneas, no que diz respeito aos responsáveis pela execução das mesmas, a proposta não atende a particularidades importantes dos *workflows* de longa duração. Elas não desenvolvem, por exemplo, um suporte adaptativo para auxiliar em cenários como mudanças em regras de negócio ou mecanismos para reuso de dados.

Uma outra proposta, cujo foco principal também é o fluxo de execução de *workflows*, mas voltada especificamente à descentralização deste processamento, é encontrada no trabalho de FernÁndez, Tedeschi e Priol (2016). A solução se baseia no *paradigma químico*, um estilo de programação inspirado em uma metáfora baseada na Química. Neste paradigma, os

dados são encarados como *moléculas* que estão *flutuando* em uma *solução química* e que se submetem a *reações* cujas regras constituem o programa propriamente dito. Desta forma, novas moléculas (dados) são geradas de maneira que apenas quando a solução inteira se encontra em um estado *inerte* é que se pode considerar o processamento encerrado. A proposta de Fernandez, Tedeschi e Priol utiliza a linguagem *HOCL (Higher Order Chemical Language)* (BANÂTRE; FRADET; RADENAC, 2006) na sua implementação. Esta linguagem traduz elementos típicos de programação (e.g., entidades, constantes, rotinas) em elementos do *paradigma químico*.

A Figura 3.2 apresenta a visão geral da proposta de Fernández, Tedeschi e Priol. Os chamados *Chemical Web Services (ChWS1, ChWS2, ChWS3, ChWS4)* são estruturas responsáveis diretamente pela execução de passos (*tasks*) do *workflow*. Cada um deles possui um invocador de serviços (*service caller*) que realiza a invocação dos *web services (ws)* que efetivamente executarão as atividades do *workflow*. O *engine químico (Chemical engine)* é um interpretador da linguagem *HOCL* que traduz toda a execução dos *workflows* segundo o paradigma químico. Finalmente, o espaço de armazenamento local *(storage space)* guarda informações importantes para a sincronização e coordenação geral da execução distribuída das atividades do *workflow*, ação realizada pelo *Multiset*.

Figura 3.2: Estrutura de Distribuição da Execução de *Workflows*



Fonte: (FERNÁNDEZ; TEDESCHI; PRIOL, 2016)

O trabalho traz importantes contribuições para a execução de *workflows*, uma vez que a distribuição da execução das atividades permite um melhor uso de infraestruturas distribuídas como nuvens computacionais. No entanto, não há suporte a características típicas dos WLDs, como, por exemplo, adaptações na forma de reconfigurações dinâmicas com gerenciamento de dados e estados para evitar desperdícios de recursos computacionais utilizados em longos processamentos.

3.2 Adaptação em WLDs

No que diz respeito à adaptação, é possível encontrar muitos trabalhos relacionados com os *workflows* comuns (SIEBERT, 1999; WESKE, 2001; CHUNG et al., 2003; NARENDRA, 2004; MÜLLER; GREINER; RAHM, 2004; ARDISSONO et al., 2006; DANG et al., 2008; SAMIRI et al., 2017). Entretanto, a aplicação destas abordagens diretamente nos *workflows* de longa duração não é uma estratégia interessante pois, conforme explicado no Capítulo 1, os WLDs possuem particularidades que devem ser levadas em consideração no gerenciamento de suas execuções. Ao mesmo tempo, nos trabalhos que tratam de WLDs e adaptação, é possível se observar dois tipos principais de estratégias. Alguns utilizam composições e reconfigurações dinâmicas de serviços enquanto outros se baseiam em migrar dados e objetos associados à execução de uma transação dentro de infraestruturas que dão suporte a esta execução.

3.2.1 Composições e Reconfigurações Dinâmicas

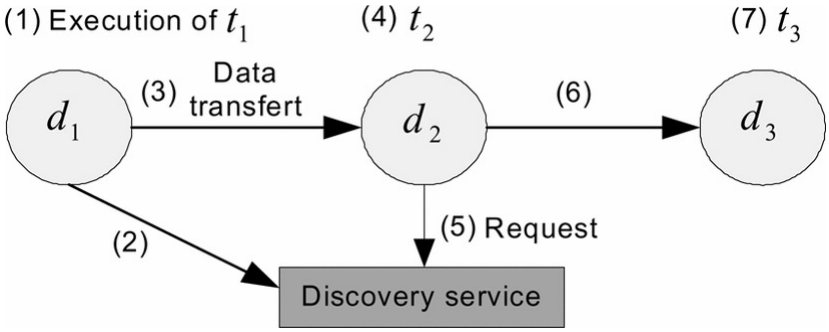
O trabalho de Montagut, Molva e Golega (2008) apresenta um protocolo para a execução dos chamados *workflows* pervasivos: tipos especiais de *workflows* que permitem a execução de processos de negócio em ambientes distribuídos e ubíquos (WEISER, 1995).

A proposta de Montagut, Molva e Golega define um modelo que descreve propriedades de *web services* pertencentes a diferentes fornecedores e um protocolo transacional para o acompanhamento das execuções de *workflows* modelados. Este protocolo é responsável por gerir a execução de processos de negócio através do monitoramento das atividades a serem executadas e a reconfiguração dinâmica quando falhas em componentes são observadas na execução dos *workflows*. Desta forma, a execução do processo de negócio pode atender a requisitos previamente estabelecidos por um cliente (e.g., custo e disponibilidade). Além da escolha dinâmica dos serviços que executarão o *workflow* considerando os requisitos dos clientes, a abordagem também considera as restrições definidas pelos projetistas dos serviços. Um exemplo de restrição é a definição de que uma atividade deve ser atômica.

A Figura 3.3 apresenta uma visão geral dos passos executados em um *workflow* de acordo esta proposta. Os números entre parênteses representam a sequência de ações para a execução das tarefas (t_1, t_2, t_3) ao longo da transação. Os círculos (d_1, d_2, d_3) representam *parceiros de negócio*, cada um dos responsáveis por executar tarefas específicas do *workflow*. Cada uma destas tarefas, antes de ser executada, precisa ser associada a um serviço encontrado a partir de um mecanismo de busca, identificado na figura pelo *Discovery service*. Na Figura 3.3, o *parceiro* que inicia o *workflow* (d_1) executa instruções (tarefa t_1) e só após a busca pelo serviço responsável pela tarefa do próximo *parceiro* (passo (2)) é que ocorre uma transferência de dados para que este *parceiro* seguinte (d_2) possa executar a tarefa (t_2) sob sua responsabilidade. Estas ações de descobertas e invocações de serviços com eventuais transferências de dados ocorrem até a conclusão do *workflow*.

Esta proposta possui uma abrangência que cobre desde aspectos dinâmicos para a com-

Figura 3.3: Modelo para a Execução de *Workflows* Pervasivos



Fonte: (MONTAGUT; MOLVA; GOLEGA, 2008)

posição dos *workflows* até reconfigurações necessárias para minimizar transtornos provenientes de falhas. Entretanto, ela é bastante ligada ao uso de *web services*. Além disso, não há uma preocupação em se estabelecer mecanismos que garantam uma alta disponibilidade para todo o ambiente de gerenciamento destas composições e reconfigurações que ocorrem no ambiente de execução destes *workflows*. Por exemplo, não existe, no trabalho, um suporte explícito à distribuição de componentes do ambiente ao longo de infraestruturas virtualizadas seguindo uma abordagem *as a Service*.

Ainda neste contexto de composições e reconfigurações dinâmicas, Juhnke, Dornemann e Freisleben (2009) propõem um mecanismo para aumentar a robustez na execução de processos de negócio através da composição de *web services*. Neste caso, a troca de componentes que apresentam problemas na execução é realizada pela própria infraestrutura de suporte a esta execução.

Em transações de longa duração, a propagação da falha de um serviço por todo o *workflow* pode implicar em transtornos como a perda de recursos utilizados para a geração de resultados e o custo elevado para executar novamente toda a transação. Em cenários assim, a troca de um componente com problema pode tornar mais eficiente a execução destes processos. A proposta apresentada por Juhnke, Dornemann e Freisleben utiliza a linguagem *WS-BPEL* e os engenhos de execução sem alterá-los. A solução utiliza mecanismos de extensão presentes em um destes engenhos, denominado *ActiveBPEL*, e, a partir deles, são implementadas ações para que a infraestrutura de execução possa identificar, classificar e tratar falhas ocorridas ao longo da execução de uma transação. Um módulo de tolerância a falhas é implementado, associado à infraestrutura e assume a responsabilidade destas ações.

O trabalho também dá suporte ao conceito de políticas que direcionam o tratamento de falhas. Elas são úteis para especificar, por exemplo, a quantidade de tentativas que podem ser realizadas para um determinado componente ou o tipo de recurso computacional a ser usado: um serviço dedicado, um ambiente de nuvem e assim por diante.

É possível afirmar que esta proposta foca no uso de adaptações, tais como reconfigurações dinâmicas, para obter mais robustez e tolerância a falhas na execução de transações longas. Entretanto, outros aspectos importantes para este contexto, como um gerenciamento de estados

de objetos ou o provisionamento de recurso computacionais, não recebem atenção.

Outro trabalho que também lida com reconfigurações dinâmicas é a solução de Assuncao e Cunha (2013). Ele estabelece um mecanismo de identificação e tratamento de falhas aplicável a *workflows* de longa duração cujas atividades são executadas a partir de serviços computacionais implantados em um ambiente de nuvem. Neste trabalho, as reconfigurações dinâmicas de serviços são utilizadas para realizar a correção em erros que eventualmente podem ocorrer ao longo do processamento de um WLD. A abordagem impede que uma falha em um ponto específico do *workflow* implique necessariamente no reprocessamento completo de todas as atividades que o compõem, permitindo uma modificação pontual, como a troca de um serviço defeituoso por outro, no local da falha e a continuidade da execução do processamento daquele ponto em diante. Esta proposta gera informações, como a ocorrência de uma falha na execução de um *workflow*, que podem ser lidas pelos usuários responsáveis pela execução dos WLDs. Com isso, permite-se a realização de interferências manuais que auxiliem a continuidade do processamento, como o uso de rotinas computacionais específicas, criação de máquinas virtuais para dar suporte às tarefas a serem executadas e assim por diante.

Este trabalho também reforça aspectos como a robustez do ambiente de execução de WLDs, sendo capaz de dar continuidade a processamentos de *workflows* mesmo sob condições adversas, como na ocorrência de uma falha. Entretanto a proposta não trata eventos importantes para o contexto dos WLDs, como mudanças em regras de negócio. Além disso, apesar de lidar com serviços presentes em nuvens computacionais, esta infraestrutura de nuvem não é utilizada para garantir, por exemplo, uma maior elasticidade e disponibilidade para os componentes que constituem o ambiente de gerenciamento da execução dos WLDs.

Já o trabalho de Olmsted (2015) propõe uma solução para garantir alta disponibilidade, consistência e durabilidade dos dados em transações longas envolvendo composições de *web services*. Para conseguir atender a estes requisitos, a proposta trabalha com *web services* presentes em diferentes *clusters* (SADASHIV; KUMAR, 2011), onde também estão os serviços de bancos de dados envolvidos com o armazenamento de informações.

A solução analisa as tarefas que compõem uma transação e as distribui pelos *clusters* de maneira que elas sejam submetidas à execução em série ou concorrentemente de acordo com as requisições presentes nas tarefas. A *Unified Modeling Language* (*UML*) (BOOCH; RUMBAUGH; JACOBSON, 2005) é utilizada para compor modelos relativos a estas transações, incluindo, nestes artefatos, informações relacionadas à semântica das tarefas a serem executadas. Componentes especializados analisam estes modelos para garantir a distribuição das tarefas pelos *clusters* onde as mesmas serão executadas. Aspectos como a largura de banda da rede ou a distância para o acesso a estes *clusters*, elementos que podem influenciar no resultado final da execução da transação, também podem ser considerados na distribuição das tarefas.

Esta proposta torna mais eficiente a execução de *workflows* de longa duração, mas, além de restrição ao uso de *web services*, não possui mecanismos importantes para o contexto dos WLDs, como o aproveitamento de dados de resultados intermediários do *workflow* para serem

reusados após uma ação de reconfiguração, por exemplo.

Outro trabalho que lida com composições e reconfigurações dinâmicas é o de Stein, Payne e Jennings (2011), onde são propostas adaptações baseadas em provisionamento de recursos e redundância dos provedores responsáveis pelos serviços que executam as tarefas dos *workflows*. Os serviços que executarão as atividades são escolhidos à medida que o *workflow* vai tendo seus passos executados. Para a escolha dos provedores destes serviços, um algoritmo se responsabiliza pela avaliação de diversos parâmetros, como o custo (financeiro) pelo uso dos serviços, a confiabilidade e a disponibilidade associadas a um provedor e assim por diante. Estes parâmetros são identificados a partir de informações que os próprios provedores disponibilizam através, por exemplo, de contratos propostos e estabelecidos com os clientes ou ainda através de históricos de uso dos serviços providos por estes fornecedores, incluindo as próprias experiências passadas dos clientes interessados em utilizar novamente os recursos oferecidos por aqueles fornecedores.

Neste trabalho, quando um serviço falha, a existência de múltiplos provedores permite que serviços alternativos possam ser selecionados para substituir os que falharam. Esta escolha dos substitutos também passa por uma análise dos mesmos parâmetros que foram utilizados na seleção dos serviços que inicialmente foram definidos para executar as atividades.

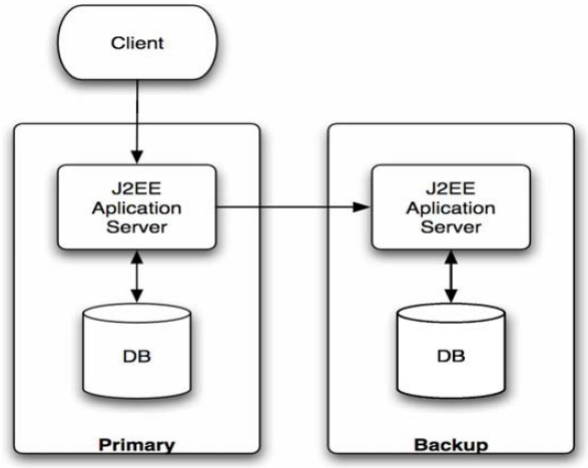
O trabalho melhora o nível de robustez e as chances de sucesso na execução do processo de negócio mesmo na presença de situações adversas, como a falha em um provedor específico que está atendendo às atividades de um determinado *workflow*. Entretanto, a proposta de Stein, Payne e Jennings não foca em particularidades específicas dos WLDs, deixando de lado aspectos importantes para o contextos destes tipos especiais de *workflows*, como um mecanismo de gerenciamento de dados para reuso de resultados intermediários gerados ao longo do processamento antes de uma interrupção por uma falha, por exemplo.

3.2.2 Migrações em *Workflows* de Longa Duração

Perez-Sorrosal et al. (2006) propõem o aumento da disponibilidade de uma solução envolvendo transações de longa duração através da replicação de recursos computacionais, conforme apresentado na Figura 3.4. Nela, é possível observar uma estrutura primária de servidores de aplicação e de banco de dados e a de *backup*, que passa a dar suporte ao processamento das transações em caso de indisponibilidade da primeira. Com isso, há a garantia de alta disponibilidade da infraestrutura para a execução de transações, onde dados e objetos envolvidos nos processamentos podem migrar entre dois serviços redundantes. Os autores também partem da premissa que transações longas devem satisfazer integralmente às propriedades ACID. Neste caso, são necessárias estratégias mais elaboradas e flexíveis, que evitem o desperdício de recursos computacionais. Abortar uma transação que já passou um tempo considerável gerando resultados, desprezando os mesmos e reiniciando a transação por completo, são ações indesejáveis para este contexto.

Uma vez que o trabalho foca na replicação de recursos, outra argumentação apresentada é de que não adianta trabalhar com redundância em apenas um nível da aplicação, como por exemplo na camada relativa ao banco de dados. Se forem perdidas informações relevantes, tais como estados de objetos que representam entidades relativas a um negócio, em outra camada da solução, isto também compromete ações como o reaproveitamento de resultados parciais gerados pelo processamento interrompido. Sendo assim, Perez-Sorrosal et al. propõem um conjunto de algoritmos para replicação de estruturas presentes no servidor de aplicação e também da camada de banco de dados.

Figura 3.4: Replicação de Recursos Computacionais para Suporte à Execução de WLDs



Fonte: (PEREZ-SORROSAL et al., 2006)

As transações longas também são tratadas neste trabalho como sendo compostas por subtransações. Estas últimas são consideradas ações atômicas, ou seja, indivisíveis em operações de confirmação ou desfazimento. Estas operações liberam recursos a serem acessados por outros serviços. Quando serviços que fazem parte das composições que executam processos de negócio falham por conta de problemas no servidor de aplicação, o processamento pode ser assumido por outro servidor. Isto é possível porque há, na infraestrutura, um mecanismo onde estados de objetos são transpostos de um servidor de aplicação (primário) para o outro (*backup*). A solução pode, então, continuar a execução da transação em um servidor diferente de onde ela foi iniciada. O mesmo ocorre com as informações presentes nos bancos de dados, uma vez que eles estão sincronizados.

A dependência da plataforma Java torna a aplicabilidade do trabalho mais restrita. Além disso, apesar da proposta aumentar a eficiência do ambiente de suporte às transações longas, o custo para se replicar recursos pode inviabilizar o uso da abordagem em cenários onde há restrições orçamentárias, por exemplo. No contexto dos WLDs, este efeito colateral relativo ao custo ainda pode se tornar mais presente, uma vez que, conforme discutido nas Seções 1.1 e 2.3, o tempo elevado de processamento muitas vezes implica em quantidade grande de recursos necessários à execução destes *workflows*. Replicá-los significa multiplicar algo que já tem um volume considerável.

Diferentemente da proposta de Perez-Sorrosal et al., Frank, Fong e Lam (2010) não trabalham com replicações de recursos computacionais, mas utilizam também o conceito de migração. No caso, objetos (e.g., dados, estados) pertencentes às *instâncias de workflows* (definidas na Seção 2.2) migram entre componentes de suporte à execução de transações quando ocorrem interrupções ou mudanças em processos de negócio.

Quando alterações em um processo de negócio são encaminhadas para um ambiente de execução através de um novo *workflow*, isto pode invalidar parcialmente ou totalmente instâncias que estejam ativas. Em casos de *workflows* de longa duração, o consumo de recursos computacionais é normalmente elevado em virtude do tempo de processamento das transações. Aproveitar tarefas e resultados já executados e gerados é uma estratégia que gera benefícios como a diminuição do desperdício de recursos computacionais.

Sendo assim, neste trabalho, uma vez que um novo processo chega ao ambiente, ocorre uma análise das versões do *workflow*: a antiga, que está sendo executada, e a nova, que deverá servir de base para as futuras execuções. São identificadas as compatibilidades entre as instâncias através da análise das tarefas que compõem os modelos, sendo checadas informações tais como fluxos de dados entre tarefas ou a própria ligação entre os serviços que fazem parte da composição. Com base no resultado das análises, é montado um plano de migração, com as indicações do que pode ser transposto de uma instância antiga para uma nova, permitindo o reuso de resultados previamente criados, transposição de estados de objetos de um processo em execução para outro e assim por diante.

A proposta também sugere que estas migrações envolvendo todas as instâncias ativas devam ser feitas de forma contínua, o que já pode ser considerado um processo longo, dado o consumo de tempo envolvido em contextos onde há muitas instâncias e análises para se submeterem às transposições.

O trabalho de Frank, Fong e Lam atende a aspectos como a robustez através das migrações de instâncias, já que estas ações evitam o reprocessamento completo de transações longas. O desperdício de recursos computacionais é minimizado ou evitado por completo também. Estas ações de reconfiguração dinâmica envolvendo migrações de instâncias de serviços e processos de negócio presentes no trabalho podem ser encontradas em outras propostas relacionadas à tolerância a falhas (ZHAO; ZHANG, 2009) ou até mesmo migração de sistemas corporativos (PLOOM; SCHEIT; GLASER, 2012).

Apesar das vantagens apresentadas anteriormente, a estratégia de se utilizar múltiplas instâncias pode significar um uso mais intenso de recursos computacionais. Mecanismos de reconfiguração que atuam para realizar ajustes em uma instância em execução, ao invés de criar uma nova ou sobrecarregar outra existente com novos processamentos, pode ajudar na redução de custos associados, por exemplo, ao uso da infraestrutura de suporte à execução do *workflow*.

3.3 Outras Abordagens sobre *Workflows* de Longa Duração

Ali e Reiff-Marganiec (2012) propõem um trabalho sobre a identificação de falhas que ocorrem durante a execução de transações de longa duração. O mecanismo de captura destas falhas para um eventual tratamento é incorporado à lógica de negócio presente na transação através de definições relacionadas a dependências comportamentais entre os componentes que executam o processo de negócio modelado.

Ali e Reiff-Marganiec definem um *workflow* como um conjunto de nós de processamento e consideram transações longas como sendo compostas por múltiplas subtransações aninhadas. Uma hierarquia é montada com base na dependência entre os nós e análises são feitas para identificar a propagação de erros ao longo dos elementos que fazem parte do processamento da transação.

O mecanismo desenvolvido foi construído de maneira que ele reage a estímulos e/ou eventos. Transições no estado de execução de um componente pertencente à composição responsável pela transação, por exemplo, promovem a intervenção da solução implementada. O mecanismo atua de forma contínua até que a transação de longa duração tenha sua execução encerrada em um estado que seja consistente tanto do ponto de vista de sistema como do ponto de vista de negócio.

A proposta também classifica os nós de processamento em *vitais* e *não-vitais* de acordo com a relevância da ocorrência de falhas nos mesmos. Uma análise da dependência e interligação entre nós de processamento e subtransações identifica os caminhos percorridos na propagação de falhas ocorridas em nós da composição.

A proposta é útil no tratamento de falhas que se propagam em transações de longa duração, aumentando a robustez de um ambiente de execução de um WLD, por exemplo. No entanto, embora tenha o foco na identificação de falhas, não há na abordagem a elaboração de mecanismos para o tratamento das mesmas.

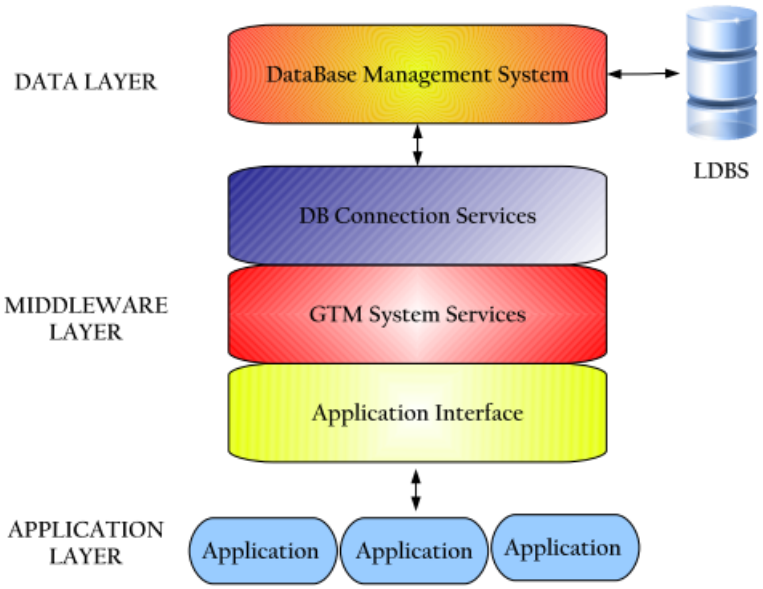
Outro trabalho sobre transações longas é o de Amato et al. (2014). Esta proposta está relacionada a um aspecto importante para a garantia da integridade e da robustez no contexto dos WLDs: o acesso concorrente a recursos computacionais como um registro em um banco de dados. Para isso, o núcleo da solução está baseado em um *middleware* denominado *GTM-middleware* (*Global Transaction Manager - middleware*) (CHIANESE et al., 2008) que cria um *contexto virtual* para as transações em execução. Este contexto compreende informações manipuladas fora dos serviços de bancos de dados que, quando precisam ser efetivadas nestes serviços, o *GTM* se responsabiliza por fazê-lo.

Através desta intervenção, esta camada intermediária é capaz de realizar controles sobre travamentos de recursos, gerenciar desconexões entre componentes envolvidos com as transações, inatividade de usuários, promover a execução em série de tarefas para reduzir efeitos da concorrência por recursos computacionais ou até mesmo intermediar a continuação de alguma transação suspensa previamente. A comunicação do que deve ser efetivamente armazenado nos

repositórios é realizada por uma transação especial denominada *SST* (*Secure System Transaction*).

A Figura 3.5 apresenta a visão geral da solução proposta. Ela mostra uma arquitetura com uma camada de banco de dados e uma camada relativa a aplicações, tais como soluções de software distribuídas, além do *middleware* responsável pelo *contexto virtual*.

Figura 3.5: Arquitetura para Suporte a Transações Longas Envolvendo *GTM-middleware*



Fonte: (AMATO et al., 2014)

Apesar de sua relevância para cenários de execução dos WLDs, a proposta de Amato et al. pode consumir muitos recursos computacionais caso o volume de dados administrados pelo *contexto virtual* seja elevado, algo comum em processamentos de *workflows* de longa duração.

Também lidando com alta disponibilidade, mas através da replicação de objetos na execução de *workflows*, é como Schäfer et al. (2016) estabelece seu trabalho. Nesta proposta, há replicações das instâncias do *workflow* em execução. Ou seja, para garantir maior robustez e alta disponibilidade na execução do *workflow*, ela é realizada em paralelo por múltiplos mecanismos de execução denominados de *execution engines*.

Este trabalho propõe o sistema *HAWKS (system for Highly Available executions of WorKflowS)*, que possui algoritmos específicos para realizar o sincronismo entre as múltiplas instâncias executando o mesmo *workflow* e evitando, assim, inconsistências como a geração de resultados finais divergentes pelas diferentes instâncias. Os estados das atividades são gerenciados para garantir a consistência nos resultados intermediários gerados pelas execuções dos *workflows*. Com esta replicação de instâncias, quando um *execution engine* para de funcionar por causa de uma falha na infraestrutura onde o mesmo estava implantado, por exemplo, outro *execution engine* dá continuidade à execução do *workflow*. Desta forma, há um aumento na garantia de que o processamento chegará ao final mesmo com a ocorrência de falhas ao longo da execução do *workflow*. Compensações são utilizadas quando uma atividade é interrompida antes da sincronização com as demais instâncias replicadas.

A replicação de recursos, conforme já discutido anteriormente, é capaz de gerar um acréscimo nos custos (e.g., valores a serem pagos pelo uso de uma infraestrutura). Além disso, a proposta de Schäfer et al. não considera as particularidades dos WLDs quanto à grande demanda de tempo, sendo voltada particularmente ao contexto dos *workflows* comuns.

Ainda no contexto de propostas ligadas aos WLDs, um trabalho relacionado à simulação dos chamados *processos de negócio de longa duração* é uma interessante contribuição de Stankevicius e Vasilecas (2016). Estes tipos especiais de processos são diretamente ligados aos *workflows* de longa duração, uma vez que sua principal característica também é o longo tempo para a conclusão de seus processamentos. A solução de Stankevicius e Vasilecas se baseia no fato de que simular a execução de processos de negócio é uma estratégia interessante para que mudanças de processos corporativos possam ser realizadas de forma mais segura, uma vez que as simulações são capazes de mostrar problemas e vantagens na adoção das mudanças em cenários reais.

Este trabalho se baseia em processos de negócio sendo simulados a partir de modelos orientados a eventos. Ou seja, as atividades são iniciadas a partir de eventos (e.g, uma ligação realizada por um cliente para uma empresas de seguros para dar entrada em um sinistro) específicos e os seus resultados também são expressos na forma de eventos. Além disso, as transações são divididas em tarefas menores de forma que ações de *rollback* ou compensações em operações já confirmadas possam ser realizadas.

As simulações, como os próprios autores discutem, são sempre alvos de discussões a respeito de suas vantagens quando confrontadas com os custos e esforços envolvidos na construção dos modelos e análises de possíveis resultados para as diversas possibilidades de execução das atividades que compõem os processos de negócio. Entretanto, a complexidade crescente na estrutura e na dinâmica destes processos torna a tarefa de antever as implicações de eventuais mudanças neles um desafio considerável. Neste contexto, as simulações podem ser muito úteis ao permitir entender os impactos das mudanças.

Embora útil como ferramenta de simulação, esta proposta tem alcance limitado na execução real de processos de negócio.

Finalmente, no que diz respeito ao uso da computação em nuvem, trabalhos como o de Wang et al. (2014) ou o de Esteves e Veiga (2016) suportam a execução de *workflows* utilizando uma estratégia *como serviço* (*as a Service*). Entretanto, elas não atendem a particularidades advindas do elevado consumo de tempo observadas nos *workflows* de longa duração, como, por exemplo, múltiplas estratégias de adaptação de acordo com cenários identificados na execução das atividades, sendo mais voltadas aos *workflows* comuns.

3.4 Sumário dos Trabalhos Relacionados

Todos os trabalhos apresentados neste capítulo possuem alguma relação com o suporte à execução de *workflows* de longa duração. A Tabela 3.1 apresenta um sumário destes trabalhos

considerando cinco critérios: suporte à adaptação, mecanismo de adaptação implementado, orientação a serviço, independência de plataforma e infraestrutura elástica.

Tabela 3.1: Quadro Comparativo - Trabalhos Associados à Execução de WLDs

Trabalho	Suporte à Adaptação	Mecanismo de Adaptação	Orientado a Serviço	Independência de Plataforma	Infraestrutura Escalável
(PEREZ-SORROSAL et al., 2006)	Sim	Migração de execução de tarefas	Sim	Não	*cluster*
(MONTAGUT; MOLVA; GOLEGA, 2008)	Sim	Composição e reconfiguração dinâmicas de serviços	Sim	Sim	-
(JUHNKE; DORNEMANN; FREISLEBEN, 2009)	Sim	Reconfiguração dinâmica	Sim	Sim	nuvem
(HU et al., 2010)	Sim	Redirecionamento de fluxo de execução	Não	Sim	-
(MENG; ARBAB, 2010)	Sim	Redirecionamento de fluxo de execução	Sim	Sim	-
(FRANK; FONG; LAM, 2010)	Sim	Migração de execução de tarefas	Não	Sim	-
(STEIN; PAYNE; JENNINGS, 2011)	Sim	Composição e reconfiguração dinâmicas de serviços	Sim	Sim	-
(ALI; REIFF-MARGANIEC, 2012)	Não	-	Sim	Sim	-
(ASSUNCAO; CUNHA, 2013)	Sim	Reconfiguração dinâmica	Sim	Sim	nuvem
(KOKASH; ARBAB, 2013)	Sim	Redirecionamento de fluxo de execução	Sim	Sim	-
(AMATO et al., 2014)	Não	-	Sim	Sim	-
(YANG et al., 2014)	Sim	Redirecionamento de fluxo de execução	Sim	Sim	-
(OLMSTED, 2015)	Sim	Composição dinâmica de serviços	Sim	Sim	*cluster*
(FERNÁNDEZ; TEDESCHI; PRIOL, 2016)	Sim	Redirecionamento de fluxo de execução	Sim	Sim	nuvem
(SCHäFER et al., 2016)	Sim	Redirecionamento de fluxo de execução	Sim	Sim	-
(STANKEVICIUS; VASILECAS, 2016)	Não	-	Sim	Sim	-

O *Suporte à Adaptação* considera a existência ou não de qualquer mecanismo de adaptação relacionado à execução de *workflows* no trabalho. Quando ocorre este suporte, esta informação é complementada na coluna *Mecanismo Adaptativo*, que especifica o mecanismo implementado. A coluna *Orientação a Serviço* indica se o trabalho se baseia em princípios do paradigma de Computação Orientada a Serviço. A *Independência de Plataforma* especifica se o trabalho é limitado a alguma tecnologia específica (e.g., linguagem de programação) ou plataforma tecnológica (e.g., *API* oferecida por algum provedor de nuvem). Finalmente, a última coluna (*Infraestrutura Escalável*) indica se o trabalho faz referência explícita ao uso de infraestruturas associadas à escalabilidade de aplicações, tais como *clusters* ou nuvens computacionais.

Ao se observar os trabalhos relacionados na Tabela 3.1, nota-se que considerável parte das propostas apresentam mecanismos adaptativos. Isto ressalta a relevância da adaptação para o contexto da execução dos WLDs. A adaptação é capaz de levar mais robustez à execução dos *workflows*, uma vez que problemas como a falha em rotinas associadas às atividades do *workflow* podem ser sanados sem necessariamente abortar e reiniciar o *workflow* do início. As estratégias envolvendo estas adaptações são comumente associadas a mecanismos que interferem no fluxo de execução dos *workflows*, reconfigurações dinâmicas com implementações associadas a ações mais restritas, como a troca de serviços e o reinício da atividade associada a estes elementos, ou ainda a migrações de objetos na infraestrutura de execução. Entretanto, não se observa uma abordagem onde, por exemplo, haja uma abrangência maior no que diz respeito a opções de adaptação e aspectos como os descritos abaixo sejam atendidos simultaneamente:

- Múltiplos mecanismos adaptativos estejam presentes para atender não apenas a falhas mas também a eventos como mudanças de regras de negócio;
- As adaptações sejam planejadas e executadas de acordo com o que é observado na execução do processamento;
- Quiescência, reuso de dados e estados sejam integrados às adaptações; e

- As adaptações sejam aplicáveis a serviços não atrelados a um único tipo de implementação (como a restrição a *web services* observada em algumas propostas).

Outro aspecto que também é observado ao analisar a Tabela 3.1 é a presença de conceitos de *SOC* na implementação das propostas. Isto reforça a importância deste paradigma no desenvolvimento de soluções para o contexto dos WLDs. O uso dos princípios da orientação a serviços permite que as soluções se tornem mais robustas, adaptáveis, modulares e facilmente mantidas. Isto decorre de vantagens como a facilidade com que serviços podem ser adicionados, removidos ou substituídos em uma solução que segue os princípios de *SOC*. A interoperabilidade entre os serviços baseada apenas em interfaces sem se preocupar com detalhes internos de implementação ou com aspectos da infraestrutura onde se encontram implantados é outra vantagem de se utilizar a orientação a serviços. Todos estes fatores são importantes para a execução de processos de negócio onde a longa duração aumenta as possibilidades de problemas como a interrupção em atividades ou falhas de infraestruturas. *Workflows* podem ter suas atividades sendo processadas através de serviços oferecidos por diferentes fornecedores, escolhidos entre os que melhor sejam capazes de implementar soluções que atendam às especificações estabelecidas para cada um dos serviços computacionais necessários para se executar um processo de negócio.

A coluna referente à independência de plataforma também mostra uma tendência na área de WLDs que diz respeito a soluções que não são fortemente dependentes de uma linguagem de programação ou de uma plataforma de desenvolvimento como, por exemplo, uma *API* fornecida na infraestrutura de nuvem de um provedor qualquer. Quando uma solução de software é construída, naturalmente, linguagens de programação, infraestruturas de hardware e software (e.g., SGBDs) são selecionados. Entretanto, as contribuições presentes na solução, como a arquitetura ou os algoritmos para execução das rotinas mais importantes estabelecidas na proposta, não devem ser atreladas a nenhum daqueles elementos computacionais mencionados. Isso restringe o escopo do uso das soluções. Algo que ajuda esta independência é o uso apropriado dos conceitos de *SOC*. Uma solução definida em função de serviços em uma arquitetura genérica capaz de ser implementada em qualquer linguagem de programação ou infraestrutura de suporte à execução, como nuvens ou *clusters* torna-se uma contribuição mais interessante

Finalmente, a última coluna da Tabela 3.1 mostra que, para o contexto dos *workflows* de longa duração, poucas propostas estabelecem, de forma explícita, o uso de infraestruturas computacionais capazes de prover elasticidade e escalabilidade às soluções nela implantadas, como é o caso dos *clusters* e das nuvens computacionais. Além disso, as soluções que trazem referências a estas infraestruturas normalmente as direcionam para as atividades dos *workflows* propriamente ditas. Ou seja, a distribuição dos processamentos é restrita aos componentes e serviços que entram na composição dos *workflows*. Não se observa, nestes casos, uma solução onde o ambiente responsável pelo gerenciamento da execução de WLDs possua uma arquitetura cuja estrutura e dinâmica sejam capazes de se beneficiar das propriedades elásticas como as observadas em nuvens computacionais, por exemplo. Nestas infraestruturas, recursos computacionais, como máquinas virtuais ou serviços de armazenamento, podem ser alocados

e utilizadas de acordo com a demanda, de forma a garantir aspectos como disponibilidade e escalabilidade. Um ambiente de execução de WLDs, ao utilizar estas características inerentes a estes ambientes em benefício próprio, poderia ser disponibilizado também como um serviço. Ou seja, a demanda pela execução de *workflows* poderia ter os recursos computacionais necessários aos serviços responsáveis por esta execução também sendo colocados à disposição sob demanda, não apenas focando nas atividades dos *workflows*.

Sendo assim, a proposta de um ambiente de execução de WLDs que adote princípios de *SOC*, ofereça mecanismos adaptativos múltiplos, associados a elementos como quiescência, gerenciamento de dados e estados, organizados dentro de uma arquitetura de software que permita a disponibilização deste ambientes como uma solução *as a Service*, constitui um desafio relevante para o contexto do uso de *workflows* de longa duração.

3.5 Considerações Finais

Trabalhos relacionados ao contexto dos *workflows* de longa duração foram apresentados ao longo deste capítulo. Este levantamento é importante como parte do processo de estudo do domínio relativo à proposta contida nesta tese de doutorado. A partir destes estudos, desafios e lacunas existentes no contexto dos WLDs puderam ser identificados e utilizados como base para o projeto e implementação do *LONGWisE4cloud* que serão apresentados nos próximos capítulos deste trabalho.

4

PRINCIPAIS CONCEITOS E ARQUITETURA DO *LONGWISE4CLOUD*

Este capítulo apresenta uma visão geral do *LONGWisE4cloud*, detalhes da solução, além de características estruturais e dinâmicas de sua arquitetura. O capítulo introduz inicialmente os princípios básicos usados na definição do ambiente. Em seguida, apresenta-se uma visão geral do ambiente de suporte à execução de WLDs. Por fim, são apresentados a arquitetura e detalhes dos seus componentes.

4.1 Princípios Básicos

O *LONGWisE4cloud* é um ambiente para execução de *workflows* de longa duração que possui suporte à adaptação e escalabilidade. Estes aspectos reforçam outras características desejáveis nestes ambientes, tais como desempenho e robustez. O ambiente também é capaz de monitorar e controlar estes *workflows* enquanto eles estão sendo executados.

Um aspecto importante para o *LONGWisE4cloud* é a definição do que vem a ser *longa duração*. Os WLDs foram apresentados na Seção 2.3 mas, em particular, para o desenvolvimento da solução proposta nesta tese, um WLD é um *workflow* que se enquadra nas seguintes características:

- O *workflow* está necessariamente associado a um processo de negócio;
- O tempo para executar o *workflow* é longo o suficiente para que ocorram, por exemplo, alterações no ambiente de execução ou mudanças nas regras de negócio que invalidam parcialmente ou completamente a geração de resultados pelo *workflow*; e
- O tempo para reprocessamento completo do *workflow* é inaceitável.

Considerando estas características, serão inicialmente detalhados os princípios que devem ser seguidos na construção de um ambiente de suporte à execução deste tipo de *workflow*. Estes princípios foram identificados a partir do estudo de conceitos básicos (Capítulo 2) e da análise de soluções existentes (Capítulo 3). Uma breve descrição a sobre como o *LONGWisE4cloud* oferece suporte a cada um destes princípios será inicialmente realizada. O detalhamento dos componentes, serviços e implementações que permitem este suporte será apresentado ao longo das demais seções deste capítulo e no Capítulo 5.

- ***Computação Orientada a Serviços***

O uso de princípios da Computação Orientada a Serviços (Seção 2.5) no desenvolvimento do *LONGWisE4cloud* é motivado por três razões principais:

- Soluções desenvolvidas por terceiros, tais como serviços de armazenamento ou criptografia, podem ser utilizados, acelerando não apenas o desenvolvimento da solução, mas também a montagem dos *workflows*. Tanto o ambiente como os WLDs são implementados como composições de serviços;
- A modularidade e o fraco acoplamento dos componentes responsáveis pelas atividades dos *workflows* facilitam a troca destes componentes em tempo de execução; e
- O uso destes princípios permitiram que o ambiente de execução pudesse ser disponibilizado em uma abordagem *as a Service*, onde a demanda estabelece o uso efetivo de recursos computacionais de hardware e software.

Sendo assim, estes princípios e benefícios podem ser observados em soluções presentes no *LONGWisE4cloud* que atendem aos demais aspectos cobertos pelo ambiente, conforme será apresentado a seguir.

- ***Quiescência***

De acordo com o que foi discutido na Seção 2.6.1, a quiescência é uma propriedade que envolve a passividade de um *nó* de processamento onde não há transações sendo executadas ou que venham a ser iniciadas considerando o uso daquele nó. No contexto de adaptações no *LONGWisE4cloud*, a quiescência possui um papel importante, uma vez que os serviços responsáveis por executar as atividades destes *workflows* são nós envolvidos em transações. Desta forma, as restrições impostas pela quiescência permitem a execução de mudanças de uma maneira segura, menos sujeita a inconsistências nos dados, por exemplo.

- ***Adaptação***

A adaptação no *LONGWisE4cloud* permite a execução de WLDs com a possibilidade, por exemplo, de se trocar um serviço responsável por uma atividade de um *workflow*. Desta forma, reforça-se a possibilidade de sucesso na execução destes processos de negócio, uma vez que falhas ou mudanças em regras de negócio que exijam alterações nas composições dos WLDs são passíveis de serem feitas no ambiente. Algumas mudanças podem ser aplicadas antes mesmo do início da execução dos *workflows*, quando referências a serviços podem ser trocadas em virtude da indisponibilidade, momentânea ou não, de serviços no ambiente de execução. Outras adaptações ocorrem quando os *workflows* já estão em execução. Nestas últimas, são

promovidas alterações nas composições de serviços que constituem os WLDs à medida que os mesmo são executados, evitando interrupções e o reprocessamento completo dos *workflows*. Estas adaptações são planejadas com base em análises envolvendo a própria execução dos *workflows*, de forma que ocorram sem efeitos colaterais indesejáveis, como inconsistências nos dados processados pelos WLDs.

- ***Escalabilidade***

LONGWisE4cloud é capaz de acomodar o aumento na demanda pela execução de WLDs através da alocação dinâmica de recursos computacionais, tais como máquinas virtuais disponíveis em uma infraestrutura de nuvem onde a solução está implantada. A elasticidade presente em ambientes de nuvem ajuda a garantir esta escalabilidade à medida que recursos computacionais são disponibilizados para a demanda gerada pelo processamento do *workflow*. O uso de múltiplos serviços replicados e distribuídos para gerenciar a execução dos WLDs ajuda a evitar uma sobrecarga no ambiente de execução como um todo e reforça a garantia de disponibilidade do *LONGWisE4cloud*. Estes serviços podem ser implantados, por exemplo, em máquinas virtuais diferentes, de forma que eles possam receber novas requisições para executar processos de negócio sem interferir na execução de outros WLDs já presentes no ambiente.

- ***Robustez***

No *LONGWisE4cloud*, robustez é a garantia de executar WLDs mesmo sob condições adversas, tais como falhas na infraestrutura de comunicação ou indisponibilidade temporária de serviços. Para obter robustez, o *LONGWisE4cloud* adota um mecanismo que envolve monitoramento, detecção e tratamento de falhas capaz de identificar e eliminar problemas em serviços. A adaptação desempenha um papel importante neste processo de garantia da robustez, uma vez que a maior parte das ações executadas pelo ambiente em casos críticos, como a falha em serviços mencionada, envolve mudanças (e.g., trocas de serviços) nas instâncias dos *workflows* sendo executados.

- ***Heterogeneidade***

No contexto do *LONGWisE4cloud*, heterogeneidade refere-se ao uso de serviços cujo processamento pode ser realizado por rotinas implementadas em diferentes linguagens de programação, utilizando soluções com arquiteturas distintas, interconectadas através de múltiplos protocolos de comunicação e sendo executados em várias infraestruturas de suporte, tais como ambientes de nuvem pertencentes a diferentes provedores. Como WLDs normalmente extrapolam as fronteiras de uma organização, sua execução envolve diferentes contextos. Consequentemente, os serviços responsáveis por suas atividades geralmente envolvem algumas ou até mesmo todas as diferenças citadas anteriormente. Sendo assim, eles precisam cooperar para uma execução apropriada do *workflow*. No que diz respeito à adaptação, mecanismos adaptativos

devem ser capazes de lidar, por exemplo, com a troca de serviços implementados em diferentes linguagens de programação ou que estão sendo executados em infraestruturas diferentes. No *LONGWisE4cloud*, a heterogeneidade é resolvida através do uso de princípios de *SOC*, como a orientação a serviços na arquitetura do ambiente e também com o uso de serviços na composição dos *workflows* executados nele. Estes serviços podem encapsular a invocação a rotinas remotas que apresentam os aspectos de heterogeneidade mencionados anteriormente, como linguagens de programação distintas ou infraestruturas de execução diferentes.

- ***Provisionamento de Recursos***

Para o *LONGWisE4cloud*, este provisionamento envolve a garantia da existência de recursos de hardware e/ou software para a execução das atividades que constituem os WLDs. O suporte a este aspecto no ambiente se baseia nos seguintes elementos:

- Parâmetros acompanhando o WLD que indicam a necessidade de recursos de hardware específicos (e.g., uma máquina virtual exclusiva para o processamento do *workflow*) ou de software (e.g., a exigência de um serviço dedicado de banco de dados);
- Adaptadores para realizar a troca de serviços associados ao *workflow* de forma a atender parâmetros estabelecidos para ele, como o uso de serviços de banco de dados específicos; e
- Componentes capazes de direcionar a execução do WLD para atender, por exemplo, à demanda pela exclusividade no uso de elementos de hardware. Um usuário pode querer, por exemplo, que *workflows* pertencentes a outros usuários não sejam executados em uma máquina virtual onde os seus WLDs estão sendo processados.

Com esta solução, o *LONGWisE4cloud* é capaz de garantir, por exemplo, que regras estabelecidas por projetistas dos *workflows* e capazes de aumentar o desempenho na execução dos WLDs sejam atendidas.

- ***Gerenciamento de Estados***

Conforme discutido na Seção 1.2, a longa duração dos WLDs aumenta o risco de problemas tais como a degradação no desempenho de um serviço ou a interrupção de sua execução. Gerenciar os estados destes serviço pode ser útil em situações onde, por exemplo, ele precisa ser trocado por outro e a execução de uma atividade deve continuar do ponto onde ela foi interrompida. Neste caso, o estado do serviço antigo pode ser restaurado no serviço novo. *LONGWisE4cloud* é capaz de monitorar, armazenar e recuperar o estado dos serviços associados a um WLD através de componentes específicos.

- ***Gerenciamento de Dados***

Ambientes de execução de WLDs precisam lidar com dados usados e gerados enquanto os *workflows* são executados. Em situações críticas, o reuso ou o descarte destes dados é crucial para o desempenho geral do *workflow*. Migração, transformação ou reuso de dados gerados sobre transações muito demoradas são importantes para evitar desperdício ou mau uso de recursos computacionais aplicados para gerar estes dados. Com a necessidade de um gerenciamento de dados, *LONGWisE4cloud* inclui um mecanismo para coletar dados de transações envolvidas na execução dos WLDs. Este mecanismo identifica detalhes estruturais de esquemas de dados utilizados na execução do *workflow* e os adapta através de mudanças como alterações em estruturas de tabelas. Assim, é possível aproveitar resultados intermediários do WLD em reconfigurações dinâmicas, por exemplo. Rotinas providas através de parâmetros pelo usuário do *workflow* também podem ser utilizadas nestas ações envolvendo dados gerados e processados pelo WLD.

- ***Segurança***

Segurança neste contexto está relacionada à proteção da execução e dos dados dos serviços que compõem o WLD contra acessos não autorizados. Com a longa duração, os recursos computacionais alocados para a execução do *workflow* podem estar expostos a ações mal intencionadas, tais como ataques para interrupção de serviços ou apropriação indevida de dados. A preocupação com privacidade, integridade e confiabilidade das informações processadas, nestes casos, se torna maior. O suporte à segurança no *LONGWisE4cloud* se dá a partir do momento em que o ambiente utiliza princípios de *SOC* na definição de sua arquitetura e na composição do WLD. De acordo com o discutido na Seção 2.5, uma das vantagens no uso do paradigma da orientação a serviços é poder escolher, entre diversos fornecedores e opções de implementação, o serviço que melhor atende a uma determinada funcionalidade. Sendo assim, o uso destes princípios de *SOC* permite que sejam selecionados serviços que utilizem, por exemplo, criptografia de dados em sua implementação. O *LONGWisE4cloud* possui um mecanismo capaz de selecionar serviços que estejam cadastrados em um registro presente no ambiente. Propriedades estabelecidas pelo usuário responsável pelo *workflow* podem direcionar esta seleção dos serviços que serão utilizados na execução do WLD.

- ***Integração***

No contexto do desenvolvimento do *LONGWisE4cloud*, integração significa que mecanismos como a adaptação e escalabilidade, aspectos principais do sistema, não trabalham isolados, mas cooperando com o restante do ambiente. O uso integrado de soluções para todos os demais aspectos mencionados anteriormente aumenta o benefício que eles trazem à execução de WLDs. Por exemplo, ter um mecanismo de reuso de dados para aproveitar resultados intermediários de um *workflow* é uma característica importante para evitar perda destes resultados parciais, e, consequentemente, desperdício de esforços para produzi-las. Associando este mecanismo com a distribuição de componentes do ambiente de gerenciamento de execução por diversas máquinas

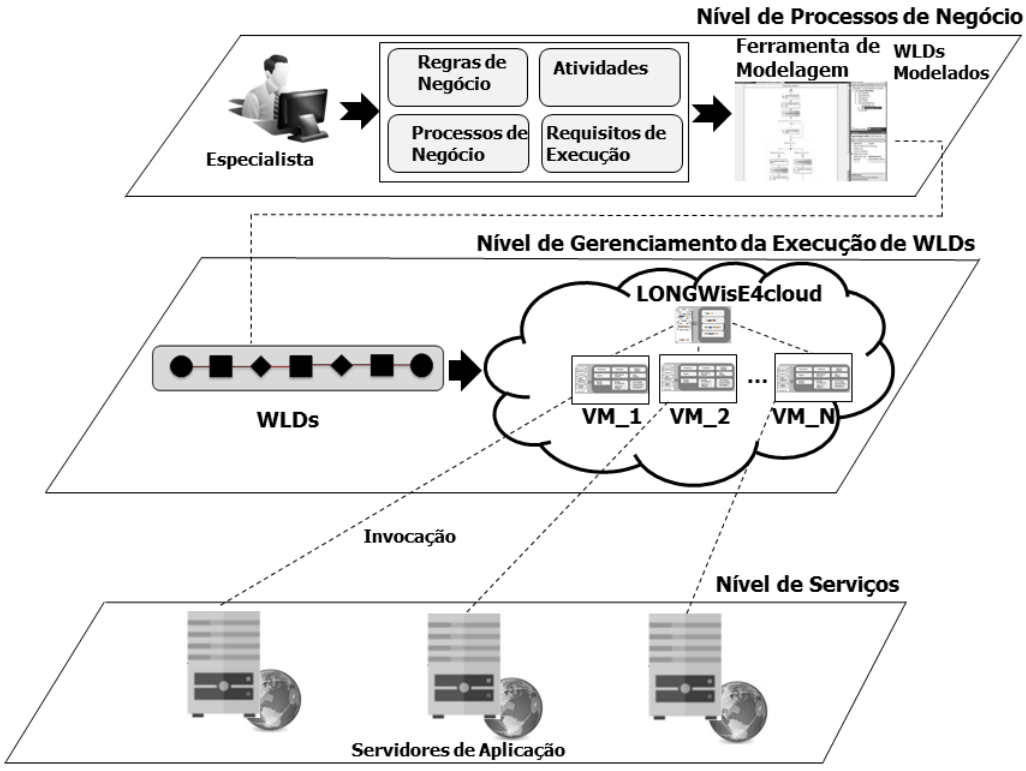
virtuais disponíveis na infraestrutura de execução evita a sobrecarga do ambiente. Isto favorece escalabilidade, garantindo a execução de *workflows* mesmo com o aumento da demanda.

Sendo assim, *LONGWisE4cloud* integra os mecanismos que satisfazem a todas as características mencionadas. A solução proposta utiliza quiescência, gerenciamento de dados e estados para garantir a integridade na execução de WLDs. Robustez é suportada pela adoção de mecanismos adaptativos de forma que mudanças em tempo de execução possam garantir a geração de resultados por um WLD mesmo na presença de situações adversas como a falha de um serviço. O uso de princípios de *SOC* é essencial para a heterogeneidade e escalabilidade como vai ser detalhado nas seções seguintes.

4.2 Visão Geral

Para o contexto desta tese, o suporte à execução de WLDs pertence a um cenário composto por três níveis, como pode ser visto na Figura 4.1.

Figura 4.1: Contexto de Uso do *LONGWisE4cloud*



Fonte: O próprio autor.

No *Nível de Processos de Negócio*, os *workflows* são modelados por especialistas capazes de projetar processos de negócio da empresa. Eles criam *workflows* que representam estes processos e são utilizados na sua automação e também na sua execução. No nível de *Gerenciamento da Execução de WLDs* é onde os WLDs modelados são submetidos como um conjunto de referências a serviços capazes de serem executados e onde o *LONGWisE4cloud* está situado. Os componentes do *LONGWisE4cloud*, por sua vez, podem ser implantados em máquinas virtuais

para melhorar o desempenho da execução dos WLDs. Finalmente, o *Nível de Serviços* inclui os serviços responsáveis por executar as atividades dos *workflows*.

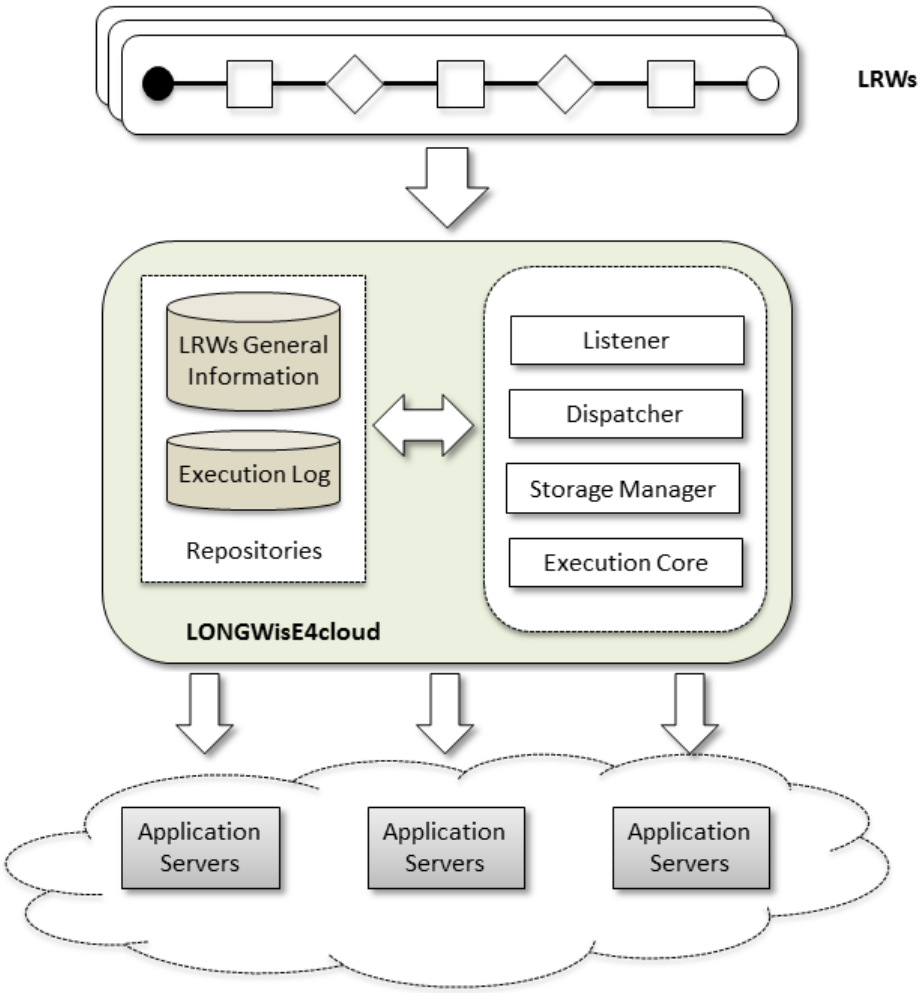
4.3 Aspectos Arquiteturais

Como mencionado anteriormente, o *LONGWisE4cloud* foi projetado de acordo com os princípios de *SOC*. Logo, os serviços providos pelo ambiente se integram para atender aos princípios apresentados na Seção 4.1. A estrutura geral da arquitetura do *LONGWisE4cloud*, assim como aspectos dinâmicos da mesma, serão detalhados nas seções a seguintes.

4.3.1 Estrutura

Os componentes que constituem a arquitetura geral do *LONGWisE4cloud* são apresentados na Figura 4.2.

Figura 4.2: Arquitetura Geral do *LONGWisE4cloud*



Fonte: O próprio autor.

- *Execution Core*

O *Execution Core* consiste de um conjunto de componentes responsáveis por gerenciar a execução dos WLDs. O ambiente pode trabalhar com um ou vários *cores* simultaneamente, cada um deles denominado uma instância do núcleo de execução (*ECI - Execution Core Instance*). O número de *ECIs* é estabelecido pelos usuários que administram o sistema durante a configuração do ambiente de acordo com diferentes aspectos, tais como o número de *workflows* executados nas instâncias existentes, tempo de resposta das atividades destes *workflows* ou a requisição de clientes que exigem que seus processos sejam executados isoladamente. Novas *ECIs* podem ser adicionadas ao *LONGWisE4cloud* em tempo de execução, bastando para isso se alterar a configuração do ambiente com dados destas novas *ECIs* (e.g., endereços IP das máquinas virtuais onde foram implantadas). Os detalhes estruturais e dinâmicos do *Execution Core* serão apresentados nas Seções 4.3.3 e 4.3.4.

- *Listener*

Este componente é responsável pelo serviço de recebimento dos WLDs executados pelo ambiente. Antes de repassar estes *workflows* aos componentes que gerenciam efetivamente a execução dos mesmos, parâmetros de execução e.g., protocolos de comunicação, número máximo de instâncias do *workflow* que podem ser executadas em uma *ECIs*, são armazenados nos repositórios do *LONGWisE4cloud*. O *Listener* é responsável também por armazenar a data e a hora em que os *workflows* foram submetidos ao ambiente assim como os seus usuários responsáveis. Este registro pode ser utilizado como um histórico de execuções dos WLDs gerenciados pelo ambiente. Outras operações sobre *workflows* submetidos ao ambiente também são recebidas e respondidas pelo *Listener*, como, por exemplo, o cancelamento da execução de instâncias de um WLD.

- *Dispatcher*

Este componente é responsável por receber os WLDs do *Listener* e direcioná-los para as *ECIs* de forma que eles possam ser efetivamente executados. A escolha sobre qual *ECI* será responsável pela execução de um WLD segue critérios que variam desde o número de *workflows* sendo executados nestas instâncias até requisitos dos usuários, tais como a necessidade de executar o *workflow* em uma *ECI* onde só existam instâncias pertencentes ao usuário responsável por aquele *workflow*. Operações que devem ser realizadas sobre *workflows* em execução (e.g., cancelamento, pausa) recebidas pelo *Listener* também são encaminhadas pelo *Dispatcher* às *ECIs*.

O *Dispatcher* é um componente diretamente associado a dois aspectos importantes do *LONGWisE4cloud* mencionados na Seção 4.1: a escalabilidade e o provisionamento de recursos. Ele atende ao primeiro através da distribuição dos WLDs de acordo com critérios como os citados anteriormente. Sendo assim, ele se torna um elemento fundamental para acomodar as execuções destes *workflows* de acordo com a demanda em uma infraestrutura distribuída, como máquinas virtuais em um ambiente de nuvem. O provisionamento de recursos também é atendido

a partir do momento que exigências como, por exemplo, uma *ECI* exclusiva para um WLD, são satisfeitas a partir da capacidade que o *Dispatcher* tem de distribuir a execução dos *workflows* que chegam ao *LONGWisE4cloud* atendendo a este tipo de parametrização imposta pelo usuário.

- *Storage Manager*

Este componente formata, armazena e recupera dados solicitados pelos demais componentes do ambiente, trabalhando como um *proxy* para facilitar o acesso aos repositórios que fazem parte do *LONGWisE4cloud*. Qualquer elemento que necessite armazenar ou coletar informações nestes repositórios pode, assim, abstrair detalhes relacionados à comunicação com os mesmos, tais como: *drivers*, linguagens e consulta ou *frameworks* associados à persistência de dados. O *Storage Manager* é um componente importante para a heterogeneidade suportada pelo ambiente. Uma vez que ele isola os demais componentes do *LONGWisE4cloud* de detalhes de implementação dos repositórios, diferentes soluções (e.g., como SGBDs de diversos fornecedores) podem ser utilizadas sem que isso interfira no restante do ambiente.

- *LRWs General Information*

O objetivo principal deste repositório é armazenar dados gerais dos WLDs executados pelo *LONGWisE4cloud*, tais como: a data e o horário que um *workflow* chegou ao ambiente, o usuário que o submeteu para execução, referências às listas de atividades de cada *workflow*, e restrições que devem ser aplicadas na sua execução, como a impossibilidade de executá-lo junto a outros WLDs em uma mesma *ECI*. Operações, tais como a a suspensão ou o cancelamento da execução do *workflow*, que chegam ao ambiente através do *Listener* também são registradas neste repositório.

- *Execution Log*

Este repositório armazena dados da execução dos WLDs, tais como: tempo de processamento dos *workflows* em diferentes *ECIs*, falhas em serviços e reconfigurações realizadas em instâncias específicas. Neste repositório também são armazenadas informações básicas a respeito das *Execution Core Instances*, tais como: identificação das *ECIs*, referência a sua localização dentro da infraestrutura de execução do *LONGWisE4cloud* (e.g., endereço IP de uma máquina virtual). O armazenamento de informações no *Execution Log* é realizado pelo serviço de monitoramento existente nas *ECIs*, que será apresentado na Seção 4.3.3. O *Execution Log* tem um papel importante na robustez (ver Seção 4.1) do *LONGWisE4cloud*. Com base nas informações presentes neste repositório, falhas em serviços podem ser identificadas e tratadas pelo ambiente através de adaptações.

- *Application Servers*

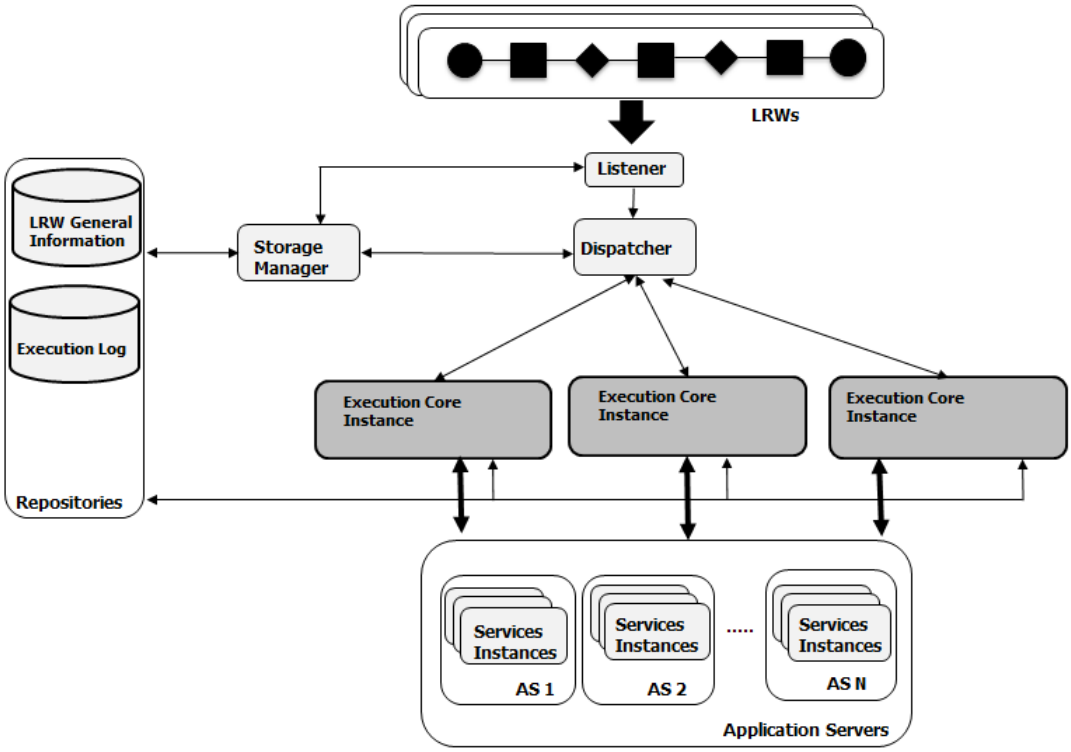
Estes servidores executam os serviços responsáveis pelas atividades dos *workflows* submetidos ao *LONGWisE4cloud*. Os *Application Servers* podem estar implantados em infraestruturas elásticas como as nuvens computacionais.

4.3.2 Dinâmica

A Figura 4.3 apresenta as interações entre os componentes do *LONGWisE4cloud*. As setas representam estas interações definidas como invocações e respostas de mensagens presentes nas interfaces dos componentes.

O *Listener* se comunica apenas com os repositórios, utilizando o *Storage Manager* como intermediário, e com o *Dispatcher*. Este último, por sua vez, se comunica com as *ECIs* para encaminhamento dos *workflows* e de operações a serem executadas sobre eles. Nesta figura também é possível observar a ligação entre o ambiente e alguns elementos externos, como os serviços que executam atividades dos *workflows*, implantados em servidores de aplicação acessados pelas *ECIs*. O acesso destas *ECIs* aos repositórios do *LONGWisE4cloud* também ocorre pela necessidade de registro de informações relativas à execução dos WLDs.

Figura 4.3: Interações Entre Componentes do *LONGWisE4cloud*



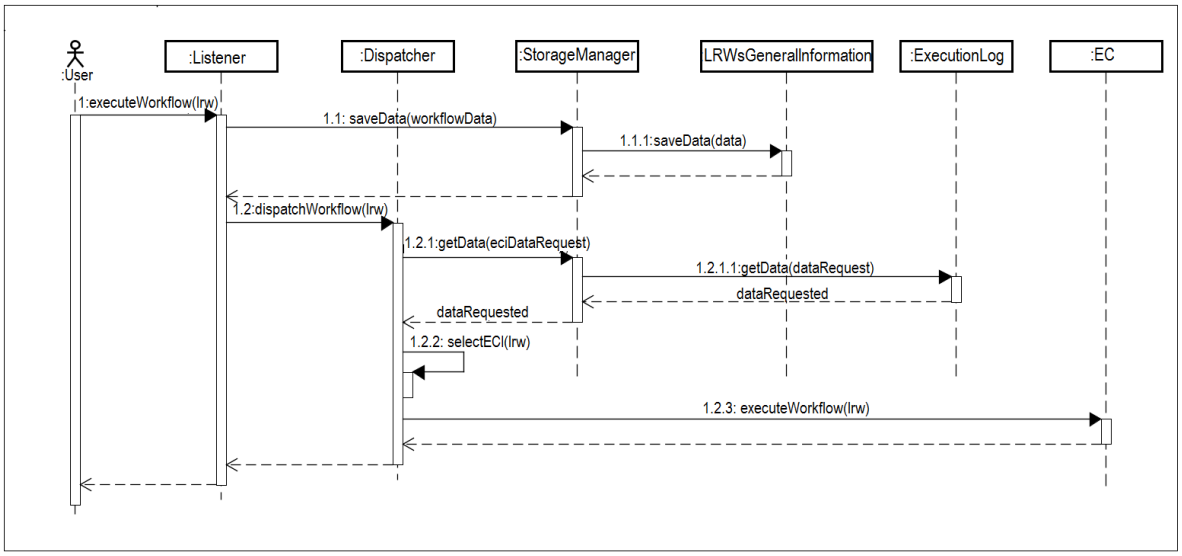
Fonte: O próprio autor.

Um aspecto dinâmico importante do *LONGWisE4cloud* é o processo que compreende desde a chegada de um WLD ao ambiente até seu encaminhamento para uma *ECI* com o consequente início da execução deste *workflow*.

A Figura 4.4 mostra que o processo de execução de um WLD é iniciado a partir do momento em que o *LONGWisE4cloud* recebe um *workflow* através do *Listener* (*executeWorkflow(lrw)*). Em seguida, o *Listener* realiza o armazenamento das informações do WLD no repositório *LRWs General Information*. Entre estas informações estão o identificador do usuário que submeteu o *workflow* ao ambiente, datas e horários de submissão, referências das atividades

que o compõem, parâmetros e restrições para sua execução. Estes dados são levados efetivamente ao *LRWs General Information* através do *Storage Manager*, que recebe-as, formata-as e executa as instruções para inclui-las no repositório. A sequência de invocação dos métodos *saveData(workflowData)* e *saveData(data)* representa estas ações de registro que ocorrem no ambiente quando um WLD precisa ser executado. Os parâmetros *workflowData* e *data* correspondem, respectivamente, às informações do *workflow* que será executado e a estas mesmas informações formatadas e prontas para serem incluídas no repositório.

Figura 4.4: Sequência para Início da Execução de um WLD no *LONGWisE4cloud*



Fonte: O próprio autor.

Após este primeiros passos de recebimento e registros iniciais do WLD, o *workflow* é encaminhado para o *Dispatcher* (*dispatchWorkflow(lrw)*). O *Dispatcher*, conforme já apresentado na Seção 4.3.1, direciona o *workflow* para uma *ECI* de forma que ele possa ter sua execução orquestrada. Antes disso, o *Dispatcher* analisa para qual *ECI* disponível no ambiente ele encaminhará o WLD. Para realizar esta análise, ele usa o *Storage Manager* para recuperar informações armazenadas no *Execution Log* sobre a execução dos *workflows* ao longo das *ECIs* (*getData(eciDataRequest)* e *getData(dataRequest)*). O parâmetro *eciDataRequest* indica ao *Storage Manager* o tipo de informação que se está buscando. O parâmetro *dataRequest* consiste de uma consulta formatada a ser enviada ao repositório.

A partir dos resultados da invocação da análise (*selectECI(lrw)*), o *Dispatcher* verifica, por exemplo, se há uma *ECI* sem nenhum processamento sendo executado e outra com dezenas de atividades em andamento. Neste caso, a primeira deverá ser priorizada por ele no direcionamento do *workflow*. Por outro lado, se um WLD chega com a restrição que ele deve ser executado em uma *ECI* dotada apenas de *workflows* pertencentes ao mesmo usuário que o submeteu ao ambiente, o *Dispatcher* deverá atender a esta exigência. Sendo assim, se já houver uma *ECI* apenas com processos do usuário ou sem nenhum processamento sendo realizado, ela

será utilizada. Caso contrário, o *Dispatcher* fará direcionamentos de novos *workflows* para outras *ECIs* até que uma delas seja finalmente exclusiva daquele usuário ou não apresente mais nenhum processamento. Isto pode acontecer também com novas *ECIs* acrescentadas ao ambiente. Processo similar ocorre quando a parametrização do WLD exige que o mesmo tenha uma *ECI* exclusiva para ele. Finalmente, o método *runWorkflow(lrw)* representa a invocação, partindo do *Dispatcher*, para que a *ECI* escolhida execute o *workflow* passado como parâmetro.

4.3.3 Aspectos Estruturais da *ECI*

Conforme mencionado na Seção 4.3.1, o *Execution Core* é responsável pelo gerenciamento da execução dos *workflows* de longa duração submetidos ao *LONGWisE4cloud*. Trata-se de um componente importante para a escalabilidade do ambiente, uma vez que múltiplas instâncias deste componente (*ECIs*) podem ser implantadas em infraestruturas distribuídas e elásticas, como os ambientes virtualizados encontrados em nuvens computacionais, para atender à demanda pela execução de *workflows*. O gerenciamento da execução dos WLDs se baseia na orquestração dos serviços que compõem estes WLDs. O uso do *Execution Core* se responsabilizando pela sequência de execução das atividades dos *workflows* simplifica o desenvolvimento da solução, uma vez que novos serviços podem ser acrescidos a este *core* de forma localizada, respeitando, assim, princípios de *SOC* como a modularidade. A robustez do *LONGWisE4cloud* também é reforçada com o uso das *ECIs*, uma vez que em situações críticas, tais como falhas em componentes de um *core*, os problemas decorrentes podem ser isolados em apenas uma parte da solução.

A Figura 4.5 apresenta os componentes do *Execution Core*.

- *ECIListener*

Este componente é responsável por receber os *workflows* enviados pelo *Dispatcher*, em um processo detalhado na Seção 4.3.2, antes de repassar os WLDs para os componentes que efetivamente promoverão suas execuções. Outras operações sobre *workflows* já submetidos ao ambiente também são recebidas e respondidas pelo *ECIListener*, como o cancelamento da execução de instâncias de um WLD numa *ECI*.

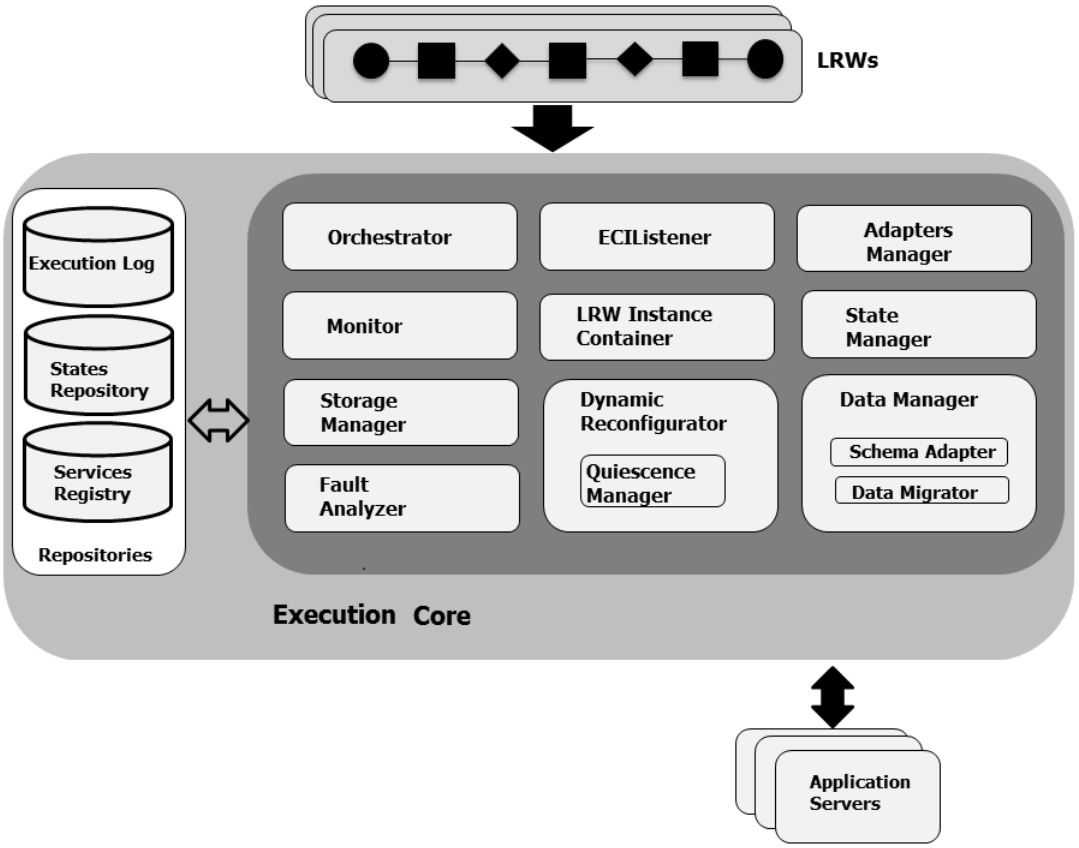
- *LRW Instance Container*

Este componente armazena as *instâncias* dos WLDs submetidas ao ambiente. Todas as ações de monitoramento, gerenciamento e controle do processamento destes *workflows* são realizadas através do acesso a estas *instâncias* dentro deste *container*.

- *Orchestrator*

A função do *Orchestrator* é coordenar a invocação dos serviços que executam as atividades definidas nos WLDs.

Figura 4.5: Estrutura de Componentes do *Execution Core*



Fonte: O próprio autor.

- *Adapters Manager*

A responsabilidade do *Adapters Manager* é controlar os componentes que realizam adaptações no *workflow* antes que ele tenha sua execução iniciada pelo *Orchestrator*. Estes componentes são os *adaptadores*, elementos capazes de trocar referências a serviços que compõem os WLDs. Eles podem direcionar uma atividade para ser executada por um serviço específico que atende a requisitos estabelecidos na definição do *workflow*, como o uso de um banco de dados pré-definido. Conforme descrito na Seção 4.1, este direcionamento de atividades é capaz de dar suporte ao provisionamento de recursos. O uso dos adaptadores é um mecanismo de adaptação mais simples do que a reconfiguração dinâmica, que será explicada adiante nesta seção, uma vez que a lógica dos adaptadores é baseada em trocas de referências a serviços considerando informações que acompanham os *workflows* e que são aplicadas antes de sua execução começar.

- *Dynamic Reconfigurator*

Este é o componente responsável por realizar reconfigurações dinâmicas nos *workflows* que estão sendo processados no ambiente. Reconfigurações são acionadas por eventos como alterações em regras de negócio identificadas a partir da chegada de novas *versões* de um *workflow* ao ambiente ou ainda quando um serviço responsável por uma atividade falha. Ou

seja, o *Dynamic Reconfigurator* tem um papel importante na robustez do *LONGWisE4cloud*. A partir das adaptações realizadas por ele, falhas e mudanças em regras de negócio podem ser tratadas pelo ambiente, evitando ações como o reinício do processamento de um WLD, algo que, conforme discutido na Seção 1.2, pode acarretar transtornos para o contexto onde o *workflow* está sendo aplicado. Além disso, embutido no *Dynamic Reconfigurator* existe o *Quiescence Manager*, cujo objetivo é garantir que todos os componentes envolvidos em uma mudança durante um processo de reconfiguração estejam em um *estado quiescente* (ver Seção 2.6.1).

- *Fault Analyzer*

A função deste componente é analisar continuamente os registros do *Execution Log* para identificar a ocorrência de falhas na execução dos WLDs. O *Fault Analyzer* também tem uma participação direta no mecanismo relativo à robustez do *LONGWisE4cloud*. Uma vez que o componente detecta uma falha registrada no *Execution Log*, ele aciona o *Dynamic Reconfigurator* para que este último possa executar adaptações direcionadas a sanar a falha, seguindo os passos detalhados na Seção 4.3.4.

- *Monitor*

O objetivo do *Monitor* é coletar continuamente dados a partir dos WLDs em execução e alimentar não apenas o *Execution Log* como também o *States Repository*. Como exemplo das informações coletadas estão o tempo de execução dos WLDs, de suas atividades e a ocorrência de algum tipo de falha no processamento quando submetidos às *ECIs*. O armazenamento deste histórico pode ser consultado posteriormente por componentes do próprio *LONGWisE4cloud* ou por usuários do ambiente. A análise destas informações servem de base para a execução de ações como a criação de novas *ECIs* ou a desativação de *ECIs* que não estão conseguindo executar apropriadamente os *workflows*, e.g., em virtude de falhas na infraestrutura onde estão implantadas. O monitoramento provido por este componente também faz parte dos recursos do *LONGWisE4cloud* voltados à robustez do ambiente. As falhas em serviços associadas ao processamento, por exemplo, são identificadas com base nas informações coletadas pelo *Monitor* e tratadas através de mecanismos adaptativos presentes no ambiente.

- *Storage Manager*

Este componente é idêntico ao introduzido na Seção 4.3.1. Ele serve de intermediário para todos os componentes da *ECI* que precisam armazenar ou recuperar informações nos repositórios que fazem parte do *LONGWisE4cloud*. O *Storage Manager* recebe as requisições e promove a busca, inserção ou alteração dos dados nos componentes de armazenamento do ambiente de execução de WLDs.

- *State Manager*

O objetivo deste componente é garantir que os estados dos serviços envolvidos na execução de processos de negócio sejam apropriadamente armazenados e recuperados em ações críticas, tais como uma reconfiguração dinâmica. Ele também é responsável por prevenir a ocorrências de problemas como, por exemplo, a perda de estados intermediários relacionados aos elementos de processamento mencionados anteriormente. Há situações onde o *State Manager* precisa das informações coletadas pelo *Monitor* para poder realizar, por exemplo, a restauração do estado de um serviço em outro que irá continuar o processamento do primeiro. Um exemplo destas situações é quando o serviço a ser substituído não mais responde a chamadas em sua interface. Neste caso, o *State Manager* não tem como recuperar o estado do serviço imediatamente antes de trocá-lo, tendo que utilizar informações prévias de monitoramento coletadas pelo *Monitor*. Quando a troca ocorre com o componente a ser substituído ainda ativo, o próprio *State Manager* pode fazer a captura antes da substituição, garantindo que o estado a ser recuperado é mais próximo do momento da permuta.

- *Data Manager*

Este gerenciador de dados é capaz de realizar alterações em esquemas de dados e executar migrações de informações entre serviços de armazenamento. Ele se torna fundamental em cenários onde o reuso de resultados intermediários do WLD é uma alternativa desejável. O *Data Manager* é o responsável pelo gerenciamento de dados mencionado na Seção 4.1. Ele depende de informações a respeito de esquemas e dados existentes em propriedades dos *workflows* e discutidas com mais detalhes no Capítulo 5. O *Data Manager* também é capaz de executar rotinas relacionadas com o reuso, migração e transformações de dados que podem ser submetidos como parâmetros dos WLDs.

- *Services Registry*

Repositório responsável por armazenar referências a serviços que podem ser utilizados pelo *LONGWisE4cloud*.

- *States Repository*

Componente que armazena estados dos WLDs, atividades e serviços. Estas informações podem ser utilizadas, por exemplo, quando uma reconfiguração é necessária e o estado de um componente precisa ser restaurado de forma que um processamento possa ser continuado após uma interrupção para mudanças.

- *Execution Log*

Trata-se do mesmo componente de armazenamento apresentado na Seção 4.3.1. O repositório é único para todo o *LONGWisE4cloud* independente do número de *ECIs* ativas em um determinado instante.

Além destes componentes que fazem parte do *LONGWisE4cloud*, é importante salientar que há também requisitos para os serviços que são utilizados para a execução das atividades dos *workflows* cuja execução será gerenciada pelo ambiente. As interfaces destes serviços, por exemplo, devem ser capazes de responder a algumas mensagens, de forma a garantir que o *LONGWisE4cloud* possa interagir apropriadamente com eles. Estas mensagens podem ser agrupadas em quatro conjuntos principais: *Contract, Executing, Monitoring* e *Change Signaling*.

Mensagens *Contract* compreendem os métodos de acesso às informações dos serviços, tais como: aspectos de segurança (por exemplo, algoritmos de criptografia, uso de chaves assimétricas) e mecanismos de armazenamento utilizado (por exemplo, banco de dados relacional, *NoSQL*). Estas informações são utilizadas pelo *LONGWisE4cloud* para selecionar serviços a serem utilizados em adaptações, por exemplo. As mesnagens do tipo *Executing* são referentes às ações mais diretamente ligadas à execução dos serviços, como o início, suspensão, retomada do andamento (após uma suspensão) ou o cancelamento destas execuções. Por sua vez, as mensagens do tipo *Monitoring* incluem um conjunto de métodos voltados às ações de monitoramento dos serviços. Finalmente, mensagens de *Change Signaling* definem métodos que funcionam como ponto de entrada para sinalização de mudanças tais como a necessidade do serviço iniciar as ações visando a entrada em um estado de quiescência.

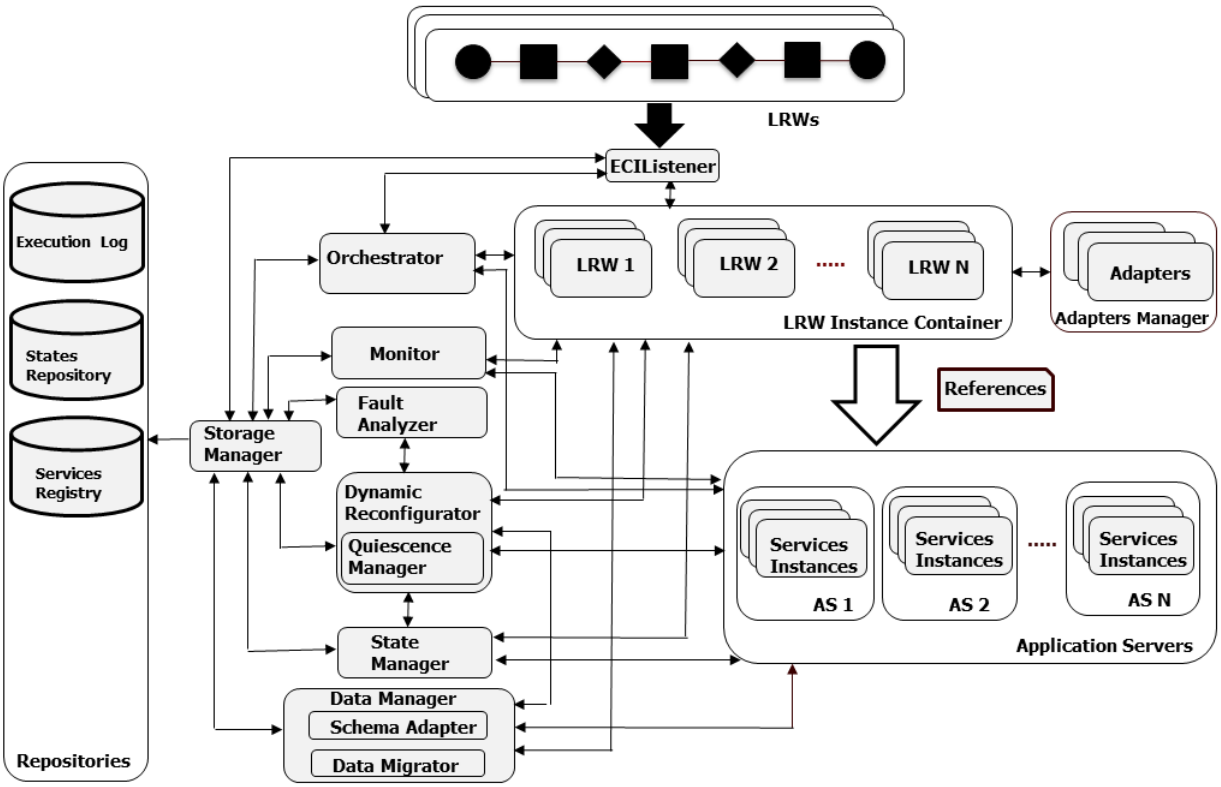
4.3.4 Aspectos Dinâmicos de uma *ECI*

A Figura 4.6 apresenta as múltiplas interações entre os componentes do *Execution Core* para a realização de suas funcionalidades voltadas ao gerenciamento da execução de WLDs. Como exemplo destas funcionalidades estão a invocação dos serviços responsáveis pelas atividades dos *workflows* ou as adaptações suportadas pelo *LONGWisE4cloud*. As setas representam as interações entre os componentes, definidas como invocações e respostas de mensagens presentes nas interfaces destes elementos, em uma dinâmica que não apenas promove a execução de processamentos, mas também gera fluxos de informações entre os componentes apresentados. Os detalhes destas funcionalidades, ações e fluxos serão detalhados adiante nesta seção.

O *Storage Manager* interage, conforme mencionado anteriormente, com todos os componentes interessados em acessar os repositórios de armazenamento do *LONGWisE4cloud*. O *ECIListener* interage com o *Storage Manager* e com o *LRW Instance Container* para, respectivamente, armazenar informações sobre as requisições que chegaram à *ECI* (e.g., executar um WLD) e encaminhar estas requisições aos componentes responsáveis por realizá-las. O *LRW Instance Container* é acessado por muitos componentes, tais como o *Orchestrator* ou o *Monitor*, uma vez que, a partir das instâncias dos WLDs, é possível interagir com os serviços responsáveis pelas atividades dos *workflows*. Esta interação é possível porque estas instâncias possuem referências a estes serviços computacionais.

Um dos processos mais importantes relativos à dinâmica de uma *ECI* diz respeito à

Figura 4.6: Detalhes Dinâmicos de uma *ECI*



Fonte: O próprio autor.

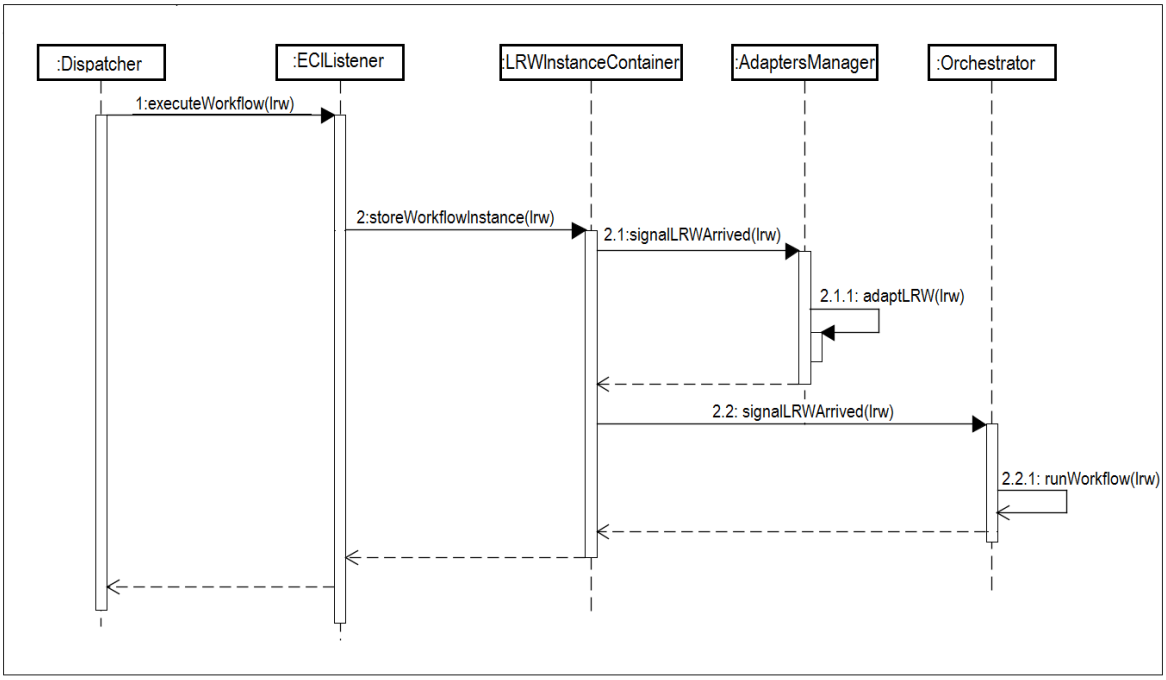
execução dos WLDs. A Figura 4.7 mostra uma diagrama *UML* de sequência detalhando os passos do processo de chegada de um WLD a uma *ECI* e o início da execução do *workflow*.

A execução de um WLD em uma *ECI* do *LONGWisE4cloud* inicia quando o *ECIListener* recebe o *workflow* do *Dispatcher* (*runWorkflow (lrw)*). Em seguida, o WLD é repassado ao *LRW Instance Container*. Uma vez armazenados neste componente, ele alerta o *AdaptersManager* e o *Orchestrator* sobre a chegada de um *workflow* para ser executado pelo núcleo do *LONGWisE4cloud*. A mensagem *signalLRWArrived(lrw)* representa estas sinalizações. Inicialmente, os *Adapters* podem mudar a composição de acordo com informações existentes na definição do *workflow*, levando a uma adaptação conforme explicado na Seção 4.3.3. Em seguida, o *Orchestrator* finalmente inicia o processamento do WLD. Há um retorno ao *Dispatcher* informando que o WLD está em execução. O usuário que submeteu o *workflow* tomará conhecimento de que a submissão (feita através do *Listener*) está completa e ele poderá acompanhar a execução de seu processo de negócio a partir de informações geradas pelo ambiente.

Enquanto os WLDs que chegaram ao *LONGWisE4cloud* estão tendo suas atividades executadas, o *Monitor* coleta informações já mencionadas anteriormente sobre o processamento destes *workflows* para, por exemplo, registrar o histórico destas execuções no *Execution Log* do *LONGWisE4cloud*. Estes dados são úteis para diversas ações relacionadas ao contexto do ambiente, como o acompanhamento, por parte de seus usuários, da execução dos processos de negócio submetidos ou ainda na tomada de decisões do *Dispatcher* sobre o direcionamento de

workflows para as *ECIs*, conforme já explicado na Seção 4.3.2.

Figura 4.7: Sequência de Início da Execução de um WLD em uma *ECI*



Fonte: O próprio autor.

Enquanto as instâncias de WLDs armazenadas no *LRW Instance Container* são executadas pelo *Orchestrator*, o *Dynamic Reconfigurator* pode identificar a necessidade de adaptações. Elas ocorrem, por exemplo, quando uma regra de negócio é alterada e esta mudança chega ao ambiente através de novas *versões* de um *workflow*, ou quando um serviço responsável por uma atividade se torna problemático, ficando muito lento ou não mais respondendo a invocações.

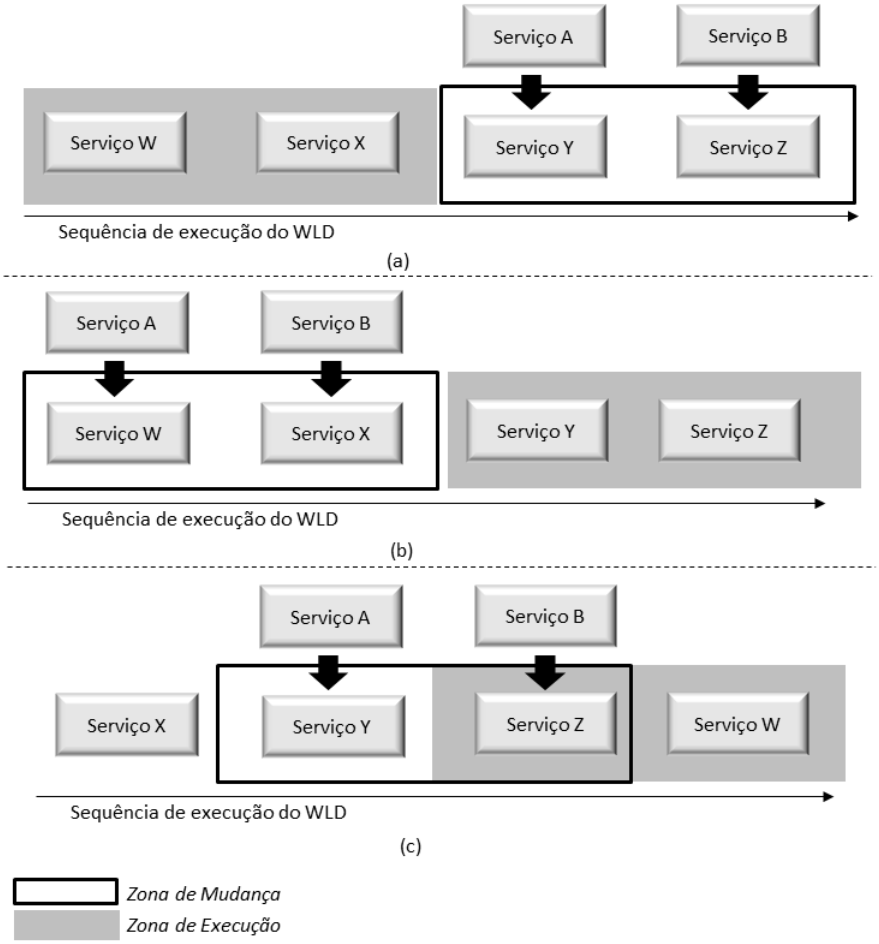
4.3.4.1 Zonas de Execução e Mudança

Um aspecto importante para as adaptações que ocorrem com os *workflows* em execução é o uso do conceito de *zonas*, denominadas *zona de execução* e *zona de mudança*. A primeira compreende o conjunto de serviços que estão efetivamente sendo executados quando a necessidade de uma adaptação é detectada. A segunda é composta por serviços que mudam (e.g., substituídos, removidos e assim por diante) no processo de adaptação.

A posição relativa destas duas zonas define três diferentes *cenários de reconfiguração* (*CR1*, *CR2* e *CR3*), conforme mostrado na Figura 4.8. Assumindo que os serviços *A* e *B* vão substituir outros dois serviços em execução, um cenário é associado ao conjunto de ações executadas pelo *Dynamic Reconfigurator*: remover ou substituir componentes, reutilizar resultados intermediários do WLD ou até mesmo reiniciar o processamento.

A identificação do *cenário de reconfiguração* é realizada pelo *Dynamic Reconfigurator* através de um conjunto de passos, onde o *workflow antigo* (que já tem instâncias em execução

Figura 4.8: Cenários de Reconfiguração Dinâmica



Fonte: O próprio autor.

no ambiente) é comparado ao *workflow novo* (que deverá ter suas instâncias executadas) através da análise das suas respectivas listas de atividades, verificando-se, nestas listas, quais atividades do *workflow antigo* devem ser modificadas para ele se igualar ao *novo*. As mudanças a serem aplicadas sempre ocorrem nas instâncias dos *workflows antigos*, uma vez que eles é quem devem ser adaptados para que suas instâncias ativas sejam concluídas com o atendimento, por exemplo, a novas regras de negócio definidas para o processo que eles modelam. Quando as atividades que estão sendo executadas no *workflow antigo* são *anteriores* às que devem ser modificadas, então o *CR1* é estabelecido. Se as atividades em execução no *wokflow antigo* são *posteriores* às mudanças, então o *CR2* é identificado. Finalmente, quando há interseção entre as atividades a serem modificadas e as que estão sendo executadas então o *CR3* é o cenário com o qual os componentes envolvidos na adaptação terão que lidar.

De acordo com o cenário de reconfiguração identificado, os componentes que fazem parte do *LONGWisE4cloud* executam uma série de ações para que a adaptação ocorra. A primeira delas é o *planejamento da adaptação*, realizado pelo *Dynamic Reconfigurator*. Este planejamento compreende não apenas a identificação do cenário de reconfiguração e a definição de ações específicas para cada cenário. Ele também envolve a análise de parâmetros submetidos por

usuários, conforme será visto mais adiante, que podem conter desde rotinas de auxílio para situações mais complexas, como o reuso de dados, até sinalizações para que todos os *workflows* associados a um processo de negócio sejam abortados. Este último tipo de ação pode ocorrer, por exemplo, em virtude da inviabilidade, detectada por projetistas, de adaptá-los a novas regras de negócio que devem ser refletidas nos WLDs. Outros fatores, como, por exemplo, o esforço de se realizar o reuso de dados ou simplesmente trocar um serviço por outro e reiniciar sua execução, são o foco da análise. Estas comparações podem ser realizadas tanto pelos usuários como pelo próprio *LONGWisE4cloud* com base em informações já existentes nos históricos de execuções de rotinas presentes nos repositórios do ambiente.

Para os cenários de reconfiguração discutidos anteriormente, as ações realizadas pelo *LONGWisE4cloud* são as seguintes:

- *CR1 - zona de execução antecedendo a zona de mudança* - Figura 4.8 (a)

O *Dynamic Configurator*, através do *Quiescence Manager*, sinaliza, no *LRW Instance Container*, quais instâncias e respectivas atividades serão submetidas à adaptação, alterando propriedades destas entidades. Assim, componentes como o *Orchestrator* tomam conhecimento do processo de adaptação e não invocam atividades que estejam envolvidas com as alterações. Neste contexto, o *Quiescence Manager* é o responsável direto por estas sinalizações porque ele é que garante que os princípios da quiescência, discutidos na Seção 2.6.1, sejam atendidos. Ele não só é capaz de identificar todos os nós que devem estar associados à adaptação, como também envia um sinal para os serviços relacionados a ela de maneira a impedir que eles iniciem processos ligados ao WLD. Quando a reconfiguração é finalizada após as trocas, remoções ou inclusão de novos serviços na *zona de mudança*, os serviços que haviam sido previamente notificados pelo *Quiescence Manager* para não aceitarem requisições ou iniciarem processos, são informados, também através de suas interfaces, que podem voltar a aceitar invocações dentro da execução do *workflow*. As instâncias do WLD e suas respectivas atividades tem suas propriedades novamente alteradas para que reflitam que estão em um processamento *normal*, i.e., não estão mais sendo adaptadas.

- *CR2 - zona de execução após zona de mudança* - Figure 4.8 (b)

O *Dynamic Reconfigurator* analisa a existência de dados já gerados pelo WLD que podem, eventualmente, ser reutilizados em uma adaptação. Informações coletadas pelo *Monitor* a respeito da execução dos *workflows* auxiliam nesta tarefa. Tipos de dados e regras de negócio podem tornar este reuso inviável, uma vez que novos serviços ou novas regras de negócio podem ser totalmente incompatíveis com os resultados previamente gerados pelo WLD. Quando estas incompatibilidades são detectadas (e.g., através de um parâmetro submetido junto ao *workflow*), a execução das instâncias do *workflow* é abortada e os processamentos reiniciados a partir da *zona de mudança*. Conforme mencionado anteriormente, parâmetros passados pelo usuário nas novas *versões* do *workflow* podem indicar, por exemplo, que o processamento deve ser abortado

por completo. Alternativamente, podem vir, entre estes parâmetros, rotinas específicas que promovam o reuso dos dados já gerados e indiquem que o processamento possa continuar do ponto onde se encontra. Além disso, se atividades existentes na *zona de execução* possuem uma associação direta com atividades da *zona de mudança*, ações de suspensão de serviços e sinalizações de instâncias e atividades de forma análoga às descritas nas adaptações envolvendo o *CR1* são realizadas. Esta associação está ligada, por exemplo, à dependência que um serviço pode ter dos dados gerados por outro, algo que pode ser identificado pelo ambiente a partir de informações contidas no próprio *workflow*. Estas suspensões e sinalizações também estão ligadas ao atendimento aos princípios de quiescência e a uma garantia de que a execução de atividades de um *workflow* não causará, por exemplo, inconsistências em dados que são processados quando ações de reuso ou transformação de dados estão sendo executadas paralelamente.

- *CR3 - Interseção das zonas de mudança e execução* - Figure 4.8 (c)

A posição relativa da interseção tem uma influência direta no *planejamento da adaptação* realizado pelo *Dynamic Reconfigurator*, uma vez que ela pode gerar cenários ainda mais específicos como os descritos a seguir. As ações relacionadas adiante são tomadas considerando que as análises previamente descritas nesta seção e realizadas pelo *Dynamic Reconfigurator* levaram à conclusão de que a adaptação é possível de ser realizada. Além disso, embora sejam feitas menções a reusos e retomadas de processamentos, para os cenários apresentados adiante também são válidas as explicações feitas anteriormente nesta seção, onde informações analisadas no *planejamento da adaptação* (e.g., parâmetros dos usuários, análises de históricos de execução de rotinas existentes no ambiente) podem levar o *LONGWisE4cloud* a realizar, por exemplo, o reprocessamento completo de atividades que fazem parte do *workflow* após trocas de componentes.

- *CR3.1* - A *zona de execução* da composição contém a *zona de mudança*

  Os serviços da *zona de mudança* têm sua execução suspensa e sinalizações em instâncias e atividades são realizadas de forma similar ao que ocorre para adaptações relativas aos cenários previamente explicados. Ações como trocas de serviços, reuso de dados e estados são feitas e, assim, os processamentos são retomados dos pontos onde foram suspensos.

- *CR3.2* - A *zona de mudança* contém a *zona de execução*

  Neste caso, quem tem sua execução suspensa são os serviços da *zona de execução*. Sinalizações em instâncias e atividades ocorrem de forma similar ao que acontece quando o cenário *CR2* é observado, uma vez que serviços já finalizados e anteriores à *zona de execução* (mas dentro da *zona de mudança*) podem ter uma associação com os que ainda estão em execução, assim como explicado para o cenário *CR2*. Após as ações relativas à adaptação serem concluídas, os processamentos são retomados dos pontos onde foram suspensos.

- *CR3.3* - Há uma coincidência completa entre as zonas

  Este cenário é observado, por exemplo, quando um único serviço, responsável por uma atividade de um *workflow*, tem que substituir outro. Os serviços da *zona de mudança* têm sua execução suspensa conforme mencionado nos cenários anteriores. Sinalizações em instâncias e atividades para identificar que estão passando por um processo de adaptação também ocorrem de forma idêntica à descrita anteriormente. Os processamentos continuam a partir dos pontos de interrupção.

- *CR3.4* - A *zona de mudança* tem seu início antes da *zona de execução* e sua porção final é *interna* à *zona de execução*

  Os serviços da *zona de mudança* que estão dentro da *zona de execução* juntamente com os que estão diretamente associados a eles têm sua execução suspensa de forma idêntica ao que ocorre quando o cenário *CR2* é observado. As mesmas sinalizações, interrupções e ações de adaptação (trocas de serviços, reuso de dados, estados e assim por diante) já mencionadas são realizadas antes dos processamentos interrompidos serem retomados.

- *CR3.5* - A *zona de mudança* tem seu início *internamente* à *zona de execução* e termina após esta última.

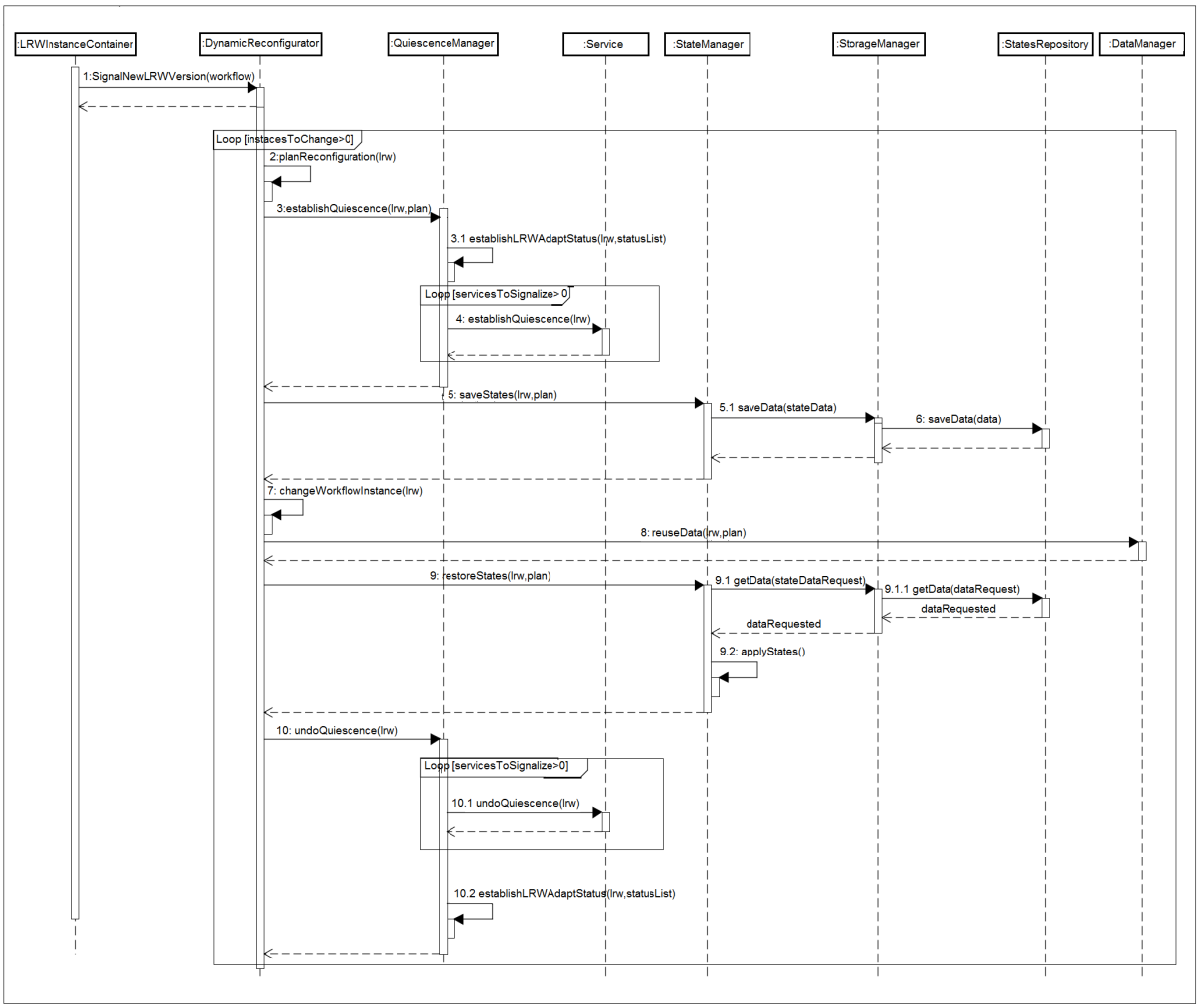
  Neste caso, os serviços da *zona de mudança* que estão dentro da *zona de execução* têm sua execução suspensa. As mesmas sinalizações, interrupções e ações de adaptação (permutas de serviços, reuso de dados, estados e assim por diante) também mencionadas anteriormente são realizadas e processamentos suspensos são retomados.

4.3.4.2 Reconfigurações Dinâmicas

A Figura 4.9 mostra, em um diagrama *UML* de sequência, as mensagens trocadas para detalhar ações que são executadas quando uma reconfiguração dinâmica é realizada. A adaptação apresentada é aplicada quando o contexto envolve a chegada de uma nova *versão* de um WLD e há a possibilidade de mudanças envolvendo identificação de cenários, reusos de dados, estados e retomada de processamentos em instâncias ativas do *workflow*. Vale salientar que a chegada de novas *versões* do *workflow* com sinalizações de alterações em regras de negócio não é a única forma de se iniciar um processo de reconfiguração. Conforme já discutido anteriormente, o mau funcionamento de um serviço é um outro exemplo de ocorrência capaz de acarretar mudanças em WLDs sendo executados e será discutida mais adiante nesta seção.

No caso do cenário da Figura 4.9, considera-se que a ação que sinalizou o *Dynamic Reconfigurator* sobre a necessidade de uma adaptação foi iniciada a partir do *LRW Instance Container* através da mensagem *signalNewLRWVersion(workflow)*, onde o parâmetro é referência ao *workflow* cujas instâncias (referenciadas nos demais métodos por *lrw*) se submeterão ao

Figura 4.9: Sequência de Reconfiguração Dinâmica no *LONGWisE4cloud*



Fonte: O próprio autor.

processo de reconfiguração. Estas novas *versões* são, na verdade, *workflows* que representam processos de negócio que já estão tendo suas execuções gerenciadas pelo ambiente, mas que chegaram com alterações que impactam em instâncias ativas, como a necessidade da troca de um serviço responsável por uma atividade em virtude de mudanças em regras de negócio.

Após tomar conhecimento da chegada de uma nova *versão* de um *workflow*, o *Dynamic Reconfigurator* planeja e executa, para cada instância ativa, conforme já explicado anteriormente, as ações de reconfiguração que são necessárias. O *loop* mais externo da Figura 4.9 define estas ações. No *LONGWisE4cloud*, o conjunto de passos deste *loop* é executado em uma *thread* independente para cada instância a ser modificada.

A mensagem *planReconfiguration(lrw)* representa o planejamento da reconfiguração. Nela, são feitas comparações entre as versões dos WLDs, o que deve e/ou pode ser modificado nas instâncias em andamento, ou se vieram parâmetros junto da nova *versão* para auxiliar neste tipo de adaptação. Um exemplo típico destes parâmetros podem ser referências a serviços que ajudam (ou realizam) o processo de reuso de dados ou até mesmo uma sinalização de que todas as

instâncias devem ser abortadas e reiniciadas conforme já explicado anteriormente nas descrições dos cenários de reconfiguração.

No caso apresentado na Figura 4.9, o resultado dos planejamentos estabelece que haverá mudanças nas instâncias em execução com o consequente reuso de dados e estados. O *Dynamic Reconfigurator* invoca o *Quiescence Manager* (*establishQuiescence(lrw,plan)*) para que este último garanta a *tranquilidade* estabelecida pela quiescência envolvendo os serviços a serem modificados na composição do *workflow*. O atributo *plan* contém informações importantes resultantes do planejamento da adaptação, como a *zona de mudança* identificada e a referência à instância que disparou o processo de reconfiguração. O *Quiescence Manager* atualiza o *status* das instâncias e atividades de WLDs que serão submetidas a ações de reconfiguração (*establishLRWAdaptStatus(lrw,statusList)*) para evitar, por exemplo, que o *Orchestrador* interfira nestas instâncias e atividades enquanto a reconfiguração está sendo realizada. O *Quiescence Manager* também invoca métodos específicos (*establishQuiescence(lrw)*) nas interfaces dos serviços associados a estas atividades. Uma vez que a quiescência é estabelecida, os estados dos serviços do WLD envolvidos na mudança são salvos pelo *State Manager* no *States Repository* através do *Storage Manager*. Assim, na retomada da execução, será possível fazer com que o *workflow* seja reiniciado do ponto onde o processamento se encontrava quando do início da reconfiguração. As mensagens *saveStates(lrw,plan)*, *saveData(stateData)* e *saveData(data)* representam este fluxo de ações.

As inclusões, remoções e trocas de serviços são realizadas pelo *Dynamic Reconfigurator* (*changeWorkflowInstance(lrw)*) atuando diretamente nas referências destes serviços existentes nas instâncias dos *workflows*. Em seguida, o *Dynamic Reconfigurator* invoca o *Data Manager* (*reuseData(lrw,plan)*) para que o mesmo assuma ações relacionadas a dados dos *workflows*, como o reuso de resultados intermediários utilizando recursos próprios ou através de rotinas providas pelos usuários através de parâmetros dos WLDs. Uma vez que o reuso de dados é concluído, os estados de serviços salvos pelo *State Manager* são restaurados para que o processamento do *workflow* possa ser retomado do ponto onde o mesmo foi interrompido. Estas informações são recuperadas a partir do *States Repository*. As mensagens *restoreStates(lrw, plan)*, *getData(stateDataRequest)*, *getData(dataRequest)* e *applyStates()* representam esta sequência de ações relativas aos estados. O *Quiescence Manager* desfaz a suspensão na execução dos serviços (*undoQuiescence(lrw)*) realizada antes das mudanças começarem a ser aplicadas e atualiza os *status* das instâncias e atividades dos *workflows* (*establishLRWAdaptStatus(lrw,statusList)*) para que componentes como o *Orchestrator* possam voltar a interagir normalmente com estes processos em andamento.

Finalmente, vale salientar que, embora os cenários de reconfiguração descritos anteriormente sejam fundamentais para casos onde novas *versões* de *workflows* chegam ao ambiente, para as adaptações associadas a falhas no processamento de WLDs o *planejamento* foca em trocar os componentes problemáticos que deveriam estar sendo executados. A sinalização das falhas ao *Dynamic Configurator* é feita pelo *Fault Analyzer* baseada em informações de monito-

ramento conforme já explicado na Seção 4.3.3. Nestes casos, apenas os cenários que envolvem mudanças em componentes ativos fazem sentido. Definições de cenários como os associados a alterações em atividades que sequer estejam sendo processadas se tornam irrelevantes. As ações de reuso de dados e estados para estas situações seguem as mesmas regras, passos e restrições descritas anteriormente, assim como o uso de rotinas que podem ser repassadas pelo usuário para casos críticos onde nem a reconfiguração dinâmica consegue reverter a situação de falha. Estas rotinas podem, por exemplo, realizar limpezas em conjuntos de dados que não devem continuar existindo quando o *workflow* tiver sido abortado por causa de uma falha que não pôde ser sanada.

4.4 Considerações Finais

Neste capítulo, foram apresentados detalhes gerais do projeto do *LONGWisE4cloud*. Aspectos a serem cobertos pelo ambiente foram listados e detalhados. Características estruturais e dinâmicas de sua arquitetura, traduzidas na forma de componentes e serviços providos por eles juntamente com suas interações para atender às funcionalidades e aspectos necessários à execução de *workflows* de longa duração, foram detalhadas também. Todos estes elementos, definidos a partir de estudos e conceitos identificados durante a construção deste projeto de pesquisa e apresentados em capítulos anteriores, foram utilizados como base para a implementação que será discutida no Capítulo 5.

5

IMPLEMENTAÇÃO DO *LONGWISE4CLOUD*

Este capítulo apresenta detalhes a respeito da implementação do *LONGWisE4cloud*. O capítulo inicia apresentando os elementos do padrão *OSGi* utilizados no desenvolvimento do ambiente. Em seguida, são detalhadas as classes e objetos principais do *LONGWisE4cloud*. Finalmente, são detalhadas as implementação dos repositórios que fazem parte do ambiente de suporte à execução de WLDs.

5.1 *OSGi* na Implementação do *LONGWisE4cloud*

O *LONGWisE4cloud* foi desenvolvido usando *Open Services Gateway initiative* (*OSGi*), um padrão Java, apresentado em detalhes no Apêndice B, voltado ao desenvolvimento de aplicações baseadas em componentes e orientadas a serviços. O uso do *OSGi* foi fundamental para a simplificação da construção do *LONGWisE4cloud* pelas seguintes razões:

- As aplicações desenvolvidas seguindo o padrão *OSGi* tem seus componentes (*bundles*) armazenados em um repositório (*container OSGi*) que é capaz de controlar o ciclo de vida destes elementos, realizando, dinamicamente, ações como a instalação, ativação, desativação e remoção dos mesmos. O *LONGWisE4cloud* tem todos os seus componentes implantados neste *container*. Desta forma, a inclusão de novos serviços no ambiente capazes de prover suporte à execução dos WLDs, a remoção de funcionalidades ou a suspensão temporária de aspectos podem ser realizadas de forma simplificada, apenas utilizando estes recursos previamente descritos.

- O padrão oferece suporte a múltiplas versões de componentes coexistindo. Com isso, a evolução do *LONGWisE4cloud* foi realizada de forma incremental. Componentes e serviços com versões antigas e novas puderam coexistir até o momento em que apenas as funcionalidades mais recentes foram deixadas ativas no ambiente para dar suporte à execução de WLDs.

- A existência de um registro associado ao *container* onde serviços podem ser inseridos, buscados com base em propriedades e filtros ou removidos também de forma dinâmica. Além dos componentes que fazem parte do ambiente, os elementos que entram

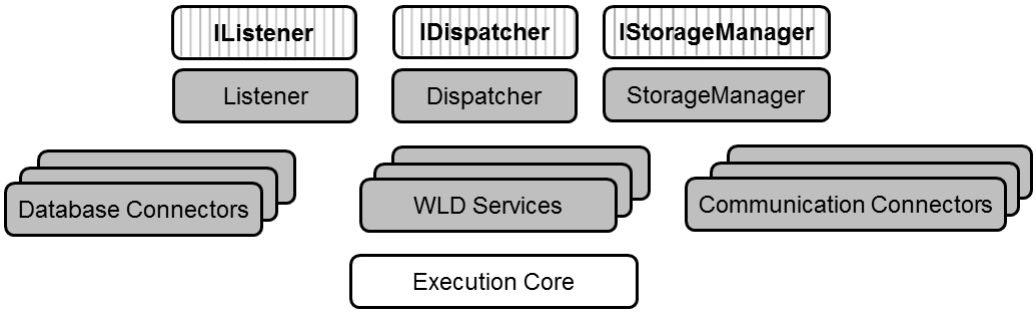
na composição dos WLDs também são registrados neste repositório do *container*. Desta forma, as execuções dos WLDs, além de ações como as relacionadas com as adaptações, que incluem buscas e substituições de serviços, podem ser facilitadas, uma vez que os recursos de armazenamento e filtragem associados a este registro de serviços simplificam a implementação destas ações;

- O padrão estabelece o gerenciamento de dependência entre os componentes. No caso do *LONGWisE4cloud* , a construção do ambiente, conforme já mencionado anteriormente, se deu a partir da integração de componentes responsáveis por prover os serviços relacionados à execução de WLDs. Quando um componente depende de outro que não está disponível no *container*, automaticamente o primeiro também fica indisponível para uso. Assim, falhas pela inexistência de componentes necessários à execução de serviços são evitadas, já que apenas quando as dependências são satisfeitas é possível invocar os serviços implementados.

- No padrão *OSGi*, há o controle da visibilidade de recursos, tais como classes e objetos, existentes nos componentes da aplicação. No *LONGWisE4cloud*, o encapsulamento de funcionalidades é obtido a partir destas visibilidades controladas de elementos de programação. Com isso, apenas classes, objetos e, consequentemente, serviços, explicitamente disponibilizados nos componentes podem ser invocados por usuários e por outros componentes.

A Figura 5.1 apresenta os *bundles OSGi* que compõem o *LONGWisE4cloud*.

Figura 5.1: *Bundles* do *LONGWisE4cloud*



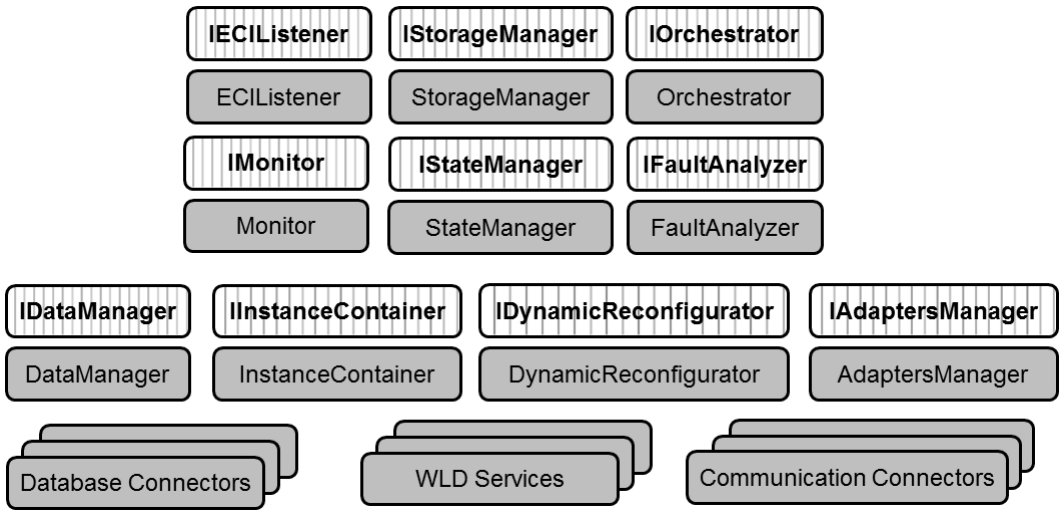
Fonte: O próprio autor.

Os elementos hachurados representam as *interfaces* dos serviços que compõem o ambiente. Esta separação entre a definição das *interfaces* e a implementação efetiva dos serviços em componentes diferentes é uma prática capaz de trazer vantagens quando se trabalha com *OSGi*. Utilizando esta estratégia, é possível, por exemplo, trocar o provedor ou até mesmo usar, quando for o caso, múltiplos provedores para uma mesma definição de interface de serviço sem necessariamente envolver trocas e mudanças no componente que contem esta definição.

Os *Database Connectors* são *drivers* e bibliotecas para acesso a bancos de dados específicos como Oracle ou *MySQL*. Os *WLD Services* são referências aos diversos *bundles* responsáveis

por prover os serviços associados às atividades dos *workflows* que chegam para serem executados no ambiente. *Communication Connectors* são os *bundles* que empacotam *APIs*, objetos e *frameworks* que viabilizam a comunicação entre serviços através do uso de mecanismos como arquivos *XML* e *RESTful Web Services* (RICHARDSON; RUBY, 2007). Por exemplo, um destes conectores permite o uso da *API* Java JAX-RS, voltada à implementação de *RESTful Web Services*, junto a componentes *OSGi*. Finalmente o *Execution Core* está associado a outro conjunto de *bundles* detalhados na Figura 5.2.

Figura 5.2: *Bundles* do *Execution Core*



Fonte: O próprio autor.

IStorage Manager, *Storage Manager*, *Database Connectors*, *WLD Services* e *Communication Connectors* estão replicados nas diversas *ECIs* existentes no ambiente, uma vez que elas podem, conforme já explicado no Capítulo 4, se encontrar distribuídas em máquinas virtuais diferentes em um ambiente de nuvem, por exemplo.

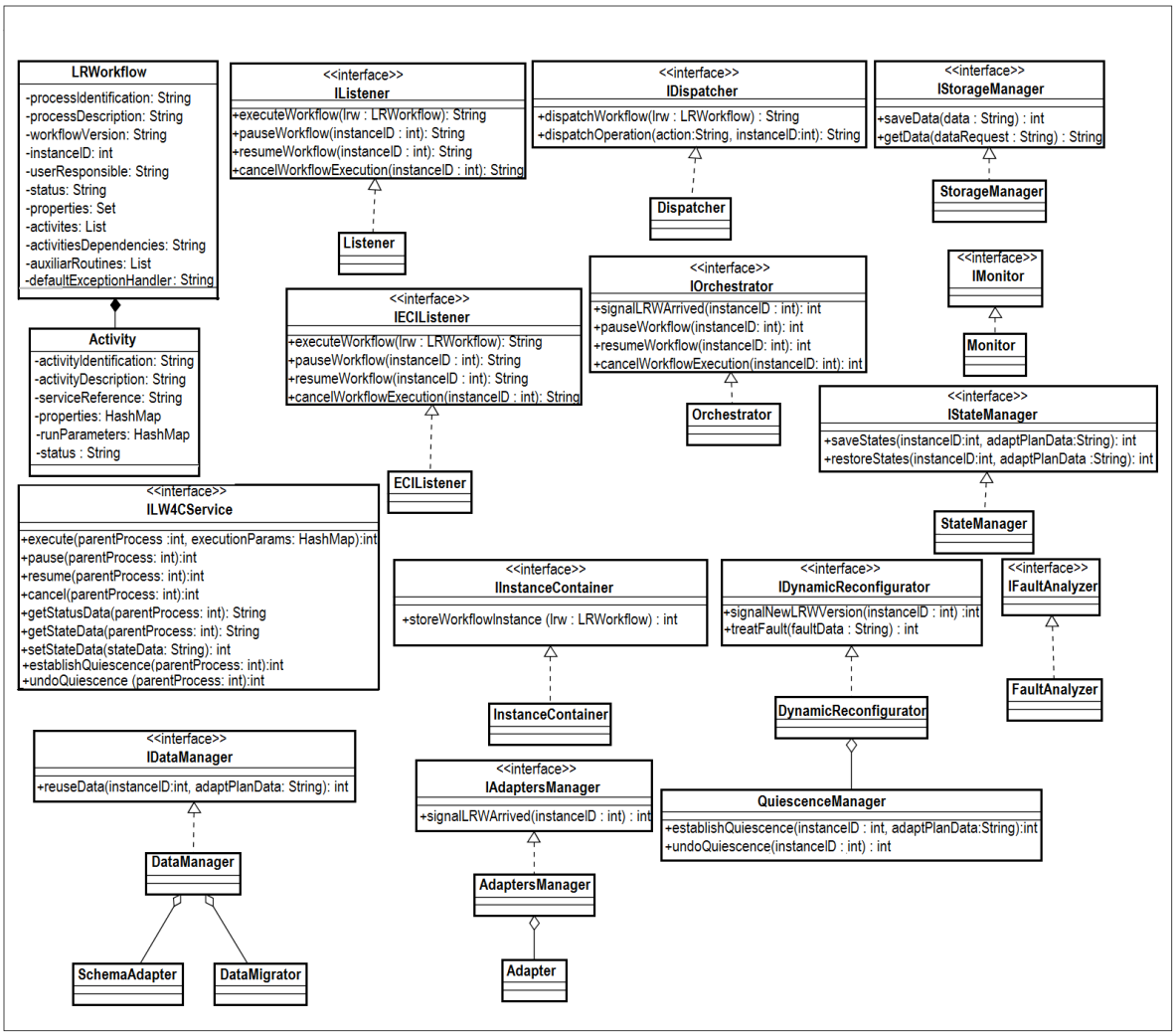
O desenvolvimento do *LONGWisE4cloud* utilizou o *Framework Equinox* (EQUINOX, 2017) como plataforma *OSGi*. Este *framework* contempla as especificações do padrão.

5.2 Classes e Objetos

A Figura 5.3 apresenta as principais classes que compõem o *LONGWisE4cloud* juntamente com seus atributos e métodos mais relevantes para entendimento de aspectos estruturais e dinâmicos do ambiente.

A classe *LRWorkflow* representa o WLD a ser submetido a uma execução gerenciada pelo *LONGWisE4cloud*. O identificador do processo de negócio (*processIdentification*) ao qual ele está relacionado assim como uma descrição a respeito deste processo (*processDescription*) são atributos desta classe. A *versão* do *workflow* (*workflowVersion*) é um atributo que é utilizado, por exemplo, para determinar mudanças no processo de negócio a ele associado. Conforme já mencionado na Seção 4.3.4, versões diferentes chegando ao ambiente podem disparar processos

Figura 5.3: Classes no *LONGWisE4cloud*



Fonte: O próprio autor.

de adaptação em instâncias ativas do *workflow*. O atributo *instanceID* da classe *LRWorkflow* é o identificador único associado ao WLD no *LONGWisE4cloud*. Ele é atribuído ao *workflow* assim que ele chega ao ambiente para ser executado. O objeto constitui, a partir deste instante, uma *instância* de um *workflow*. Este identificador é utilizado, conforme pode ser observado em parâmetros de métodos de diversas classes presentes no diagrama, como uma forma de se estabelecer qual instância será submetida a uma determinada ação executada por um dos componentes do ambiente. O usuário responsável (*userResponsible*) pela submissão do *workflow* para ser executado também está presente na classe associada aos WLDs. O *status* do *workflow* indica como está seu estado em relação à execução, podendo se encontrar *em espera* (para iniciar a execução), *executando*, *suspenso*, *cancelado*, *em reconfiguração*, *abortado* ou *concluído*.

O atributo *properties* constitui um conjunto de informações que servem para direcionar a execução do *workflow*. Elas são importantes para diversas ações que o *LONGWisE4cloud* realiza como, por exemplo, o direcionamento dos WLDs para as *ECIs* feito pelo *Dispatcher* apresen-

tado na Seção 4.3.1 ou o *planejamento* da adaptação executado pelo *Dynamic Reconfigurator* explicado na Seção 4.3.4. Uma lista das propriedades possíveis de serem utilizadas atualmente pode ser observada na Tabela 5.1.

Tabela 5.1: Propriedades para Execução de WLDs

Propriedade	**Direcionamento da Execução**
EXCLUSIVE_ECI	A instância deve ser executada em uma *ECI* sem nenhuma outra instância ativa
EXCLUSIVE_USER_ECI	A instância deve ser executada em uma *ECI* onde só existam instâncias do mesmo usuário que o responsável por ela
EXCLUSIVE_DB_SERVICE	A instância deve ser executada apenas por serviços dedicados de bancos de dados
ABORT_ALL_INSTANCES	Determina que o ambiente deve cancelar a execução de todas as instâncias ativas daquele *workflow* ao iniciar sua execução
APPLY_ROUTINES_REUSE	Determina que as rotinas auxiliares referenciadas no *workflow* devem ser aplicadas quando reusos de dados são necessários
KEEP_ACTIVE_INSTANCES	As instâncias ativas não devem passar por mudanças

A lista de atividades (objetos da classe *Activity*) que compõem o processo modelado e devem ser executadas é armazenada no atributo *activities*. As dependências entre atividades estão armazenadas no atributo *activitiesDependencies*. Estas dependências são importantes para que o mecanismo de execução do *workflow* possa saber, por exemplo, que uma atividade só pode ser iniciada após o término de outra ou que uma destas atividades depende de dados gerados por outra. Finalmente, a lista de atividades presente no atributo *auxiliarRoutines* são referências às atividades que devem ser utilizadas em situações particulares, como, por exemplo, na reconfiguração dinâmica de instâncias de WLDs onde o reuso de dados pode precisar de implementações específicas para serem realizadas. Finalmente, o atributo *defaultExceptionHandler* é uma referência a um serviço que deve ser invocado em situações críticas, como, por exemplo, uma falha em uma atividade que não consegue ser sanada pelo mecanismo de adaptação do *LONGWisE4cloud*. Este serviço implementa a interface *ILW4CService*, explicada adiante. Neste caso da invocação disparada por uma falha não contornada, ele receberá todas as informações sobre a atividade que falhou (identificadores, propriedades, parâmetros e assim por diante), para que possa tomar medidas voltadas a garantir aspectos como a eliminação de dados que não devam mais existir em virtude do cancelamento da execução do WLD.

A classe *Activity*, conforme mencionado anteriormente, representa as atividades que compõem os WLDs executados pelo ambiente. Os atributos *activityIdentification* e *activityDescription* identificam e descrevem a atividade, respectivamente. As informações *serviceReference* e *properties* são utilizadas para localizar, junto ao registro de serviços presentes no ambiente, qual serviço estará associado à atividade. O primeiro é uma referência à identificação de uma classe (ou interface), enquanto o segundo é constituído por pares compostos por um *nome* e um *valor*, o que justifica o uso de um *HashMap* para implementá-lo. Uma vez que o serviço é localizado, a execução da atividade pode ser realizada. Estas *propriedades* registradas junto aos

serviços fazem parte dos *Contracts* descritos na Seção 4.3.3. Versões de serviços utilizados, presença de recursos de segurança (e.g., criptografia de dados), tipos de armazenamento suportados pelo serviço (SGBD relacionais, *NoSQL*, bancos de dados dedicados ou compartilhados) são exemplos de informações que podem estar presentes entre estas propriedades.

O atributo *runParameters* também é definido com um *HashMap* onde há registros de nomes de parâmetros e seus respectivos valores que serão utilizados na invocação do serviço responsável por uma atividade do WLD. O *Orchestrator* é um componente que realiza estas ações de buscas e invocações de serviços à medida que um *workflow* é executado, dentro de um cenário já descrito na Seção 4.3.4. Finalmente, o *status* da atividade diz respeito ao seu estado atual dentro do contexto de execução do *workflow*, podendo estar *ativa*, *suspensa*, *passiva* (princípio da quiescência utilizado em reconfigurações dinâmicas, por exemplo), *inativa*, *em falha* ou *concluída*.

A interface *ILW4CService* permite que o *LONGWisE4cloud* possa interagir com os serviços responsáveis por executar as atividades que compõem os WLDs. O método *execute* recebe dois parâmetros na sua invocação. O primeiro (*parentProcess*) é um identificador utilizado para controlar múltiplas invocações do serviço. Considerando que um determinado serviço pode estar à disposição para ser invocado, simultaneamente ou não, a partir de diversos *workflows* pertencentes a distintos *clientes*, um *identificador de processo* deve ser utilizado para identificar unicamente um processamento específico daquele serviço. No contexto do *LONGWisE4cloud*, o *identificador de instância* do WLD é utilizado para esta função. O segundo parâmetro é uma lista com nomes e valores de parâmetros utilizados na execução do serviço. Estes parâmetros estão associados à classe *Activity* em seu atributo *runParameters*.

Os métodos *pause*, *resume* e *cancel* são responsáveis por realizar intervenções na execução do serviço. Estas três mensagens, juntamente com o método *execute*, constituem as ações relacionadas ao conjunto *Executing* da interface dos serviços mencionado na Seção 4.3.3.

O método *getStatusData* retorna dados da execução do serviço, tais como o percentual de completude do processamento, se o mesmo está ativo, suspenso, em quiescência ou inativo. Estes método faz parte do conjunto *Monitoring* referenciado na Seção 4.3.3.

As funções dos métodos *getStateData* e *setStateData* são, respectivamente, retornar e restaurar os estados relativos aos serviços que executam as atividades dos WLDs. Estes métodos são importantes em ações como reconfigurações dinâmicas onde, por exemplo, o estado de um serviço pode ter que ser restaurado após uma troca de um componente. Estas informações são de responsabilidade do serviço.

O propósito de *establishQuiescence* é implementar as ações relacionadas à quiescência, tais como suspensão das atividades correntes e bloqueio de novas invocações externas que disparem transações relativas à execução de processos de negócio. Este método compõe o grupo *Change Signaling* mencionado na Seção 4.3.3.

As demais classes e interfaces presentes no diagrama, assim como os métodos presentes nas mesmas, representam componentes e ações que já foram apresentadas e discutidas nas Seções

4.3.1, 4.3.2, 4.3.3 e 4.3.4.

Atualmente, existem implementações distintas para o *Listener* que permitem a comunicação com o componente de acordo com particularidades dos clientes do *LONGWisE4cloud*. Uma destas implementações é para a invocação de seus métodos *localmente*, o que facilita sua interação com outros objetos que tenham sido criados, por exemplo, a partir do mesmo *container OSGi* em que ele se encontra, e.g., um formulário para *desktop*. Outra implementação permite a comunicação com o *Listener* através de *RESTFul web services*, com o encaminhamento de um *workflow*, por exemplo, para ser executado pelo *LONGWisE4cloud* através do envio de um arquivo *XML* cuja estrutura se baseia no arquivo *XSD* apresentado no Anexo C.

Conforme já mencionado anteriormente, os *bundles* que compõem as *ECIs* podem estar na mesma máquina virtual dos demais *bundles* do *LONGWisE4cloud*. Neste caso, as referências a todos os componentes do ambiente podem ser diretamente acessadas através do uso de recursos, como o registro de serviços, de um único *Equinox* implantado na máquina virtual. O *Listener* invoca o *Dispatcher* que, por sua vez, invoca diretamente o *ECIListener* para acionar a execução de um *workflow* ou ainda solicitar a suspensão ou o cancelamento de uma instância ativa de um WLD.

Entretanto, quando as *ECIs* estão distribuídas por múltiplas máquinas virtuais, então o *ECIListener* passa a ser invocado pelo *Dispatcher* através do uso de um mecanismo baseado em *web services*. O encaminhamento de um *workflow* para ser executado em uma máquina remota, por exemplo, é realizado através de um *XML* com a mesma estrutura previamente citada para a comunicação remota com o *Listener*. As identificações das diversas *ECIs* juntamente com as *URLs* para invocação dos serviços providos pelo *ECIListener* são registradas no próprio *Dispatcher*, ação realizada manualmente por um usuário do *LONGWisE4cloud* com acesso direto ao serviço provido por este componente.

5.3 Repositórios

Os repositórios *LRWs General Information*, *Execution Log* e *States Repository* do *LONGWisE4cloud* foram definidos com o *MySQL* conforme descrito nas tabelas 5.2, 5.3 e 5.4.

Tabela 5.2: *LRWs General Information*

Atributo	**Tipo**	**Descrição**
LGI_ID	INTEGER	Chave primária da entidade
LGI_DATETIME	DATETIME	Data e hora da criação do registro
LGI_USER_ID	VARCHAR(20)	Usuário associado ao WLD
LGI_PROCESS_ID	VARCHAR(25)	Identificador de processo
LGI_PROCESS_DESCRIPTION	VARCHAR(100)	Descrição de processo
LGI_INSTANCE_ID	INTEGER	Identificador de instância
LGI_OPERATION	VARCHAR(8)	Identificador da operação a ser realizada no WLD
LGI_ECI_ID	VARCHAR(10)	*ECI* responsável pelo WLD
LGI_WLD_XML	TEXT	WLD completo em XML

O *Services Registry* é uma exceção, pois reusou-se o registro de serviços *OSGi* existente no *Equinox*. Embora este registro seja povoado a partir de implementações realizadas com componentes Java, estes elementos podem encapsular chamadas a serviços remotos implantados em outras infraestruturas, como nuvens computacionais públicas ou privadas, e implementados em linguagens de programação diversas. Além disso, o fato do *framework Equinox* disponibilizar o acesso a este registro de serviços diretamente aos *bundles OSGi*, os componentes do *LONGWisE4cloud* não precisam do *Storage Manager* para interagir com este repositório do ambiente.

A estrutura dos bancos de dados relacionais não está completamente normalizada pois a ideia é que as entidades trabalhem como tabelas de *fatos* em uma *modelagem dimensional* (KIMBALL; ROSS, 2002), algo que acelera ações como a inclusão de registros assim como a recuperação de informações por parte de ferramentas que fazem processamentos e análises *online*.

Tabela 5.3: *Execution Log*

Atributo	**Tipo**	**Descrição**
EXL_ID	INTEGER	Chave primária da entidade
EXL_DATETIME	DATETIME	Data e hora do registro da informação WLD
EXL_ECI_ID	VARCHAR(10)	Identificador da *ECI*
EXL_ECI_LOCATION	VARCHAR(30)	Localização da *ECI*
EXL_INSTANCE_ID	INTEGER	Identificador de instância de WLD
EXL_ACTIVITY_ID	VARCHAR(25)	Identificador da atividade
EXL_STATUS_INSTANCE	VARCHAR(10)	*Status* da instância
EXL_STATUS_ACTIVITY	VARCHAR(10)	*Status* da atividade
EXL_LOG_COMPLEMENT	VARCHAR(50)	Complemento do registro de *log*

Os tipos de registros que podem ser armazenados na Tabela 5.3 são definidos a partir de um domínio que cobre as diversas naturezas de informações monitoradas pelo ambiente quando da execução de WLDs. Sendo assim, os dados podem ser registrados na tabela indicando que uma determinada atividade (e respectiva instância) está em execução, suspensa, inativa e assim por diante. O complemento armazena informações extras tais como o percentual de completude da atividade em execução ou o motivo que levou a abortar o processamento completo do WLD.

Tabela 5.4: *States Repository*

Atributo	**Tipo**	**Descrição**
STR_ID	INTEGER	Chave primária da entidade
STR_DATETIME	DATETIME	Data e hora do registro da informação WLD
STR_INSTANCE_ID	INTEGER	Identificador de instância de WLD
STR_ACTIVITY_ID	VARCHAR(25)	Identificador da atividade
STR_STATE_DATA	TEXT	Registro de estados em formato texto

Os estados armazenados na Tabela 5.4 são, conforme já explicado na Seção 5.2, dados que podem restaurar o estado de um componente quando ações como reconfigurações dinâmicas

são executadas no ambiente.

5.4 Considerações Finais

Este capítulo apresentou detalhes a respeito da implementação do *LONGWisE4cloud*. Para este trabalho de desenvolvimento do ambiente utilizou-se o *OSGi* em virtude das facilidades oferecidas pelo padrão, tais como o controle do ciclo de vida de componentes, registro de serviços e automação de ações como iniciar, pausar ou continuar um processamento de acordo com a ativação ou inativação de componentes no ambiente. O *framework Equinox* foi utilizado como implementação do padrão *OSGi*. Os repositórios do *LONGWisE4cloud* foram definidos com um SGBD relacional (*MySQL*) à exceção do repositório de referências a serviços que utilizou o registro do próprio *OSGi*. Embora implementado na plataforma Java, o ambiente é capaz de trabalhar com serviços remotos cuja invocação pode ser realizada a partir dos componentes registrados para serem responsáveis pela execução de atividades associadas aos WLDs submetidos para a execução no *LONGWisE4cloud*.

6

AVALIAÇÃO EXPERIMENTAL

Este capítulo apresenta os experimentos realizados para avaliar o impacto, em termos de desempenho, que mecanismos do *LONGWisE4cloud* acarretam na execução de *workflows* de longa duração. Os experimentos foram projetados, executados e descritos seguindo o passo-a-passo definido em Jain (1991). Todos os WLDs utilizados na avaliação são executados por uma operadora de planos de saúde brasileira com clientes distribuídos pelo país inteiro e vinculados a diversas subsidiárias associadas à operadora. Na Seção 6.1 é detalhado o experimento que avalia o impacto do mecanismo de reconfiguração dinâmica do *LONGWisE4cloud* no desempenho da execução dos WLDs. A Seção 6.2 detalha o experimento definido para avaliar como o provisionamento de recursos influencia o desempenho do processamento dos WLDs. Na Seção 6.3, é analisado o impacto da distribuição do gerenciamento da execução dos WLDs no desempenho da execução destes *workflows*.

6.1 Reconfiguração Dinâmica

O objetivo desta avaliação experimental é medir o impacto do mecanismo de reconfiguração dinâmica na execução de *workflows* de longa duração.

O ambiente de gerenciamento de execução de WLDs utilizado nesta avaliação tem três diferentes políticas de reconfiguração. A primeira simplesmente troca componentes quando uma adaptação é necessária, sem reutilização de dados ou estados, reprocessando atividades associadas a estes componentes a partir do início. A segunda realiza a troca de componentes e reusa dados gerados pelo WLD, dando continuidade às atividades do ponto onde foram interrompidas. A terceira analisa os cenários de reconfiguração (Seção 4.3.4), dados a serem reaproveitados, esforços para reuso e reprocessamentos de atividades e executa as ações de reconfiguração necessárias.

A métrica adotada nesta avaliação é o *tempo de processamento*, que é o tempo transcorrido desde o início até a conclusão da execução do *workflow*. Ele é uma métrica natural em WLDs devido à longa duração que é característica a estes tipos particulares de *workflows*.

Existem alguns elementos (*parâmetros*) que afetam a métrica escolhida, tais como configuração de hardware, disponibilidade de recursos de rede e sistemas operacionais envolvidos

no cenário de processamento dos *workflows*. Parâmetros adicionais, tais como a política de reconfiguração adotada, medidas de segurança (e.g., algoritmos de criptografia), quantidade de dados processados pelo WLD e o próprio *workflow* também foram considerados. Finalmente, como a reconfiguração dinâmica é foco desta avaliação, o ponto onde uma atividade do *workflow* é interrompida para iniciar uma reconfiguração estabelece o tempo de execução já transcorrido da atividade até o início desta reconfiguração. Este ponto de interrupção também é um parâmetro capaz de afetar o tempo total de processamento de um WLD.

Considerando o objetivo desta avaliação, três parâmetros foram escolhidos para variar (os chamados *fatores* (JAIN, 1991)): *política de reconfiguração*, *volume de dados* processado pelo *workflow* e o *ponto de interrupção* da atividade que será o alvo das ações executadas pela reconfiguração dinâmica. O primeiro foi escolhido por ser o foco principal das análises do experimento. A quantidade de dados processados pelas atividades do WLD é, normalmente, um aspecto crítico quando se analisa seu desempenho. Finalmente, o ponto de interrupção de uma atividade que se submeterá a mudanças é um aspecto essencial para a escolha das ações executadas durante a reconfiguração. Isto ocorre porque quanto mais tempo de execução já estiver transcorrido, maior a probabilidade de existir um grande volume de informações já geradas e isto tem impacto direto no aproveitamento de resultados intermediários.

A Tabela 6.1 mostra os fatores e seus respectivos *níveis*, que são os valores assumidos pelos fatores no experimento. Conforme mencionado anteriormente, três políticas de reconfiguração foram consideradas: *Troca de componentes e reprocessamento (TCR)*, *Troca de componentes e reuso de dados (TCRED)* e a *Solução de reconfiguração completa do LONGWisE4cloud (SRL4C)*. A primeira apenas troca componentes e reinicia serviços. A segunda reaproveita dados já gerados pelo *workflow*. A terceira corresponde ao uso de todos os recursos associados à reconfiguração dinâmica implementados no *LONGWisE4cloud* e apresentados no Capítulo 4. O volume de dados está associado ao número de clientes pertencentes a diferentes subsidiárias da operadora. Este número classifica estas subsidiárias em diferentes categorias ou *portes*. Devido ao seu impacto no tempo total de processamento do *workflow*, a atividade escolhida para se submeter a mudanças é a mais longa do WLD. Finalmente, os pontos de interrupção da atividade submetida a mudanças ao longo do processo de reconfiguração foram estabelecidos na forma de percentuais relativos ao tempo total estimado de processamento da atividade. Os percentuais escolhidos ajudam a entender o impacto da reconfiguração quando a atividade é interrompida não apenas no seu início e próximo ao seu final (valores extremos), mas também em pontos intermediários de execução da mesma.

Tabela 6.1: Fatores e Níveis - Avaliação da Reconfiguração Dinâmica

Fator	**Níveis**
Política de Reconfiguração	*TCR, TCRED, SRL4C*
Número de Clientes	*10.000*, *50.000*, *100.000*
Ponto de Interrupção da Atividade-Alvo	*5%*, *25%*, *50%*, *75%*, *95%*

O WLD adotado como *carga de trabalho* está associado ao processo de negócio mostrado na Figura 6.1. Este processo é utilizado atualmente pela operadora, que tem cerca de sessenta subsidiárias com mais de um milhão de clientes distribuídos em todo o país. Ele é responsável por gerar as cobranças mensais enviadas aos clientes.

O *workflow* possui cinco atividades executadas através de componentes *OSGi* responsáveis por passos específicos do processo de cobrança:

- *Ajustar Valores dos Planos de Saúde*: mensalmente, um conjunto de clientes tem seus planos de saúde ajustados em termos de valores a serem pagos de acordo com regras contratuais: renovação anual do plano, mudanças em idades dos clientes que implicam em novos valores a serem cobrados e assim por diante;

- *Gerar Valores Fixos*: Define os valores que devem ser pagos mensalmente pelos clientes independente se eles se submeteram a procedimentos médicos. Para se obter estes valores, planos e contratos de clientes devem ser analisados;

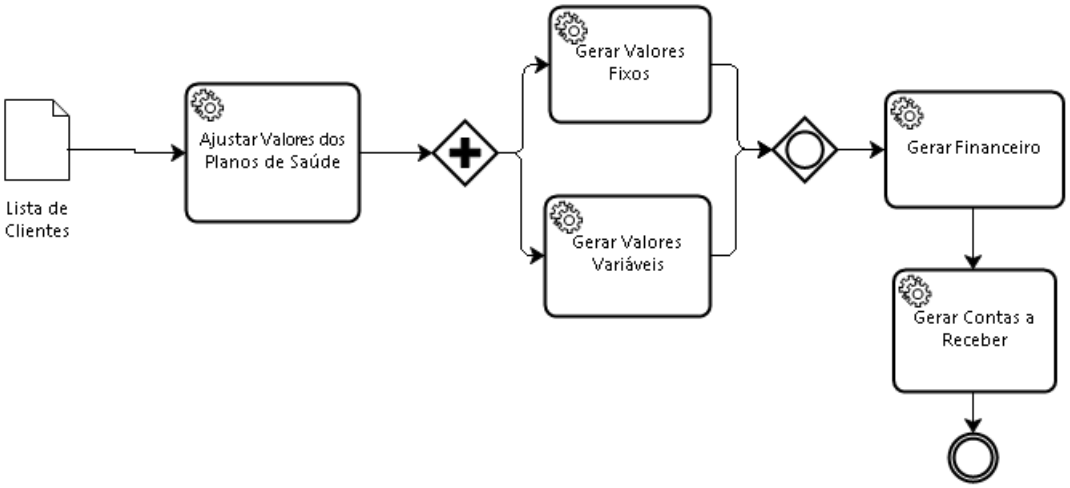
- *Gerar Valores Variáveis*: Define a quantia a ser paga pelos clientes de acordo com procedimentos médicos a que se submeteram. Trata-se de um valor variável porque depende de regras contratuais e da presença de atendimentos médicos realizados no período de cobrança que está sendo processado. Esta atividade normalmente consome uma quantidade de tempo razoável em virtude da quantidade de informações que devem ser analisadas para se chegar aos valores que devem ser cobrados. Esta é a atividade mais longa do *workflow*, consumindo, em média, de 50% a 60% do tempo total de processamento;

- *Gerar Financeiro*: agrega os valores fixos e variáveis em um conjunto reduzido de registros que é utilizado como base para dedução de impostos, ajustes manuais e outras manipulações de dados financeiros que podem ser realizadas por auditores de contas, por exemplo; e

- *Gerar Contas a Receber*: utiliza os resultados da geração dos dados financeiros para criar um conjunto ainda mais reduzido de registros que será aplicado na impressão de boletos, confecção de arquivos de títulos de cobrança para bancos ou envio de cobranças por meios eletrônicos como o *e-mail*.

Este *workflow* utilizado nesta avaliação experimental se enquadra no conceito de WLD proposto na Seção 4.1. Em virtude de sua duração, um reprocessamento completo do *workflow* é, geralmente, inaceitável. Dependendo do volume de dados sendo processados, muitas horas podem ser necessárias para que o processo seja concluído. Uma interrupção com o consequente reprocessamento pode implicar em atrasos na geração de cobranças, impossibilidade de geração de registros contábeis, adiamento de ações de auditoria, além de outros problemas operacionais

Figura 6.1: *Workflow* para Cobrança de Clientes



Fonte: O próprio autor.

que podem gerar danos colaterais tais como perdas financeiras com o pagamento de horas-extras a funcionários da organização.

Para a realização deste experimento, o ambiente de gerenciamento de execução de WLDs foi implantado em uma máquina com as características apresentadas na Tabela 6.2.

Tabela 6.2: Configurações de *Hardware* para Implantação do *LONGWisE4cloud*

Item	**Configuração**
Processador	*Intel Core i5* com 4 *núcleos*
Memória *RAM*	8Gb
Sistema Operacional	*Windows 7 Ultimate - 64 bits*

O SGBD que dá suporte aos serviços associados ao *workflow* é o *Oracle 11g*. *Stored procedures* escritas em *PL/SQL* são utilizadas nos processamentos associados às atividades.

Para cada combinação possível dos níveis e fatores mostrados na Tabela 6.1 foram realizadas trinta medidas do tempo de execução do *workflow* submetido a reconfigurações conforme sugerido por Cochran, Cochran e Cox (1992).

Quando uma instância do WLD está sendo executada, uma mudança na composição é forçada para simular uma situação que pode ocorrer em um cenário real: uma nova regra de negócio relacionada à geração de valores de cobrança foi apresentada para a organização e invalidou um dos antigos serviços responsável por processar estes registros.

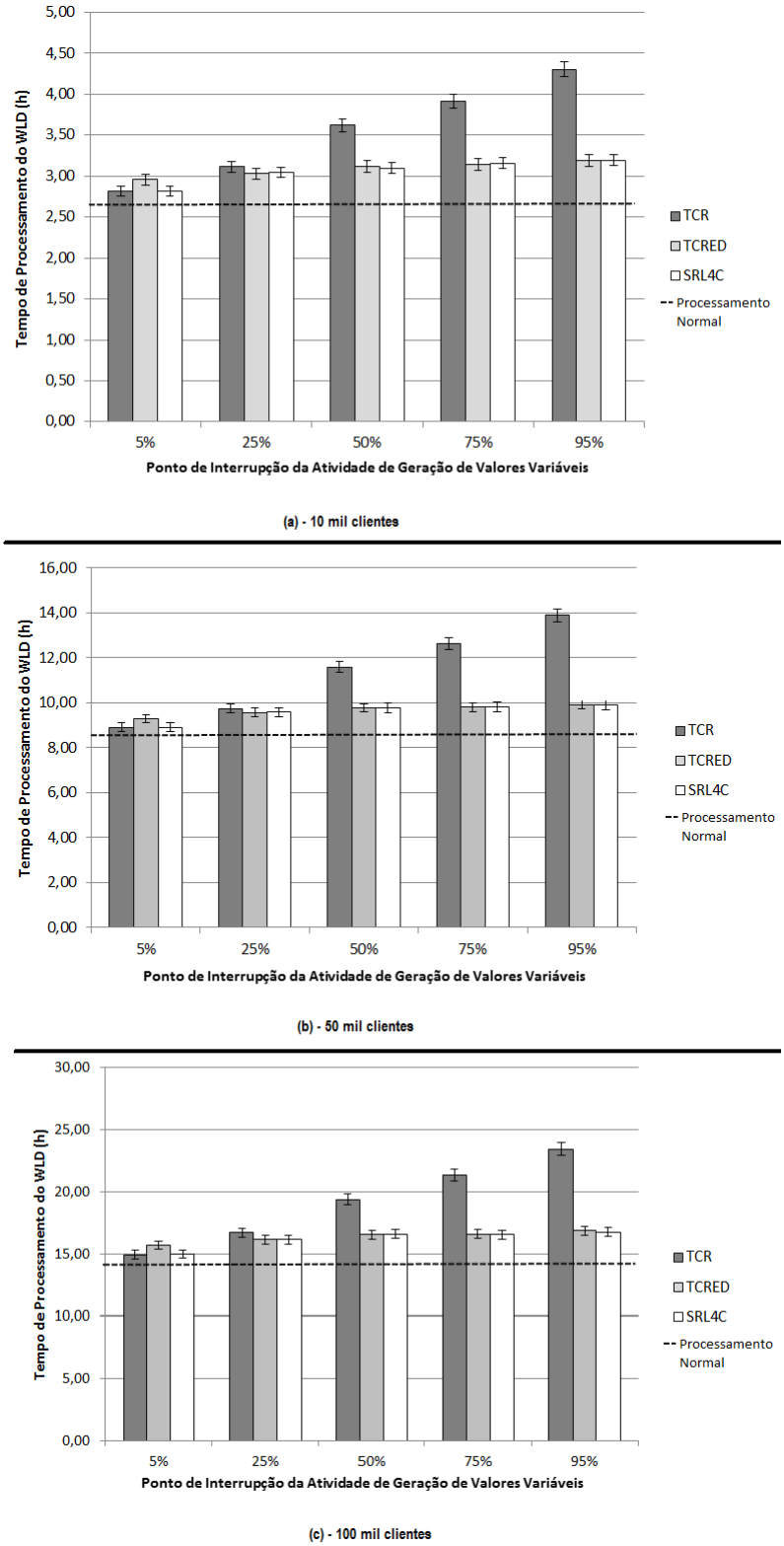
As mudanças no *workflow* foram sinalizadas ao ambiente de execução precisamente quando a atividade *Gerar Valores Variáveis* está sendo executada. A escolha da interrupção foi realizada em virtude do tempo de processamento da atividade que, conforme já discutido nesta seção, é o mais longo entre todas as que compõem o *workflow*.

O reuso de dados para este experimento é realizado através do uso rotinas que têm histórico de execução no ambiente, algo que favorece o cálculo da estimativa de tempo da sua

execução.

Os gráficos da Figura 6.2 apresentam os tempos de processamento do WLD obtidos ao executar o experimento com os fatores previamente definidos.

Figura 6.2: Tempos de Processamento na Reconfiguração Dinâmica



Fonte: O próprio autor.

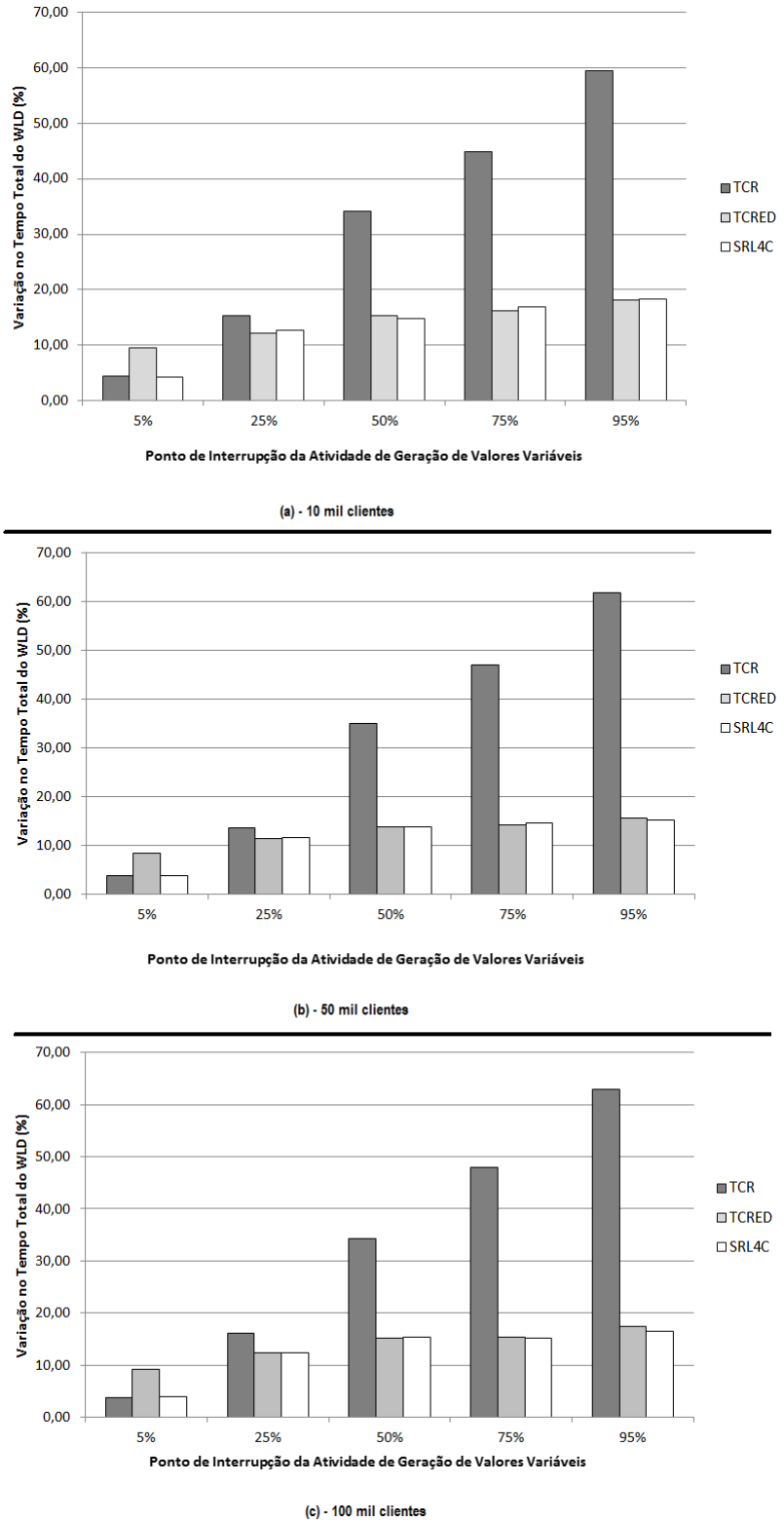
Os tempos de processamento do *workflow* sem interrupções também são apresentados. Há a separação dos valores para as subsidiárias com 10 mil (Figura 6.2 (a)), 50 mil (Figura 6.2 (b)) e 100 mil (Figura 6.2 (c)) clientes. Todos os tempos de processamento registrados nos gráficos são, na verdade, médias relativas às trinta medidas realizadas para cada uma das combinações de fatores, conforme explicado na descrição do experimento. O coeficiente de variação, razão entre o desvio padrão e a média das medidas, está representado na forma de pequenos traços dispostos sobre as colunas dos gráficos. Embora haja um valor específico deste coeficiente para cada tempo registrado nestes gráficos, os valores observados neste experimento ficaram entre 2,1% e 2,7%, o que implica em um conjunto consideravelmente homogêneo de amostras para o contexto dos *workflows* de longa duração.

Já os gráficos da Figura 6.3 mostram informações complementares derivadas da Figura 6.2 que são as *variações* impostas pelo uso das políticas de reconfiguração quando combinadas aos demais fatores do experimento. Estas variações são calculadas em termos percentuais e correspondem aos acréscimos de tempo relativos às execuções *normais* (sem interrupção) do WLD. Uma variação de 50% em um *workflow* que leva normalmente 10 horas para ser concluído implica que o *workflow* teve seu tempo de execução ampliado para 15 horas.

Quando se analisa a política baseada em trocas e reprocessamentos (TCR), observa-se que, quanto mais alto o percentual de completude da atividade interrompida, maiores os tempos de resposta envolvidos na execução do *workflow*. O número de instruções (implementadas nas atividades) a serem reprocessadas impacta diretamente neste acréscimo. O aumento no número de clientes de 10 mil para 50 mil e depois para 100 mil também incrementa o tempo de processamento. Isto pode ser constatado observando os números relacionados às colunas associadas à TCR na Figura 6.2. Por exemplo, para um mesmo ponto de interrupção (95%) da atividade de geração de valores variáveis, o tempo total de processamento do WLD com o uso da TCR passa de pouco mais de 4 horas quando 10 mil clientes estão envolvidos no processamento para mais de 23 horas quando 100 mil clientes são utilizados pelas atividades do *workflow*.

Ao se observar o comportamento dos tempos de processamento e as variações envolvendo o mecanismo baseado em reuso (TCRED), há uma mudança significativa quando é feita uma comparação com os resultados da política TCR. Inicialmente, para um mesmo número de clientes envolvidos no processamento, os diferentes momentos de interrupção da atividade de geração de valores variáveis não geram uma grande diferença no tempo de processamento total do *workflow*. Este fato pode ser visto nos valores apresentados nos gráficos da Figura 6.2, onde, para uma subsidiária com 100 mil clientes, por exemplo, a diferença entre os tempos totais de processamento, quando se compara a interrupção da atividade em 5% e 95%, é de apenas 1,17 horas. Esta mesma diferença chega a 8,51 horas quando a política TCR é aplicada. Quando se analisa os percentuais mostrados na Figura 6.3, vê-se que a maior da diferença entre os acréscimos no tempo de processamento, considerando um mesmo número de clientes e a interrupção em 5% e 95% (os extremos para os níveis deste fator), chega a 8,15% apenas. Como não há reprocessamento de atividades, esta diferença é causada basicamente pelo esforço

Figura 6.3: Variação nos Tempos de Processamento na Reconfiguração Dinâmica



Fonte: O próprio autor.

relacionado ao reuso de dados. Quanto mais longo for o tempo que a atividade interrompida ficou em execução antes de ser parada, há mais dados a serem aproveitados. Desta forma, o esforço para se aplicar este reuso gera uma diferença no tempo total de processamento, representada nos

gráficos da Figura 6.2 pelas valores das colunas que ficam acima da linha pontilhada referente ao processamento sem interrupção. Esta diferença possui valores distintos para cada combinação do número de clientes e ponto de interrupção da atividade de geração de valores variáveis. No entanto, ele é menor, na maioria dos casos, do que quando se aplica o reprocessamento total da atividade. Este fato pode ser constatado observando as diferenças nos valores apresentados nos gráficos para as duas estratégias de reconfiguração (TCR e TCRED). Ao observar os gráficos da Figura 6.3, pode-se identificar, por exemplo, que ao lidar com 100 mil clientes no processamento do WLD e se interromper a atividade em 95% de sua completude, a diferença em termos de acréscimo imposto pela política no tempo total de processamento chega a 45%. Ou seja, neste caso, a TCR impõe 45% a mais de tempo no processamento *normal* (sem interrupção) do *workflow* do que a TCRED.

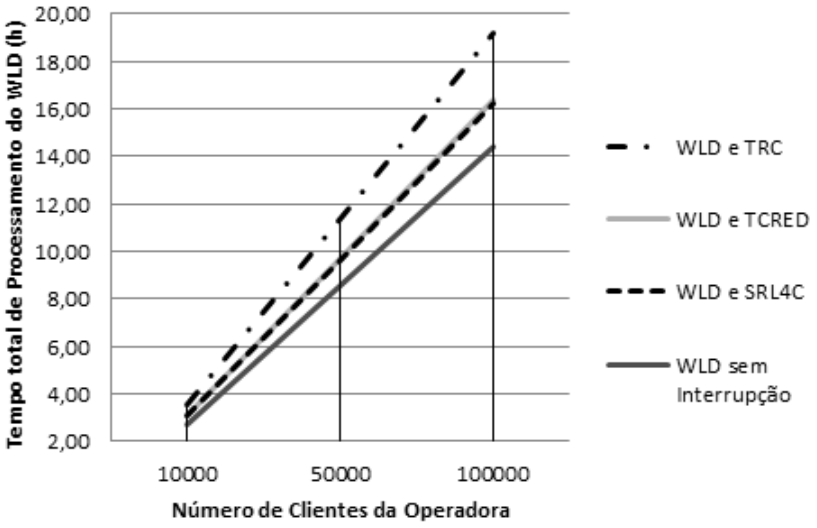
Observando-se os gráficos das Figuras 6.2 e 6.3, há um comportamento interessante quando a interrupção ocorre com 5% da atividade *gerar de valores variáveis* processada. Neste caso, independente do número de clientes envolvidos na execução do *workflow*, reprocessar a atividade do início é mais eficiente do que reutilizar os dados já gerados. Isto pode ser constatado verificando a diferença entre os valores dos tempos de execução referentes às diferentes estratégias de reconfiguração no ponto de interrupção mencionado. Por exemplo, no caso da combinação de fatores *10 mil clientes* e a interrupção da atividade em 5%, o acréscimo de tempo em relação ao processamento *normal* do *workflow* é de 4,35% quando se utiliza a TCR. Este acréscimo é de mais de 9,44% quando a TCRED é aplicada. Isso ocorre porque, no caso dos dados e rotinas utilizados no experimento, as instruções relacionadas ao reuso acabam exigindo um tempo para serem concluídas que, quando comparado ao tempo de se reiniciar a atividade, este último se mostra mais curto.

Em situações como esta, a política de reconfiguração SRL4C faz diferença na otimização dos tempos de processamento. Como se pode observar nos gráficos, na maioria das combinações dos fatores do experimento, os tempos de processamento (Figura 6.2) e variações (Figura 6.3) envolvendo o SRL4C são basicamente os mesmos dos relacionados ao TCRED. Isto ocorre porque a SRL4C aplica a estratégia de reuso de dados, o que já implica, conforme explicado anteriormente, em ganhos no que diz respeito a tempos de processamento. Entretanto, ao se observar estes tempos de processamento e variações quando a interrupção se dá aos 5%, é possível notar que os valores do SRL4C se tornam quase os mesmos do TCR. Isto acontece porque, neste cenário em particular, o reprocessamento como política de reconfiguração é utilizado. A capacidade do *LONGWisE4cloud* de planejar a reconfiguração com base em cenários, históricos de processamento de rotinas e parâmetros de processamento, torna o ambiente mais eficiente em termos de consumo de tempo quando situações similares ao que ocorreu neste experimento são observadas na execução do WLD. Basta ver, por exemplo, na Figura 6.3, que ao se lidar com 100 mil clientes, se a estratégia de reuso que é, a princípio, conforme já discutida, mais eficiente para a maioria das combinações de fatores deste experimento, fosse aplicada, haveria um acréscimo de quase 6% com relação ao tempo total de processamento do *workflow* sem interrupção. Com a

capacidade da política SRL4C de fazer escolhas para cada cenário, este acréscimo foi evitado. Embora este incremento em particular seja pequeno em termos absolutos, a capacidade da SRL4C de planejar a reconfiguração permite ganhos maiores como, por exemplo, quando o reuso é utilizado em detrimento ao reprocessamento para eliminar percentuais mais altos de incrementos de tempos de processamento, conforme já discutido. Considerando cenários onde diversas execuções de *workflows* e interrupções são observadas, estas diferenças acumuladas podem significar economia de tempo conforme será explicado mais adiante.

A Figura 6.4 apresenta uma visão consolidada relativa ao tempo de processamento do *workflow* nas sucessivas execuções realizadas ao longo desta avaliação experimental. Os valores no gráfico dizem respeito às médias de tempos gastos para a execução do WLD de acordo com todas as políticas de reconfiguração definidas para o experimento e combinando-as com as quantidades de clientes utilizados para gerar as cobranças associadas ao *workflow*. Entraram no cálculo destas médias os cinco pontos de interrupção da atividade *gerar valores variáveis* também definidos para o experimento. A informação sobre os tempos de execução do *workflow* sendo executado sem interrupção é apresentada para mostrar a diferença entre estes valores e os gerados quando a reconfiguração é realizada pelos diferentes mecanismos. A linha relativa ao reuso praticamente se confunde com o uso da política presente no *LONGWisE4cloud*. Isto ocorre porque, conforme discutido anteriormente, o reuso de valores é a estratégia mais adotada pelo ambiente ao longo do experimento, já que se mostra uma alternativa para se evitar o aumento de tempo imposto por reprocessamentos de instruções quando, por exemplo, uma interrupção ocorre no momento que uma atividade já estava quase finalizada. Mas, sempre é bom lembrar que, além de possuir a capacidade de reutilizar dados, o *LONGWisE4cloud* também é capaz de escolher o reprocessamento de uma atividade, quando este se mostrar mais eficiente para reduzir tempo de processamento ou for uma opção de quem está submetendo mudanças às instâncias ativas de um WLD.

Figura 6.4: Média de Tempos de Processamento em Reconfigurações



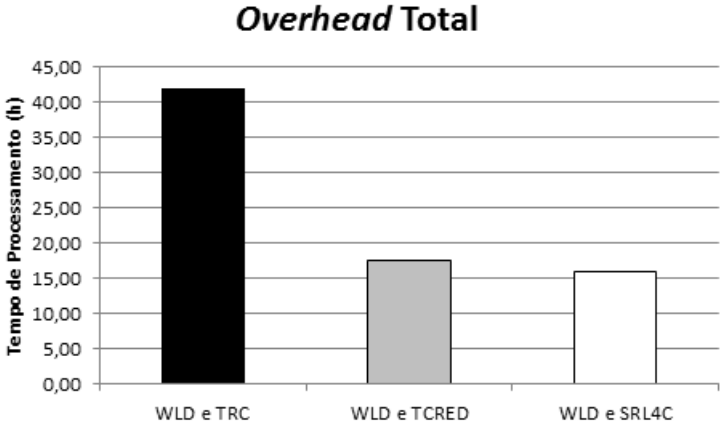
Fonte: O próprio autor.

A Figura 6.5 mostra o *overhead* total referente ao tempo de processamento do WLD do experimento causado pelas ações de reconfiguração. O cálculo deste *overhead* é feito para cada política de reconfiguração utilizada de acordo com a Fórmula 6.1. T_{N_NC} corresponde ao tempo normal (sem interrupção) de processamento do *workflow* para cada número de clientes (*NC*) diferente utilizado no experimento. $T_{R\_NC\_PI}$ diz respeito ao tempo de processamento do *workflow* para cada política de reconfiguração (*R*), número de clientes (*NC*) e ponto de interrupção (*PI*) utilizados. A diferença entre este dois tempos indica o *overhead* imposto pela reconfiguração. O somatório interno apresentado na Fórmula 6.1 calcula o acúmulo deste *overhead* para cada ponto de interrupção da atividade de gerar valores variáveis. O somatório externo deduz, finalmente, o *overhead* total considerando todas as subsidiárias com diferentes números de clientes.

$$\sum_{NC=10mil}^{100mil} \sum_{PI=5\%}^{95\%} (T_{N\_NC} - T_{R\_NC\_PI}) \tag{6.1}$$

É possível observar que, no caso deste experimento, o *overhead* causado pelo reprocessamento de instruções causa um acúmulo total quase três vezes superior aos outros dois mecanismos de reconfiguração. Mais uma vez os números apresentados pelo reuso e pela política de reconfiguração do *LONGWisE4cloud* estão bem próximos, com uma diferença de 8%. Esta similaridade já foi explicada anteriormente, mas vale ressaltar mais uma vez que, além do mecanismo de reuso ser uma vantagem presente no mecanismo do *LONGWisE4cloud*, o fato dele poder realizar um planejamento de ações de reconfiguração o torna mais eficiente na economia de tempos de processamento. No caso deste experimento, ao optar pelo reprocessamento em algumas situações já descritas anteriormente, o mesmo apresentou uma vantagem de cerca de 2 horas, em termos de *overhead* total, em relação ao mecanismo de reuso incondicional.

Figura 6.5: *Overhead* em Reconfigurações Dinâmicas



Fonte: O próprio autor.

Esta informação a respeito de *overhead* é um aspecto importante. Considerando que um *workflow* pode ser executado múltiplas vezes em um contexto corporativo, este acúmulo de horas

pode significar um aumento nos custos envolvidos com a execução de processos de negócio. Por exemplo, se a infraestrutura para executar o *workflow* é um ambiente de nuvem cuja cobrança é comumente associada ao tempo de uso de seus componentes virtualizados, utilizar menos horas significa pagar menos ao provedor deste ambiente.

Baseado nos resultados obtidos nesta avaliação experimental, é possível afirmar que o uso de um mecanismo de reconfiguração com os recursos apresentados pelo *LONGWisE4cloud* traz vantagens para o contexto da execução de WLDs. Se for observado o *overhead* total gerado pelas diferentes estratégias de reconfiguração aplicadas à execução do *workflow*, por exemplo, o ambiente reduziu em cerca de 60% os tempos de processamento acumulados nas execuções do *workflow* quando comparado à abordagem tradicional de reprocessamento de atividades. Estes tipos de ganhos obtidos relativos à eficiência em processamentos computacionais podem ser associados a benefícios aos negócios de uma organização, tais como agilidade ou redução de custos de processos corporativos.

6.2 Provisionamento de Recursos

O objetivo deste experimento é avaliar o impacto do mecanismo de provisionamento de recursos do *LONGWisE4cloud* sobre a execução de WLDs.

Para este experimento, trabalhou-se com o ambiente de gerenciamento de execução de WLDs com e sem o mecanismo de provisionamento de recursos ativo. O suporte ao provisionamento de recursos do *LONGWisE4cloud* se baseia nos *adaptadores* apresentados nas Seções 4.3.3 e 4.3.4. Eles são capazes de trocar serviços associados a um *workflow* para atender a requisitos específicos estabelecidos para ele, como, por exemplo, a necessidade de se trabalhar com serviços de segurança (e.g, criptografia de dados) ou ainda exigir que um determinado recurso, como um serviço de armazenamento de dados, seja utilizado de forma exclusiva pelas atividades que compõem o WLD.

A métrica adotada nesta avaliação também é o *tempo de processamento*, que consiste no intervalo de tempo transcorrido desde o início até a conclusão da execução do *workflow*. Conforme já estabelecido na Seção 6.1, esta é uma métrica natural em WLDs.

Da mesma forma, assim como discutido na Seção 6.1, alguns parâmetros são capazes de interferir nos valores da métrica escolhida. Entre eles podem ser citados a configuração de hardware que dá suporte à execução dos *workflows*, a disponibilidade de recursos de rede para comunicações remotas, sistemas operacionais e serviços de gerenciamento de banco de dados envolvidos no cenário de processamento dos *workflows*. Parâmetros adicionais, tais como a quantidade de dados processados pelo WLD, a existência de mecanismos que garantam os recursos necessários à execução das atividades dos WLDs e o próprio *workflow* sendo executado também foram considerados.

Observando o objetivo desta avaliação, três fatores foram escolhidos para este experimento específico: *mecanismo de provisionamento de recursos*, *serviço de banco de dados* e o

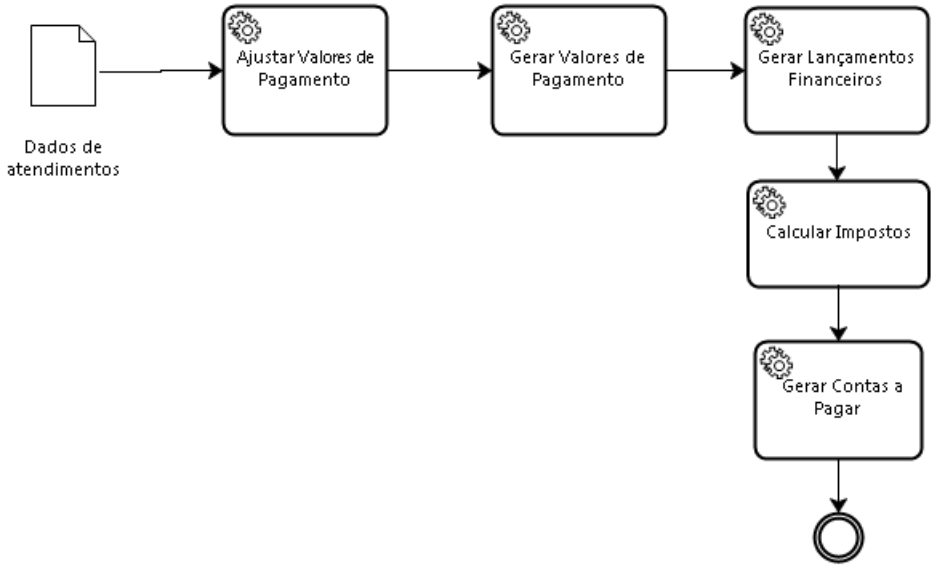
volume de dados processado pelo *workflow*. A Tabela 6.3 mostra estes fatores e seus respectivos *níveis*. O primeiro foi escolhido por ser o foco principal das análises do experimento e, conforme já mencionado, ele pode estar ausente ou presente. O serviço de gerenciamento de banco de dados pode ser *dedicado* ou *compartilhado*. No primeiro caso, atividades que solicitam o uso do serviço possuem exclusividade no uso de seus recursos como áreas de memória e processos de gerenciamento de dados. No segundo, há um compartilhamento destes elementos envolvendo vários usuários simultâneos do serviço. Finalmente, a quantidade de dados processados pelas atividades do WLD é, em geral, um aspecto crítico para a análise de seu desempenho, já que quanto mais dados forem gerados e processados, mais tempo normalmente é necessário para se concluir a execução do *workflow*. No caso deste experimento, o volume de dados está associado ao número de clientes associados a diferentes subsidiárias da operadora de planos de saúde.

Tabela 6.3: Fatores e Níveis - Avaliação do Provisionameto de Recursos

Fator	**Níveis**
Provisionamento de Recursos	*presente, ausente*
Serviço de Gerenciamento de BD	*dedicado, compartilhado*
Número de Clientes	*10.000*, *50.000*, *100.000*

A Figura 6.6 apresenta o processo de negócio associado ao WLD adotado como *carga de trabalho*. Assim como o processo da Seção 6.1, ele também é utilizado atualmente pela operadora. Este *workflow* em particular é responsável por gerar os pagamentos mensais encaminhados aos prestadores de serviços médicos, tais como médicos, clínicas e hospitais que atendem os beneficiários da operadora.

Figura 6.6: *Workflow* para Pagamento de Prestadores de Serviços Médicos



Fonte: O próprio autor.

O *workflow* possui cinco atividades executadas através de componentes *OSGi* responsáveis por passos específicos do processo de pagamento:

- *Ajustar Valores de Pagamento*: mensalmente, um conjunto de profissionais e prestadores de serviços médicos tem os valores de procedimentos associados ao seu pagamento ajustados de acordo com regras específicas: negociações individuais, mudanças em contratos ou até mesmo o estabelecimento de novas regulamentações governamentais que influenciem o cenário;

- *Gerar Valores de Pagamento*: define os valores que devem ser pagos aos prestadores de acordo com os procedimentos médicos realizados por eles. Para deduzir estes valores, todos os atendimentos devem ter seus detalhes analisados (cliente envolvido, a hora da execução, a região do país onde houve o atendimento, materiais e medicamentos envolvidos além de outras informações pertinentes a estes eventos). Neste passo, o valor de cada item referente a um atendimento gera um registro individual de maneira a avaliar ações tais como auditorias médicas;

- *Gerar Lançamentos Financeiros*: agrega os valores de pagamento em um conjunto reduzido de registros que serão utilizados como base para ajustes manuais, geração de relatórios gerenciais, auditorias e assim por diante. O agrupamento de valores, gerados de forma detalhada no passo anterior, se dá através de critérios diversos, tais como a natureza do registro (serviço médico, material, medicamento, taxas, diárias);

- *Calcular Impostos*: calcula os impostos que devem ser retidos pela operada de plano de saúde em virtude de legislações tributárias municipais, estaduais e federais. Estes valores deverão ser repassados aos governos responsáveis por cada tributo e são armazenados no mesmo repositório dos lançamentos financeiros; e

- *Gerar Contas a Pagar*: utiliza todos os registros financeiros para gerar um conjunto ainda mais reduzido de informações que são utilizadas no pagamento dos prestadores de serviços médicos. Estes valores podem ser utilizados na construção de documentos a serem enviados aos bancos para depósito dos valores nas contas dos prestadores.

O *workflow* utilizado nesta avaliação experimental também se enquadra no conceito de WLD proposto na Seção 4.1. Ele representa um processo de negócio real de uma operadora de planos de saúde. Além disso, o tempo que normalmente leva para ser concluído é um fator que acarreta uma maior probabilidade de ocorrência de mudanças em regras de negócio que atingem diretamente as instâncias ativas do *workflow*. Também em virtude de sua duração, um reprocessamento completo do *workflow* é, geralmente, algo inaceitável. Dependendo do volume de dados sendo processado, muitas horas podem ser necessárias para que o processo seja executado até o final. A interrupção com o consequente reprocessamento pode implicar em atrasos na geração de pagamentos, impossibilidade de geração de registros na contabilidade da empresa, adiamento de ações de auditoria, além de problemas operacionais já mencionados na Seção 6.1, como perdas financeiras com o pagamento de horas-extras a funcionários.

Quando o serviço de banco de dados não é dedicado, processos são executados no SGBD para que a concorrência pelos recursos do serviço de banco de dados aconteça. Algumas rotinas relacionadas com o dia a dia da operadora foram executadas em paralelo com o *workflow* do experimento:

- Geração de cobrança como apresentada na Seção 6.1;
- Processamento de solicitações de autorizações para que beneficiários possam se submeter a procedimentos médicos especializados; e
- Exportação de arquivos para órgãos reguladores, como dados para a Receita Federal relativo ao pagamento de prestadores.

A escolha destes processos ocorreu para simular situações reais que ocorrem na operadora. Enquanto os processamentos de pagamento são executados, diversas outras rotinas relativas a outros processos de negócio são disparadas por usuários que utilizam os sistemas de gestão de planos de saúde das operadoras. Além disso, as rotinas citadas compartilham dados utilizados no processo de pagamento, o que implica em acesso concorrente por dados compartilhados por toda a organização, algo capaz de interferir em tempos de processamento dos processos concorrentes.

Para a realização deste experimento, o ambiente de gerenciamento de execução de WLDs foi instalado em uma máquina com as mesmas configurações apresentadas na Tabela 6.2. No caso deste experimento, o SGBD que dá suporte às atividades do *workflow* também é um *Oracle 11g* onde *stored procedures* escritas em *PL/SQL* são utilizadas nos processamentos associados a estas atividades.

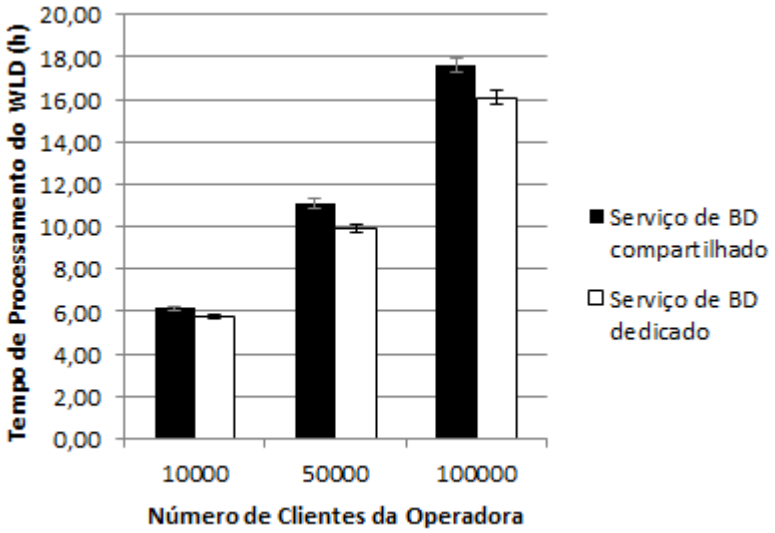
Assim como no experimento descrito na Seção 6.1, há, para cada combinação possível dos níveis e fatores mostrados na Tabela 6.3, a realização de trinta medidas do tempo de execução do *workflow* submetido a execuções sucessivas.

O *workflow* chega ao ambiente com a sinalização, através de seus parâmetros de execução, de que as atividades devem ser executadas usando-se um SGDB dedicado. Para simplificar o experimento e eliminar resultados irrelevantes para a avaliação, algumas combinações de fatores foram suprimidas nas execuções do *workflow*. Por exemplo, a combinação entre o mecanismo de provisionamento de recursos *presente* e o serviço *compartilhado* de banco de dados assim como o mecanismo *ausente* e o serviço *dedicado* de banco de dados. Quando um *workflow* demanda um serviço *dedicado* e o mecanismo de provisionamento está presente, então o SGBD dedicado é encontrado e utilizado. Além disso, quando este mecanismo se encontra *ausente*, um serviço *compartilhado* é aplicado na execução do *workflow*. No primeiro caso, se um serviço dedicado de banco de dados não fosse encontrado por inexistir ou não estar registrado junto ao ambiente de gerenciamento da execução de WLDs, seria um problema associado a algo que vai além do ambiente propriamente dito, constituindo um cenário que não serviria para avaliar seus recursos no contexto deste experimento. No segundo caso, se um serviço dedicado de banco de dados foi utilizado como suporte às atividades de um *workflow* mesmo sem o ambiente possuir um

mecanismo que direcione a escolha para este serviço, também seria um cenário que não serviria para avaliar recursos do ambiente, pois não se tratou de algo direcionado por ele, mas apenas uma situação casual ou fora do escopo de atuação do mesmo.

A Figura 6.7 apresenta os tempos de processamentos do WLD para as diferentes combinações dos níveis referentes aos fatores definidos. Observa-se, nos resultados, que há um ganho de desempenho quando se usa a estratégia de se dedicar um serviço de gerenciamento de banco de dados para o processamento dos pagamentos que, em termos percentuais, chega a 10,9% (para os processamentos envolvendo 50 mil clientes). Este ganho também é incrementado, em termos de tempo de processamento, à medida que o número de clientes da base da operadora aumenta. Este incremento ocorre porque, com mais clientes, há um aumento do número de atendimentos analisados para o pagamento de prestadores.

Figura 6.7: Resultados do Experimento Relativo ao Provisionamento de Recursos



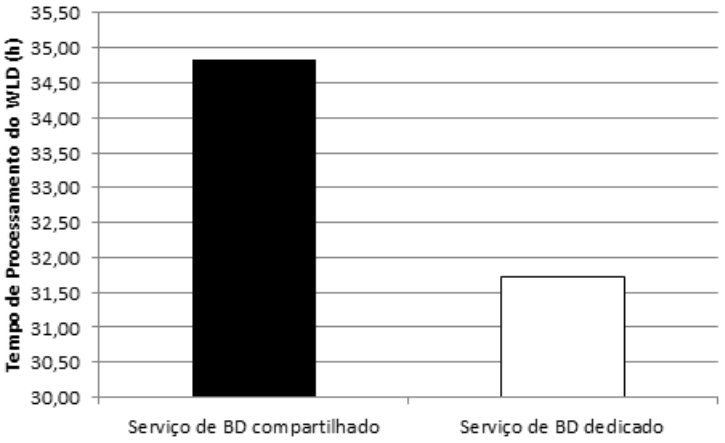
Fonte: O próprio autor.

Assim como no experimento da Seção 6.1, os tempos de processamento registrados no gráfico da Figura 6.7 são médias relativas a trinta medidas realizadas para cada uma das combinações de fatores. Os coeficientes de variação observados para cada uma destas medidas lançadas no gráfico ficaram entre 1,75% e 1,91%, o que também implica em um conjunto consideravelmente homogêneo de amostras para o contexto dos *workflows* de longa duração.

A Figura 6.8 apresenta as somas de todos os tempos realizados pelas execuções do *workflow* de pagamento de prestadores para as combinações de fatores definidas neste experimento. Quando se calcula a diferença total de tempos de processamento considerando o uso compartilhado e dedicado do serviço de gerenciamento de banco de dados, encontra-se, como pode ser visto no gráfico da Figura 6.8, um valor de 3,12 horas.

Embora, em termos absolutos, este *overhead* possa parecer um valor relativamente baixo, também deve ser levado em consideração o contexto nos quais estes WLDs são utilizados. A médio e longo prazos, diversos processamentos podem ser executados dentro das subsidiá-

Figura 6.8: *Overhead* em Provisionamento de Recursos



Fonte: O próprio autor.

rias associadas à operadora cujos dados foram aplicados nos processamentos do experimento. Conforme mencionado na Seção 6.1, o uso de uma infraestrutura que dê suporte a estes processamentos, como uma nuvem computacional, pode ter um custo diretamente associado ao tempo de utilização de seus recursos de hardware e software. Reduzir estes custos ao longo de múltiplos processamentos de *workflows* pode significar um ganho financeiro para uma organização. Além disso, horas de processamento podem significar o adiamento da conclusão de uma rotina importante, como o pagamento de prestadores, que, por sua vez, pode resultar em impactos negativos para a operadora, como a insatisfação por parte destes prestadores caso estas situações se tornem recorrentes.

Sendo assim, a presença de uma estratégia de provisionamento de recursos computacionais é importante para o contexto dos *workflows* de longa duração.

6.3 Escalabilidade

O objetivo deste terceiro experimento é avaliar o impacto do mecanismo de distribuição do *LONGWisE4cloud* na execução dos WLDs.

Para isto, configurou-se o ambiente com o gerenciamento da execução dos WLDs centralizado em um único componente de processamento e também com a distribuição deste gerenciamento em componentes implantados em máquinas virtuais distintas conforme a arquitetura proposta para o *LONGWisE4cloud* e cujos detalhes foram apresentados na Seção 4.3.

A métrica adotada nesta avaliação é o *tempo de processamento*, que aqui também consiste no intervalo de tempo transcorrido desde o início até a conclusão da execução dos *workflows* de longa duração submetido ao ambiente de gerenciamento.

Assim como também já foi discutido nos experimentos anteriores, alguns parâmetros podem interferir nos valores desta métrica escolhida. Recursos computacionais que dão suporte à execução dos *workflows* tais como memória e processadores ou máquinas virtuais, a disponibilidade de recursos de rede para a realização de comunicações remotas, softwares básicos como

sistemas operacionais e serviços de gerenciamento de banco de dados envolvidos no cenário de processamento dos *workflows*. A quantidade de dados processados pelo WLD, o número de núcleos de processamentos envolvidos no gerenciamento dos *workflows* submetidos ao ambiente e os próprios *workflows* sendo executados também foram considerados.

Tendo em vista o objetivo do experimento, dois fatores foram escolhidos: *número de máquinas virtuais* e o *volume de dados* processado pelo *workflow*. Cada máquina tem uma *ECI* (apresentada nas Seções 4.3.3 e 4.3.4) implantada capaz de gerenciar WLDs. Quando há apenas uma máquina virtual, simula-se o gerenciamento *monolítico*, *stand alone* ou centralizado em um único componente de gerenciamento. Assim como nos experimentos anteriores, a quantidade de dados processados pelas atividades dos WLDs é, em geral, um aspecto crítico para a análise de seu desempenho.

A Tabela 6.4 mostra os fatores selecionados para o experimento com seus respectivos *níveis*. Para o número de máquinas virtuais, as quantidades escolhidas variam de apenas uma (processamento *stand alone*) a três. Considerando a infraestrutura de hardware que irá dar suporte à execução do WLD e os volumes de dados e instruções processadas, estes níveis já são capazes de permitir a observação de impactos nos valores obtidos para a métrica escolhida. O volume de dados está associado às diferentes categorias (também já explicado nos experimentos anteriores) das subsidiárias da operadora.

Tabela 6.4: Fatores e Níveis - Avaliação da Escalabilidade

Fator	**Níveis**
Número de Máquinas Virtuais	*1VM, 2VM, 3VM*
Número de Clientes	*10.000*, *50.000*, *100.000*

Dois WLDs relativos aos processos apresentados nas Figuras 6.1 e 6.6 são adotados como *carga de trabalho*. Ambos se enquadram, também como já explicado anteriormente, no conceito de WLD proposto na Seção 4.1. O uso dos dois simultaneamente simula uma situação comum na operadora, onde cobranças e pagamentos são comumente processados em paralelo.

Para a realização deste experimento, as máquinas virtuais são instanciadas em um servidor com as configurações apresentadas na Tabela 6.5.

Tabela 6.5: Configurações de *Hardware* do Ambiente de Virtualização

Item	**Configuração**
Processador	*Intel Xeon* com 16 *núcleos*
Memória *RAM*	64Gb
Sistema Operacional (*hipervisor*)	*ESXi*

Cada máquina virtual possui as configurações equivalentes às apresentadas na Tabela 6.2. O SGBD que dá suporte às atividades dos *workflows* é um *Oracle 11g*. Cada uma das instâncias dos *workflows* sendo executadas utiliza, para suporte as suas atividades, um servidor de banco de

dados dedicado implantado em máquinas à parte com as configurações apresentadas na Tabela 6.2.

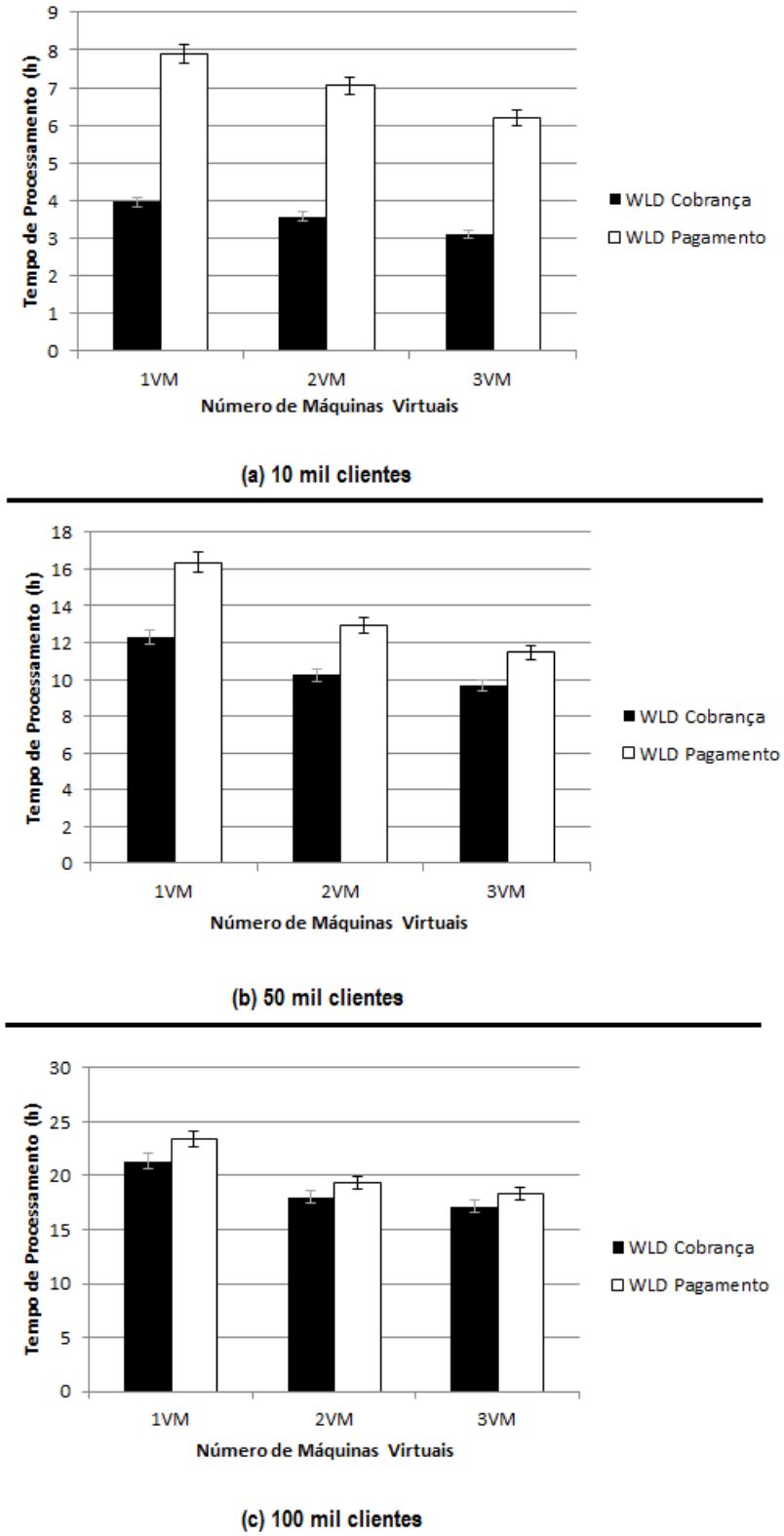
Como sempre ocorre nos experimentos apresentados nesta tese, para cada combinação possível dos níveis e fatores mostrados na Tabela 6.4, ocorre a realização de trinta medidas do tempo de execução dos *workflows* submetidos a execuções simultâneas sucessivas. As médias destas medidas, juntamente com o coeficiente de variação, são apresentadas nos resultados do experimento.

Para cada *workflow* utilizado nesta avaliação, dez instâncias são executadas simultaneamente. Com isso, o ambiente de gerenciamento da execução destes WLDs terá 20 instâncias ativas sendo monitoradas e controladas por ele. De forma a provocar um aumento do consumo de recursos computacionais por parte dos componentes que constituem o ambiente de gerenciamento, reconfigurações dinâmicas são acionadas para todas estas instâncias ativas. O evento que irá iniciar estas adaptações é algo similar ao que ocorre no experimento da Seção 6.1: a atividade mais longa de cada *workflow* é interrompida quando está próxima de sua conclusão (95% de processamento já realizado). No caso do *workflow* de cobrança, esta atividade é a *Gerar Valores Variáveis*, responsável por mais da metade do tempo de processamento total do WLD. No caso do *workflow* de pagamento, a atividade mais crítica, em termos de consumo de tempo, é a *Gerar Valores de Pagamento*, onde também se observa um consumo de tempo que chega a mais de 50% do tempo total de execução do *workflow*. Novas versões destes WLDs chegam ao ambiente fazendo o mecanismo de adaptação ser acionado conforme explicado na Seção 4.3.4. Rotinas relacionadas ao reuso de dados já gerados pelos *workflows* estão presentes no ambiente com histórico de processamento registrado, algo análogo ao que ocorreu no experimento da Seção 6.1. Com este cenário configurado para o processamento dos WLDs, o reuso será a ação a ser escolhida pelo mecanismo de reconfiguração, conforme já observado no experimento da Seção 6.1. Sendo assim, 20 instâncias em paralelo sendo reconfiguradas implica em uma carga suficiente para causar impacto nos tempos de processamentos destes WLDs quando estes processamentos são impostos à infraestrutura de suporte colocada à disposição do ambiente de gerenciamento de execução de *workflows* .

Na Figura 6.9 são apresentados os tempos de processamento observados para as combinações dos diferentes níveis dos fatores. Os gráficos para os processamentos em bases de dados com 10 mil (Figura 6.9 - (a)), 50 mil (Figura 6.9 - (b)) e 100 mil (Figura 6.9 - (c)) clientes estão individualizados para uma melhor análise dos valores de tempo registrados. Como há dez instâncias simultâneas para cada *workflow* utilizado no experimento, os valores dizem respeito à media do tempo de processamento destas instâncias.

O *workflow* de pagamento possui um tempo de execução mais elevado em virtude das regras de negócio envolvidas ao longo das suas atividades. Além de dados de clientes registrados nos atendimentos, muitos dados cadastrais referentes aos prestadores de serviços também são avaliados para se realizar valorações de itens a serem pagos, tais como consultas, materiais médicos ou taxas hospitalares, ou ainda a suspensão de pagamentos de atendimentos

Figura 6.9: Resultados do Experimento Relativo à Escalabilidade



Fonte: O próprio autor.

por irregularidades como a realização de um procedimento fora da área de especialização de um profissional. Quanto maior a base de clientes envolvidos nos processamentos, maiores também os

tempos de processamento observados. Para confirmar isto, basta se observar, por exemplo, que o processamento da cobrança para 10 mil clientes e com o uso de apenas uma máquina virtual chega próximo a 4 horas para concluir sua execução. Este mesmo processamento, quando realizado por uma máquina virtual, mas com 100 mil clientes envolvidos, chega a ultrapassar as 21 horas para ser concluído. Este aumento também já foi explicado em experimentos como o da Seção 6.1, onde a existência de mais clientes normalmente implica em mais dados de atendimentos a serem processados para a geração de valores. As ações de reuso de dados impostas pelas reconfigurações dinâmicas também elevam estes tempos. A comprovação desta afirmação pode ser feita ao se comparar os tempos registrados nestes gráficos com aqueles observados nos experimentos das Seções 6.1 e 6.2, onde os tempos *normais* (sem interrupção) destes *workflows* são apresentados. Esta comparação é razoável, pois os SGBDs que dão suporte às atividades dos *workflows* através da execução de *stored procedures* estão em infraestruturas praticamente idênticas em todos os experimentos. O tempo *normal* (sem interrupções) de processamento para a geração da cobrança em uma base de 10 mil clientes, de acordo com os resultados apresentados na Seção 6.1 é de 2,7 horas. Neste experimento relativo à escalabilidade, o processamento chega a quase 4 horas.

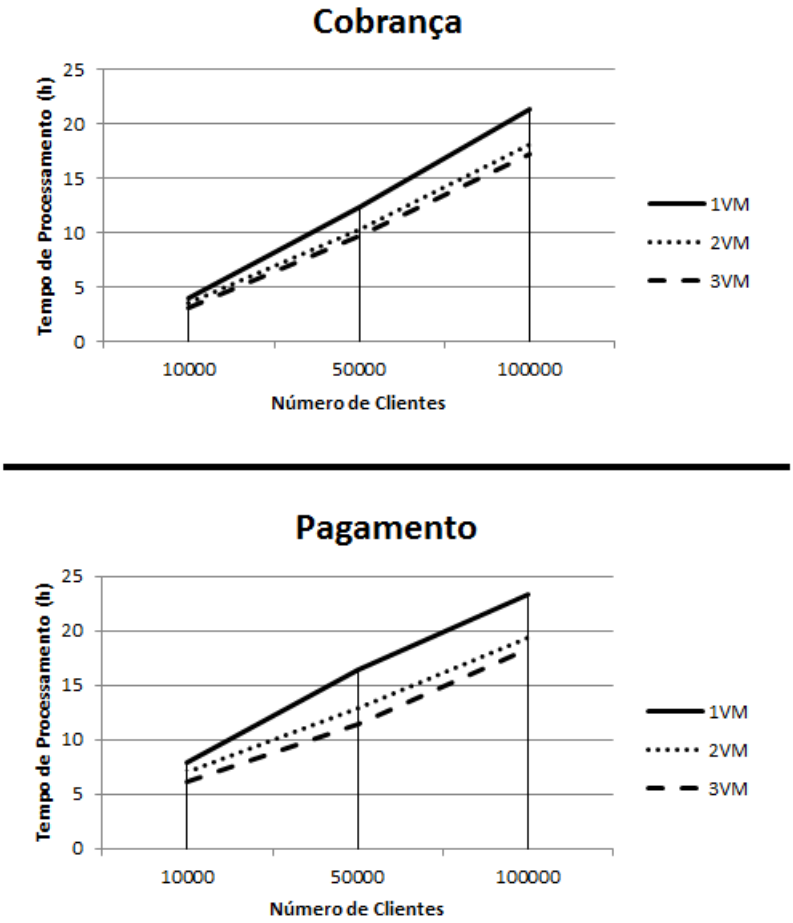
A relevância de se distribuir o gerenciamento da execução dos WLDs fica evidente quando se observa os tempos de processamento à medida que o número de máquinas virtuais que dão suporte ao ambiente de gerenciamento aumenta. Como as ações de reconfiguração impõem aos componentes do ambiente um trabalho significativo, o aumento de recursos computacionais à disposição para se realizar estas adaptações se reflete em redução nos tempos de execução dos *workflows*. Ao se analisar, por exemplo, os tempos de processamento para se realizar os pagamentos em uma base com 100 mil clientes, observa-se que há uma redução de cerca de 23 horas para pouco mais de 18 horas quando o número de máquinas virtuais aumenta de um para três, o que significa um ganho de mais de 21% em termos de economia de tempo de processamento.

Conforme já mencionado nos experimentos anteriores, os tempos de processamento registrados nos gráficos da Figura 6.9 são médias calculadas sobre as trinta medidas realizadas para cada uma das combinações de fatores. O coeficiente de variação observado nestas medidas ficou entre 3,2% e 3,7%. Ou seja, mais uma vez se trata de um conjunto consideravelmente homogêneo de amostras para o contexto dos WLDs. Os pequenos traços dispostos sobre as colunas representam esta variação observada para as medidas realizadas.

Os gráficos da Figura 6.10 apresentam os tempos totais consumidos para cada *workflow* utilizado no experimento.

As atividades voltadas ao pagamento, conforme já mencionado, consomem um tempo maior de processamento. Entretanto, para ambos os WLDs, é possível ver a diferença de tempo quando se utiliza mais de uma máquina virtual para gerenciar a execução destes *workflows*. Esta diferença é sempre observada para todas as categorias de operadoras utilizadas. Por exemplo, ao se considerar a cobrança para 10 mil clientes, observa-se que o tempo de processamento é

Figura 6.10: Resultados da Escalabilidade por *Workflow*



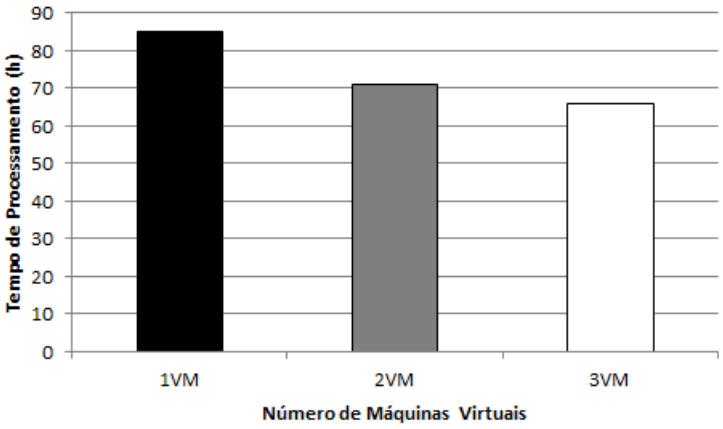
Fonte: O próprio autor.

reduzido em mais de 20% quando se aumenta o número de máquinas virtuais de um para três. No caso do pagamento para uma subsidiária com 50 mil clientes, esta mesma variação no número de máquinas virtuais gera, conforme já mencionado anteriormente, uma economia de cerca de 30% em termos de tempos de processamento. Finalmente, ao se observar a subsidiária com 100 mil clientes, a economia dos tempos de processamento para a geração da cobrança e do pagamento é de 19% e 21%, respectivamente.

Finalmente, na Figura 6.11 é possível observar os tempos totais consumidos pelos processamentos dos *workflows* submetidos ao ambiente de gerenciamento de execução. As colunas representam a soma dos tempos de processamento dos WLDs relativos à cobrança e ao pagamento apresentados nos gráficos anteriores de forma separada. Observa-se que o uso de apenas uma máquina virtual impõe um consumo de tempo consideravelmente mais elevado do que quando o gerenciamento do processamento é distribuído por duas ou três máquinas virtuais. A diferença chega a 23% quando se compara o uso de uma e três máquinas virtuais.

Quando se observam gráficos como o da Figura 6.11, é possível afirmar ainda que, em termos de valores absolutos, há uma diferença de quase 20 horas de processamento quando se

Figura 6.11: Resultados da Escalabilidade por *Workflow*



Fonte: O próprio autor.

compara o uso de uma estratégia *stand alone* para a aplicação de três máquinas virtuais como suporte à execução. Entretanto, considerando-se que se trata de uma média de tempo calculada com base em 20 instâncias em paralelo, estes números podem ser multiplicados por um fator próximo a 20 em termos de valores reais de uso de recursos de hardware e software para suporte a estas execuções. Assim, esta diferença de quase 20 horas se torna uma economia de tempo ainda mais relevante para o cenário observado no experimento.

Diante dos resultados obtidos, é possível afirmar que a distribuição do gerenciamento da execução dos *workflows* de longa duração por diversas máquinas virtuais é um fator relevante para ganhos de desempenho na execução destes *workflows*. A solução proposta pelo *LONGWisE4cloud* para resolver esta distribuição, com o uso das *ECIs*, constitui um mecanismo que atende a esta distribuição, tornado-o, assim, um importante elemento para fins de melhoria de desempenho na execução de WLDs. O *LONGWisE4cloud* é capaz de usufruir dos benefícios das infraestruturas que oferecem a virtualização de recursos, como as nuvens computacionais, onde elementos como máquinas virtuais são colocados à disposição, alocados ou desalocados de acordo com as demandas de processamento associadas aos usuários da infraestrutura.

6.4 Considerações Finais

Neste capítulo, foram apresentados experimentos para avaliar os mecanismos existentes no *LONGWisE4cloud* para dar suporte à reconfiguração dinâmica, ao provisionamento de recursos e à escalabilidade na execução de *workflows* de longa duração. Os WLDs utilizados são relacionados a processos de negócio de uma operadora de planos de saúde com diversas subsidiárias e com abrangência nacional. Na execução dos experimentos foram consideradas subsidiárias com diferentes números de clientes, de forma que os impactos do uso do *LONGWisE4cloud* no desempenho de WLDs em subsidiárias de diferentes categorias pudessem ser analisados. Nos três experimentos realizados foi possível identificar ganhos, em termos de desempenho, na execução de *workflows* de longa duração quando submetidos ao *LONGWisE4cloud*.

Estes ganhos apresentados nos experimentos são importantes sinalizações de que as adaptações e a escalabilidade proporcionadas pelo *LONGWisE4cloud* são capazes de reduzir tempos de processamentos dos WLDs. No contexto destes *workflows*, esta economia de tempo pode ter impactos significativos em uma organização que utiliza o *LONGWisE4cloud* na execução de seus WLDs. Reduzir este tempo significa diminuir despesas. Além do aspecto financeiro, acelerar a conclusão de um *workflow* é capaz de trazer benefícios operacionais diversos para uma organização. Funcionários podem ter suas atividades aceleradas, processos envolvendo clientes podem ser acelerados.

7

CONCLUSÕES E TRABALHOS FUTUROS

Esta tese de doutorado apresentou o *LONGWisE4cloud*, um ambiente especialmente projetado e implementado para gerenciar a execução dos *workflows* de longa duração. Inicialmente, o capítulo sumariza as contribuições de pesquisa do trabalho. Em seguida, são apresentadas as limitações da proposta. Finalmente, o capítulo é concluído com a identificação de trabalhos futuros para dar continuidade ao ambiente proposto.

7.1 Contribuições

- *Um ambiente adaptativo para execução de WLDs como serviço*

A principal contribuição desta tese é o *LONGWisE4cloud*, um ambiente de suporte à execução de WLDs que atende às características particulares destes tipos especiais de *workflows*, advindas principalmente da longa duração. Como exemplo destas particularidades podem ser citadas: a necessidade de mecanismos que atendam a mudanças em regras de negócio durante a execução destes *workflows*, suporte adaptativo, escalabilidade, robustez, provisionamento de recursos computacionais, gerenciamento de dados e estados, além de mecanismos que lidem com a heterogeneidade requerida nos contextos de uso dos WLDs. Conceitos relacionados à computação orientada a serviços também foram utilizados no desenvolvimento do ambiente não apenas para satisfazer aos aspectos mencionados, mas para integrá-los em uma solução disponibilizada como serviço (*as a service*).

Sendo assim, o *LONGWisE4cloud*pode ser utilizado como ambiente de referência para o desenvolvimento de novas soluções associadas ao contexto destes tipos particulares de *workflows*, já que se trata de uma proposta que integra elementos essenciais para contemplar aspectos que precisam ser atendidos quando eles são submetidos à execução.

- *Arquitetura para gerenciar a execução de WLDs*

A arquitetura orientada a serviço favorece aspectos como a robustez, heterogeneidade e a manutenibilidade. A substituição de um serviço defeituoso, por exemplo, pode ser feita de maneira simples, dadas as características de modularidade e fraco acoplamento propiciados pela *SOC*. Além disso, uma vez que a comunicação entre os serviços deve ser independente de

linguagens de programação utilizadas para implementá-los ou infraestrutura tecnológicas onde estão implantados, a possibilidade de uso de múltiplas tecnologias, como serviços de nuvem distintos, também se torna uma característica da arquitetura final. O próprio aperfeiçoamento da solução, com a inclusão de novas funcionalidades para atender a novos requisitos, pode ser feita através de uma estratégia baseada na simples incorporação de novos serviços à arquitetura já existente.

O uso dos princípios de *SOC* e a distribuição do processamento dos WLDs provida pelo ambiente também permitiu que o *LONGWisE4cloud* fosse construído na forma de uma solução como um serviço, ou seja, capaz de acomodar as demandas pela execução destes *workflows* através do uso racional de recursos de infraestrutura (e.g., máquinas virtuais) colocados a sua disposição na infraestrutura onde ele se encontra implantado.

Finalmente, a arquitetura proposta para o *LONGWisE4cloud* é genérica o suficiente para ser implementada através do uso de diversas tecnologias, tais como Java (existente atualmente) ou .NET. A definição estrutural dos componentes, o estabelecimento dos serviços por eles providos e a interação entre os mesmos, observada a partir dos aspectos dinâmicos da arquitetura, foram realizados sem o direcionamento a nenhuma linguagem ou plataforma de desenvolvimento específica.

- *Definição de um conjunto de requisitos para a execução de WLDs*

A elaboração da solução proposta nesta tese permitiu a criação de um conjunto de requisitos importantes que devem ser atendidos por ambientes responsáveis pela execução de *workflows* de longa duração. Este conjunto foi definido a partir do estudo do domínio dos WLDs, realizado com o levantamento de trabalhos e soluções existentes associado a este contexto.

A adaptação, a heterogeneidade, a robustez, o uso de princípios de *SOC*, o gerenciamento de dados e estados ao lado dos princípios da quiescência em reconfigurações dinâmicas, a segurança, o provisionamento de recursos computacionais e a escalabilidade são elementos importantes deste conjunto. Além disso, a integração das soluções que atendem a estes requisitos também é um aspecto importante a ser coberto. Através desta integração é possível obter um aumento da eficiência destas soluções quando estiverem executando suas funções.

- *Adaptação de aplicações orientadas a serviço com quiescência, gerenciamento de estados, dados e análise de cenários*

O desenvolvimento do *LONGWisE4cloud* permitiu o estabelecimento de um mecanismo adaptativo que pode ser usado em cenários mais complexos, como, por exemplo, as particularidades na execução de *workflows* de longa duração. Estas adaptações muitas vezes demandam um planejamento com base em detalhes da execução dos WLDs observados no instante que ela será realizada.

O primeiro aspecto relevante deste mecanismo adaptativo é que ele age sobre composições de serviços. Além disso, quando se trata de realizar modificações em fluxos de execução, é

necessário observar o cenário em que a execução se encontra. Este cenário é definido a partir da análise dos serviços que estão efetivamente ativos, quais já foram concluídos e quais ainda serão processados. Com base nisso é possível que o mecanismo adaptativo defina quais as possíveis mudanças que devem ser realizadas. Neste contexto, a quiescência é um conceito que ajuda a estabelecer um estado de *tranquilidade* para a realização das mudanças estabelecidas pela adaptação, garantindo que a execução de instruções do WLD não interfira nas ações relativas a estas mudanças, evitando, assim, problemas como inconsistências em informações geradas. Finalmente, o gerenciamento de dados e estados permitem que trocas de serviços possam ser acompanhadas de ações como o reuso de dados já gerados e a continuidade de um processamento do ponto onde havia sido interrompido. Desta forma, as adaptações ocorrem de maneira mais confiável no que diz respeito a evitar problemas tais como inconsistências nos dados gerados pelos WLDs quando submetidos a estas mudanças.

- *Caracterização dos workflows de longa duração*

Durante o esforço de desenvolvimento do *LONGWisE4cloud*, estabeleceu-se uma definição aplicável aos *workflows* de longa duração. Esta caracterização é importante não apenas para que aspectos, particularidades e requisitos para a execução destas abstrações sejam definidos, mas também para que, ao se observar um *workflow* qualquer, o mesmo possa ser classificado como sendo ou não de longa duração e, assim, saber se ele será capaz de se beneficiar de soluções voltadas especificamente a estes tipos particulares de *workflows*.

- *Distribuição do ambiente de gerenciamento*

Os componentes do núcleo do *LONGWisE4cloud* diretamente associados ao gerenciamento da execução dos *workflows* podem ser instanciados e implantados em infraestruturas virtualizadas de forma a acomodar necessidades por recursos, tais como elementos de hardware ou software, em uma estratégia *as a Service*.

Sendo assim, o *LONGWisE4cloud*, através de sua arquitetura orientada a serviços, permite que rotinas remotas se responsabilizem pela execução das atividades dos *workflows*. Mas, além disso, ele é capaz de acomodar a demanda por processamentos de WLDs através da distribuição dos próprios componentes que fazem parte do ambiente. Desta forma, estes processamentos podem, por exemplo, ser implantados em diferentes máquinas virtuais, estruturas comuns em ambientes de nuvem, o que traz, como até foi sinalizado em avaliações realizadas para este tese, ganhos de desempenho para o contexto da execução dos WLDs.

7.2 Limitações

- *Ausência de mecanismos de segurança*

De acordo com o que foi apresentado nesta tese, a segurança é um aspecto importante no contexto dos WLDs, já que a longa duração os deixa expostos por mais tempo a ações inconvenientes, tais como ataques visando o acesso a informações sigilosas. No *LONGWisE4cloud*, a segurança é suportada a partir da possibilidade de se escolher serviços, tanto para o ambiente como para as atividades dos *workflows*, que atendam a requisitos de segurança, tais como criptografia de dados. Não há, na proposta, o uso explícito de padrões, arquiteturas, protocolos ou *frameworks* comumente utilizados atualmente na construção de softwares seguros. Apesar disso, os princípios de *SOC* aplicados no desenvolvimento do ambiente facilitam a integração destes aspectos de segurança, uma vez que estas extensões ao ambientes pode ser realizadas a partir da inclusão e integração de novos serviços junto aos já existentes.

- *Heterogeneidade dependente de fornecedores de serviços*

Conforme explicado neste trabalho, a heterogeneidade é um importante requisito a ser atendido em ambientes de suporte à execução de WLDs. Através dela, é possível obter a integração de rotinas computacionais implementadas em linguagens de programação diferentes, implantadas em infraestruturas de hardware distintas, utilizando múltiplos sistemas operacionais e assim por diante.

Neste trabalho, não são desenvolvidos protocolos, linguagens ou sistemas de *middleware* que permitam a interoperabilidade de componentes que apresentem incompatibilidades em características como as citadas. A heterogeneidade está presente no *LONGWisE4cloud* no uso de um componente que abstrai detalhes a respeito dos repositórios utilizados no ambiente, permitindo que múltiplas soluções sejam utilizadas para implementá-los sem maiores reflexos para os demais componentes. Fora este elemento específico, a heterogeneidade é dependente dos recursos utilizados na implementação dos serviços que atendem às funcionalidades do ambiente e às atividades dos WLDs. Em virtude dos princípios de *SOC* utilizados no desenvolvimento do ambiente, há a possibilidade de se encapsular chamadas a rotinas que estejam, por exemplo, remotamente implantadas em uma infraestrutura diferente de onde o ambiente se encontra.

- *Provisionamento restrito a parametrizações de WLDs*

A longa duração dos WLDs aumenta o desafio de se desenvolver estratégias para o provisionamento de recursos de hardware e/ou software de forma que haja ganhos no desempenho da execução destes *workflows* e ao mesmo tempo evite-se o desperdício ou mal uso destes recursos computacionais.

No *LONGWisE4cloud*, este provisionamento é restrito ao atendimento a parâmetros específicos que podem vir associados aos WLDs, como o uso de um serviço de armazenamento dedicado ou a execução isolada de um WLD em uma *ECI*. Não há, na proposta, um mecanismo de provisionamento de recursos que permita a reserva específica de elementos de hardware (e.g., quantidade de processadores ou montante de memória) ou de software (e.g., uso temporário de um software como serviço (*as a Service*) de terceiros em uma infraestrutura qualquer). Também

não existe no *LONGWisE4cloud* uma solução que realize previsões relativas à necessidade destes recursos com antecedência, utilizando, por exemplo, informações que indiquem aumento em tempos de processamentos que precisam de mais recursos computacionais para serem reduzidos.

- *Reuso de dados provido pelo ambiente restrito a SGBDs relacionais*

O mecanismo de reuso de dados implementado no *LONGWisE4cloud*, utilizado quando, por exemplo, um serviço é substituído por outro em caso de falhas, é capaz de realizar mudanças em esquemas de dados e transportar informações de um esquema para outro apenas quando estes esquemas estão implantados em SGBDs relacionais. Sendo assim, este mecanismo não suporta, por exemplo, a chamada *persistência poliglota*. Neste tipo de persistência, dados processados e gerados por uma aplicação podem ser armazenados em diversos tipos de repositórios como, por exemplo, SGBDs relacionais, bancos de dados *NoSQL* ou ainda em alguma solução de armazenamento específica oferecida por um provedor de nuvem. Apesar desta restrição existente no *LONGWisE4cloud*, vale salientar que os WLDs submetidos ao ambiente podem ser associados a rotinas específicas destinadas a ações de reuso em adaptações. Estas rotinas podem lidar com bases de dados diferentes de SGBDs relacionais e até mesmo dar suporte à *persistência poliglota*, caso tenham sido implementadas com tal finalidade.

- *Inexistência de mecanismos de self-healing*

O *LONGWisE4cloud* apresenta mecanismos de monitoramento, identificação de falhas, adaptações para contornar estas falhas e eventuais mudanças em regras de negócio associadas aos *workflows* em execução. Entretanto, o ambiente não apresenta soluções elaboradas para adaptações e tratamentos de falhas envolvendo os próprios componentes do ambiente. Por exemplo, caso uma *ECI* que esteja ativa no ambiente se torne indisponível de forma abrupta em virtude de uma falha na máquina virtual onde ela estava implantada, o ambiente não possui recursos nem mecanismos implementados para dar continuidade a todos os processamentos que estavam sendo tocados pela *ECI*. Uma exceção é lançada no *Dispatcher* ao tentar acessar a *ECI* que se tornou indisponível e ações manuais, realizadas pelos operadores do sistema, deverão contornar a situação. O desenvolvimento de estratégias que utilizem estados intermediários e monitoramentos dos próprios componentes do *LONGWisE4cloud* de forma que este tipo de falha pudesse ser sanada automaticamente pelo ambiente seria um melhoramento importante para a robustez na execução dos WLDs.

- *Inexistência de negociações e acompanhamento de parâmetros de QoS*

O gerenciamento dos parâmetros relativos à qualidade de serviços, ou *QoS* (*Quality of Service*), é algo muito comum hoje em dia em aplicações distribuídas sobre ambientes de rede. Através deles, é possível se negociar com fornecedores de serviços de infraestrutura aspectos como a disponibilidade ou a confiabilidade das soluções oferecidas. Em geral, esta negociação

envolve valores explícitos, tais como o percentual de falhas aceitável quando uma determinada infraestrutura é utilizada para implantar um sistema. O *LONGWisE4cloud* não oferece suporte automatizado para a negociação e o acompanhamento destes parâmetros.

7.3 Trabalhos Futuros

- *Aperfeiçoar os algoritmos para a distribuição dos processamentos dos WLDs*

Conforme já mencionado, o *LONGWisE4cloud* é capaz de distribuir seus próprios componentes de forma a acomodar melhor a demanda pela execução de *workflows* de longa duração. Entretanto, este mecanismo, hoje, trabalha com um algoritmo que observa a quantidade de WLDs sendo executados em cada *ECI* e analisa tempos médios de processamento para identificar onde uma nova instância será executada. Além disso, ele é capaz de atender a exigências de usuários responsáveis pelos *workflows*, como não permitir que um *workflow* seja executado junto a outros WLDs de outros usuários.

Aperfeiçoar o mecanismo de criação e distribuição das *ECIs* em infraestruturas virtualizadas, assim como a alocação destes WLDs ao longo das mesmas é um esforço importante para tornar o *LONGWisE4cloud* ainda mais eficiente na execução destes *workflows*. Informações como estimativas de conclusão de atividades, custos (inclusive financeiros) associados ao uso de recursos computacionais, a disponibilidade e a confiabilidade destes recursos, muitas vezes definidas pelos próprios fornecedores destes recursos, podem ser integrados no desenvolvimento de algoritmos que permitam ao ambiente estabelecer quantas e em quais plataformas as *ECIs* serão instanciadas e para quais infraestruturas os *workflows* serão direcionados.

- *Suporte a novos aspectos na execução de WLDs*

Como forma de aperfeiçoar o *LONGWisE4cloud*, novos aspectos desejáveis à execução dos *workflows* de longa duração podem ser identificados e o suporte aos mesmos pode ser integrado aos demais recursos já existentes no ambiente.

Neste contexto, um aspecto que pode ser citado é a *análise dos custos* associados à execução dos WLDs. Através desta análise, ações como a identificação dos valores envolvidos com o pagamento pelo uso de infraestruturas de suporte à execução de aplicações, tais como as nuvens computacionais, poderiam ser realizadas. Estes valores poderiam ser confrontados com restrições impostas pelos usuários responsáveis pelos *workflows*. Desta forma, o ambiente gerenciaria a execução dos WLDs de maneira a atender requisitos financeiros estabelecidos nos cenários onde estes *workflows* estão sendo utilizados.

Outro aspecto que traria benefícios à execução de WLDs no *LONGWisE4cloud* seria o suporte à *antifragilidade*. Softwares que apresentam esta característica são capazes de se aperfeiçoar quando enfrentam falhas na execução de suas rotinas, através de ações como a mudança em funcionalidades já existentes ou a inclusão de novos mecanismos entre os já existentes em sua implementação.

- *Uso de novas tecnologias e padrões arquiteturais*

Frequentemente são propostas novas tecnologias associadas a aspectos como o armazenamento de informações ou a comunicação remota entre componentes de aplicações. Novos padrões arquiteturais para aplicações de software também surgem com diferentes propósitos, tais como aumento na robustez da solução ou a obtenção de uma maior facilidade na manutenção e aprimoramento dos sistemas. Neste cenário de inovações, um novo padrão arquitetural conhecido como *arquitetura de microserviços* (do inglês *microservices architecture*) tem sido cada vez mais adotado em diversos contextos, entre os quais se destacam os cenários corporativos. O padrão estabelece a construção de soluções baseadas em serviços computacionais especializados, fracamente acoplados, que agregam um número reduzido de funcionalidades e que se comunicam entre si através de protocolos simples e comumente utilizados em ambientes como a Internet (e.g, REST). O *LONGWisE4cloud* poderia adotar este padrão arquitetural não apenas na composição dos WLDs, onde os microserviços se responsabilizariam pelas atividades dos *workflows*, como também na implementação do ambiente propriamente dito. Com isto, ele incorporaria as vantagens acarretadas pelo uso do padrão, tais como o baixo acoplamento de serviços e a alta modularidade observadas nas soluções construídas com base nele.

- *Uso de padrões relacionados à modelagem de processos de negócio*

Atualmente, as soluções do *LONGWisE4cloud* lidam, basicamente, com objetos Java capazes de representar *workflows* e atividades associados a processos de negócio que devem ser executados no ambiente.

Entretanto, atualmente, padrões como *BPMN* são comumente utilizados na modelagem de processos de negócio com a consequente execução destes processos de forma automatizada por ambientes especializados para este fim, como os *BPMS*s. A integração do *LONGWisE4cloud* a ambientes de execução *BPMN*, por exemplo, tornará a solução mais abrangente e também mais alinhada com ferramentas e técnicas atuais associadas à modelagem de processos de negócio. Isto implicará em avanços no que diz respeito às funcionalidades do ambiente associadas ao contexto dos *workflows* de longa duração.

- *Novos cenários de reconfiguração*

Identificar novos cenários de reconfiguração que podem levar a adaptações dos *workflows* à medida que os mesmos são executados também constitui uma melhoria importante para o *LONGWisE4cloud*. Quanto mais opções de adaptação existirem no ambiente, maior a capacidade do mesmo de lidar com situações particulares críticas, tais como a interrupção abrupta de um serviço associado a uma atividade de um WLD. Estes eventos podem levar a efeitos indesejados, tais como a perda de resultados intermediários gerados por um WLDs antes de uma interrupção em sua execução. Mais estudos a respeito do comportamento da execução de *workflows* de

longa duração pode levar à identificação de novos e mais complexos cenários que exijam novas estratégias de adaptação.

REFERÊNCIAS

AALST, W. M. P. van der; HOFSTEDE, A. H. M. ter; WESKE, M. Business process management: A survey. In: AALST, W. M. P. van der; WESKE, M. (Ed.). *Business Process Management*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. p. 1–12. ISBN 978-3-540-44895-2.

ABPMP-BR. *BPM CBOK Versão 3.0: Guia para o Gerenciamento de Processos de Negócio - Corpo Comum de Conhecimento - ABPMP BPM CBOK V3.0*. 1. ed. [S.l.]: ABPMP – Association of Business Process Management Professionals – Brasil, 2013. 35 p. ISBN 978-1-4905-1659-2.

ALI, M. S.; REIFF-MARGANIEC, S. Autonomous failure-handling mechanism for wf long running transactions. In: *Services Computing (SCC), 2012 IEEE Ninth International Conference on*. [S.l.: s.n.], 2012. p. 562–569.

AMATO, F. et al. A middleware for improving concurrency of long running transactions. In: *Complex, Intelligent and Software Intensive Systems (CISIS), 2014 Eighth International Conference on*. [S.l.: s.n.], 2014. p. 588–595.

AMAZON-EC2. *Amazon Elastic Compute Cloud - EC2*. 2017. [Online; acessado em 23-outubro-2017]. Disponível em: <https://aws.amazon.com/pt/ec2/>.

AMAZON-S3. *Amazon Simple Storage Service*. 2017. [Online; acessado em 23-outubro-2017]. Disponível em: <https://aws.amazon.com/pt/s3/>.

ANJU, K. S. et al. Business process reengineering of the workflows in intensive care unit supported with a tablet pc based automation system. In: *2013 Third International Conference on Advances in Computing and Communications*. [S.l.: s.n.], 2013. p. 265–268.

ANJUM, M.; BUDGEN, D. Modelling the design for an soa system to control a small scale energy zone. In: *2012 IEEE 36th Annual Computer Software and Applications Conference Workshops*. [S.l.: s.n.], 2012. p. 538–543.

ARBAB, F. Reo: A channel-based coordination model for component composition. *Mathematical. Structures in Comp. Sci.*, Cambridge University Press, New York, NY, USA, v. 14, n. 3, p. 329–366, jun. 2004. ISSN 0960-1295. Disponível em: <http://dx.doi.org/10.1017/S0960129504004153>.

ARDISSONO, L. et al. Adaptive medical workflow management for a context-dependent home healthcare assistance service. *Electronic Notes in Theoretical Computer Science*, v. 146, n. 1, p. 59 – 68, 2006. ISSN 1571-0661. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1571066106000144>.

ASSUNCAO, L.; CUNHA, J. Dynamic workflow reconfigurations for recovering from faulty cloud services. In: *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*. [S.l.: s.n.], 2013. v. 1, p. 88–95.

AWS-EB. *AWS Elastic Beanstalk*. 2017. [Online; acessado em 23-outubro-2017]. Disponível em: <https://aws.amazon.com/pt/elasticbeanstalk/>.

AZURE. *Microsoft Windows Azure*. 2017. [Online; acessado em 23-outubro-2017]. Disponível em: <https://azure.microsoft.com/pt-br/>.

BANÂTRE, J.-P.; FRADET, P.; RADENAC, Y. Generalised multisets for chemical programming. *Mathematical. Structures in Comp. Sci.*, Cambridge University Press, New York, NY, USA, v. 16, n. 4, p. 557–580, ago. 2006. ISSN 0960-1295. Disponível em: <http://dx.doi.org/10.1017/S0960129506005317>.

BOCCHI, L.; LANEVE, C.; ZAVATTARO, G. Formal methods for open object-based distributed systems: 6th ifip wg 6.1 international conference, fmoods 2003, paris, france, november 19.21, 2003. proceedings. In: ____. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. cap. A Calculus for Long-Running Transactions, p. 124–138. ISBN 978-3-540-39958-2. Disponível em: <http://dx.doi.org/10.1007/978-3-540-39958-2_9>.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. *Unified Modeling Language User Guide, The (2Nd Edition) (Addison-Wesley Object Technology Series)*. [S.l.]: Addison-Wesley Professional, 2005. ISBN 0321267974.

BRUNI, R. et al. A conceptual framework for adaptation. In: *Proceedings of the 15th International Conference on Fundamental Approaches to Software Engineering*. Berlin, Heidelberg: Springer-Verlag, 2012. (FASE'12), p. 240–254. ISBN 978-3-642-28871-5. Disponível em: <http://dx.doi.org/10.1007/978-3-642-28872-2_17>.

BUTLER, M. J.; FERREIRA, C. A process compensation language. In: *Proceedings of the Second International Conference on Integrated Formal Methods*. London, UK, UK: Springer-Verlag, 2000. (IFM '00), p. 61–76. ISBN 3-540-41196-8. Disponível em: <http://dl.acm.org/citation.cfm?id=647982.743549>.

CAIRES, L.; FERREIRA, C.; VIEIRA, H. T. A process calculus analysis of compensations. In: KAKLAMANIS, C.; NIELSON, F. (Ed.). *Proceedings of the Fourth Symposium on Trustworthy Global Computing, TGC'08*. [S.l.]: Springer-Verlag, 2009. (Lecture Notes in Computer Science, 5474), p. 87–103.

CHENG, B. H. C. et al. Software engineering for self-adaptive systems: A research roadmap. In: ____. *Software Engineering for Self-Adaptive Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 1–26. ISBN 978-3-642-02161-9. Disponível em: <https://doi.org/10.1007/978-3-642-02161-9_1>.

CHIANESE, A. et al. Pre-serialization of long running transactions to improve concurrency in mobile environments. In: *Data Engineering Workshop, 2008. ICDEW 2008. IEEE 24th International Conference on*. [S.l.: s.n.], 2008. p. 129–136.

CHUNG, P. W. H. et al. Knowledge-based process management - an approach to handling adaptive workflow. *Knowl.-Based Syst.*, v. 16, n. 3, p. 149–160, 2003. Disponível em: <http://dx.doi.org/10.1016/S0950-7051(02)00080-1>.

COCHRAN, W. G.; COCHRAN, W.; COX, G. M. *Experimental designs*. [S.l.]: Wiley, 1992. (Wiley Classics Library). ISBN 9780471545675.

DANG, J. et al. An ontological knowledge framework for adaptive medical workflow. *Journal of Biomedical Informatics*, v. 41, n. 5, p. 829–836, 2008. Disponível em: <http://dblp.uni-trier.de/db/journals/jbi/jbi41.html#DangHHT08>.

DECKER, M. et al. Modeling mobile workflows with bpmn. In: *2010 Ninth International Conference on Mobile Business and 2010 Ninth Global Mobility Roundtable (ICMB-GMR)*. [S.l.: s.n.], 2010. p. 272–279.

DROPBOX. *DropBox*. 2017. [Online; acessado em 23-outubro-2017]. Disponível em: <https://dropbox.com/>.

EQUINOX. *Eclipse Equinox Home Page*. 2017. [Online; acessado em 23-outubro-2017]. Disponível em: <http://www.eclipse.org/equinox/>.

ESTEVES, S.; VEIGA, L. Waas: Workflow-as-a-service for the cloud with scheduling of continuous and data-intensive workflows. *The Computer Journal*, v. 59, n. 3, p. 371–383, 2016. Disponível em: <http://comjnl.oxfordjournals.org/content/59/3/371.abstract>.

FELIX. *Apache Felix Home Page*. 2017. [Online; acessado em 23-outubro-2017]. Disponível em: <http://felix.apache.org/>.

FERME, V. et al. Workflow management systems benchmarking: Unfulfilled expectations and lessons learned. In: *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. [S.l.: s.n.], 2017. p. 379–381.

FERNÁNDEZ, H.; TEDESCHI, C.; PRIOL, T. A chemistry-inspired workflow management system for decentralizing workflow execution. *IEEE Transactions on Services Computing*, v. 9, n. 2, p. 213–226, March 2016. ISSN 1939-1374.

FRANK, D.; FONG, L.; LAM, L. A continuous long running batch orchestration model for workflow instance migration. In: *Services Computing (SCC), 2010 IEEE International Conference on*. [S.l.: s.n.], 2010. p. 226–233.

GARCIA-MOLINA, H.; SALEM, K. Sagas. In: *Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: ACM, 1987. (SIGMOD '87), p. 249–259. ISBN 0-89791-236-5. Disponível em: <http://doi.acm.org/10.1145/38713.38742>.

GEORGAKOPOULOS, D.; PAPAZOGLOU, M. P. Overview of service-oriented computing. In: ____. *Service-Oriented Computing*. MIT Press, 2008. p. 1–28. ISBN 9780262273671. Disponível em: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6285523>.

GMAIL. *Google Gmail*. 2017. [Online; acessado em 23-outubro-2017]. Disponível em: <https://gmail.com/>.

GONG, R. et al. Modelling for business process design: a methodology based on causal loop diagram. In: *Systems, Man and Cybernetics, 2004 IEEE International Conference on*. [S.l.: s.n.], 2004. v. 7, p. 6149–6154 vol.7. ISSN 1062-922X.

GREENFIELD, P. et al. Compensation is not enough [fault-handling and compensation mechanism]. In: *Enterprise Distributed Object Computing Conference, 2003. Proceedings. Seventh IEEE International*. [S.l.: s.n.], 2003. p. 232–239.

HAO, Y.; ZHONG, F. The extension of long-running transactions compensation language-stac. In: *Computer Science and Network Technology (ICCSNT), 2011 International Conference on*. [S.l.: s.n.], 2011. v. 2, p. 713–717.

HEROKU. *Heroku Platform*. 2017. [Online; acessado em 23-outubro-2017]. Disponível em: <https://www.heroku.com/>.

HU, Q. et al. A long-running transaction model of workflow. In: *Biomedical Engineering and Informatics (BMEI), 2010 3rd International Conference on*. [S.l.: s.n.], 2010. v. 6, p. 2618–2622.

IBM. *IBM Business Process Management*. 2018. [Online; acessado em 12-fevereiro-2018]. Disponível em: <https://www.ibm.com/support/knowledgecenter/pt-br/SSFTN5_8.5.6/com.ibm.wbpm.main.doc/topics/ibmbmp_overview.html>.

JAIN, R. The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling. In: ____. [S.l.]: Wiley, 1991. (Wiley professional computing), p. 26–28. ISBN 978-0-471-50336-1.

JAYAKODY, J. A. D. C. A. et al. A framework for business process re-engineering to reduce the number of processes - a case study of national blood transfusion services. In: *2016 International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT)*. [S.l.: s.n.], 2016. p. 208–214.

JUHNKE, E.; DORNEMANN, T.; FREISLEBEN, B. Fault-Tolerant BPEL Workflow Execution via Cloud-Aware Recovery Policies. In: *Software Engineering and Advanced Applications, 2009. SEAA '09. 35th Euromicro Conference on*. [S.l.: s.n.], 2009. p. 31–38. ISSN 1089-6503.

KHALAF, R.; KELLER, A.; LEYMANN, F. Business processes for Web Services: Principles and applications. *IBM Systems Journal: Celebrating 10 Years of XML*, Online, v. 45, n. 2, p. 425–446, maio 2006.

KIKUCHI, S. Performance estimation for business workflows on public cloud offerings using probabilistic model checker. In: *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*. [S.l.: s.n.], 2014. p. 317–326.

KIMBALL, R.; ROSS, M. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. 2nd. ed. New York, NY, USA: John Wiley & Sons, Inc., 2002. ISBN 0471200247, 9780471200246.

KOKASH, N.; ARBAB, F. Formal design and verification of long-running transactions with extensible coordination tools. *IEEE Transactions on Services Computing*, v. 6, n. 2, p. 186–200, April 2013. ISSN 1939-1374.

KRAMER, J.; MAGEE, J. The evolving philosophers problem: dynamic change management. *Software Engineering, IEEE Transactions on*, v. 16, n. 11, p. 1293–1306, Nov 1990. ISSN 0098-5589.

LEMOS, R. de et al. Software engineering for self-adaptive systems: A second research roadmap. In: ____. *Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 1–32. ISBN 978-3-642-35813-5. Disponível em: <https://doi.org/10.1007/978-3-642-35813-5_1>.

LIU, F. et al. *NIST Cloud Computing Reference Architecture: Recommendations of the National Institute of Standards and Technology (Special Publication 500-292)*. USA: CreateSpace Independent Publishing Platform, 2012. ISBN 1478168021, 9781478168027.

LU, M.; CAI, Q.; LI, H. A method of transition from bpmn to bpel. In: *Advanced Intelligence and Awareness Internet (AIAI 2011), 2011 International Conference on*. [S.l.: s.n.], 2011. p. 372–375.

MACHADO, C.; AREIAS, C.; CUNHA, J. C. Soasales: A soa system for research purposes. In: *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*. [S.l.: s.n.], 2016. p. 17–24.

MACLENNAN, E.; BELLE, J.-P. V. Factors affecting the organizational adoption of service-oriented architecture (SOA). *Information Systems and e-Business Management*, 2014.

MARPLES, D.; KRIENS, P. The Open Services Gateway Initiative: an introductory overview. *Communications Magazine, IEEE*, v. 39, n. 12, p. 110–114, Dec 2001. ISSN 0163-6804.

MENG, S.; ARBAB, F. A Model for Web Service Coordination in Long-Running Transactions. In: *Service Oriented System Engineering (SOSE), 2010 Fifth IEEE International Symposium on*. [S.l.: s.n.], 2010. p. 121–128.

MONTAGUT, F.; MOLVA, R.; GOLEGA, S. T. The pervasive workflow: A decentralized workflow system supporting long-running transactions. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, v. 38, n. 3, p. 319–333, May 2008. ISSN 1094-6977.

MOTAHARI-NEZHAD, H. R.; STEPHENSON, B.; SINGHA, S. Outsourcing Business to Cloud Computing Services: Opportunities and Challenges. *IEEE IT Professional, Special Issue on Cloud Computing*, v. 11, n. 2, set. 2009.

MSOFFICE365. *Microsoft Office 365*. 2017. [Online; acessado em 23-outubro-2017]. Disponível em: <https://www.office.com/>.

MÜLLER, R.; GREINER, U.; RAHM, E. Agentwork: a workflow system supporting rule-based workflow adaptation. *Data & Knowledge Engineering*, Elsevier, v. 51, n. 2, p. 223–256, 2004.

NAMI, M. R.; SHARIFI, M. A survey of autonomic computing systems. In: ____. *Intelligent Information Processing III: IFIP TC12 International Conference on Intelligent Information Processing (IIP 2006), September 20–23, Adelaide, Australia*. Boston, MA: Springer US, 2007. p. 101–110. ISBN 978-0-387-44641-7. Disponível em: <https://doi.org/10.1007/978-0-387-44641-7_11>.

NARENDRA, N. Flexible support and management of adaptive workflow processes. *Information Systems Frontiers*, v. 6, n. 3, p. 247–262, 2004. ISSN 1572-9419. Disponível em: <http://dx.doi.org/10.1023/B:ISFI.0000037879.05648.ca>.

NITTO, E. D. et al. A journey to highly dynamic, self-adaptive service-based applications. *Autom. Softw. Eng.*, v. 15, n. 3-4, p. 313–341, 2008.

NURHAYATI; FITRISARI, D. Business process automation of document filing based alfresco for referral bpjs patient (case study: Bpjs center of dharmais cancer hospital). In: *2016 International Conference on Informatics and Computing (ICIC)*. [S.l.: s.n.], 2016. p. 406–410.

OASIS. *Web Services Business Process Execution Language - Version 2.0 Specification*. [S.l.], 2007.

OLMSTED, A. Long running, consistent, web service transactions. In: *10th International Conference for Internet Technology and Secured Transactions, ICITST 2015, London, United Kingdom, December 14-16, 2015*. [s.n.], 2015. p. 139–144. Disponível em: <http://dx.doi.org/10.1109/ICITST.2015.7412074>.

OMG. *Business Process Model and Notation (BPMN) Version 2.0*. [S.l.], 2011.

Oracle. *Oracle BPM - Business Process Management*. 2018. [Online; acessado em 12-fevereiro-2018]. Disponível em: <http://www.oracle.com/us/technologies/bpm/overview/index.html>.

OROSZ, I.; OROSZ, T. Business process reengineering project in local governments with erp. In: *2012 7th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)*. [S.l.: s.n.], 2012. p. 371–376.

OSGi-ALLIANCE. *OSGi Alliance Home Page*. 2017. [Online; acessado em 23-outubro-2017]. Disponível em: <http://www.osgi.org/Main/HomePage>.

PAPAZOGLOU, M. P.; HEUVEL, W.-J. V. D. Service oriented architectures: approaches, technologies and research issues. *The VLDB journal*, Springer, v. 16, n. 3, p. 389–415, 2007.

PATHIRAGE, M. et al. A Multi-tenant Architecture for Business Process Executions. In: *Web Services (ICWS), 2011 IEEE International Conference on*. [S.l.: s.n.], 2011. p. 121–128.

PEREZ-SORROSAL, F. et al. Highly Available Long Running Transactions and Activities for J2EE Applications. In: *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on*. [S.l.: s.n.], 2006. p. 2–2. ISSN 1063-6927.

PLOOM, T.; SCHEIT, S.; GLASER, A. Methodology for migration of long running process instances in a global large scale bpm environment in credit suisse's soa landscape. In: *Software Engineering (ICSE), 2012 34th International Conference on*. [S.l.: s.n.], 2012. p. 977–986. ISSN 0270-5257.

QI, Y. n.; CHU, Z. f. The impact of supply chain strategies on supply chain integration. In: *Management Science and Engineering, 2009. ICMSE 2009. International Conference on*. [S.l.: s.n.], 2009. p. 534–540.

Red Hat. *Red Hat JBoss BPM Suite*. 2018. [Online; acessado em 12-fevereiro-2018]. Disponível em: <https://www.redhat.com/pt-br/technologies/jboss-middleware/bpm>.

RICHARDSON, L.; RUBY, S. *Restful Web Services*. First. [S.l.]: O'Reilly, 2007. ISBN 9780596529260.

ROBINSON, J. W.; ZUVIRIA, N. M.; VINITA, P. E. Automated analysis of workflow cloud-based business process using map reduce algorithm. In: *Computing Communication Networking Technologies (ICCCNT), 2012 Third International Conference on*. [S.l.: s.n.], 2012. p. 1–7.

ROSENBERG, J.; MATEOS, A. *The Cloud at Your Service*. 1st. ed. Greenwich, CT, USA: Manning Publications Co., 2010. ISBN 1935182528, 9781935182528.

SADASHIV, N.; KUMAR, S. M. D. Cluster, grid and cloud computing: A detailed comparison. In: *Computer Science Education (ICCSE), 2011 6th International Conference on*. [S.l.: s.n.], 2011. p. 477–482.

SALEHIE, M.; TAHVILDARI, L. Self-adaptive software: Landscape and research challenges. *ACM Trans. Auton. Adapt. Syst.*, ACM, New York, NY, USA, v. 4, n. 2, p. 14:1–14:42, maio 2009. ISSN 1556-4665. Disponível em: <http://doi.acm.org/10.1145/1516533.1516538>.

SAMIRI, M. Y. et al. Toward a self-adaptive workflow management system through learning and prediction models. *Arabian Journal for Science and Engineering*, v. 42, n. 2, p. 897–912, 2017. ISSN 2191-4281. Disponível em: <http://dx.doi.org/10.1007/s13369-016-2372-3>.

SAURABH, K.; RANJAN, R. Cloud management simulation and design. In: *2012 UKSim 14th International Conference on Computer Modelling and Simulation*. [S.l.: s.n.], 2012. p. 522–527.

SCHäFER, D. R. et al. Hawks: A system for highly available executions of workflows. In: *2016 IEEE International Conference on Services Computing (SCC)*. [S.l.: s.n.], 2016. p. 130–137.

SIADAT, S. H.; SHAMASBI, S. M. Reengineering purchase request process of tam iran khodro company using best practices. In: *2015 7th Conference on Information and Knowledge Technology (IKT)*. [S.l.: s.n.], 2015. p. 1–10.

SIEBERT, R. An open architecture for adaptive workflow management systems. *Journal of Integrated Design and Process Science*, IOS Press, v. 3, n. 3, p. 29–41, 1999.

SILVIA; SUHARDI; YUSTIANTO, P. Business process improvement of district government innovation service: Case study cimahi tengah district of cimahi. In: *2016 International Conference on Information Technology Systems and Innovation (ICITSI)*. [S.l.: s.n.], 2016. p. 1–6.

STANKEVICIUS, K.; VASILECAS, O. An approach on long running business process modelling and simulation. In: *2016 Open Conference of Electrical, Electronic and Information Sciences (eStream)*. [S.l.: s.n.], 2016. p. 1–4.

STEIN, S.; PAYNE, T. R.; JENNINGS, N. R. Robust execution of service workflows using redundancy and advance reservations. *IEEE Transactions on Services Computing*, v. 4, n. 2, p. 125–139, April 2011. ISSN 1939-1374.

SUPING, X. et al. Design of workflow cloud platform and its application in business systems. In: *2016 8th International Conference on Information Technology in Medicine and Education (ITME)*. [S.l.: s.n.], 2016. p. 521–525.

VAQUERO, L. M. et al. A break in the clouds: Towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, p. 50–55, 2009.

VAZ, C.; FERREIRA, C.; RAVARA, A. Dynamic recovery of long running transactions. In: KAKLAMANIS, C.; NIELSON, F. (Ed.). *4th International Symposium on Trustworthy Global Computing (TGC 2008)*. [S.l.]: Springer-Verlag, 2009. (Lecture Notes in Computer Science, 5474), p. 201–215.

VERGIDIS, K.; TIWARI, A.; MAJEED, B. Business Process Analysis and Optimization: Beyond Reengineering. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, v. 38, n. 1, p. 69–82, Jan 2008. ISSN 1094-6977.

WANG, J. et al. Workflow as a service in the cloud: Architecture and scheduling algorithms. *Procedia Computer Science*, v. 29, p. 546 – 556, 2014. ISSN 1877-0509. 2014 International Conference on Computational Science. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877050914002269>.

WEISER, M. Human-computer interaction. In: BAECKER, R. M. et al. (Ed.). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995. cap. The Computer for the 21st Century, p. 933–940. ISBN 1-55860-246-1. Disponível em: <http://dl.acm.org/citation.cfm?id=212925.213017>.

WESKE, M. Formal foundation and conceptual design of dynamic adaptations in a workflow management system. In: *Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 7 - Volume 7*. Washington, DC, USA: IEEE Computer Society, 2001. (HICSS '01), p. 7051–. ISBN 0-7695-0981-9. Disponível em: <http://dl.acm.org/citation.cfm?id=820759.821911>.

WIERINGA, R. J. *Design science methodology for information systems and software engineering*. [S.l.]: Springer, 2014.

XIAOLAN, W.; HONG, H. Research on long running transaction consistency based on csp trace semantic. In: *Information Processing (ISIP), 2010 Third International Symposium on*. [S.l.: s.n.], 2010. p. 148–152.

YANG, P. et al. Architecture design of the workflow execution system containing humanware services. In: *Proceedings of the 11th IEEE International Conference on Networking, Sensing and Control*. [S.l.: s.n.], 2014. p. 54–57.

YIMIN, Z.; PING, S.; QI, S. Design of public health management system based on soa architecture. In: *2017 29th Chinese Control And Decision Conference (CCDC)*. [S.l.: s.n.], 2017. p. 3381–3384.

YIN, J. et al. pi;rbt-calculus protocol for web service long running transactions. In: *Communications and Information Technology, 2005. ISCIT 2005. IEEE International Symposium on*. [S.l.: s.n.], 2005. v. 2, p. 1540–1544.

ZENG, S. et al. An evidence based analytical approach for process automation: Case study in finance and administration process delivery services. In: *2011 IEEE 8th International Conference on e-Business Engineering*. [S.l.: s.n.], 2011. p. 286–292.

ZHAO, W.; ZHANG, H. Proactive service migration for long-running byzantine fault-tolerant systems. *IET Software*, v. 3, n. 2, p. 154–164, 2009. Disponível em: <http://dx.doi.org/10.1049/iet-sen.2008.0065>.

APÊNDICE

A

PUBLICAÇÕES DO AUTOR

Publicações realizadas pelo autor juntamente com seus orientadores e relacionadas com o trabalho desenvolvido nesta tese:

- **Junior, M. S. S.**, Rosa, N. S. e Lins, F. (2014). Execution Support to Long Running Workflows. (pp. 496–503). In: *14th IEEE International Conference on Computer and Information Technology (CIT)*.
- **Junior, M. S. S.**, Rosa, N. S. e Lins, F. (2015). xlongwise: An environment to support the execution of long running workflows. (pp. 1-8). In: *IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA),*.

B

OSGI

OSGi (*Open Services Gateway Initiative*) (OSGi-ALLIANCE, 2017) é um padrão de desenvolvimento que permite a construção de aplicações dinâmicas e modulares (MARPLES; KRIENS, 2001). Uma aplicação *OSGi* é composta por um conjunto de componentes interconectados e cujas dependências, uma vez definidas pelos desenvolvedores, são resolvidas pela própria plataforma. Ações como parar ou reiniciar componentes são resolvidas pelo próprio ambiente onde a aplicação é executada à medida que outros componentes tornam-se indisponíveis ou disponíveis. Sendo assim, para a implementação de composições dinâmicas de serviços, o uso de ambientes *OSGi* se torna uma escolha interessante.

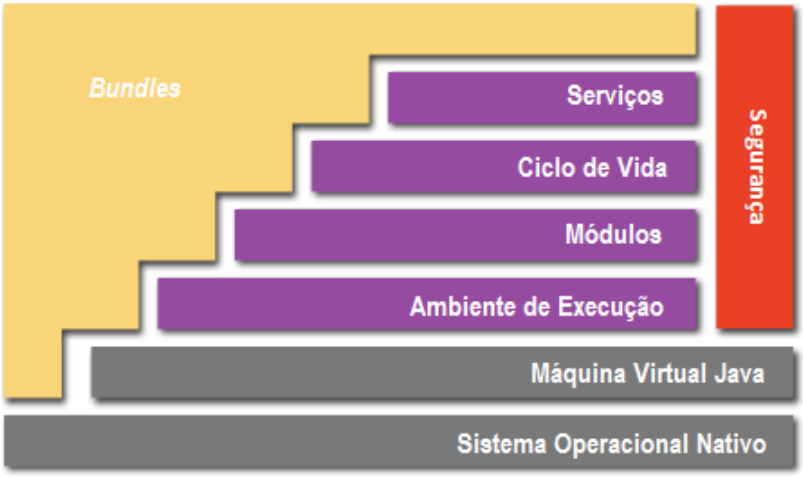
A Figura B.1 apresenta a arquitetura em camadas do padrão *OSGi* (OSGi-ALLIANCE, 2017). Nela, é possível encontrar os termos abaixo descritos:

- *Bundles*: São os componentes *OSGi* implementados pelos desenvolvedores;
- Serviços: esta camada é responsável pela conexão dos *bundles* de forma dinâmica oferecendo um modelo baseado em publicação, busca e conexão para objetos Java;
- Ciclo de vida: A *API* destinada a instalar, iniciar, parar, atualizar e desinstalar *bundles*;
- Módulos: Estabelece como um *bundle* pode importar e exportar código;
- Segurança: Camada responsável por aspectos de segurança;
- Ambiente de execução: Responsável por definir quais métodos e classes estão disponíveis em uma plataforma específica.

O uso de soluções *OSGi* é facilitado pelos *frameworks*, ambientes colaborativos onde são implantados os *bundles* para a sua execução. Como exemplos destes *frameworks* estão o Eclipse Equinox (EQUINOX, 2017) e o Apache Felix (FELIX, 2017).

No contexto dos *workflows* de longa duração, onde não apenas os processos de negócio são executados a partir de composições de serviços especializados mas o próprio ambiente de gerenciamento destas execuções também é constituído por serviços computacionais, o uso

Figura B.1: Arquitetura em Camadas do Padrão OSGi



Fonte: (OSGi-ALLIANCE, 2017)

de *OSGi* é capaz de reforçar aspectos de flexibilidade e robustez, por exemplo. Isto se dá a partir do momento que a escolha e a troca de componentes para executar estes processos de negócio é facilitada. Além disso, falhas na execução de processos ocasionadas pela ausência ou instabilidade de componentes também podem ser mais facilmente contornadas, minimizadas ou evitadas com o auxílio da plataforma. Isto implica em ganhos de desempenho e até vantagens financeiras, como a economia nos custos de uso de infraestruturas computacionais, para cenários que precisam dos resultados destes processamentos, como é o caso dos ambientes corporativos.

C

ESQUEMAS XML UTILIZADOS

Na Figura C.1, encontra-se o esquema que define o *XML* enviado para o *ECIListener* quando ocorre a solicitação para a execução de um WLD via *Web Service*.

Figura C.1: Definição de Esquema XML para WLD

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="lrWorkflow" type="lrWorkflow"/>

  <xs:complexType name="lrWorkflow">
    <xs:sequence>
      <xs:element name="activitiesList" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="activity" type="activity" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="activitiesDependecies" type="xs:string" minOccurs="0"/>
      <xs:element name="auxiliarRoutinesList" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="auxiliarRoutine" type="activity" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="defaultExceptionHandler" type="xs:string" minOccurs="0"/>
      <xs:element name="instanceID" type="xs:int"/>
      <xs:element name="processDescription" type="xs:string" minOccurs="0"/>
      <xs:element name="processIdentification" type="xs:string" minOccurs="0"/>
      <xs:element name="propertiesList" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="property" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="userResponsible" type="xs:string" minOccurs="0"/>
      <xs:element name="workflowVersion" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="activity">
    <xs:sequence>
      <xs:element name="activityDescription" type="xs:string" minOccurs="0"/>
      <xs:element name="activityIdentification" type="xs:string" minOccurs="0"/>
      <xs:element name="properties">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="entry" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="key" minOccurs="0" type="xs:string"/>
                  <xs:element name="value" minOccurs="0" type="xs:string"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="runParameters">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="entry" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="key" minOccurs="0" type="xs:string"/>
                  <xs:element name="value" minOccurs="0" type="xs:string"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="serviceReference" type="xs:string" minOccurs="0"/>
      <xs:element name="status" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Fonte: O próprio autor.