Gerando Planos de Rollback mais Eficientes em Sistemas de Gerenciamento de Mudanças em TI*

Alan Diego dos Santos, Guilherme Sperb Machado, Weverton Luis da Costa Cordeiro, Fabrício Girard Andreis, Juliano Araujo Wickboldt, Roben Castagna Lunardi, Cristiano Bonato Both, Luciano Paschoal Gaspary, Lisandro Zambenedetti Granville

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre, RS – Brasil
{adsantos, gsmachado, weverton.cordeiro, fgandreis
jwickboldt, rclunardi, cbboth, paschoal, granville}@inf.ufrgs.br

Abstract. *Change Management systems aim to coordinate and deploy changes over modern IT (Information Technology) infrastructures. However, unexpected failures may occur during the deployment of changes, leading the managed infrastructure to an inconsistent state. In order to minimize the effects caused by those failures, these systems should support compensate activities as well as rollback actions. Mostly, the generation of rollback plans is made in a rudimentary manner, undoing change activities by reversing the change plan in the same order that they were executed. Optimized rollback plans may potentially decrease the IT infrastructure recovery time and improve the use of computational resources. In this paper, it is presented a solution to the generation of optimized rollback plans in IT Change Management systems, focusing on the algorithm to achieve them. The obtained results show that optimized rollback plans are executed in a faster way, also using computational resources rationally.*

Resumo. *Sistemas de gerenciamento de mudanças visam a coordenação e execução de modificações em infra-estruturas de TI de organizações modernas. Durante a execução das mesmas, no entanto, falhas podem ocorrer, levando a infra-estrutura gerenciada a um estado inconsistente. Para minimizar os efeitos dessas falhas, tais sistemas devem suportar atividades compensatórias, bem como o retrocesso (rollback) das mudanças que falharam. Nesse último caso, o rollback é feito de forma rudimentar, desfazendo-se as atividades de uma mudança na ordem contrária em que foram executadas. Planos de rollback otimizados, por sua vez, têm potencial para diminuir o tempo de recuperação de uma infra-estrutura de TI, além de melhor utilizar os recursos computacionais disponíveis. Neste artigo, é apresentada uma solução para a geração de planos de rollback otimizados em sistemas de gerenciamento de mudanças em TI, descrevendo um algoritmo com esse propósito. Os resultados alcançados permitem observar que os planos gerados são executados mais rapidamente, além de utilizar os recursos computacionais de maneira mais racional.*

1. Introdução

Para oferecer serviços com qualidade, organizações de grande porte freqüentemente empregam sofisticadas infra-estruturas de Tecnologia da Informação (TI). O gerenciamento

*Este trabalho foi desenvolvido em colaboração com HP Brasil P&D.

e manutenção dessas infra-estruturas são complexos e demandam altos custos para a instituição. Por esse motivo, a implantação de boas práticas e políticas de gerenciamento torna-se importante, já que essa adoção aumenta a qualidade dos processos de TI e, conseqüentemente, a qualidade dos serviços prestados. Com o intuito de auxiliar as instituições na adoção de políticas de gerenciamento, a *Office for Government Commerce* (OGC) criou a *Information Technology Infrastructure Library* (ITIL) [ITIL 2008], que introduz um conjunto de processos e boas práticas para auxiliar as organizações na condução adequada de suas infra-estruturas de TI.

Dentre os vários processos descritos pela ITIL, o *gerenciamento de mudanças* é aquele que descreve como mudanças em TI devem ser planejadas, avaliadas e executadas. No livro ITIL *Service Transition* [ITIL 2007] é definido que as mudanças devem ser expressas em requisições de mudanças (*Requests for Change* - RFCs), que são documentos que descrevem e documentam as mudanças solicitadas, mas não informam como elas devem ser implantadas. A partir de uma RFC, deve ser criado posteriormente um plano de mudanças (*change plan* - CP), este sim um *workflow* contendo a definição de atividades encadeadas que devem ser executadas sobre a infra-estrutura de TI para que a mudança requisitada seja efetivamente implantada. Esse *workflow* é computado respeitando uma série de dependências entre as diversas atividades a serem executadas sobre a infra-estrutura gerenciada [Cordeiro *et al.* 2008].

De acordo com a ITIL, para que as mudanças descritas na RFC sejam implantadas, é necessária sua aprovação por parte de um Comitê de Avaliação de Mudança (*Change Advisory Board* - CAB). O CAB avalia, dentre vários aspectos, a existência de atividades especiais que devem ser tomadas em caso de falha durante a execução do CP; esse conjunto de atividades especiais é denominado *plano de remediação*. Existem basicamente dois tipos distintos e complementares de planos de remediação: planos de compensação e planos de *rollback*. Os *planos de compensação* têm o objetivo de tratar as falhas ocorridas, compensando as atividades com erro, de modo a possibilitar que as mudanças descritas na RFC sejam, mesmo com a existência de falhas, implantadas na infra-estrutura de TI [Machado *et al.* 2009]. Os *planos de rollback*, por sua vez, executam o retorno da infra-estrutura de TI ao estado anterior à implantação da RFC [Machado *et al.* 2008b], retrocedendo o efeito das atividades do CP original. Neste artigo, estamos particularmente interessados em investigar estratégias para aprimoramento dos planos de *rollback*. Tal investigação se faz necessária porque os planos de *rollback* têm impacto direto no tempo de indisponibilidade dos serviços de uma infra-estrutura de TI; planos de *rollback* mal projetados podem gerar indisponibilidades de serviços prejudicialmente longas para os serviços das corporações.

Para retornar a infra-estrutura de TI ao estado imediatamente anterior à implantação de uma RFC, as atividades do CP que aconteceram antes de uma falha precisam ser desfeitas, evitando assim que a infra-estrutura de TI evolua um estado inconsistente. Algumas estratégias podem ser adotadas para reverter uma RFC que falhou. O modo mais rudimentar de realizar o retrocesso é especificar um plano de *rollback* onde as atividades são desfeitas exatamente na ordem inversa em que foram executadas no CP original [Machado *et al.* 2008a]. Apesar dessa abordagem mais rudimentar ser eficaz, otimizações podem ser feitas nos planos de *rollback*, de forma a reduzir o tempo de retrocesso e os recursos computacionais utilizados. Essa meta pode ser alcançada, por exemplo, eliminando atividades de reversão que não alteram o resultado final do retro-

cesso. Isso pode ser observado, por exemplo, ao se executar o *rollback* de atividades seqüenciais de instalação e configuração de um servidor DNS; para desinstalar um servidor DNS não é necessário proceder com nenhuma desconfiguração. Logo, a atividade de desconfiguração (*i.e.*, atividade de reversão) não precisa ser executada no *rollback*, mesmo existindo uma atividade de configuração (*i.e.*, atividade original) associada.

Este artigo apresenta uma nova solução para a criação de planos de *rollback*, de modo que os planos gerados sejam mais eficientes em relação ao tempo de retrocesso e utilização de recursos computacionais. Objetivando aumentar essa eficiência, nossa abordagem é caracterizada por (i) levar em consideração não só dependências entre atividade do CP original, mas também dependências entre atividades de *rollback* e (ii) remover atividades desnecessárias à execução do CP em questão, diminuindo o número de atividades do plano de *rollback* gerado. Para avaliar os planos de *rollback* gerados por nossa estratégia, um algoritmo, a ser apresentado neste artigo, foi implementado e anexado ao módulo de suporte a remediação do sistema CHANGELEDGE [Machado *et al.* 2008b].

O restante deste artigo está organizado da seguinte maneira. Na Seção 2, é revisado o suporte a planos de remediação em sistemas de TI, ligando este trabalho com pesquisas anteriores. A solução proposta é descrita na Seção 3, detalhando o algoritmo para geração automática de planos de *rollback* aprimorados. O sistema CHANGELEDGE e a incorporação ao mesmo do algoritmo proposto são apresentados na Seção 4, enquanto estudos de caso para avaliação de nossa proposta são mostrados na Seção 5. Por fim, este artigo é encerrado na Seção 6, onde conclusões e trabalhos futuros são apresentados.

2. Trabalhos Relacionados

Atualmente, diversas técnicas para a remediação de falhas são estudadas, podendo-se destacar, dentre elas, o *rollback*, que objetiva o retorno do sistema para o estado consistente anterior à operação executada. Uma das técnicas mais utilizadas de *rollback* é o *backup* das informações do estado anterior a execução de uma operação. Em caso de falha durante essa operação, o estado anterior é recuperado, utilizando-se o *backup* criado anteriormente. Utilizando essa técnica, o protocolo NETCONF [Enns *et al.* 2004], desenvolvido pela *Internet Enginnering Task Force* (IETF), incorporou a definição de transações em uma tarefa de configuração, evitando que dispositivos afetados evoluam para estados inconsistentes.

Contudo, essa abordagem se torna não realista no gerenciamento de infra-estruturas de TI, pois a relação de dependências entre serviços, aplicações e dispositivos é complexa. Em vista disso, a utilização de *rollback* em nível de rede é necessária. Desse modo, a coordenação dessas ações pode ser centralizada em um sistema específico que é conhecedor da infra-estrutura de TI como um todo. Com isso, é obtido um maior controle sobre as ações de *rollback*. Em um trabalho passado, foi proposto o sistema PBNM (*Policy-Based Network Management*) [Alves *et al.* 2006], onde falhas na implantação de políticas de QoS (*Quality of Service*) geram ações que retornam os dispositivos envolvidos para um estado anteriormente conhecido, utilizando uma versão adaptada do protocolo *two-phase commit*. Assim, o sistema PBNM tem o conhecimento da falha em um determinado ponto, e então relaciona os dispositivos inerentes a política para que as configurações dos mesmos sejam retrocedidas.

Para a utilização de *rollback* no *gerenciamento de mudanças* em infra-estruturas

de TI, foram propostos, em trabalhos anteriores deste grupo de pesquisa, mecanismos para (i) detecção de falhas durante a execução de mudanças [Machado *et al.* 2008b], e (ii) para a representação de reversibilidade de atividades [Machado *et al.* 2009]. Também, nessa mesma linha de pesquisa, foi proposto um algoritmo para a geração automática de planos de *rollback* [Machado *et al.* 2008a], o qual tem como base o processo de inversão do *change plan*, substituindo as atividades originais por atividades que desfazem as ações executadas anteriormente. Os planos de *rollback* gerados por esse algoritmo se mostraram eficazes, mas o número de atividades nos planos de *rollback* é alto, o que pode elevar os custos para a execução dos mesmos.

Candea *et al.* [Candea *et al.* 2004] propuseram o projeto *Recovery-Oriented Computing* (ROC), que foca na construção de sistemas com um rápido e alto poder de recuperação de falhas, ao invés de defender o processo de construção de sistemas imunes a qualquer tipo de erro. No projeto ROC, são discutidos vários tipos de técnicas para a recuperação de falhas, sendo que podemos encaixar *rollback em sistemas de gerenciamento de mudanças* como sendo um tipo de *system-level undo/redo*. Nessa classificação, um sistema controlador desfaz (ou refaz) ações em dispositivos, com base em ações que foram executadas anteriormente. Porém, como já mencionado nesta seção, o trabalho a ser apresentado neste artigo também leva em consideração dependências da infra-estrutura de TI para produzir um plano de *rollback*. Em termos de eficácia, Candea *et al.* mostra que técnicas como o *system-level undo/redo* solucionam uma grande quantidade de falhas, quase se equiparando com a recuperação de forma manual. Em termos de eficiência, essa técnica têm a tendência de também se aproximar ao modo de recuperação executada de forma manual, tornando-se uma abordagem custosa.

3. Solução para Suporte a Remediação

Nesta seção, descrevemos a solução para suporte a remediação em sistemas de gerenciamento de mudanças em infra-estruturas de TI. Primeiramente, apresentaremos uma arquitetura genérica de sistemas de gerenciamento de mudanças em TI, no qual o suporte a remediação é explorado. Em seguida, explicaremos a definição de reversibilidade em RFCs e como essa classificação é definida pelos administradores/operadores nas diferentes partes de uma RFC. Por fim, apresentamos o algoritmo para geração de planos de *rollback*.

3.1. Arquitetura Conceitual

Apesar de existirem diferentes arquiteturas para o gerenciamento de mudanças em infra-estruturas de TI, pode-se identificar um conjunto de componentes funcionais básicos, que, agrupados, formam uma arquitetura genérica. Em trabalhos anteriores [Machado *et al.* 2009], é proposta uma arquitetura com componentes especializados que oferecem o suporte a remediação. Assim, na Figura 1, é apresentada a arquitetura resultante, ressaltando os componentes necessários para o suporte à remediação.

Basicamente, a especificação de uma RFC começa quando o *change requester*, interagindo com o *change designer*, descreve suas necessidades em um documento de alto nível. O *change designer* é uma ferramenta que auxilia o *change requester* a preencher a RFC de forma clara e consistente. É importante lembrar que a RFC é um documento que descreve quais as mudanças são necessárias, mas não como essas devem ser implantadas.

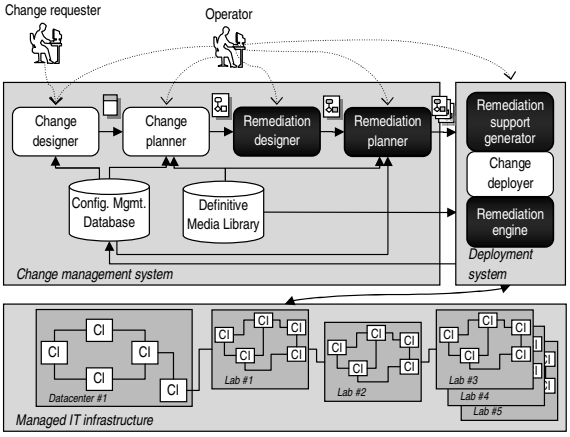


Figura 1. Arquitetura genérica para um sistema de gerenciamento de mudanças

Com a requisição preenchida, o operador define um *change plan* preliminar, contendo as macro-atividades necessárias para a implantação da mudança.

A saída do *change designer* é um *change plan* que necessita ser refinado. O *change planner* é responsável por esse refinamento, adicionando ao *workflow* as atividades executáveis necessárias para a implantação da mudança. O algoritmo responsável por esse refinamento está fora do escopo deste artigo, e já foi explicitado em outros trabalhos [Cordeiro *et al.* 2008]. Com o propósito de construir um *change plan* executável, o *change planner* precisa consultar duas bases de dados: o *Configuration Management Database* (CMDB) e a *Definitive Media Library* (DML). Para descobrir quais o elementos precisam ser manipulados, o *change planner* consulta o CMDB, que mantém as informações atualizadas sobre a infra-estrutura gerenciada. A DML é um repositório com as versões autorizadas de *software* e *hardware* para uso na infra-estrutura, bem como suas relações de interdependência. Por exemplo, o CMDB informa quais *softwares* estão instalados em um determinado servidor, e a DML informa, durante um processo de instalação, que uma aplicação *e-Commerce* necessita de um *Web server* e um banco de dados para funcionar de maneira correta.

Na próxima etapa, a RFC é enviada para o *remediation designer*, onde o operador/administrador determina quais atividades são reversíveis ou irreversíveis, de acordo com as características de cada atividade. É importante ressaltar que as ações descritas por atividades reversíveis podem ser desfeitas, de modo que a infra-estrutura retorne ao estado anterior à execução da atividade. As ações descritas por atividades irreversíveis não podem ser desfeitas e, por isso, a infra-estrutura não pode retornar ao estado anterior à mudança. Com o intuito de facilitar a geração de planos de remediação, as atividades reversíveis podem ser agrupadas, formando *reversible groups*. Na Seção 3.2, essa classificação é descrita com mais detalhes.

Na próxima etapa, o *change plan* é enviado ao *remediation planner*, o qual é responsável pela criação dos planos de remediação. Esses planos são baseados nas informações definidas pelo operador/administrador do sistema, que determina quais atividades são reversíveis ou irreversíveis, bem como os dados atualizados sobre a infra-estrutura, fornecidos pelo CMDB, e as informações sobre os itens que podem ser utilizados na infra-estrutura, fornecidas pela DML. Para a criação dos planos de *rollback*, é uti-

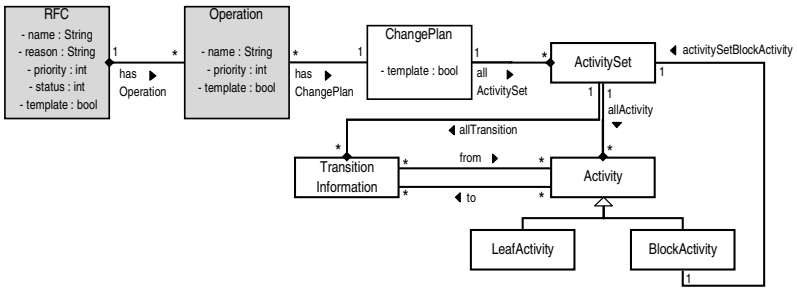


Figura 2. Modelo para representação da organização de uma RFC

lizado o algoritmo descrito na Seção 3.3. Esse algoritmo baseia-se nos *reversible groups*, além das informações citadas anteriormente. A geração de planos de compensação está fora do escopo deste trabalho.

Após a criação dos planos de remediação, a RFC é enviada ao *deployment system*, que é responsável pela implantação da mudança. Internamente, o *deployment system* é composto de três componentes: o *remediation support generator*, o *change deployer* e o *remediation engine*. O *remediation support generator* é responsável pela adição de estruturas para suportar ações de remediação durante a implantação da mudança. O *change deployer* tem o objetivo de implantar a mudança, gerenciando a execução do *change plan*. No caso de falha durante essa execução, o *remediation engine* é acionado e é sua responsabilidade a execução dos procedimentos de remediação, seguindo os planos consolidados anteriormente pelo *remediation planner*.

3.2. Classificando as RFCs

Como descrito nas seções anteriores, para utilizar os planos de remediação é necessário que o operador/administrador do sistema forneça as informações necessárias para que o *remediation planner* possa gerar esses planos. Essas informações definem qual o tipo de remediação é requerido pelas atividades de uma RFC. Para entender como essas informações são descritas, é necessário o entendimento da organização interna de uma *Request for Change*. Uma RFC é composta de uma ou mais operações. A operação é uma parte independente da mudança e, por isso, duas ou mais operações podem ser executadas em paralelo. Cada operação é associada a um único *change plan*, que é formado por um ou mais *activities sets*. Esses *activities sets* são utilizados para agrupar atividades encadeadas, formando um *workflow*, que representa o fluxo de atividades que devem ser executadas sobre a infra-estrutura. Na Figura 2, pode-se observar modelo que representa a organização descrita anteriormente.

Dessa forma, é possível notar que a hierarquia de uma *Request for Change* é representada dessa maneira: RFC, operações, *change plan*, *activities sets* e atividades. O operador pode determinar se cada umas dessas estruturas é reversível ou irreversível. Se a RFC for classificada como reversível, significa que são utilizados para a remediação planos de *rollback* e/ou planos de compensação, uma vez que as mudanças executadas sobre a infra-estrutura de TI podem ser desfeitas. Caso a RFC seja definida como irreversível, somente planos de compensação são utilizados como remediação, já que as mudanças executadas sobre a infra-estrutura não podem ser desfeitas. Quando o operador determina que a RFC é reversível, as operações também serão reversíveis, bem como os *change plans*, *activities sets* e as atividades pertencentes a essa RFC. No entanto, ao

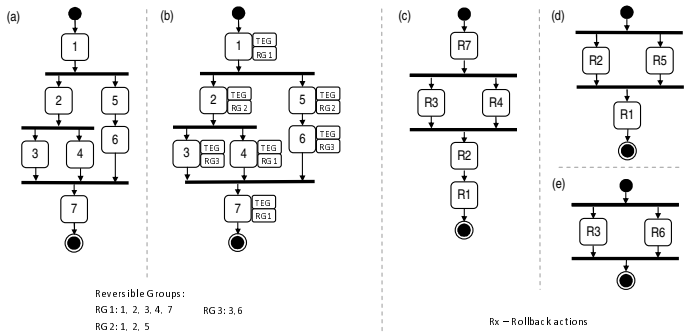


Figura 3. *change plan* com seus *reversible groups* e os respectivos planos de *rollback*.

classificar a RFC como irreversível, algumas estruturas internas podem ser classificadas como reversíveis. Por exemplo, uma RFC é composta de duas operações, denominadas OP1 e OP2. Se a operação OP1 for classificada como reversível e OP2, como irreversível, a mudança descrita na RFC será irreversível, visto que as ações irreversíveis contidas em OP2 são executadas durante a execução da RFC.

Em certos casos, o contexto da mudança necessita que partes diferentes do mesmo *change plan* tenham ações de remediação diferentes. Por exemplo, no *workflow* mostrado na Figura 3-a, o contexto da mudança pode requerer que, quando a atividade 4 falhar, sejam desfeitas as atividades 1, 2, 3 e 4. Já se uma falha ocorrer na atividade 5, é necessário o *rollback* das atividades 1, 2 e 5. Para suprir essa necessidade são utilizados os *reversible groups*. Em caso de falha de uma atividade pertencente a um *reversible group*, todas as outras atividades pertencentes a esse mesmo grupo serão desfeitas. Por exemplo, no *workflow* descrito na Figura 3-a, pode-se destacar três *reversible groups*: RG1, RG2 e RG3. Nesse caso, se a atividade 7 falhar, as atividades pertencentes à RG1, ou seja, as atividades 1, 2, 3, 4 e 7, serão desfeitas.

Para evitar o *rollback* em cascata [Machado *et al.* 2008b] quando a falha ocorre em uma atividade pertencente a mais de um *reversible group*, ou seja, o *rollback* de dois *reversible groups* diferentes que contém a mesma atividade, é utilizado o *Treatment Execution Group* (TEG). O TEG define qual o *reversible group* será utilizado em caso de falha na referida atividade. Na Figura 3-b, podemos ver os TEGs de cada atividade do *change plan*, bem como os *reversible groups* associados. Dessa forma, se uma falha acontecer na atividade 2, somente as atividades 1, 2 e 5 serão desfeitas, visto que essas atividades pertencem ao grupo RG2. Dessa forma, o plano de *rollback* para a atividade 2 é mostrado na Figura 3-d. Nas Figuras 3-c e 3-e, são mostrados o planos de *rollback* para as atividades 1, 4 e 7 e o plano para as atividades 3 e 6, respectivamente.

3.3. Algoritmo para a Geração de Planos de *Rollback*

Como descrito anteriormente, para a geração dos planos de *rollback*, o operador/administrador do sistema necessita definir quais atividades serão reversíveis ou irreversíveis, de acordo com as características particulares de cada uma, bem como definir, se necessário, os *reversible groups*. De posse dessas informações, o *remediation planner* pode gerar os planos de *rollback*. O primeiro passo dessa geração é a inversão do *change plan*. Essa inversão é feita utilizando a classe *TransitionInformation*, apresentada na Figura 2. Essa estrutura representa as transições de um *change plan*, tendo um atributo *from*

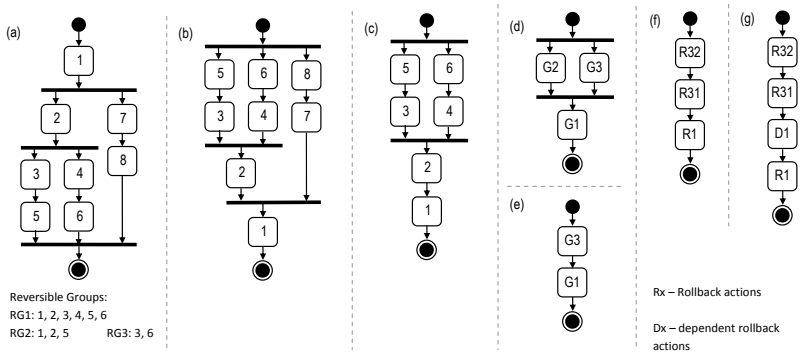


Figura 4. Passos para a criação dos planos de *rollback*

e um atributo *to*, que apontam, respectivamente, para as atividades origem e destino da transição. Desse modo, para que o algoritmo inverta o *workflow*, basta que sejam trocadas a origem e o destino de cada transição. Como exemplo, pode-se observar o *workflow* original, na Figura 4-a, e o *workflow* invertido, na Figura 4-b.

Após a inversão do *workflow* original, é feita a análise das atividades contidas no plano gerado, coletando as informações necessárias. Essas informações estão definidas nas estruturas reversíveis descritas na Seção 3.2. Essa análise tem o objetivo de remover atividades que (i) não são reversíveis, (ii) não pertencem ao *reversible group* ou (iii) são executadas, no *change plan*, após a atividade para qual o plano está sendo gerado. Dessa forma, as atividades que não são reversíveis são removidas do plano, uma vez que não possuem ações de *rollback*. As atividades que não pertencem ao *reversible group* que está sendo processado no momento são descartadas, de modo a manter o mecanismo de utilização desses grupos, descrito na Seção 3.2. Além disso, as atividades que são executadas após a atividade para qual o plano está sendo gerado também são removidas, já que, em caso de falha dessa atividade, suas ações ainda não terão sido executadas no *change plan*. Por exemplo, no *workflow* descrito na Figura 4-a, se a atividade 4 falhar, a atividade 6 ainda não terá sido executada, não sendo necessário, dessa forma, uma ação de *rollback* para a atividade 6. Essa identificação é feita de maneira recursiva, iniciando da atividade para qual o plano está sendo gerado, percorrendo as atividades até o final do *workflow*. Na Figura 4-c, pode-se observar o plano resultante dessa análise, considerando a geração das ações de *rollback* para a atividade 5. Nesse exemplo, nota-se que as atividades descartadas foram removidas do *workflow*, sem contudo, alterar o fluxo de execução. Para efetuar essa remoção, é necessário a identificação das atividades ligadas a atividade a ser removida, identificando também o sentido das ligações entre elas. Após, todas atividades que antecedem a atividade a ser removida são ligadas com todas aquelas que a sucedem. Ao utilizar essa remoção de atividades, o fluxo original do *workflow* é mantido.

Com o intuito de descrever as próximas etapas, é necessário o entendimento dos estados de cada elemento gerenciável. Cada elemento possui um conjunto de estados possíveis, que descreve o atual funcionamento do elemento gerenciado, como esse deve ser manipulado e os estados alcançáveis apartir do estado atual. Como esses estados descrevem o funcionamento dos elementos, eles influenciam os serviços prestados pelos mesmos. Para que ocorram transições entre os estados, é necessária a execução de uma sequência de ações sobre o elemento. As informações referentes ao estado atual de cada elemento devem estar populados no CMDB. Por exemplo, um software qualquer possui

4 estados bem definidos: *deployable*, *installable*, *executable* e *running*. Para oferecer um determinado serviço, o software necessita estar no estado *running*. Com o intuito de evoluir para esse estado, partindo do estado *installable*, esse elemento necessita passar pelo estado *executable*. Para que essa transição ocorra, o software precisa ser instalado em um sistema computacional. Essa ação pode ser dividida em vários passos, como o *download* do pacote de instalação, a instalação propriamente dita e a configuração da aplicação. Dessa forma, o software passa do estado *instalable* para o estado *executable*, onde pode ser executado no sistema computacional.

A próxima etapa consiste em agrupar as atividades do *change plan*, de forma a identificar as macro-atividades que as originaram durante o refinamento descrito na Seção 3.1. Para efetuar esse agrupamento, é necessário agrupar as ações executadas sobre cada elemento, de forma a identificar o objetivo das mudanças executadas sobre cada elemento, determinando os estados pelo qual o elemento evolui. Dessa forma, são criadas macro-atividades, com o intuito de desfazer as mudanças executadas sobre o elemento em questão. Por exemplo, na execução do *change plan*, um *web server* migra do estado *installable* para o estado *executable*. Durante a análise recém descrita, esses estados são identificados e uma macro-atividade visando a transição do estado *executable* para o *installable* é criada. Pode-se observar, na Figura 4-d, o *workflow* resultante do agrupamento de atividades do fluxo descrito na Figura 4-c. Nesse exemplo, as atividades 1 e 2 formam o grupo G1. Por sua vez, o grupo G2 é formado pelas atividades 3 e 5 e o grupo G3 é formado pelas atividades 4 e 6.

Após a identificação desses grupos, é realizada a identificação e remoção das atividades desnecessárias a execução do *rollback*. Para isso, é preciso a identificação de certas relações entre os elementos afetados pela mudança. Um exemplo desse tipo de relação é a aquela existente entre o elemento contido e o elemento recipiente. A premissa dessa interação é que os elementos contidos podem estar, no máximo, no mesmo estado dos elementos recipientes. Por exemplo, uma aplicação é instalada em um sistema operacional. Logo, o sistema operacional *contém* a aplicação. Desse modo, se o sistema operacional estiver no estado *running*, a aplicação poderá ou não estar no estado *running*. No entanto, se o sistema operacional estiver em *executable*, a aplicação, necessariamente, estará *executable*. Dessa forma, se for executada uma atividade para desligar o sistema operacional, não será necessária uma ação para desligar a aplicação, uma vez que essa ação será executada, de maneira indireta, ao se desativar o sistema operacional. O *workflow* resultante dessa análise pode ser observado na Figura 4-e.

Dessa forma, obtemos um plano de *rollback* preliminar, contendo as ações de *rollback* em alto-nível de abstração. Esse plano deve então ser refinado. Em trabalhos anteriores [Cordeiro *et al.* 2008], foi explicitado um algoritmo para tal refinamento. Com o intuito de refinar os planos de *rollback*, esse algoritmo, de forma recursiva, substitui os grupos de atividades por atividades executáveis. Ao realizar essa substituição, já são adicionadas as dependências das atividades contidas no plano gerado, de forma que o plano de *rollback* respeite as dependências entre as atividades que o compõe. Podemos observar, na Figura 4-g, o resultado da aplicação desse algoritmo.

4. Implementação e Protótipo

Para avaliar a solução proposta, foi desenvolvido um protótipo que implementa o suporte à remediação em gerenciamento de mudanças em infra-estruturas de TI. Nesse protótipo,

foi utilizado *Java 2 Standard Edition* (J2SE) para implementar os módulos descritos na Seção 3.1. Para representar as informações contidas no *Configuration Management Database* (CMDB) e na *Definitive Media Library* (DML) foi utilizado o *Common Information Model* (CIM) [DMTF 2008], que descreve os vários elementos de uma infra-estrutura de TI e suas diversas relações. Com o intuito de executar as ações sobre os elementos envolvidos em cada mudança, foi utilizado o padrão *Web services*, devido a três fatores: a facilidade de comunicação entre os processos, sua aceitação, tanto pelo meio acadêmico quanto pelo meio industrial, e a possibilidade de utilização de *Business Process Execution Language* (BPEL) [OASIS Standards 2007], utilizado para a orquestração de *Web services*. Para implantar as mudanças sobre a infra-estrutura de TI, o *change plan* é mapeado para um documento BPEL, que é executado por um BPEL *engine*. Nessa implementação, foi utilizado o ActiveBPEL [Active Endpoints 2007]. Por sua vez, no lado dos elementos gerenciáveis, assumimos que existe uma interface de gerenciamento através de *Web services*. Essa interface pode seguir as especificações do *Configuration Description, Deployment, LifeCycle Management* (CDDLM) [CDDLM Working Group 2007], por exemplo.

Primeiramente, o *remediation support generator* recebe um *change plan* com as definições de reversibilidade, explicadas na Seção 3.2, e os planos de remediação gerados pelo *remediation planner*. Baseando-se nas informações contidas no *change plan*, o *remediation support generator* identifica os elementos afetados pela mudança e faz uma consulta aos *Web services* desses elementos, com o intuito de verificar a descrição dos serviços disponibilizados por cada *Web service*. Esses serviços são descritos em documentos WSDL (*Web Service Description Language*). Após a análise dos documentos WSDL, o *change plan* é convertido para um documento BPEL. Nessa etapa, os planos de remediação também são traduzidos para documentos BPEL, de modo a serem utilizados pelo *remediation engine* em caso de falha. No documento BPEL descrevendo o *change plan*, o *remediation support generator* adiciona estruturas denominadas *fault handlers* [Machado *et al.* 2008b], com o intuito de detectar falhas ocorridas durante a implantação da mudança. Então, é gerado um documento, denominado *Process Deployment Descriptor* (PDD), necessário para a execução do *workflow* pelo ActiveBPEL. Dessa forma, os documentos gerados seguem para o *change deployer*.

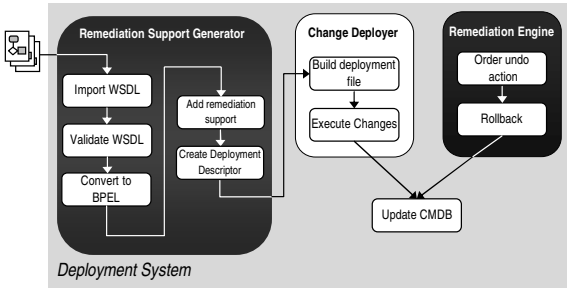


Figura 5. Arquitetura interna do *deployment system*

O *change deployer* expande as funcionalidade do BPEL *engine*, com o intuito de adaptar as necessidades do projeto em relação ao *deployment system*. Primeiramente, o *change deployer* gera um arquivo contendo todos os documentos necessários para a execução do BPEL (WSDLs, BPEL, PDD) e, então, o ActiveBPEL é invocado para executar o *workflow*. No protótipo desenvolvido, o ActiveBPEL é responsável pela execução do *change plan*, a detecção das falhas e a execução das ações de remediação.

Em caso de falhas, o *remediation engine* é invocado. Esse componente é responsável pela execução das ações de remediação definidas pelo *remediation planner*. Nessa implementação, os planos de remediação são traduzidos para documentos BPEL e são executados pelo ActiveBPEL. Para cada atividade executada sobre um elemento gerenciado, o CMDB, descrito na Seção 3.1, é atualizado, afim de manter essa base de dados coerente com o estado atual da infra-estrutura de TI.

5. Avaliação

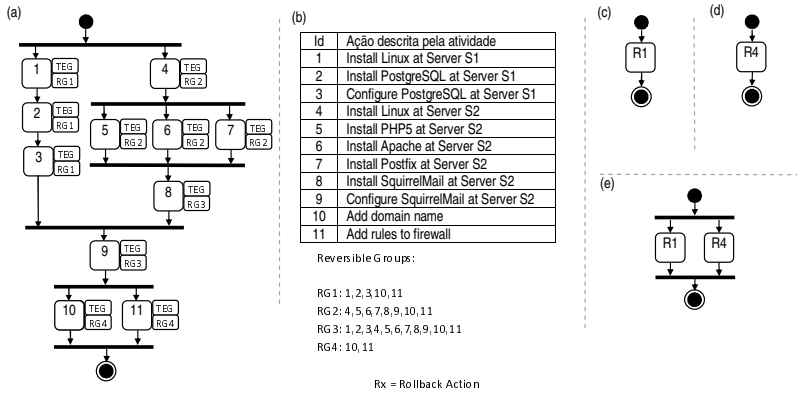
Com o intuito de avaliar a solução proposta, bem como testar sua viabilidade técnica, foi elaborado um experimento com base em situações reais. A descrição do cenário e da requisição de mudança é apresentada na Seção 5.1. A análise da solução proposta, por sua vez, é descrita na Seção 5.2.

5.1. Cenário

Em uma reunião administrativa de uma companhia que tem como negócio fornecer conexões dedicadas para o acesso à Internet, foi mostrado que há uma acentuada perda de clientes durante o primeiro semestre do ano. Em uma pesquisa encomendada, foi identificado que os clientes preferem a concorrente que fornece outros serviços além da conexão para o acesso à internet. Esses serviços adicionais variam desde conteúdo exclusivo até contas de *webmail*. Na tentativa de solucionar o problema identificado, foi decidido que, em uma jogada de *marketing*, a companhia lançará um serviço de *webmail*. O acesso ao serviço não só irá se conter aos clientes da companhia, mas também se expandirá ao registro de contas gratuitas.

Hoje em dia a infra-estrutura da empresa conta com diversos elementos, dentre eles roteadores de borda, um servidor DNS, e um *firewall* dedicado. Essa infra-estrutura é responsável por atender perfeitamente as requisições feitas pelos usuários, também respeitando os termos descritos pelos contratos de SLA (*Service Level Agreement*) das conexões cedidas pela companhia. Para o lançamento do serviço de *webmail*, mudanças na infra-estrutura gerenciada são requeridas, de modo que (i) o novo serviço de *webmail* seja disponibilizado e (ii) os termos descritos pelo SLA continuem sendo respeitados. Para que esses dois objetivos sejam alcançados, foram alocados dois novos servidores equipados com processadores *Quad Core Xeon* e 8 GiB de RAM. Um desses servidores será utilizado para armazenamento de dados, como as informações referentes a cada conta de e-mail, enquanto o outro será utilizado como servidor Web, disponibilizando o serviço de *webmail* através da rede. Tendo em vista esses componentes e a infra-estrutura já disponibilizada pela empresa, as mudanças necessárias são: a instalação/configuração do servidor de dados (servidor S1) e do servidor Web (servidor S2), de forma a disponibilizar o novo serviço; e a configuração do servidor DNS juntamente com a configuração do *firewall*, de modo a possibilitar o acesso dos clientes ao mesmo.

Para que essa mudança seja implantada, foi criada uma RFC composta de uma única operação, cujo *change plan* é descrito na Figura 6-a. As ações de cada atividade do *change plan* estão descritas na tabela da Figura 6-b. Os *reversible groups* utilizados foram criados de forma que atividades que falhassem em um determinado elemento não influenciassem as atividades executadas corretamente sobre outros elementos. Desse modo, foram criados *reversible groups* com as atividades referentes a cada servidor: RG1, contendo as atividades do servidor S1, e RG2, contendo as atividades do servidor S2. Como



Id	Ação descrita pela atividade
1	Install Linux at Server S1
2	Install PostgreSQL at Server S1
3	Configure PostgreSQL at Server S1
4	Install Linux at Server S2
5	Install PHP5 at Server S2
6	Install Apache at Server S2
7	Install Postfix at Server S2
8	Install SquirrelMail at Server S2
9	Configure SquirrelMail at Server S2
10	Add domain name
11	Add rules to firewall

Figura 6. *Change plan* para a publicação de um serviço de *webmail* (a,b) e os planos de *rollback* para os *reversible groups* RG1 (c), RG2 (d), RG3 (e) e RG4 (f).

as atividades de instalação e configuração do *webmail* (representados pelos identificadores 8 e 9) são críticas para o processo de mudança, foi decidido usar um terceiro *reversible group* (RG3). Esse fato ocorre pois falhas nessas atividades podem afetar algum outro elemento instalado anteriormente, comprometendo a mudança como um todo. Ainda, com o intuito de evitar a disponibilização do serviço para o acesso dos clientes em caso de falha durante a execução do *change plan*, as atividades 10 e 11 pertencem a todos os *reversible groups*. Para evitar que falhas nessas atividades desencadeassem *rollback* das atividades executadas sobre o servidor de dados e o servidor Web, as atividades que configuram os itens de rede formam um grupo a parte, denominado RG4.

5.2. Análise

Para avaliar a eficácia e eficiência dos planos de *rollback* gerados pelo algoritmo proposto, foram criadas quatro máquinas virtuais, utilizando a ferramenta *open-source* de virtualização *Virtual Box*. Cada uma dessas máquinas foi devidamente configurada, de modo a emular a infra-estrutura descrita na seção anterior. As atividades de instalação de sistemas operacionais foram executadas automaticamente, pois existia uma imagem pré-instalada do sistema operacional utilizado. Foram configurados *Web Services*, de modo que as atividades descritas pelo *change plan* pudessem ser executadas automaticamente com a utilização de documentos BPEL. Dessa forma, foi possível a avaliação da solução proposta, analisando parâmetros como a quantidade de atividades e o tempo de execução dos planos de *rollback* gerados. Com a obtenção desses parâmetros, é possível a comparação com as medidas obtidas em testes utilizando a abordagem descrita por Machado *et al.* [Machado *et al.* 2008a].

O *change plan* descrito foi executado quatro vezes, de forma que se pudesse avaliar sua execução e a dos planos de *rollback*. Na primeira, não foi injetada nenhuma falha, de modo a verificar se a mudança seria implantada de maneira consistente. Nas outras execuções foram injetadas falhas, para que fosse possível avaliar os planos de *rollback*. Para analisar os planos gerados, foram injetadas falhas em atividades pertencente a cada *reversible group*. A fim de testar o *reversible group* RG1, foi injetada uma falha na atividade 2, cuja ação é instalar o *PostgreSQL server* no servidor S1. O plano de *rollback* gerado está descrito na Figura 6-c, onde R1 é a ação de *rollback* para a atividade 1, ou seja, desinstalar o *Linux* do servidor S1. Com o intuito de testar o plano gerado para o

grupo RG2, descrito na Figura 6-d, foi injetado uma falha na atividade 6, cuja ação é instalar o *Apache* no servidor S2. O plano de *rollback* utilizado contém apenas a atividade relativa a desinstalação do sistema operacional *Linux* no servidor S2 (atividade 4), simbolizada pela atividade R4. Em outro teste, foram injetadas falhas na atividade 9, de forma que o plano gerado descrito na Figura 6-e fosse executado. Assim, esse último plano de *rollback* contém as atividades para a desinstalação do *Linux* nos servidores S1 e S2.

Tabela 1. Comparação entre a abordagem proposta (Solução 1) e a abordagem anterior (Solução 2)

Exemplos	Tempos de execução (min)		Número de atividades	
	Solução 1	Solução 2	Solução 1	Solução 2
RG1	11	15	1	2
RG2	13	23	1	4
RG3	14	28	2	9

Na Tabela 1, mostra-se o tempo de execução e o número de atividades dos planos de *rollback* gerados pela solução proposta (Solução 1) e dos planos gerados pela abordagem descrita por Machado *et al.* [Machado *et al.* 2008a] em cada um dos exemplos citados anteriormente. Apesar das duas soluções gerarem planos eficazes, que retornaram a infra-estrutura ao estado anterior à execução da mudança, o método proposto gerou planos menores, contendo um menor número de atividades, e foi executado em menor tempo. Dessa forma, os recursos computacionais foram melhor utilizados, uma vez que foram usados por menos tempo e em menor quantidade. Por exemplo, utilizando o grupo RG3, o plano gerado pela Solução 1 possui 9 atividades e foi executado em 28 minutos, enquanto o plano gerado pelo método proposto possui apenas 2 atividades e foi executado em 14 minutos. Portanto, houve uma economia de 14 minutos, diminuindo o uso dos recursos computacionais em 50% do tempo.

6. Conclusão

Neste artigo, discutiu-se como organizações implantam mudanças em suas infra-estruturas de TI, utilizando, para isso, sistemas de gerenciamento de mudanças, bem como a importância do suporte a remediação nesses sistemas. Devido a complexidade intrínseca das infra-estruturas de TI atuais, as mudanças tendem a envolver uma quantidade maior de elementos gerenciáveis, aumentando, dessa forma, a probabilidade de falhas. Portanto, com base em trabalhos passados desenvolvidos por esse grupo de pesquisa [Machado *et al.* 2009] [Machado *et al.* 2008a], foi desenvolvido um algoritmo, descrito neste trabalho, para a geração de planos de *rollback* otimizados. Com o intuito de gerar esses planos, o operador/administrador do sistema deve determinar quais atividades são reversíveis ou irreversíveis, conforme a remediação necessária para cada ação. Caso alguma falha ocorra durante a execução de uma atividade reversível, ações de *rollback* serão executadas para retornar a infra-estrutura para o estado anterior à execução da mudança.

Os resultados obtidos demonstraram que os planos de *rollback* gerados pelo algoritmo proposto se mostraram mais eficientes que abordagens anteriores, diminuindo o número de atividades necessários para a execução do *rollback*. Com essa diminuição, os planos gerados necessitaram de menos tempo para retornar a infra-estrutura para um estado consistente e utilizaram os recursos computacionais de maneira mais racional, quando comparados a planos gerados utilizando abordagens propostas anteriormente.

Apesar disso, detectamos que possíveis otimizações são aplicáveis, e servem como trabalhos futuros. Uma dessas otimizações é a adequação dos planos de *rollback* aos recursos (*e.g*, humanos ou recursos computacionais) disponíveis para a execução da RFC, de forma a diminuir o tempo de retorno da infra-estrutura ao estado consistente anterior a execução da mudança.

Referências

Active Endpoints (2007). *ActiveBPEL Open Source Engine*. http://www.activebpel.org.

Alves, R., Granville, L., Almeida, M., e Tarouco, L. (2006). A Protocol for Atomic Deployment of Management Policies on QoS-Enabled Networks. In *6th IEEE International Workshop on Ip Operations and Management (IPOM 2006)*.

Candea, G., Brown, A. B., Fox, A., e Patterson, D. (2004). Recovery-oriented computing: Building multitier dependability. *Computer*, 37(11):60–67.

CDDLM Working Group (2007). *Configuration Description, Deployment, and Lifecycle Management*. http://forge.gridforum.org/projects/cddlm-wg.

Cordeiro, W., Machado, G., Andreis, F., Santos, A., Both, C., Gaspary, L., Granville, L., Bartolini, C., e Trastour, D. (2008). A Runtime Constraint-Aware Solution for Automated Refinement of IT Change Plans. In *Proceedings of 19th Ifip/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2008)*.

DMTF (2008). *Common Information Model*. DMTF.

Enns, R. *et al.* (2004). NETCONF Configuration Protocol.

ITIL (2007). *ITIL - Information Technology Infrastructure Library: Service Transition Version 3.0*. Office of Government Commerce (OGC).

ITIL (2008). ITIL - Information Technology Infrastructure Library (ITIL). http://www.itil-officialsite.com/.

Machado, G., da Costa Cordeiro, W., dos Santos, A., Both, C., Gaspary, L., Granville, L., Bartolini, C., Sahai, A., Trastour, D., e Saikoski, K. (2008a). Algoritmo para geração automática de ações de rollback em sistemas de gerenciamento de mudanças em TI. In *Brazilian Symposium on Computer Networks and Distributed Systems (SBRC 2008)*.

Machado, G., da Costa Cordeiro, W., dos Santos, A., Wickboldt, J., Castagna, R., Andreis, F., Both, C., Gaspary, L., Granville, L., Bartolini, C., e Trastour, D. (2009). Refined Failure Remediation for IT Change Management Systems. In *Proceedings of Miniconference of 11th IFIP/IEEE International Symposium on Integrated Network Management (IM 2009)*.

Machado, G., Daitx, F., Cordeiro, W., Both, C., Gaspary, L., Granville, L., Bartolini, C., Sahai, A., Trastour, D., e Saikoski, K. (2008b). Enabling rollback support in IT change management systems. In *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, pages 347–354.

OASIS Standards (2007). *Business Process Execution Language*.