ISRE: PLATAFORMA DE DEFINIÇÃO E EXECUÇÃO DE PROCEDIMENTOS DE SUPORTE EM SISTEMAS EM REDE

ACHILES CALDEIRA QUINTELLA SANTANA JUNIOR

ISRE: PLATAFORMA DE DEFINIÇÃO E EXECUÇÃO DE PROCEDIMENTOS DE SUPORTE EM SISTEMAS EM REDE

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Renato Antônio Celso Ferreira

Belo Horizonte

Dezembro de 2017

© 2017, Achiles Caldeira Quintella Santana Junior.
Todos os direitos reservados

.

Ficha catalográfica elaborada pela Biblioteca do ICEx - UFMG

Santana Junior, Achiles Caldeira Quintella.

S232i iSRE: plataforma de definição e execução de procedimentos de suporte em sistemas em rede ./ Achiles Caldeira Quintella Santana Junior. – Belo Horizonte, 2017.
xxiii, 83 f.: il.; 29 cm.

Dissertação (mestrado) - Universidade Federal de Minas Gerais – Departamento de Ciência da Computação.

Orientador: Renato Antônio Celso Ferreira

1. Computação – Teses. 2. Computação autônoma. 3. Gerenciamento de sistemas em rede. 4. Sistemas operacionais distribuídos (Computadores). I. Orientador. II. Título.

CDU 519.6*34(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

iSRE: PLATAFORMA DE DEFINIÇÃO E EXECUÇÃO DE PROCEDIMENTOS DE SUPORTE EM SISTEMAS EM REDE

ACHILES CALDEIRA QUINTELLA SANTANA JUNIOR

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

Prof. Renato Antônio Celso Ferreira - Orientador
Departamento de Ciência da Computação - UFMG

Prof. Adriano Alonso Veloso
Departamento de Ciência da Computação - UFMG

Prof. Dorgival Olavo Guedes Neto
Departamento de Ciência da Computação - UFMG

Sr. Wilton C. Padrão
Diretoria Técnica - Engetron

Belo Horizonte, 14 de Dezembro de 2017.

Dedico esse trabalho aos meus pais, pelo suporte incondicional nessa e em diversas outras iniciativas.

Agradecimentos

Ao Professor Renato, que orientou o desenvolvimento desse trabalho, compartilhando ideias, experiências e conselhos, primordiais nessa caminhada. Aos membros da banca, Adriano, Dorgival e Padrão, por aceitarem o desafio de avaliar esse trabalho e contribuir com recomendações que o tornaram melhor. Agradeço aos demais professores e funcionários do DCC, que sempre estiveram disponíveis para auxiliar em demandas relacionadas ao curso. Por fim, agradeço a minha família e a minha namorada, pelo amor e carinho dedicados durante essa trajetória.

"Ticking away the moments that make up a dull day
You fritter and waste the hours in an offhand way
Kicking around on a piece of ground in your home town
Waiting for someone or something to show you the way..."
(Pink Floyd)

Resumo

Problemas ocorridos em componentes de um sistema em rede podem causar interrupção de serviços, elevar o custo de operação e provocar prejuízos materiais e imateriais as organizações. Uma opção para minimizar o impacto de problemas consiste em adicionar mais pessoas para realizar a operação, manutenção e evolução dos componentes do sistema. Porém, essa abordagem pode ser financeiramente inviável, dentre outras coisas, em função dos custos relacionados a utilização recursos humanos. Outra opção que pode minimizar esse tipo de impacto consiste em adotar e integrar ao sistema elementos de automação que minimizem a necessidade de intervenção manual. Nesse contexto, podemos destacar duas abordagens, que não são mutuamente exclusivas, a adoção de elementos de computação autonômica e/ou *robotic process automation (RPA)*. A utilização dessas abordagens tende a ser mais escalável, centralizada, causar menor quantidade de erros por execução e oferecer maiores oportunidades de padronização quando comparadas as soluções que envolvem o aumento de equipe para suporte a operação, manutenção e evolução de um sistema em rede.

Nessa direção, observamos um conjunto de limitações nas ferramentas e trabalhos disponíveis que contemplam alguns dos princípios característicos de elementos de computação autonômica e *RPA*. Nesse trabalho apresentamos a plataforma *iService-ReliabilityEngineer (iSRE)*, na qual projetamos e implementamos, por meio de um protótipo, os mecanismos que viabilizam de forma prática algumas das características das abordagens de computação autonômica e *RPA*, sendo avaliada por meio de experimentos. Também apresentamos uma avaliação qualitativa a respeito da relevância do tema e do interesse de profissionais que atuam em atividades relacionadas a gerenciamento de sistemas em rede.

Palavras-chave: Sistemas em Rede, Auto-Gerenciamento de Sistemas de Computação, Computação Autônoma, Gerenciamento de Sistemas em Rede, Sistemas Distribuídos.

Abstract

Problems that occur in components of a network system may cause service disruption, elevate the operating costs and contribute to material and imaterial damages to organizations. One option to minimize the impact of problems on the components of the system is to add more people able to perform tasks that enable the operation, maintenance and update of those components. But, this approach can be financially not feasible, besides other facts, due to cost associated with people. Another option that can minimize the impact is to adopt and integrate to the system automation elements capable of minimize manual intervention. In this context, we can highlight two approaches, which are not mutually exclusive, being the adoption of autonomic computing elements and robotic process automation (RPA). These approaches tend to be more scalable, centralized, less error prone and to offer more opportunities to standardize process if compared to solutions that rely solely on the increase of human resources to support the operation, maintance and upgrade of the system.

On that direction, we verified a set o limitation on the tools and available related work that contemplate some of the principles that characterize autonomic computing and robotic process automation elements. In this work, we present the iServiceReliabilityEngineer (iSRE) platform, on which we design and implement, via a prototype, the mechanisms that make feasible, in a practical way, some characteristics found in autonomic computing and RPA. That prototype is evaluate through a series of experiments. We also show a qualitative analysis that highlights the theme relevance and the interest by profissionals that work with activities related to network and distributed systems management.

Keywords: Network Systems, Self-Management Computer Systems, Network Systems Management, Distributed Systems.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

No passado, somente computadores de grande porte e verticalmente integrados, denominados sistemas centralizados, eram capazes de prover serviços computacionais em escala. A partir anos 80, em função da redução dos custos e aumento no poder de processamento dos microprocessadores e a introdução das redes de alta velocidade, foi facilitada a criação de sistemas de computação compostos por um grande número de computadores conectados por redes de alta velocidade [1]. Esses sistemas podem ser denominados sistemas em rede ou sistemas distribuídos.

O advento dos serviços de *Cloud Computing*, que fazem uso intensivo de sistemas em rede, tem permitido às organizações eliminar a necessidade de execução de investimentos antecipados em *hardware* para fazer uso de capacidade computacional em escala [2]. Esse modelo de fornecimento de serviços de computação torna mais acessível o consumo de capacidade computacional e de armazenamento e permite, de maneira simples, interconectar um número crescente de componentes heterogêneos que visam proporcionar a solução para determinado problema. Além disso, possibilita a alocação de recursos de forma flexível e em resposta a demanda. Por exemplo, o provisionamento de milhares de nós de computação para se efetuar processamento paralelo em uma massa de dados pode ser feito com apenas alguns cliques.

À medida que os sistemas ficam mais interconectados e diversificados, a antecipação e tratamento em fase de projeto de eventuais problemas de tempo de execução fica mais difícil [19]. Problemas de tempo de execução não antecipados na fase de especificação do projeto podem causar interrupção de serviços, elevar o custo de operação e provocar prejuízos materiais e imateriais.

Uma opção para minimizar o impacto de potenciais problemas de tempo de execução nos componentes de um sistema em rede consiste em adicionar mais pessoas para realizar a operação, manutenção e evolução dos componentes do sistema à medida que

o número de componentes ou a carga de trabalho cresce. Ao se adotar essa opção, o custo associado as tarefas de operação, manutenção e evolução dos componentes do sistema pode torná-lo financeiramente inviável em função, por exemplo, dos gastos associados a manutenção de pessoal. Além disso, não são eliminados potenciais problemas ocasionados como resultado da execução das atividades por parte da equipe, mais suscetível a erros do que soluções automáticas.

Outra opção para minimizar o impacto de potenciais problemas de tempo de execução nos componentes de um sistema em rede consiste em adotar e integrar ao sistema elementos de automação que minimizem a necessidade de intervenção manual para realização da operação, manutenção e evolução desses componentes. Nesse contexto, podemos destacar duas abordagens que não são mutuamente exclusivas, a adoção de elementos de auto-gerenciamento de sistemas [15] e/ou *robotic process automation (RPA)* [33].

Em essência, os elementos de auto-gerenciamento de sistemas consistem em um ou mais componentes provendo capacidades de execução autônoma para um ambiente de sistemas de computação [15]. Um dos objetivos ao se adotar esse tipo de componente em um ambiente de sistemas em rede é o de liberar os administradores de sistemas de algumas das tarefas relacionadas a operação e manutenção, sem comprometer requisitos e níveis de serviço acordados [19].

O conceito de *RPA* pode ser entendido como a aplicação no negócio de um ou mais robôs em software, configurados para executar tarefas previamente executadas por pessoas, interagindo com um conjunto de componentes de um sistema de maneira análoga a que as pessoas interagiriam [33]. Essa é uma abordagem de foco industrial e está se expandindo [33]. Algumas das características associadas a processos que são automatizados por meio de *RPA* consistem na existência de um processo manual em que uma pessoa recebe como entrada um conjunto de dados por meio de um sistema, por exemplo um evento de um sistema de monitoração, processa essa entrada usando um conjunto de regras, por exemplo um manual com uma lista de comandos a se executar, e por fim atualiza um registro com o resultado do processamento executado, por exemplo, envia um email com a lista de comandos executados em um componente da rede e o respectivo resultado. Pressupõe-se que toda a interação dos robôs de *RPA* com os componentes de um sistema em rede será efetuada por meio da camada de apresentação, não exigindo mudanças estruturais nesses componentes.

A utilização dessas abordagens tende a ser mais escalável, centralizada, causar menor quantidade de erros por execução e oferecer maiores oportunidades de padronização quando comparadas as soluções que envolvem o aumento de equipe para suporte a operação, manutenção e evolução de um sistema em rede. Além disso, alinhadas

as ferramentas disponibilizadas por meio dos avanços na área de inteligência artificial, apresentam potencial de enriquecer ainda mais a experiência dos usuários, por exemplo, suportando um elemento que aprende com as interações e dispensa a necessidade de configuração manual.

Nessa direção, observamos também um conjunto de limitações nas ferramentas e trabalhos disponíveis que contemplam alguns dos princípios característicos de elementos de auto-gerenciamento de sistemas de computação e *RPA* e que possuem enfoque no suporte a execução de procedimentos como forma de resposta a eventos que indicam mudanças de estado em um ou mais componentes de um sistema em rede com capacidade de causar problemas em tempo de execução nos referidos componentes. Essas ferramentas se encaixam em um conjunto de instrumentos que apoiam atividades de operação, manutenção e evolução de sistemas em rede e algumas das limitações observadas são: capacidade restrita de interoperabilidade com outros sistemas de identificação de mudanças de estado em componentes de um sistema em rede; mecanismos incipientes de exploração dos dados e contexto do conteúdo dos eventos; funcionalidades que permitem o reuso de procedimentos são escassas; por vezes, a distribuição dos procedimentos deve ser feita dispositivo por dispositivo.

Com base no cenário observado, entendemos que a tarefa de oferecer um conjunto de mecanismos que possibilitem a execução automática de procedimentos que visam diagnosticar, prevenir, mitigar e/ou sanar problemas de tempo de execução em componentes de um sistema em rede apresenta oportunidades e desafios.

Nesse trabalho apresentamos a plataforma experimental *iServiceReliabilityEngineer (iSRE)*, onde projetamos e implementamos, por meio de um protótipo, os mecanismos que permitem experimentar de forma prática algumas das características de modelos de auto-gerenciamento de sistemas em computação e *RPA*. O iSRE propõe e implementa mecanismos que permitem a definição de procedimentos, que são executados de forma controlada, visando validar, prevenir, mitigar e/ou sanar problemas de tempo de execução em componentes de um sistema em rede. Por meio da utilização do protótipo experimental do iSRE é possível, de forma automática, responder a eventos ocorridos como parte da operação de um sistema em rede.

1.1 Objetivos

O objetivo geral desse trabalho é projetar, implementar e avaliar uma plataforma experimental que forneça mecanismos capazes de suportar a execução automática e controlada de procedimentos de suporte em um ou mais componentes de um sistema

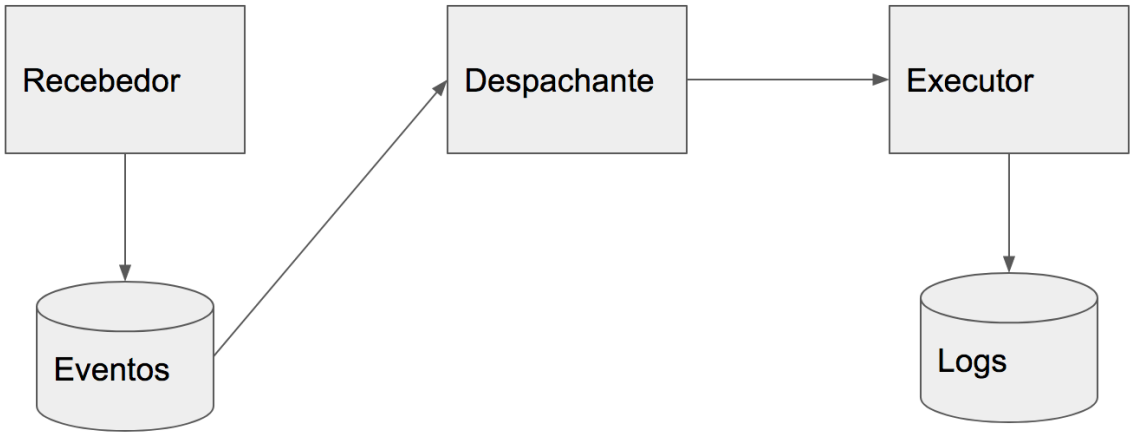

Figura 1.1: Arquitetura proposta

em rede. Alguns dos objetivos específicos são descritos a seguir nessa seção.

A plataforma deve atuar como um componente membro de um sistema em rede, sendo capaz de receber dados associados a eventos que indicam mudança de estado em um ou mais componentes desse sistema, e que são gerados com o objetivo de agregar valor as atividades de operação, manutenção e evolução. Ao receber esses eventos, a plataforma deve processá-los e analisar a aplicabilidade de se executar um ou mais procedimentos, definidos pelos usuários, como forma de reação. Caso identifique-se a necessidade de execução de um procedimento em resposta a ocorrência de um evento, a plataforma deve executar o procedimento de forma controlada. Para cumprir com esses objetivos, projetamos a arquitetura ilustrada na figura 1.1.

Uma interface de programação deve ser oferecida aos usuários, por exemplo administradores de sistema, para que eles sejam capazes de definir políticas de resposta a eventos, ocorridos nos componentes do sistema em rede.

Avaliamos a plataforma por meio da execução de experimentos, compostos por testes funcionais e não funcionais, utilizando o protótipo implementado como parte desse trabalho. Também foi efetuada uma avaliação qualitativa junto a um grupo de profissionais, com experiência na indústria, a respeito da relevância do tema e do interesse em ferramentas e iniciativas que se propõem a suportar a execução automática e controlada de procedimentos de suporte a eventos em um ambiente de sistemas em rede.

1.2 Trabalhos Relacionados

No que diz respeito a ferramentas utilizadas na operação de ambientes de sistemas em rede, podemos observar que algumas ferramentas de monitoração, com propósito genérico e que surgiram nos anos 2000 como Nagios [9], Ganglia [21], Zabbix [23] e [6] continuam sendo desenvolvidas e utilizadas para se efetuar a monitoração do estado de alguns parâmetros dos componentes administrados em determinado ambiente de sistema em rede [12]. Também podemos observar a adoção de ferramentas de monitoração focadas em ambientes de nuvem, como Amazon CloudWatch [29]. Essas ferramentas oferecem uma série recursos para monitoração de parâmetros inerentes aos componentes de um sistema em rede, mas possuem funcionalidades limitadas para execução automática de procedimentos como forma de reação a ocorrência de eventos que alteram o estado de um ou mais componentes e podem representar um potencial problema de tempo de execução.

As ferramentas Nagios[9] e Zabbix[23] possuem funcionalidades que permitem executar de forma automática procedimentos em resposta à ocorrência de eventos que alteram o estado de um ou mais componentes de um sistema em rede. Porém, ambas ferramentas não apresentam funcionalidades nativas que permitem a execução de procedimentos como forma de resposta a eventos gerados em outras ferramentas de monitoração, limitando a interoperabilidade. Além disso, não permitem a extração de conteúdo dos dados listados no evento, inibindo potencial utilização dinâmica desse conteúdo no contexto de execução dos procedimentos, o que reduz as possibilidades de reuso de procedimentos definidos pelo usuário.

Em [8], é apresentado um arcabouço chamado SHõWA, que propõe um elemento de computação autonômica especialista em aplicações web. Por meio de um conjunto de módulos, o SHõWA se propõe a executar a monitoração, análise e detecção de anomalias na aplicação ao qual está integrado. O SHõWA utiliza o modelo agente e servidor, em que um agente é instalado nos nós da aplicação web e se encarrega de coletar dados, que são posteriormente analisados utilizando correlação estatística. O objetivo da análise é identificar anomalias de desempenho e potenciais falhas em tempo de execução. Caso anomalias sejam identificadas, o SHõWA executa procedimentos de ajuste, pré configurados pelos usuários, que visam sanar a anomalia identificada.

Com foco em remediar falhas que surgem em tempo de execução em aplicações desenvolvidas no padrão *J2EE*, um protótipo que implementa um projeto de sistema de auto-remediação de falhas é proposto pelos autores em [4]. O sistema é caracterizado pela adoção e implementação de um loop de controle para gerenciamento do processo de remediação de falhas. Os componentes que integram a solução são o sistema ge-

renciado, sensores que monitoram o sistema gerenciado e mecanismos atuadores que executam ações que visam a correção de falhas de tempo de execução. Além disso, um componente controlador se encarrega de analisar os dados coletados pelos sensores e executar políticas de ajuste de configuração no sistema gerenciado por meio dos atuadores. Adicionalmente, o componente controlador possui algumas funcionalidades que têm como objetivo manter a sua alta disponibilidade.

Os trabalhos apresentados em [8] e [4], sintetizados previamente nesta dissertação, descrevem soluções onde o módulo encarregado de monitorar os componentes alvo é fortemente acoplado ao módulo que efetua a análise do conteúdo dos eventos, que contém dados a respeito da alteração de estado nos componentes alvo, e ao módulo que se encarrega de executar a validação e alteração de estado nos componentes alvo como forma de reação ao evento analisado. Essa abordagem restringe o potencial de interoperabilidade com uma série de ferramentas de monitoração e predição de eventos que estão disponíveis na academia e no mercado.

Outros trabalhos relacionados a sistemas de automação e reação a eventos são apresentados em [7], [31] e [3]. Uma lista mais completa pode ser encontrada em [16].

1.3 Contribuições

Esse trabalho apresenta cinco principais contribuições, sendo:

- definição e implementação de um sistema capaz de receber dados gerados em um ambiente de sistemas em rede, coletados a partir de diferentes fontes de monitoração e predição de estado, que aplica casamento de padrão sobre esse conteúdo e identifica a necessidade de verificação ou execução de alteração de estado nos componentes de um sistema em rede;

- implementação de um método capaz de interpretar e executar comandos com base em políticas de resposta a eventos, definidas pelos usuários, e armazenadas de forma centralizada;

- definição e implementação de uma interface de programação *(API)* que permite aos usuários definir políticas de resposta à eventos ocorridos em sistemas em rede;

- avaliação experimental dos componentes implementados;

- avaliação qualitativa junto a um grupo de profissionais a respeito da relevância do tema e do interesse em ferramentas e iniciativas que se propõem a suportar a

execução automática e controlada de procedimentos de suporte a eventos em um ambiente de sistemas em rede.

1.3.1 Organização

O conteúdo restante desse trabalho está organizado da seguinte maneira: o Capítulo 2 apresenta alguns conceitos que se relacionam com princípios e práticas exploradas no iSRE, tais como sistemas distribuídos, auto-gerenciamento de sistemas de computação e comunicação em sistemas de computação; o Capítulo 3 apresenta uma visão geral sobre a arquitetura do iSRE, descreve os principais requisitos atendidos e componentes propostos, detalha a API de programação da plataforma e ilustra a utilização com dois exemplos; o Capítulo 4 apresenta a metodologia utilizada para efetuarmos a avaliação qualitativa sobre a percepção de profissionais a respeito de soluções que propõem suportar a execução automática e controlada de procedimentos de suporte a eventos em um ambiente de sistemas em rede, juntamente dos resultados obtidos; o Capítulo 5 apresenta detalhes a respeito dos principais componentes do iSRE e de implementação da plataforma; o Capítulo 6 apresenta a metodologia utilizada para realizar a avaliação experimental desse trabalho e os resultados obtidos por meio da execução dos testes detalhados no capítulo; o Capítulo 7 apresenta as considerações finais do trabalho e discute potenciais oportunidades de trabalhos futuros.

Capítulo 2

Referencial Teórico

Neste capítulo são apresentados alguns conceitos que se relacionam com princípios explorados direta e indiretamente no iSRE. Alguns dos tópicos abordados e discutidos consistem na apresentação de conceitos e características de sistemas distribuídos na seção *Sistemas Distribuídos*. A tarefa de gerenciar sistemas distribuídos em escala exige a aplicação de métodos sistemáticos para se garantir, dentre outras coisas, a qualidade e manutenção de custos. Nesse sentido, apresentamos na seção *Gerenciamento de Sistemas em Rede* alguns dos métodos e mecanismos empregados para se efetuar o gerenciamento de sistemas em rede.

Algumas das atividades executadas como parte das tarefas de gerenciamento de sistemas em rede envolvem implantar e operar sistemas que realizam a monitoração dos componentes em operação no sistema em rede alvo de gerenciamento. Na seção *Sistemas de Monitoração* apresentamos informações sobre o papel desempenhado por sistemas de monitoração em um ambiente de sistemas em rede. Além disso, apresentamos detalhes sobre dois sistemas de monitoração utilizados atualmente.

A monitoração dos componentes de um sistema em rede muitas vezes revela atributos que são passíveis de ajuste para que se alcance melhorias ou níveis acordados de serviço no contexto de gerenciamento de um sistema em rede. Modelos que propõem que esses ajustes sejam efetuados de maneira automática e/ou autônoma são apresentados nas seções *Auto-Gerenciamento de Sistemas de Computação* e *Robotic Process Automation (RPA)*. Nessas seções apresentamos conceitos e características associadas ao auto-gerenciamento de sistemas de computação e ao *robotic process automation (RPA)*.

Por fim, na seção *Comunicação em Sistemas de Computação* são apresentados dois exemplos de mecanismos utilizados para se estabelecer comunicação entre dois objetos remotos em um sistema em rede, o protocolo *Secure Shell (SSH)* e o conceito

de *web services*. Ambos são explorados de forma prática nesse trabalho, por meio do protótipo implementado.

2.1 Sistemas Distribuídos

São várias as definições de sistemas distribuídos na literatura. Em [1], um sistema distribuído é definido como uma coleção de computadores independentes que aparecem para os usuários como um único sistema. No artigo clássico *Time, Clocks, and the Ordering of Events in a Distributed System* [20], a definição de sistema distribuído aparece como uma coleção de processos distintos e separados espacialmente, que se comunicam por meio de troca de mensagens.

Algumas das vantagens em se utilizar sistemas distribuídos para prover serviços computacionais incluem [1, 30]:

- facilidade para integrar em um único sistema diferentes aplicações executando em diferentes computadores;
- grande escalabilidade quando bem projetados;
- maior facilidade para compartilhamento de recursos;
- potencial aumento no poder de computação, confiabilidade e comunicação.

Em um sistema distribuído de arquitetura cliente servidor, clientes enviam requisições para servidores com o objetivo de obter serviços. O servidor consiste em processos que implementam serviços como email, gerenciador de arquivos, impressão, hospedagem de aplicação web, entre outros [1]. Em contraste, em um sistema distribuído de arquitetura *peer-to-peer*, a distinção cliente-servidor não existe. Todos os nós podem tanto ofertar quanto consumir serviços. Algumas das aplicações desse tipo de sistema se concentram em entrega de conteúdo e compartilhamento de arquivos.

O aumento no número de componentes que fazem parte de um sistema distribuído e/ou um aumento no número de serviços ofertados pode aumentar a complexidade para se gerenciar e manter os níveis de serviço acordados com clientes. Algumas das características observadas em sistemas distribuídos para suportar o aumento na demanda por serviços são:

- **escalabilidade:** capacidade de absorver o crescimento na demanda por serviços e acomodar mudanças a necessidade de grandes transformações;

- **disponibilidade:** quantidade de tempo que o sistema está funcionando e acessível aos clientes. Se o cliente não consegue acessá-lo, o mesmo está indisponível;

- **desempenho:** quantidade de trabalho realizado pelo sistema comparada com a quantidade de tempo e recursos consumidos;

- **tolerância a falhas:** habilidade do sistema se comportar de maneira previsível quando falhas ocorrem.

A indisponibilidade de um sistema pode ser causada pela inabilidade desse em tolerar falhas. Distúrbios externos ou interações não antecipadas entre os componentes de um sistema distribuído podem causar falhas. Um sistema falha quando não consegue entregar o que foi projetado e está responsável por entregar. Por exemplo, um sistema distribuído falha quando está projetado para ofertar aos usuários um conjunto de serviços, mas não consegue prover um ou mais serviços de forma completa ou parcial [1].

Erros não endereçados podem levar sistemas distribuídos a falhar. A causa de erros é denominada falta. Identificar e sanar as faltas pode prevenir que erros causem falhas. Diversos eventos podem causar erros e consequentemente falhas. Uma forma de identificar eventos que possam comprometer o estado saudável de um sistema distribuído é empregar um ou mais sistemas de monitoração. Caso aconteçam eventos que indiquem a ocorrência de um desvio, ações corretivas podem ser tomadas.

2.2 Gerenciamento de Redes de Computadores

Em ambientes de sistemas distribuídos e em rede, cada vez mais complexos e compostos por um número cada vez maior de componentes, o emprego de práticas de gerenciamento de redes pode auxiliar no controle de custos, manutenção dos indicativos de qualidade e, em alguns casos, na geração de receitas. De acordo com [5], o gerenciamento de redes é constituído de atividades, métodos, procedimentos e ferramentas que se relacionam a operação, administração, manutenção e provisionamento de sistemas em rede. Nesse contexto, as tarefas de operação, administração, manutenção e provisionamento possuem como objetivo:

- **operação:** manter a rede e os serviços por ela fornecidos em execução. Uma das atividades compreendidas é a de monitoração da rede para identificação de eventuais falhas;

- **administração:** controlar os recursos disponibilizados na rede e como eles são assinalados;

- **manutenção:** desempenhar as atividades de reparo e atualização nos componentes da rede. Também inclui as atividades corretivas e preventivas, como intervenção pro-ativa, para garantir o bom funcionamento da rede;

- **provisionamento:** configurar e disponibilizar recursos aos usuários.

Segundo [5], alguns dos benefícios que podem ser alcançados por meio da adoção de práticas efetivas de gerenciamento de redes incluem a redução de custos, aumento na qualidade dos serviços prestados e geração de receitas.

No que diz respeito a custos, um dos principais objetivos do efetivo gerenciamento de redes é tornar a operação mais eficiente e os operadores mais produtivos, reduzindo-se assim o custo total de propriedade associado a rede. O custo total de propriedade associado a rede inclui o custo associado aos equipamentos e o custo associado aos insumos que são demandados para operação. Ferramentas de monitoração, prevenção e remediação de falhas são exemplos de ferramentas empregadas no gerenciamento de redes que permitem o aumento da produtividade e a redução de custos. As ferramentas de monitoração permitem identificar eventuais problemas de forma mais rápida e as ferramentas de remediação permitem a correção automática de desvios rotineiros, permitindo-se assim que a equipe de operações foque em outras demandas.

São diversas as métricas de qualidade em uma rede, podendo-se incluir tempo de resposta, níveis de disponibilidade e confiabilidade de um componente ou conjunto de componentes destinados a oferecer determinado serviço. As práticas de gerenciamento de rede fornecem alguns dos meios para se atingir a qualidade esperada pelos usuários.

A geração de receitas também pode se beneficiar do gerenciamento de redes, por exemplo, ao se automatizar o provisionamento de determinado serviço, o tempo em que uma ordem de serviço é submetida e o tempo em que é efetivada é reduzido, criando-se a capacidade para geração de receita de forma mais rápida.

2.3 Sistemas de Monitoração

Os sistemas de monitoração coletam dados do sistema monitorado de forma contínua, servindo como base para uma série de análises, por exemplo, a classificação de um desvio do estado esperado do sistema ou desempenho aquém do esperado. Diversos sistemas de monitoração estão disponíveis em formato de código aberto e são utilizados em larga

escala na indústria e academia. A seguir destacamos dois sistemas de monitoração, Graphite [14] e Nagios [9].

2.3.1 Graphite

O Graphite [14] é um sistema que permite a monitoração de desempenho de componentes de sistemas distribuído. É possível coletar nos nós um conjunto de métricas em formato de série temporal e enviar para um sistema central do Graphite, denominado Carbon, que os armazena em memória não volátil e disponibiliza para visualização. Uma *daemon* no Carbon se encarrega de escutar continuamente por requisições e receber as métricas enviadas a partir dos nós. Além disso, também é possível receber dados de ferramentas de coleta de métricas como StatsD [10] e CollectD [6]. Alguns dos dados coletados podem incluir informações sobre: carga de *cpu*, utilização de disco, consumo de memória, entre outros.

2.3.2 Nagios

O Nagios [9] oferece mecanismos de monitoração para serviços de rede, dispositivos de *hardware*, sistemas operacionais, aplicações, entre outros. Por meio de arquivos de configuração, é possível declarar um estado esperado para determinado componente, juntamente de uma série de verificações a serem executados para validar se o componente possui o estado esperado. Em caso de desvios sobre o estado esperado, eventos e alertas de notificação podem ser gerados para indicar potencial problema. É possível controlar a frequência de verificação e os níveis de alerta. Um agente é disponibilizado para executar comandos pré-definidos nos nós remotos. Exemplos de verificação incluem:

- **estado de um serviço:** verificação do estado de execução de um determinado serviço. Por exemplo, serviço de email em execução ou parado;
- **resposta a requisições remotas:** pode-se verificar se determinado serviço remoto responde a requisições. Caso não responda a uma quantidade pré determinado de requisições, o mesmo é classificado como não respondendo e um evento é gerado. Por exemplo, um nó na rede que não responde;
- **tamanho de um sistema de arquivos:** com base em um valor limite pré-determinado, verificações contínuas são executadas para validar se o tamanho de um sistema de arquivos excedeu o valor limite.

Além disso, o Nagios permite aos usuários definir que procedimentos sejam executados em caso em decorrência de eventos. O usuário deve referenciar um *script* que será executado quando o evento ocorrer. Nenhuma funcionalidade para se analisar o conteúdo do evento e determinar, com base nesse conteúdo, em tempo de execução qual o melhor procedimento a se executar é oferecida. O *script* deve estar disponível em um diretório acessível a partir da máquina na qual o evento ocorreu.

2.4 Auto-Gerenciamento de Sistemas de Computação

Segundo [19], a computação autonômica pode ser definida como sistemas de computação que podem se auto-gerenciar uma vez que administradores forneçam objetivos em alto nível. Um dos objetivos dos sistemas de auto-gerenciamento é liberar os administradores de tarefas operacionais, para que foquem em outras atividades. Nesse contexto, a capacidade de determinado sistema em se auto-remediar minimiza as interações manuais que visam diagnosticar, prevenir e corrigir problemas que ocorrem tempo de execução. Em um cenário ideal, administradores definem objetivos em alto nível e o sistema de auto-gerenciamento se encarrega de executá-los de forma autônoma.

Conforme observado em [1], existem diversas variações de sistemas de auto-gerenciamento dentro de um arcabouço de computação autonômica, mas a maioria considera que a execução de alteração de estado como forma de remediação ou prevenção de problemas em tempo de execução ocorre como resultado da utilização de um loop de controle de *feedback*. Os loops de controle de feedback são formados por três elementos, sendo:

1. um sistema de monitoração e o sistema a ser gerenciado;
2. um elemento que analisa os dados do sistema monitorado e aponta a necessidade de eventuais alterações ao seu estado;
3. um ou mais componentes que têm como objetivo alterar o estado do sistema em caso de necessidade.

O elemento a ser gerenciado pode ser um sistema distribuído típico, que fornece serviços como email, impressão, compartilhamento de arquivos, entre outros. Por meio da monitoração contínua no loop de *feedback* e da elaboração de políticas de prevenção e remediação de problemas em tempo de execução, um sistema de auto-gerenciamento

pode remediar de forma automática eventos com potencial de causar problemas em um ou mais componentes de um sistema em rede.

2.5 *Robotic Process Automation* (RPA)

O conceito de *RPA* pode ser entendido como a aplicação no negócio de um ou mais robôs em software configurados para executar tarefas previamente executadas por pessoas, interagindo com um conjunto de componentes de um sistema de maneira análoga a que as pessoas interagiriam [33]. Essa é uma abordagem da indústria e está se expandindo [33]. Alguns dos benefícios em se adotar o robôs em software variam entre redução de custos, melhoria na qualidade de processos, maior de agilidade de execução de cargas de trabalho e redução de erros.

Algumas das características associadas a processos que são automatizados por meio de *RPA* consistem na existência de um processo manual em que uma pessoa recebe como entrada um conjunto de dados por meio de um sistema, por exemplo um evento de um sistema de monitoração, processa essa entrada usando um conjunto de regras, por exemplo um manual com uma lista de comandos a se executar, e por fim atualiza um sistema com o resultado do processamento executado, por exemplo envia um email com a lista de comandos executados em um componente da rede e o respectivo resultado.

Sistemas baseados em *RPA* tendem a apresentar uma interface de programação amigável e fácil de configurar, ao mesmo tempo que não demandam criação ou substituição dos sistemas legados de um sistema em rede para que funcionem adequadamente[33]. Além disso, apresentam características que permitem a interoperabilidade com a camada de apresentação dos sistemas aos quais irão interagir.

2.6 Comunicação em Sistemas de Computação

Nessa seção são apresentados dois exemplos de mecanismos para se estabelecer comunicação entre dois objetos remotos em um sistema em rede, o protocolo *Secure Shell (SSH)* e o conceito de *web services*.

2.6.1 *Secure Shell (SSH)*

O protocolo *Secure Shell (SSH)* [34] permite estabelecer conexão segura com objetos remotos de um sistema em rede por meio de autenticação. Um canal seguro sobre uma rede insegura é estabelecido entre uma aplicação *SSH* cliente e um servidor *SSH*

remoto. Algoritmos de criptografia são utilizados para autenticar ambos os lados da conexão, automaticamente codificar os dados transmitidos e, finalmente, proteger a integridade dos dados trafegados. Duas versões principais estão disponíveis, *SSH1* e *SSH2*. As principais plataformas *Unix-like* oferecem suporte a instalação e execução de clientes e servidores *SSH*.

Uma forma de autenticação utilizando *SSH* inclui a geração automática de um par de chaves público-privada para codificar a conexão de rede e então utilizar um senha de autenticação para efetuar a autenticação. Outra forma de autenticação utiliza um par de chaves público-privadas geradas manualmente para executar a autenticação, permitindo que usuários ou processos utilizem o cliente para efetuar a autenticação junto ao servidor sem a necessidade de especificar uma senha.

Uma vez que a conexão *SSH* é estabelecida, é possível enviar comandos que serão executados localmente no objeto remoto com o qual a conexão foi estabelecida. Essa funcionalidade elimina a necessidade de instalar agentes locais para se receber e executar comandos administrativos em objetos remotos. A execução de comandos interpretados pelo interpretador de linguagem de comandos *shell* constitui um exemplo de comandos passíveis de serem utilizados. Alguns exemplos de comandos *shell* incluem:

1. ***hostname*:** exibe o nome do computador;
2. ***date*:** exibe a data hora e do sistema;
3. ***ping*:** envia uma mensagem para um computador remoto com o intuito de validar se o computador remoto pode ser alcançado e exibe o resultado, além de dados como o tempo de resposta;
4. ***uptime*:** exibe o tempo total em que o sistema está ativo desde que foi ligado.

Outras operações suportados pelo *SSH* incluem a transferência segura de arquivos e encaminhamento de portas.

2.6.2 *Web Service*

De acordo com o W3C [32], um *web service* é um sistema em software que disponibiliza um meio padronizado para que máquinas consigam se comunicar e interagir com máquinas distintas por meio de uma rede. Pode-se citar como exemplo um *web service* que implementa funcionalidade para converter o valor de uma quantia em dólar para uma quantia em real. Ao receber uma requisição de uma máquina qualquer na rede

que lista determinada quantia em dólar, o *web service* executa a conversão e retorna o valor em real. Tudo ocorre sem a necessidade de que ambas as máquinas adotem uma mesma plataforma ou façam parte de um mesmo arcabouço, o único pré-requisito para que a comunicação seja realizada com sucesso é a adoção de uma arquitetura padrão para se comunicar.

Atualmente, uma das arquiteturas mais utilizadas para implementação de *web services* é o estilo de arquitetura denominado *Representational State Transfer (REST)* . Um *web service* que implementa o padrão *REST* pode receber requisições HTTP com conteúdo em diversos formatos, por exemplo XML e JSON, processá-la e retornar uma resposta com conteúdo também em uma variedade de formatos. Algumas das características de *web services REST* incluem o alto desempenho, escalabilidade e simplicidade de implementação e consumo.

2.7 Sumário

Nesse capítulo foram apresentados alguns conceitos que se relacionam com princípios e práticas exploradas no iSRE. Alguns dos tópicos abordados e discutidos nesse capítulo consistem na apresentação de conceitos e características de sistemas distribuídos na seção *Sistemas Distribuídos*, alguns dos métodos e mecanismos empregados para se efetuar o gerenciamento de sistemas em rede e, por consequência, sistemas distribuídos. Além disso, apresentamos informações sobre o papel desempenhado por sistemas de monitoração em um ambiente de sistemas em rede e detalhamos dois sistemas de monitoração utilizados atualmente. Nas seções *Auto-Gerenciamento de Sistemas de Computação* e *Robotic Process Automation (RPA)*, apresentamos conceitos e características de dois modelos que propõem que ajustes nos componentes de um sistema em rede sejam efetuados de maneira automática e/ou autônoma. Por fim, na seção *Comunicação em Sistemas de Computação* são apresentados dois exemplos de mecanismos utilizados para se estabelecer comunicação entre dois objetos remotos em um sistema em rede, o protocolo *Secure Shell (SSH)* e o conceito de *web services*.

No próximo capítulo é apresentada uma visão geral da arquitetura do iSRE. Mostramos os principais requisitos atendidos e como a arquitetura está estruturada em torno de componentes que cooperam entre si para atingir alguns dos objetivos propostos no trabalho. Também apresentamos a descrição dos principais componentes da plataforma e algumas de suas características. Por fim, aprofundamos nos detalhes da API fornecida aos usuários para que efetuem a declaração de políticas de resposta a eventos e apresentamos dois exemplos de caso de uso em que os usuários poderiam

aplicar as funcionalidades disponibilizadas pelo iSRE.

Capítulo 3

Especificação da Arquitetura Global

Neste capítulo apresentamos uma visão geral da arquitetura do iSRE, mostramos os principais requisitos atendidos e como a arquitetura está estruturada em torno de componentes que cooperam entre si para atingir os objetivos propostos para a plataforma. Também apresentamos a descrição dos principais componentes da plataforma e aprofundamos nos detalhes da API fornecida aos usuários para que efetuem a declaração de políticas de resposta a eventos.

A arquitetura leva em consideração o objetivo geral desse trabalho que consiste em projetar, implementar e avaliar uma plataforma que viabiliza de forma prática mecanismos que suportem a execução automática e controlada de procedimentos que visam prevenir, validar, mitigar e/ou sanar potenciais problemas em tempo de execução de um ou mais componentes de um sistema em rede. Dentre outros conceitos teóricos contemplados, podemos destacar a implementação de um elemento capaz de analisar os dados de um sistema monitorado e executar alteração de estado como resultado dessa análise, alinhado ao que é proposto pelo loop de *feedback* descrito no Capítulo 2 na seção de Auto-Gerenciamento de Sistemas de Computação. Além disso, por meio da API os usuários podem declarar políticas de eventos para execução automática de modelo similar aos robôs em software propostos por *RPA*, também descrito no Capítulo 2.

Esperamos que com a utilização do protótipo experimental gerado a partir dessa arquitetura os usuários consigam definir procedimentos que serão executados de forma automática e controlada como forma de reação a ocorrência de um ou mais eventos que indicam mudança de estado em um ou mais componentes de um sistema em rede. Um contexto de aplicação para a plataforma consiste em operações compostas por sistemas em rede, que possuem diversos componentes de software interagindo entre si e evoluindo de forma dinâmica, onde um ou mais sistemas de monitoração e/ou predição

identificam potenciais falhas que ocorrem em tempo de execução. Nessa direção, deseja-se automatizar a execução de ações que respondem a gatilhos originados nos sistemas de monitoração e/ou predição, de maneira controlada e escalável.

Alguns dos termos comumente utilizados nesse trabalho são definidos abaixo com o intuito de tornar mais fácil o entendimento do conteúdo apresentado nesse e nos demais capítulos:

- **Evento:** conjunto de dados que indicam a mudança de estado de um ou mais componentes de um sistema em rede. São originados a partir de sistemas que têm como objetivo agregar valor a operação, manutenção e evolução dos componentes de um sistema em rede, por exemplo, sistemas de monitoração ou sistemas de predição de falhas. Em ambientes de sistema em rede, podem, por exemplo, ser classificados como chamados, alertas, *tickets*, incidentes, entre outros;
- **Procedimento:** conjunto de uma ou mais instruções a serem executadas em um componente de um sistema em rede como forma de reação ao recebimento de um evento;

Na próxima seção apresentamos detalhes sobre a visão geral da arquitetura do iSRE.

3.1 Visão Geral da Arquitetura iSRE

A arquitetura da plataforma iSRE é composta, de maneira geral, por quatro grupos de componentes de software que colaboram entre si utilizando memória compartilhada para oferecer os serviços que implementam os requisitos definidos como parte desse trabalho. Os principais componentes são denominados *Controlador de Eventos*, *Despachante de Ações*, *Controlador de Execução de Ações* e *Políticas de Resposta a Eventos*. Uma visão geral dos componentes da plataforma e como eles interagem entre si está ilustrada na figura 3.1. São apresentados mais detalhes sobre cada componente na seção *Descrição dos Componentes* desse capítulo.

Alguns dos requisitos considerados e utilizados como norte para elaboração da arquitetura da plataforma foram definidos com base na observação de necessidades da indústria e academia e em oportunidades identificadas no referencial teórico, levando-se em consideração alguns dos princípios de computação autonômica e *RPA*. Os principais requisitos que a plataforma iSRE deve atender são:

- **Escopo de Atuação:** deve ser capaz de interagir com componentes de um sistema em rede, por exemplo, sistemas operacionais, sistemas de banco de dados,

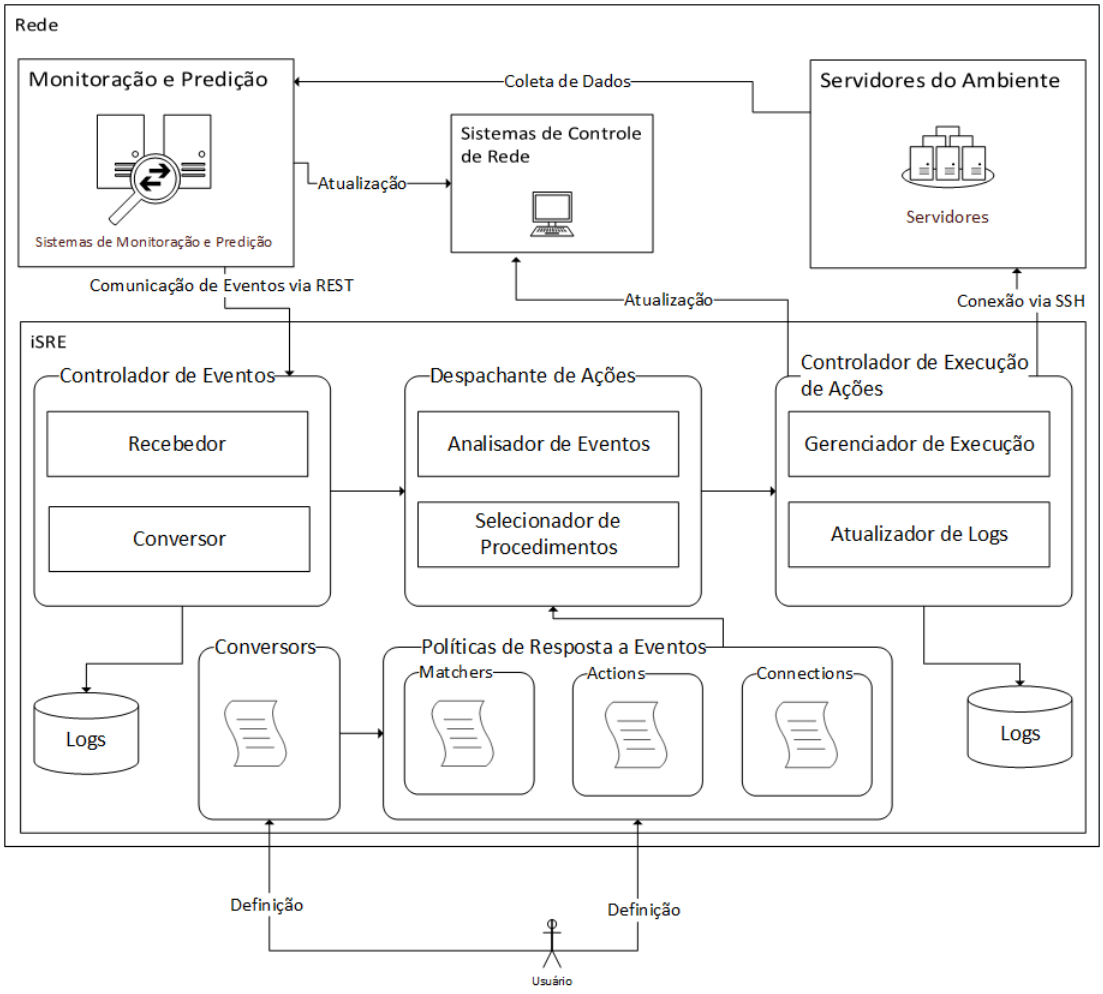

Figura 3.1: Visão geral dos componentes e suas interações

aplicações. Assim, sendo capaz de operar e interagir com componentes típicos de um sistema em rede;

- **Interoperabilidade:** ser capaz de receber e processar eventos gerados em outras ferramentas e plataformas. Portanto, não se limitando a uma ou mais ferramentas específicas;

- **Analisar Eventos:** a plataforma deve fornecer mecanismos que tornem possível a análise do conteúdo de um evento e com base nesse conteúdo e em políticas definidas pelos usuários, dispare a execução de um procedimento como forma de reação. A análise do evento deve ocorrer imediatamente após o recebimento desse na plataforma. Essa análise deve ser estruturada de forma que permita aos usuários definir procedimentos que possam ser reusados, se assim for produtivo;

- **Auditabilidade:** deve existir um mecanismo que atua como gerenciador da exe-

cução de políticas de resposta a evento e que controle a execução de comandos e as conexões com componentes remotos. Além disso, deve-se registrar em memória não volátil os comandos executados e as respostas recebidas. Isso permitirá aos usuários efetuar a verificação e validação da execução dos procedimentos, por exemplo, com o objetivo de aplicar melhoria contínua em processos ou aprofundar a análise de uma falha;

- **Escalabilidade:** a plataforma deve ser capaz de suportar o aumento no número de eventos de forma com que não seja necessário aumentar, de forma proporcional, os recursos computacionais e humanos para suportar a demanda. Portanto, a medida que os usuários ganham confiança na plataforma e alocam mais eventos ou o número de eventos aumenta naturalmente, os custos de operação não devem aumentar de maneira proporcional;

- **Produtividade:** devem existir mecanismos que facilitem o reuso de procedimentos, aumentando a produtividade dos usuários;

- **Suportar a Definição de Políticas de Resposta a Eventos:** usuários devem ser capazes de definir políticas de respostas a eventos de forma centralizada. Deve-se implementar funcionalidades que permitam ao usuário definir procedimentos a serem executadas como forma de reação eventos, regras que associem um evento a uma ação a ser executada e configurações de dados de conexão a serem utilizadas para se estabelecer conexão com componentes remotos na rede. A plataforma deve ser capaz de interpretar e executar essas políticas. Assim, os usuários são empoderados a criar de forma produtiva procedimentos a serem executados de forma automática e controlada.

A figura 3.2 ilustra o possível fluxo de um evento ao ser enviado e recebido pela plataforma iSRE. Da maneira como está estruturada a arquitetura, um protótipo experimental associado não requer que mudanças estruturais sejam efetuadas para que funcione conforme esperado em um ambiente de sistemas em rede típico. A seguir, é apresentada uma descrição que exemplifica as análises e ações executadas pela plataforma ao receber um evento:

1. uma vez que determinado evento chega ao iSRE, uma conversão do formato original do evento para o formato definido pelo iSRE pode ser aplicada, caso esteja definida pelo usuário;

2. o iSRE verifica se existe algum procedimento a ser executado em decorrência do evento recebido, utilizando técnicas de casamento de padrão para validar o

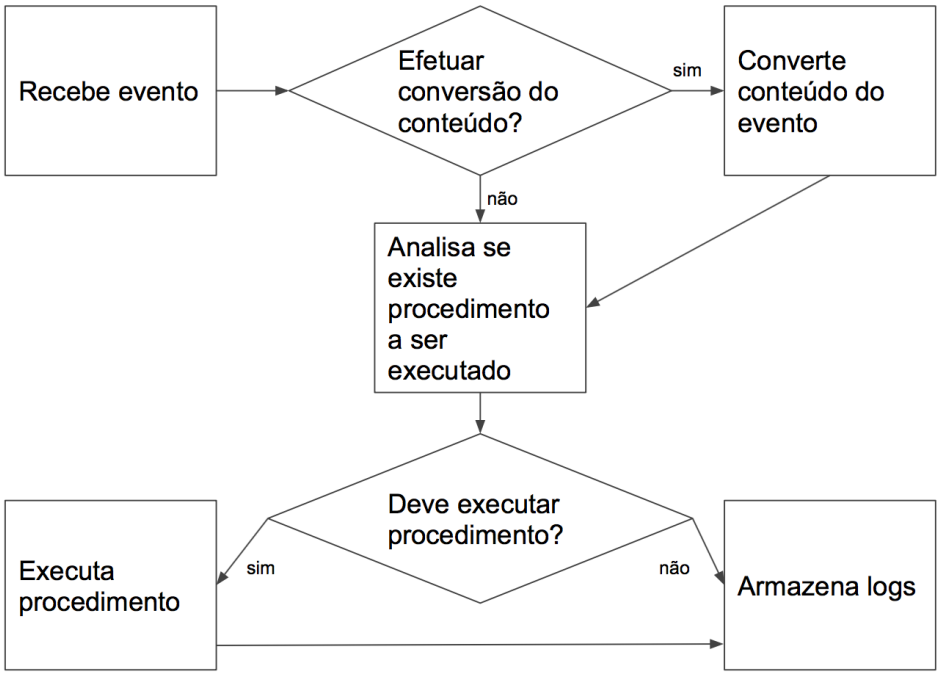

Figura 3.2: Possível fluxo de um evento ao ser enviado e recebido para a plataforma iSRE

conteúdo do evento de acordo com regras que associam eventos a procedimentos, ambos definidos pelo usuário;

3. caso exista algum procedimento a ser executado, os mecanismos para executá-lo são iniciados, por exemplo estabelecendo-se conexão com um componente remoto, visando a execução de comandos definidos no procedimento;

4. o procedimento é executado e os dados que contêm os registros dos comandos executados, as respostas recebidas ao se executar os comandos e informações adicionais são armazenados em um log de memória não volátil.

Na próxima seção apresentamos a descrição dos principais componentes da plataforma iSRE, já destacados na figura 3.1.

3.2 Descrição dos Componentes

Nesta seção são apresentados os principais grupos de componentes da plataforma iSRE, seu objetivo e informações básicas da forma como interagem. No capítulo 5 são apresentados maiores detalhes funcionais e de implementação a respeito desses componentes.

3.2.1 Controlador de Eventos

Os componentes desse grupo são responsáveis por fornecer as funcionalidades que recebem os eventos que são enviados à plataforma. Atuam de forma independente a uma ou mais ferramentas de monitoração de eventos. Transformações no conteúdo listado em um evento podem ser aplicadas com o objetivo de adequá-lo ao formato preestabelecido pela plataforma iSRE. Essas transformações são definidas pelos usuários. Portanto, é possível integrar distintas ferramentas de monitoração ou predição de eventos a plataforma.

Outro objetivo desse grupo de componentes é armazenar o conteúdo dos eventos em memória não volátil e instanciar os componentes do grupo de componentes *Despachante de Ações* na chegada dos eventos, suportando os requisitos associados a auditabilidade.

3.2.2 Despachante de Ações

Os componentes desse grupo são responsáveis por ler e manter em memória as políticas de resposta a eventos, definidas pelos usuários. Além disso, em tempo de execução, devem analisar o conteúdo dos eventos com base nas políticas de resposta a eventos, visando identificar se um procedimento deve ser executado em decorrência do evento sob análise. A análise está estruturada de forma a permitir que os usuários explorem recursos de reusabilidade de procedimentos.

Caso a análise do conteúdo de determinado evento identifique que existe um procedimento a ser executado em sua decorrência, esse grupo de componentes deve efetuar a inicialização dos mecanismos destinados a executar procedimentos, instanciando o *Controlador de Execução de Ações*.

3.2.3 Controlador de Execução de Ações

Os componentes desse grupo devem controlar a execução de procedimentos definidos pelos usuário, sendo responsáveis por interpretar e executar os comandos definidos nesses procedimentos, coletar e analisar as saídas e estabelecer conexões com os componentes remotos que são alvo desses comandos. A execução de comandos remotos faz uso do protocolo ou meio de comunicação remota estabelecida pelo usuário, permitindo a interação com diversos sistemas operacionais e componentes de software. Além disso, os componentes desse grupo devem manter e atualizar o registro dos comandos executados, as respectivas saídas e informações adicionais em arquivo não volátil, de forma alinhada aos requisitos de auditabilidade.

3.2.4 Políticas de Resposta a Eventos

Os componentes desse grupo disponibilizam APIs para que os usuários definam ou reutilizem procedimentos a serem executados em decorrência de um ou mais eventos específicos. Os procedimentos de resposta a eventos, que listam os comandos a serem executados, são definidos em arquivos denominados *Actions*. Outro objetivo dos componentes desse grupo é fornecer mecanismos para que os usuários definam parâmetros de configuração e regras de casamento de padrão que, em tempo de execução, são utilizadas para analisar o conteúdos dos eventos que chegam a plataforma, os associando a execução de *Actions*. Isso torna a associação de eventos a *Actions* dinâmica, o que permite explorar mecanismos que viabilizam o reuso de *Actions*. No iSRE essas regras e os parâmetros a elas associados são definidos em arquivos denominados *Matchers*.

Também são fornecidas funcionalidades para que os usuários declarem e definam qual tipo de conexão deve ser utilizado para se estabelecer conexão com componentes de um sistema em rede, juntamente de parâmetros relacionados a configuração para conexão remota. Os arquivos utilizados para declaração dessas configurações são denominados *Connections*. Por fim, os usuários podem definir um conjunto de regras que ao serem aplicadas em tempo de execução sobre eventos alvo efetuam a conversão do formato desses evento para o formato definido no iSRE. Essas regras são definidas em arquivos denominados *Conversors*.

3.3 API de Declaração de Políticas de Resposta a Eventos

Nessa seção são apresentadas as descrições funcionais e de implementação dos componentes que fornecem a API para que os usuários definam políticas de resposta a eventos. São apresentados os componentes que fornecem as funcionalidades para declaração de *Actions*, *Connections* e *Matchers*, que em conjunto disponibilizam os mecanismos para se definir políticas de resposta a eventos.

As *Actions*, *Connections* e *Matchers* são declaradas pelos usuários em arquivos no formato *JSON* e possuem definidas, no iSRE, classes correspondentes que servem para absorver o conteúdo dos arquivos. Em tempo de execução, os arquivos em formato *JSON* são lidos e têm seu conteúdo utilizado para instanciar um objeto da classe correspondente. Para cada arquivo de política de resposta a eventos definido pelos usuários, um objeto correspondente é instanciado no iSRE. Por exemplo, para cada arquivo de *Action* definido pelos usuários, em tempo de execução existe um objeto

Action instanciado no iSRE que contém as propriedades definidas no arquivo *Action* originalmente em formato *JSON*. O iSRE define um objeto *HashMap* [25], uma implementação de tabela hash que provê desempenho de tempo constante para as operações de inserção e recuperação de elementos, para armazenar os elementos de cada um dos tipos de arquivos que compõem as políticas de resposta a eventos. Ou seja, em tempo de execução existe uma instância do objeto *HashMap* que armazena as instâncias dos objetos *Action*, uma instância do objeto *HashMap* que armazena as instâncias dos objetos *Connection*, uma instância do objeto *HashMap* que armazena as instâncias dos objetos *Conversor* e por fim uma instância do objeto *HashMap* que armazena as instâncias dos objetos *Matcher*.

3.3.1 *Actions:* Ações de Resposta a Eventos

O iSRE define e implementa uma API que permite aos usuários a definição de gatilhos que disparam a execução de procedimentos em reação a chegada de eventos à plataforma. Os procedimentos contêm ações à serem executadas com o objetivo de verificar e/ou alterar o estado de componentes de um sistema em rede. Os procedimentos e ações são definidas em arquivos denominados *Actions*. Uma *Action* é composta por um nome, uma ou mais *Tasks* e uma ou mais *Conditionals* e uma ou mais variáveis de contexto da *Action*. Na figura 3.3 é demonstrada, como exemplo, a declaração de uma *Action* que contém variáveis de contexto, *Tasks* e *Conditionals*.

Ao declarar *Tasks*, os usuários podem definir comandos que serão executados em tempo de execução. Cada *Task* é composta por um nome, um comando e alguns atributos opcionais. Em cada *Task*, os usuários podem definir os seguintes atributos opcionais: *conditional*, *extractOutput*, *varByValue*, *varByReference*. Os comandos podem ser comandos pré-configurados do iSRE, que definem algumas operações comumente utilizadas, ou podem ser comandos suportados pelo componente alvo de execução, por exemplo comandos *bash*. A execução de uma *Task* pode depender da execução prévia de outra *Task*. As operações nativas do iSRE são:

- ***local command:*** oferece um *shell* local para execução de comandos;
- ***email:*** envia um email utilizando os parâmetros destinatário, assunto e texto;
- ***task:*** referencia uma *Action* que possui um conjunto de *Tasks*, as quais serão executadas em tempo de execução, permitindo reuso de *Actions*;
- **script:** referencia um *script* presente localmente e que deve ser executado em algum servidor remoto. Em tempo de execução o iSRE efetua a cópia do *script*

```
{
    "name":"action_exemplo.json",
    "varsDeclaration":"host=runtime, saidaPing=null, emailTecnicos=tecnicos@email.com",
    "tasks": [
        {
            "name": "Ping host",
            "command": "local ping -c4 {{host}}",
            "extractOutput": "{{saidaPing}}",
            "conditional":"Verifica saída ping"
        },
        {
            "name": "Escala o problema",
            "command": "email {{emailTecnicos}},, Ping não alcançou o servidor {{host}},,
                A saída do comando foi: {{saidaPing}}"
        },
        {
            "name": "Conecta ao servidor alvo",
            "command": "connect to endpoint conexao_exemplo"
        },
        {
            "name": "Obtém dados básicos",
            "command": "hostname; date"
        }
    ],
    "conditionals": [
        {
            "name": "Verifica saída ping",
            "atribute1": "variable",
            "evaluator": ".*(bytes from).*",
            "atribute2": "saidaPing",
            "caseTrue":"Conecta ao servidor alvo",
            "otherwise":"Escala o problema"
        }
    ]
}
```

Figura 3.3: Exemplo de declaração de uma *Action*

do servidor local para o servidor remoto, executa o *script* e coleta a saída da execução;

- ***connect to endpoint Connection:*** inicializa uma conexão remota com o objeto alvo utilizando os parâmetros de conexão definidos na *Connection* referenciada. A conexão fica aberta até que todos os comandos sejam executados ou ocorra uma interrupção de conexão não planejada. Para fins experimentais, nesse trabalho as conexões remotas são estabelecidas usando o protocolo SSH, mas a arquitetura não se restringe a esse protocolo para efetuar conexões com objetos remotos.

Os atributos opcionais *varByReference* e *varByValue* permitem aos usuários passar variáveis como referência ou por valor para a *Action* referenciada na *Task*. Caso o usuário opte por passar a variável por referência, utilizando o atributo *varByReference*, o iSRE disponibiliza a referência da variável no contexto da *Action* referenciada e alterações feitas no valor da variável enquanto em execução na *Action* referenciada serão mantidas quando o contexto de execução retornar a *Action* original. Para variáveis

passadas por valor, utilizando o atributo *varByValue*, o iSRE cria uma nova variável no contexto da *Action* referenciada, com valor similar ao da variável passada por valor, o que permite que modificações no valor da variável sejam efetuadas sem que sejam replicados além da *Action* referenciada.

O atributo opcional *extractOutput* permite aos usuários definir uma variável à qual receberá o conteúdo da saída do comando definido na *Task*. Em tempo de execução, o iSRE executa o comando definido na *Task*, lê a saída desse comando e assinala o valor para a variável declarada pelo usuário. A variável fica disponível para utilização em todo o contexto da *Action*, podendo ser utilizada em comandos definidos em outras *Tasks* e/ou passadas como referência ou por valor para outras *Actions*.

O atributo *conditional* permite fazer referência a *Conditional* definida na *Action*. As *Conditionals* possibilitam a implementação de controle de fluxo de execução entre as diferentes *Tasks* presentes em uma *Action*. Por meio das *Conditionals* é possível validar se uma ou mais regras de alteração de fluxo são satisfeitas para se direcionar o fluxo de execução. Para validar a *Conditional* o iSRE implementa um mecanismo para avaliação de casamento de padrão, onde o usuário define os atributos e a expressão regular a ser avaliada.

3.3.2 *Connections:* Declaração de Conexões

Para executar as *Actions* que verificam e/ou alteram o estado de um componente remoto de um sistema em rede, o iSRE estabelece uma conexão remota com o servidor que hospeda esse componente. O usuário pode declarar abstrações denominadas *Connections* para definir o protocolo ou meio de comunicação para se estabelecer comunicação com o objeto remoto. Uma *Connection* é um arquivo de declaração de parâmetros de conexão e tem como objetivo definir um conjunto de configurações de conexão que serão utilizadas por um ou mais *Actions* em tempo de execução para efetuar conexão com um ou mais objetos remotos.

Uma *Connection* é composta por um nome, descrição, um tipo de conexão, usuário e senha, um número de porta, uma lista de servidores, a definição de um campo para extrair informações em tempo de execução e uma expressão regular, o número de tentativas de conexão a se efetuar e o tempo de expiração de uma conexão (*timeout*). Dados sensíveis como usuário e senha são mascarados após serem declarados com o objetivo de se impor controles de segurança. Na figura 3.4 é apresentada, como exemplo, a declaração de uma *Connection* que possui listada os parâmetros de conexão requeridos em *Connection*.

O tipo de conexão estabelece o protocolo a ser utilizado. Para fins experimentais

```
{
    "name":"connection_exemplo",
    "description": "conexão de exemplo",
    "type": "user and password",
    "port": "22",
    "hosts": "runtime",
    "field": "host",
    "user": "usuario",
    "password": "*********",
    "regex": "(.*)",
    "retry": "1",
    "timeout": "500000"
}
```

Figura 3.4: Exemplo de declaração de uma *Connection*

do trabalho implementamos suporte ao protocolo SSH, mas arquitetura propõe suporte a diferentes tipos de protocolos e meios de conexão com objetos remotos. A lista de servidores pode variar entre um ou mais servidores aos quais uma conexão deve ser estabelecida. Além disso, pode ser populada em tempo de execução por meio da extração de conteúdo presente nos eventos, bastando o usuário definir o valor do parâmetro *hosts* como *runtime* e especificar o campo do evento ao qual o iSRE deve extrair o conteúdo e a expressão regular à ser utilizada. O iSRE extrai o conteúdo e assinala para a variável *hosts*, que corresponde a lista de servidores a se conectar. Também é possível que o usuário defina a porta com a qual deve se estabelecer conexão, o número de tentativas de conexão a se efetuar e o tempo de expiração para uma conexão (*timeout*).

O iSRE utiliza os parâmetros declarados na conexão para personalizar o objeto que inicia uma conexão remota. Para cada novo evento que requer uma conexão remota com algum componente do sistema em rede, uma tentativa de nova conexão é estabelecida. Conexões prévias não são reaproveitadas à nível do iSRE.

3.3.3 *Conversors:* Conversão de Formato de Eventos

A API dos *Conversors* permite aos usuários definir abstrações que são interpretadas pelo iSRE e utilizadas para aplicar transformações no conteúdo de eventos alvo. Dessa forma, os usuários podem definir regras para converter o conteúdo de eventos que originalmente não chegam ao iSRE no formato pré-estabelecido pela plataforma.

Em cada *Conversor* os usuários devem definir um nome e um conjunto de uma

```
{
        "name":"nagios conversor",

        "rules": [

        {
            "name": "Get event name",
            "regex": "(.*?): .*",
            "field": "name",
            "booleanOperator": "and"

        },
        {
            "name": "Get host",
            "regex": ".*: (.*?);.*",
            "field": "hosts",
            "booleanOperator": "and"
        },
        {
            "name": "Get description",
            "regex": ".*: \\S+;(.*?)$",
            "field": "description",
            "booleanOperator": "none"
        }
        ]
}
```

Figura 3.5: Configuração padrão de um *Conversor*

ou mais regras de casamento de padrão. A figura 3.5 apresenta um exemplo de um *Conversor* padrão. As regras de casamento de padrão são compostas por um nome, uma expressão regular, o campo no iSRE ao qual o conteúdo convertido será assinalado e um operador booleano que permite combinar uma ou mais regras. Em tempo de execução a expressão regular de uma regra é avaliada levando em consideração o conteúdo do evento alvo. Para que a transformação do conteúdo seja completada com sucesso, todas as regras definidas devem ser satisfeitas. Em seguida, o conteúdo é convertido em um objeto *Evento* no iSRE.

3.3.4 *Matchers:* Regras que Associam Eventos a *Actions*, *Connections* e *Conversors*

Com o objetivo de eliminar a necessidade de criação de uma *Action* para cada tipo de evento e oferecer oportunidades de reusabilidade de *Actions* aos usuários, o iSRE implementa um modelo dinâmico para associar uma *Action* a um evento. Por meio de arquivos denominados *Matchers*, os usuários conseguem definir qual o conteúdo que um evento deve possuir para que seja acionada uma *Action* em particular. Os usuários devem declarar regras de casamento de padrão, as quais o iSRE utiliza para validar o

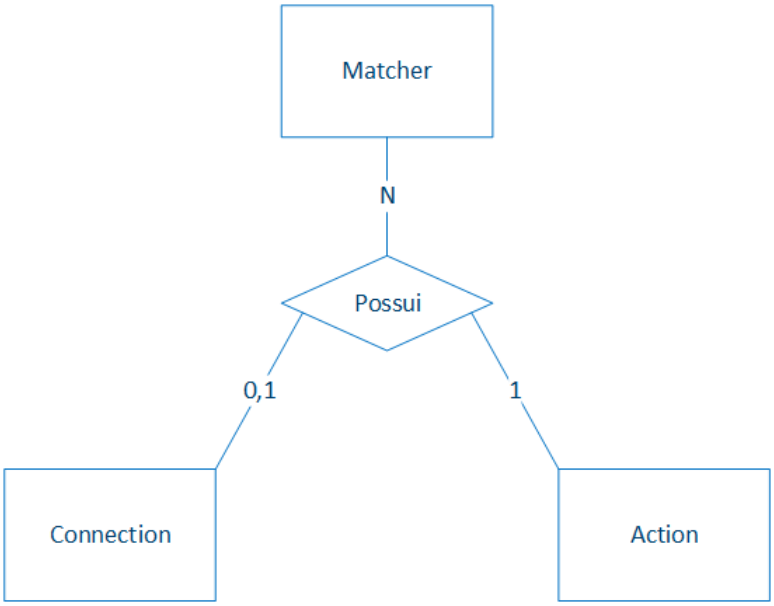

Figura 3.6: Relacionamento entre *Matcher*, *Action*, *Connection* e *Conversor*

conteúdo do evento em busca do padrão declarado. Uma *Action* pode ser referenciada em vários *Matchers*, sem a necessidade de ser duplicada.

Opcionalmente, os usuário também podem associar o *Matcher* a uma *Connection*. A relação entre *Matcher*, *Action* e *Connection* é ilustrada na figura 3.6. Além de obrigatoriamente possuir referência para uma *Action* e opcionalmente para uma *Connection*, um *Matcher* é composto por uma ou mais regras de casamento de padrão e zero ou mais *Extractors*, que são parâmetros utilizados pelo iSRE para extrair conteúdo dos eventos em tempo de execução. A figura 3.7 exibe a configuração padrão de um *Matcher*.

Do ponto de vista dos usuários, o principal objetivo em se declarar *Matchers* é o de fornecer um mecanismo que permita identificar se uma *Action* deve ser acionada em decorrência de um evento. Um *Matcher* pode possuir uma ou mais regras para efetuar casamento de padrão com o conteúdo do evento. Essas regras são compostos por um nome, uma expressão regular, um operador booleano e o nome do campo ao qual o iSRE deve recuperar o conteúdo em um dado evento para aplicar a expressão regular e validar se a mesma é verdadeira. A figura 3.8 apresenta um exemplo de declaração de um *Matcher*.

As regras de casamento de padrão utilizam expressões regulares que aderem a sintaxe padrão de expressões regulares suportada pela linguagem Java. É possível declarar mais uma regra de casamento de padrão e combina-lá com as demais regras utilizando operadores booleanos. Os operadores booleanos suportados são *AND*, *NOT* e *OR*. Uma *Action* será acionada somente se todas as regras de casamento de padrão

Figura 3.7: Configuração padrão de um *Matcher*

forem avaliadas como verdadeiro, levando-se em consideração a combinação entre mais de uma regra utilizando-se os operadores booleanos, caso assim esteja definido.

Os *Extractors* são compostos por um nome de variável, uma expressão regular e o nome de campo do evento ao qual o iSRE irá buscar por conteúdo. O iSRE utiliza a expressão regular para extrair o conteúdo do campo especificado pelo usuário no *Matcher*. Uma vez que o iSRE extrai o conteúdo do campo especificado com base na expressão regular, o mesmo é assinalado para a variável nomeada pelo usuário no *Extractor*. Essa variável é criada e disponibilizada no contexto de execução da *Action* referenciada no *Matcher*, permitindo aos usuários fazer uso de variáveis que possuem valor atribuído dinamicamente, de acordo com o conteúdo presente em cada evento processado.

3.4 Casos de uso do iSRE

Nessa seção são apresentados dois casos de uso aos quais a plataforma iSRE pode ser utilizada.

3.4.1 Processo parado de forma indevida

A seguir é apresentada a descrição de evento associado a um processo parado de forma indevida, junto de exemplos das funcionalidades do iSRE que podem ser empregadas para se responder de forma automática a esse tipo de evento.

Um exemplo de uso do iSRE se aplica em situações onde a ferramenta de monitoração do ambiente identifica que um processo que deveria estar executando para

```
{
    "name":"matcher_exemplo",
    "action": "action_exemplo.json",
    "connection": "conexao_exemplo",
    "rules": [
        {
            "name": "Verifica a descrição do evento",
            "regex": ".*(evento exemplo).*",
            "field": "description",
            "booleanOperator": "none"
        }
    ],
    "extractors": [
        {
            "varName": "host",
            "regex": "(.*)",
            "field": "hosts"
        }
    ]
}
```

Figura 3.8: Exemplo de declaração de um *Matcher*

fornecer algum serviço, por exemplo email, está parado de forma indevida. Um evento é gerado pela ferramenta de monitoração e enviado para o iSRE e demais ferramentas de gerenciamento de rede. O conteúdo desse evento é composto pelo nome do processo que está parado de forma indevida, o nome e endereço IP do servidor em que o processo foi identificado como parado, o nome do processo parado, uma descrição sobre o evento e uma severidade.

O usuário pode declarar uma *Action* genérica para tratar eventos relacionados a processos parados. Inicialmente, pode declarar uma variável de contexto da *Action* para armazenar o nome do processo parado e outra variável com a lista de interessados em receber um email informando sobre a execução da *Action*.

Nas *Tasks* dessa *Action*, o usuário pode declarar operações *local*, para consultar uma fonte de dados externa e verificar se o servidor e/ou o processo afetado estão em uma lista crítica que restringe a atuação automática de correção de problemas, junto de *Conditionals* que fazem o fluxo da execução da *Action* passar para uma *Task* de envio de email informando que não serão executados comandos para iniciar o processo pois o evento lista um servidor e/ou processo crítico. Em seguida, pode ser definida uma operação *conexão* que vai estabelecer a conexão com o servidor afetado. Adicionalmente, podem ser definidos comandos que verificam se o processo continua parado e que tentam iniciá-lo caso assim esteja. A saída desses comandos pode ser assinalada para uma variável de contexto da *Action*. O conteúdo dessa variável indica o sucesso ou não da inicialização do processo, permitindo a utilização de uma *Conditional* para direcionar o fluxo de execução da *Action* de acordo com o seguinte critério:

- **processo continua parado:** a execução segue para a última *Task*, que implementa uma operação *email* para enviar email informando sobre o estado da execução dos procedimentos e informações coletadas nas *Tasks* anteriores.

- **processo iniciado com sucesso:** a execução segue para uma *Task* que referencia uma *Action* que implementa comandos para obter informações básicas do servidor remoto (*hostname*, *date* e *uptime*), reutilizada em várias outras *Actions*, e por fim a execução também segue para a *Task*, que envia emails e foi citada no item anterior.

Para associar o evento a uma *Action*, pode-se criar um *Matcher* que declara uma regra que define o campo descrição como campo a ter conteúdo recuperado do evento e que também define uma expressão regular a ser aplicada pelo iSRE contra o campo descrição, possibilitando definir se a *Action* deve ou não ser alocada para processamento. Além disso, pode ser declarado um *Extractor* para extrair o *IP* e outro *Extractor* para extrair o nome do serviço parado e associar ambos valores à variáveis que serão utilizadas no contexto da *Action*. Também pode ser declarada no *Matcher* uma *Connection* que será utilizada se obter os dados de conexão na operação *conexão* de uma das *Tasks* da *Action*.

3.4.2 Validação de servidor que não responde na rede

A seguir é apresentada a descrição de um evento associado ao problema de um servidor não estar respondendo na rede, junto de exemplos de funcionalidades do iSRE que podem ser empregadas para se responder de forma automática a esse tipo de evento.

Ao identificar que um servidor na rede não responde a comandos de monitoração probatória por um período e número de testes que excede o limite para alertar a equipe de suporte, a ferramenta de monitoração gera um evento que é enviado para o iSRE e para as demais ferramentas de gerenciamento de rede. O conteúdo do evento é composto por o nome e endereço IP do servidor que não responde a comandos probatórios e está aparentemente inoperante, uma descrição sobre o evento e uma severidade.

O usuário pode definir uma *Action* para ser executada quando esse tipo de evento ocorrer. Na *Action*, pode ser declarada uma *Task* que utiliza a operação *local* para obter o *shell* local e executar comandos *ping* de maneira a validar se o servidor reportado no evento continua sem ser alcançado na rede, sendo que a saída dos comandos é armazenada em uma variável de contexto da *Action* declarada pelo usuário, denominada por exemplo *isServerReachable* e que possui os valores verdadeiro para o caso do servidor afetado pelo problema tenha sido alcançado ou falso caso contrário. Uma *Conditional*

pode ser declarada para avaliar o conteúdo dessa variável, permitindo assim direcionar o fluxo de execução das *Tasks* da seguinte forma:

- **servidor continua sem responder:** pode-se referenciar uma *Action* que define *Tasks* para se autenticar em um conjunto de servidores remotos na rede com o objetivo de validar se existe comunicação com servidor afetado pelo problema. A variável *isServerReachable* tem seu valor atualizado para refletir o resultado dos novos testes.

- **servidor está respondendo:** pode-se definir uma *Task* que implementa uma operação *conexão* para se conectar ao servidor afetado pelo problema e que depende do conteúdo da variável *isServerReachable* ser verdadeiro. Em seguida, pode-ser definir uma *Task* que referencia uma *Action* a qual implementa comandos para obter informações básicas do servidor remoto (*hostname*, *date* e *uptime*) e atualiza uma variável para salvar essas informações, por exemplo *basicInfos*.

A última *Task* define uma operação *email* é utilizada para enviar um email informando sobre o estado da execução da *Action*. Para informar se o servidor foi alcançado é utilizado o conteúdo da variável *isServerReachable* juntamente do conteúdo da variável *basicInfos*. Os demais logs de execução também são adicionados.

Para associar o evento a uma *Action*, pode-se criar um *Matcher* que declara uma regra que define o campo descrição como campo a ter conteúdo recuperado do evento e que também define uma expressão regular a ser aplicada pelo iSRE contra o campo descrição, possibilitando definir se a *Action* deve ou não ser alocada para processamento. Além disso, pode ser declarado um *Extractor* para extrair o *IP* do servidor que não responde na rede e associar esse conteúdo a uma variável de contexto na *Action*. Também pode ser declarada no *Matcher* uma *Connection* que será utilizada obter os dados de conexão na operação *conexão* de uma das *Tasks* da *Action*.

3.5 Sumário

Nesse capítulo foi apresentada uma visão geral da arquitetura do iSRE. Mostramos os principais requisitos atendidos e como a arquitetura está estruturada em torno de componentes que cooperam entre si para atingir alguns dos objetivos propostos nesse trabalho. Também apresentamos a descrição e algumas das características dos principais componentes da plataforma. Por fim, aprofundamos nos detalhes da API fornecida aos usuários para que efetuem a declaração de políticas de resposta a eventos

e apresentamos dois exemplos de caso de uso em que os usuários poderiam aplicar as funcionalidades disponibilizadas pelo iSRE.

No próximo capítulo apresentamos a metodologia utilizada como base para se efetuar a avaliação qualitativa sobre a percepção de profissionais a respeito de ferramentas e iniciativas que se propõem a suportar a execução automática e controlada de procedimentos de suporte a eventos em um ambiente de sistemas em rede. Discutimos o método utilizado para efetuar a pesquisa, denominado MEDS, que fornece um arcabouço sistemático para se efetuar a definição dos objetivos, coletar e analisar os dados. Também apresentamos os resultados derivados da pesquisa realizada junto aos profissionais e uma discussão a respeito desses resultados.

Capítulo 4

Avaliação Qualitativa

Nesse capítulo apresentamos a metodologia utilizada para efetuar avaliação qualitativa sobre a percepção de profissionais a respeito de ferramentas e iniciativas que se propõem a suportar a execução automática e controlada de procedimentos de suporte a eventos em um ambiente de sistemas em rede. Discutimos o método utilizado para efetuar a pesquisa, denominado MEDS, que fornece um arcabouço sistemático para se efetuar a definição dos objetivos, coletar e analisar os dados. Também apresentamos os resultados derivados da pesquisa realizada junto aos profissionais e uma discussão a respeito desses resultados.

4.1 Metodologia

Nessa seção apresentamos a metodologia utilizada para realizar a avaliação qualitativa disponível nesse trabalho. A avaliação tem como objetivo mensurar a percepção de profissionais a respeito de ferramentas e iniciativas que se propõem a suportar a execução automática e controlada de procedimentos de suporte a alertas em um ambiente de sistemas em rede.

A metodologia aplicada para realizar a análise qualitativa utiliza o Método de Explicitação de Discurso Subjacente (MEDS) [22] para nortear a pesquisa conduzida junto a profissionais que possuem experiência com o tema alvo. O MEDS consiste em um método exploratório, que trabalha explicitamente com material discursivo. Estabelece uma série de procedimentos, divididos em etapas, a serem aplicadas sistematicamente. Dentre suas características destacam-se a utilização de entrevistas semi-estruturadas e guiadas por um roteiro, que visam orientar uma conversa natural e livre, com o objetivo de extrair o que os entrevistados têm a dizer a respeito do tema [22]. Os dados coletados nas entrevistas são transcritos e analisados em busca de detalhes explícitos

e não explícitos, por exemplo, paralinguísticos, de interesse da investigação. As etapas para realização da análise qualitativa, alinhadas a sistemática proposta no *MEDS*, são destacadas a seguir:

1. **delineamento do objetivo:** são definidos os objetivos que a pesquisa visa atingir, delimitando o foco da investigação;

2. **recrutamento dos participantes:** definição do perfil dos entrevistas e da estratégia de recrutamento. O *MEDS* sugere que o recrutamento considere grupos pequenos e homogêneos;

3. **preparação para a coleta de dados:** nessa etapa são levantados os instrumentos requeridos para se realizar a coleta de dados. O *MEDS* destaca a utilização de entrevistas semi-estruturadas, guiadas por um roteiro;

4. **coleta de dados:** o *MEDS* propõe a utilização de entrevistas um a um, caracterizado conversas naturais e livres. As conversas são gravadas com consentimento dos participantes e formalizadas com a apresentação de um termo de consentimento de participação que explicita as condições da entrevista. O termo deve ser assinado por entrevistado e entrevistador;

5. **preparação para a análise dos dados:** consiste na transformação em texto dos dados coletados nas entrevistas, por meio da transcrição.

6. **análise dos dados e atualização dos resultados:** o *MEDS* possui como característica propor para a etapa de análise dos dados a aplicação de uma análise explícita de discurso, onde parte-se do princípio que qualquer atributo linguístico ou paralinguístico que é recorrente nos discursos dos participantes pode ser um indicador de algo invisível que quer se tornar visível. A análise de dados divide-se em duas etapas, sendo a primeira a análise inter-sujeitos e a segunda a análise intra-sujeito. *Insights* ganhos na segunda etapa permitem voltar a primeira etapa e onde analisa-se os conjuntos de repostas dadas por todos os entrevistados novamente. Espera-se que com esse loop, o material coletado seja dominado a fundo e detalhes ditos explicitamente e não explicitamente sejam revelados, servindo para enriquecer a pesquisa.

7. **interpretação dos resultados:** recorrências identificadas na etapa de análise dos dados levam a emersão de categorias para análise dos resultados.

4.2 Aplicação do MEDS

Nessa seção apresentamos de forma sucinta as etapas sugeridas no método MEDS que foram consideradas no trabalho. Também apresentamos uma descrição de como essas etapas foram executadas no contexto desse trabalho, juntamente dos artefatos gerados em cada em uma das etapas. A seguir são apresentados os detalhes:

- **delineamento do objetivo:** o nosso objetivo é A avaliação tem como objetivo mensurar a percepção de profissionais a respeito de ferramentas e iniciativas que se propõem a suportar a execução automática e controlada de procedimentos de suporte a alertas em um ambiente de sistemas em rede. Alinhado a isso, a pergunta principal que serviu como norte para elaboração do roteiro é "Você acredita que uma ferramenta que se propõem a suportar a execução automática e controlada de procedimentos de suporte a alertas em um ambiente de sistema em rede possui relevância no mercado?";

- **recrutamento dos participantes:** inicialmente, definimos o perfil desejado para os participantes, que consiste em profissionais de tecnologia da informação com mais de 5 anos de experiência em atividades típicas da operação de ambientes de sistemas em rede, supervisionando equipes ou atuando diretamente na administração de componentes. Com o perfil definido, utilizamos a rede de conhecidos para destacar candidatos e convidá-los a participar. Foram selecionados 4 participantes, com idade entre 28 e 45 anos e experiência de atuação variando entre 8 e 20 anos. Com o objetivo de preservar o anonimato dos profissionais, denominamos as siglas *P1, P2, P3 e P4* para nos referirmos a eles;

- **preparação para a coleta de dados:** em alinhamento ao MEDS, elaboramos um roteiro, disponível no anexo A deste trabalho, composto por 9 questões abertas que serviram de base para orientar as entrevistas executadas com os participantes, seguindo um modelo semi-estruturado. As 9 questões listadas no roteiro estão dividas em dois blocos. O primeiro bloco é composto por questões que abordam a experiência do entrevistado. O segundo bloco é composto por questões orientadas a explorar experiências e percepções dos entrevistados a respeito de iniciativas e ferramentas que propõem suportar automatização de rotinas manuais típicas da operação de ambientes de sistema em rede. Nessa etapa também elaboramos o termo de consentimento de participação na entrevista, que está disponível nos anexos desse trabalho;

- **coleta de dados** as entrevistas foram conduzidas individualmente com os participantes, gravadas e transcritas. Duas entrevistas foram conduzidas presencialmente e duas entrevistas foram conduzidas virtualmente. As entrevistas levaram entre 9 e 28 minutos. Todos os participantes assinaram o termo de consentimento de participação da entrevista, que está disponível no anexo B;

- **preparação para a análise dos dados:** nessa etapa foi transformados em texto, por meio da transcrição, o conteúdo obtido nas entrevistas. Foram tomados cuidados para transcrever em texto detalhes como hesitações e pausas, conforme mencionado em [22] tais detalhes podem ser importantes indicadores na etapa de análise dos dados;

- **análise dos dados e atualização dos resultados:** a análise dos dados seguiu a orientação disponível no MEDS. Inicialmente efetuamos a análise inter-sujeito dos dados, seguido da análise intra-sujeito. Os resultados são apresentados na seção a seguir.

4.3 Resultados

A seguir são apresentados os resultados derivados da análise dos dados obtidos nas entrevistas. Os resultados foram categorizados em em torno de subtemas que emergiram a partir de recorrências identificadas nas análises, conforme orientado em [22].

Preocupação em implantar ferramentas que suportam a automatização de rotinas manuais típicas da operação de sistemas em rede

Todos os profissionais demonstraram preocupação em tornar a operação mais eficiente por meio da implantação de soluções que visam automatizar atividades executadas de forma manual. Além disso, os profissionais destacaram a importância dessas iniciativas internamente e os movimentos que estão sendo efetuados nessa direção. O profissional menciona o seguinte: *Pro negócio da minha área da empresa é fundamental. A gente já tá correndo atrás disso. A gente já tá buscando ferramentas mais eficazes, mais eficientes para poder fazer com que a nossa produtividade aumente.*

Atividades relacionadas ao suporte a *tickets* foram mencionadas por 3 profissionais como sendo um dos alvos prioritários para se conduzir iniciativas que visam a automatizar tarefas que atualmente requerem esforço manual. O profissional *P4* mencionou que em função da característica de alguns *tickets*, as atividades necessárias para atuação poderiam ser automatizadas uma vez que estivesse disponível uma ferramenta que oferece um conjunto de mecanismos que viabilize a execução automática e contro-

lada de procedimentos: *"Eu acredito fortemente que um volume considerável de tickets não precisaria de atuação técnica, pode se colocar uma automação por trás disso."*

Portanto, podemos afirmar que existe a preocupação e interesse dos profissionais em incluir em seu portfólio de ferramentas uma plataforma que viabilize a automatização de rotinas manuais típicas da operação de sistemas em rede.

Experiência com ferramentas que suportam a execução automática e controlada de procedimentos de suporte a alertas

Os profissionais relataram variadas experiências com relação a prospecção e utilização de ferramentas que suportam a automatização da execução de procedimentos de suporte a alertas. Somente o profissional *P3* informou possuir em seu ambiente uma ferramenta em operação que fornece os mecanismos para viabilizar a execução automática de procedimentos. Os profissionais *P1* e *P4* estão buscando implementar e habilitar plataformas que fornecem essa capacidade. O profissional *P1* mencionou a relevância que uma ferramenta nesse sentido possui dentro da organização a qual ele está inserido:

> *"Estamos fazendo uma frente para nos diferenciar no mercado e já estamos estudando e investindo não só tempo, mas tentando viabilizar financeiramente a adoção desses mecanismos para que a gente consiga dentro de um ecossistema de clientes e parceiros aplicar isso e conseguir sair do outro lado com sucesso."*

O profissional *P4* mencionou que a ferramenta de monitoração em operação na organização possui algumas funcionalidades que viabilizam a execução automática de procedimentos e que estão sendo feitos estudos para iniciar a exploração dessas capacidades: *"Hoje não é explorado aqui na organização, é o próximo trabalho que quero que seja feito e pelo que sei é possível colocar uma inteligência por trás pra tomar essas ações."*

Dessa forma, ao analisarmos o conteúdo das entrevistas, percebemos que os profissionais entendem que existe espaço no ambiente para ferramentas que se propõe a fornecer mecanismos que viabilizam a execução automática de procedimentos de suporte a alerta, mas na prática essas ferramentas nem sempre estão sendo exploradas e utilizadas.

Requisitos e desafios associados a ferramentas que suportam a execução automática e controlada de procedimentos de suporte a alertas

No geral, os requisitos destacados pelos entrevistados como os mais importantes em uma ferramenta que se propõe a fornecer mecanismos que viabilizem a execução automática e controlada de procedimentos de suporte a alertas estão relacionados a interoperabilidade, auditabilidade e escalabilidade.

Sobre interoperabilidade, o profissional *P1* relata o seguinte: *"Interoperabilidade é super importante, as ferramentas precisam se comunicar"*, destacando a importância de características que possibilitem a comunicação entre diferentes componentes de um sistema em rede. A respeito dos mecanismos que garantam auditabilidade, o profissional *P4* menciona que: *[...] que seja possível auditar também eu acho importante, porque por mais que a ferramenta faça a ação, em caso de uma auditoria ou em caso de uma falha a gente vai precisar fazer um tracking bem detalhado do que aconteceu.*

Sobre escalabilidade, o profissional *P3*, profissional que possui experiência com a implantação ferramentas que se propõe a suportar a execução automática de procedimentos de suporte a alertas, destaca o seguinte:

> *Você pode começar ali pequenininho, automatizando uma ou duas tarefas e aquilo com toda certeza vai ganhar fama muito rápido. Muitas outras áreas, muitos outros problemas, vão querer ter o mesmo tipo de solução, então é importante que essa ferramenta seja o que pessoal fala de escalável, tenha escalabilidade.*

Para o profissional *P2*, um desafio importante para se implementar uma ferramenta com características mencionadas de forma prévia consiste em: *é você não tirar a flexibilidade do ambiente.* Ou seja, é importante que a ferramenta não imponha mudanças nos componentes e na estrutura do ambiente para que possa funcionar adequadamente, adicionando restrições e forçando alterações que minimizam atributos de versatilidade do ambiente.

Importância do tema

No geral, os profissionais destacaram a importância do tema. Nesse sentido, o profissional *P4* fez o seguinte comentário:

> *Eu acho o tema super interessante, eu acho que se bem desenvolvido tem muitos ganhos pra todas as organizações, ganhos em produtividade, em assertividade, em satisfação do cliente, porque a medida que você reduz erro operacional o cliente fica mais satisfeito.*

O profissional *P1* opina sobre a urgência relacionada ao tema: *É algo que realmente quem tiver na área de TI, de suporte e sustentação, se ele não olhar para isso*

o quanto antes, ele vai ficar obsoleto no mercado. Já o profissional *P3* se refere a importância da automação como um todo: *Acho que o comentário sobre essa temática de automação é que, de novo, isso não é nem futuro mais, né?! É presente, é inevitável!.* Além disso, menciona sobre as oportunidades de ganho:

> *[...] eu acho que se bem desenvolvido, tem muitos ganhos pra todas as organizações. Ganhos em produtividade, em assertividade, em satisfação do cliente, que a medida que você reduz erro operacional o cliente fica mais satisfeito.*

4.4 Discussão

A pesquisa permitiu validar que os profissionais enxergam com bons olhos o emprego de ferramentas e iniciativas que se propõem a suportar a execução automática e controlada de procedimentos de suporte a eventos em um ambiente de sistemas em rede. Foi possível identificar pontos a respeito do tema que são considerados importantes pelos profissionais entrevistados, tais como a urgência em tomar ações e preocupações com alguns dos requisitos que devem ser endereçados na ferramenta.

Dentre os pontos identificados que estão diretamente alinhados ao que foi proposto e implementado no protótipo apresentado nesse trabalho, podemos destacar a capacidade da ferramenta em fazer interface e se comunicar com diversos componentes, possuir mecanismos que permitam efetuar a auditoria dos comandos executados, apresentar características que tornem possível a absorção do aumento de demanda de carga de trabalho, não demandar mudanças nos componentes do sistema em rede com o objetivo exclusivo de tornar possível a operação da ferramenta, por exemplo, instalando agentes.

4.5 Sumário

Nesse capítulo foi apresentada a metodologia utilizada como base para se efetuar a avaliação qualitativa sobre a percepção de profissionais a respeito de ferramentas e iniciativas que se propõem a suportar a execução automática e controlada de procedimentos de suporte a eventos em um ambiente de sistemas em rede. Discutimos o método utilizado para efetuar a pesquisa, denominado MEDS, que fornece um arcabouço sistemático para se efetuar a definição dos objetivos, coletar e analisar os dados. Também apresentamos os resultados derivados da pesquisa realizada junto aos profissionais e uma discussão a respeito desses resultados.

No próximo capítulo são descritos alguns detalhes a respeito dos principais componentes do iSRE e a forma como estão implementados. São apresentados detalhes sobre os componentes que recebem e convertem eventos, analisam o conteúdo dos eventos e selecionam os procedimentos a serem executados, gerenciam a execução desses procedimentos e atualizam os logs que registram dados da execução. Além disso, apresentamos detalhes de implementação do protótipo, como a linguagem e bibliotecas utilizadas.

Capítulo 5

Detalhamento da Arquitetura

Nesse capítulo são descritos alguns detalhes a respeito dos principais componentes do iSRE e a forma como estão implementados. São apresentados detalhes sobre os componentes que recebem e convertem eventos, analisam o conteúdo dos eventos e selecionam os procedimentos a serem executados, gerenciam a execução desses procedimentos e atualizam os logs que registram dados da execução. Além disso, apresentamos detalhes de implementação do protótipo, como a linguagem e bibliotecas utilizadas.

5.1 Controlador de Eventos

Nessa seção são apresentados as detalhes de implementação dos subcomponentes Recebedor e Conversor, responsáveis por receber e converter dados associados a eventos. A figura 5.1 ilustra a estrutura do *Controlador de Eventos*.

5.1.1 Recebedor

O objetivo principal desse grupo de componentes é implementar funcionalidades que permitam a plataforma iSRE receber eventos, gerados a partir de um ou mais componentes de um sistema em rede. Para receber eventos, um serviço web escuta continuamente por chamadas *REST*, que podem ser enviadas a partir de um ou mais componentes do sistema em rede. São aceitas chamadas com conteúdo no formato *JSON*. Esse conteúdo é o conteúdo associado a um evento. Caso exista um *Conversor* definido para efetuar a conversão do padrão conteúdo do evento recebido para o padrão definido pelo iSRE, essa conversão é executada.

O iSRE implementa o Recebedor de maneira a permitir que eventos sejam recebidos e processados de maneira assíncrona. Uma vez que um evento é recebido, o mesmo

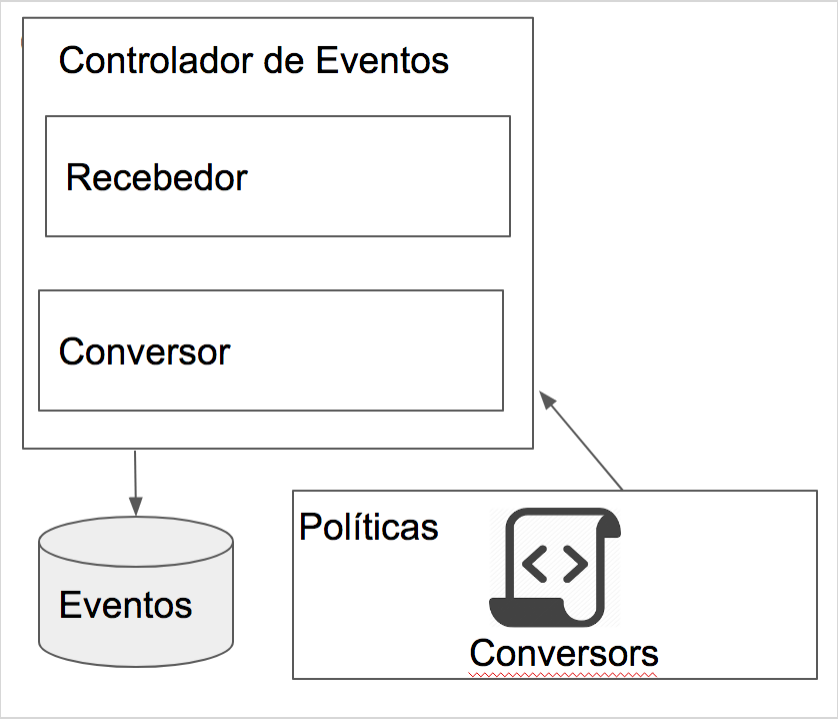

Figura 5.1: Estrutura *Controlador de Eventos*

é inserido em uma fila baseada em lista encadeada, *thread-safe*, com política *FIFO*, e disponibilizado para processamento em outros componentes do iSRE. Os eventos não possuem distinção de prioridade para processamento e uma vez iniciado o processamento, não existem mecanismos implementados pelo iSRE para interrupção proposital do processamento de um evento em função da chegada de outro evento. O *Recebedor* não fica impedido de receber novos eventos até que o evento prévio seja completamente processado em quaisquer outros componentes que compõem o iSRE. Essa característica torna possível processar diversos eventos em paralelo, fazendo com o que tempo de resposta às requisições diminua.

5.1.2 Conversor

Um dos objetivos da plataforma iSRE é ser independente de ferramentas que se encontram no primeiro grupo de elementos do loop de feedback de controle, por exemplo, ferramentas de monitoração ou predição de estado de um sistema em rede. Nessa direção, a plataforma iSRE implementa uma API que permite aos usuários definir um modelo para conversão do conteúdo dos eventos. Os usuários definem conversões em arquivos denominados *Conversors*, onde o usuário declara regras para converter o conteúdo de um evento em formato original para o formato definido no iSRE.

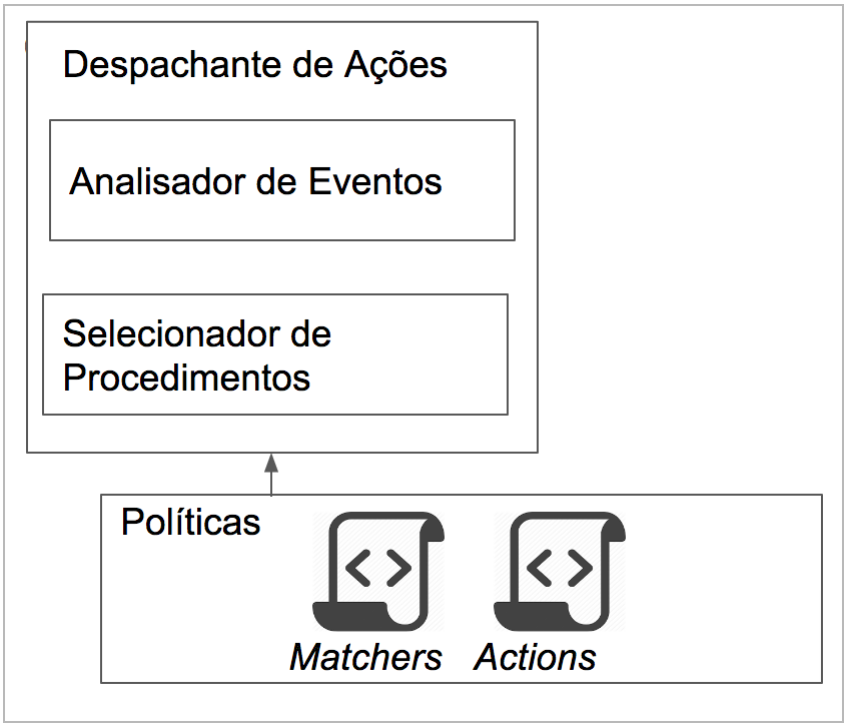

Figura 5.2: Estrutura *Despachante de Ações*

As funcionalidades de conversão permitem aos usuários integrar diversos sistemas de geração de eventos ao iSRE, o que desacopla a plataforma de suporte a resposta automatizada e controlada a eventos de uma única ferramenta de geração de eventos.

O iSRE lê, interpreta e mantém em memória as conversões definidas pelos usuários para avaliar, com a chegada de cada evento a plataforma, a necessidade de conversão do evento conforme definido pelos usuários, aplicando-a em caso positivo. O próximo passo é armazenar o evento em um log de memória não volátil e encaminhar o evento para os componentes que compõem o grupo de despacho de ações.

5.2 Despachante de Ações

Nessa seção são apresentadas detalhes de implementação dos subcomponentes Analisador de Eventos e Selecionador de Ações, responsáveis por analisar o conteúdo dos eventos e selecionar *Actions* aplicáveis. A figura 5.2 apresenta a estrutura do *Despachante de Ações*.

5.2.1 Analisador de Eventos

O objetivo do *Analisador de Eventos* é analisar o conteúdo dos eventos e identificar se existe uma *Action* a ser executado em função desse conteúdo. Ao inicializar a plataforma, o *Analisador de Eventos* lê, interpreta e mantém em memória os *Matchers* definidos pelos usuários para utilizá-los na análise do conteúdo dos eventos válidos recebidos em tempo de execução. Ao analisar um evento, as regras e parâmetros de casamento de padrão definidas nos *Matchers* são utilizadas para identificar se o evento em análise contém o conteúdo que requer a execução de uma *Action*.

Para efetuar a análise de um evento, o *Analisador de Eventos* extrai o evento da fila de eventos, seleciona o primeiro *Matcher* a ser testado e inicia a análise do conteúdo do evento aplicando os parâmetros e regras de casamento de padrão definidos nesse *Matcher* sobre o conteúdo do evento. Para cada regra de casamento de padrão, o *Analisador de Eventos* obtém do *Matcher* a expressão regular declarada e a compila utilizando funcionalidades nativas da linguagem em que o protótipo foi desenvolvido. Também obtém o nome do campo ao qual deve-se obter o conteúdo e validar se expressão regular é válida. Uma vez recuperado o nome do campo, o iSRE obtém o conteúdo disponível para esse campo no evento em análise e executa a validação da expressão regular.

Para um evento, o analisador valida se todos os *Matchers* se aplicam. Para cada *Matcher*, todas regras de casamento padrão podem ser executadas, mas não necessariamente são. Ao identificar que uma regra presente em um *Matcher* não é satisfeita dado o conteúdo do evento, o *Analisador de Eventos* descarta a análise para todas as outras regras desse *Matcher*, se existirem, e passa a validar regras presentes em outro *Matcher*.

Caso todas as regras de casamento de padrão presentes em um *Matcher* sejam satisfeitas, o *Analisador de Eventos* inicia os procedimentos para executar os componentes do *Selecionador de Ações*.

5.2.2 Selecionador de Procedimentos

Uma vez que o *Analisador de Eventos* identifica que existe uma *Action* a ser executada dado o conteúdo de um evento, o *Selecionador de Procedimentos* deve selecionar a *Action* a ser executada e instanciar os componentes responsáveis por executá-la. O nome da *Action* a ser executada é obtido por meio da leitura de um parâmetro disponível no objeto referente ao *Matcher* identificado como válido pelo *Analisador de Eventos*.

Após obter a referência de qual *Action* executar, o *Selecionador de Procedimentos* instancia os mecanismos disponibilizados pelo *Controlador de Execução de Procedimen-*

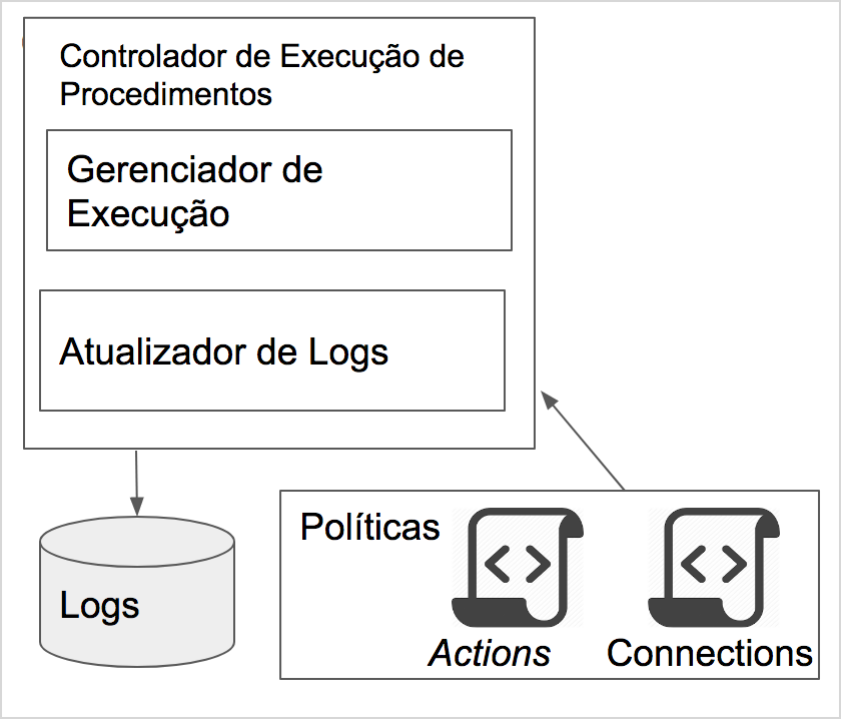

Figura 5.3: Estrutura *Controlador de Execução de Procedimentos*

tos, passando como parâmetro o nome da *Action* à ser executada. Caso também exista uma *Connection* referenciada no *Matcher*, o nome da *Connection* também é passado como referência.

5.3 Controlador de Execução de Procedimentos

Nessa seção são apresentadas as descrições dos componentes *Gerenciador de Execução* e *Verificador e Atualizador de Estado de Execução*, responsáveis por gerenciar a execução das *Actions* e atualizar um arquivo de log referente ao estado dessa execução. Na figura 5.3 é apresentada a estrutura do *Controlador de Execução de Procedimentos*.

5.3.1 Gerenciador de Execução

Os componentes do grupo *Gerenciador de Execução* são responsáveis por interpretar e executar de forma controlada as *Actions* definidas pelos usuários. Além disso, são responsáveis por estabelecer conexão com componentes remotos. O *Gerenciador de Execução* implementa os objetos que estabelecem conexão com componentes remotos, enviam comandos e recebem a saída. Para efetuar conexão, são utilizados dados

da *Connection* associada ao *Matcher* em identificado pelo *Analisador de Eventos* ou referências a *Connections* obtidas nas *Actions*.

Ao executar uma *Action*, as *Tasks* são por padrão executadas de forma sequencial, mas o fluxo de execução pode ser alterado de forma a atender as regras condicionais definidas nas *Conditionals* e/ou as regras de dependência entre *Tasks*.

O *Gerenciador de Execução* consegue interpretar a dependência entre *Tasks*, oferecendo ao usuário a possibilidade de definir que uma *Task* só seja executada caso alguma outra *Task* tenha sido executada previamente. Essa funcionalidade flexibiliza o controle sobre a execução da *Action* como um todo.

Outra funcionalidade oferecida pelo *Gerenciador de Execução* permite a manipulação de variáveis, definidas pelo usuário nas *Actions*, em tempo de execução. As variáveis declaradas pelos usuários fazem parte de todo o escopo da *Action*, o que possibilita que sejam compartilhadas e atualizadas em diferentes *Tasks*, a critério do usuário.

Caso uma conexão com objeto remoto seja estabelecida durante a execução de uma *Action*, a mesma se mantém aberta até o fim da execução de todas as *Tasks*, considerando que interrupções não esperadas não ocorram.

5.3.2 Atualizador de Logs

O objetivo dos componentes do grupo *Atualizador de Logs* é criar e registrar em um arquivo de *log* não volátil dados referentes a execução de uma *Action* pelo *Gerenciador de Execução*. Assim, os *logs* ficam disponíveis para análise posterior, por exemplo, para auditar.

Para cada execução de *Action* é criado um arquivo de *log* associado a essa execução. Durante a execução de uma *Action* pelo *Gerenciador de Execução*, são registrados no arquivo de *log* os comandos definidos na *Action*, as saídas desses comandos e dados adicionais, como o *timestamp* do instante em que o dado é registrado no arquivo.

No geral, são registrados o nome da *Action* em execução, o conteúdo completo do evento que fez a *Action* ser disparada, cada valor assumido pelas variáveis declaradas nessa *Actions* e o *timestamp* do instante em que o dado é registrado no arquivo. Adicionalmente, para cada *Task* executada, são registrados dados inerentes a *Task*. Na figura 5.4 é apresentado um exemplo dos dados registrados para uma determinada *Task*. Além disso, cria-se log em modelo de uma linha, conforme ilustrado na figura 5.5, mais amigável a leitura automática. A lista de dados registrados para cada *Task* é descrita a seguir:

- o número sequencial de controle do número de *Tasks* executadas;

```
------------------------------------------------------------


[2017-08-21 19:31:20:908]  step: 2
[2017-08-21 19:31:20:908]  task: Get hostname and date
[2017-08-21 19:31:20:908]  command: hostname;date
[2017-08-21 19:31:22:60]  exit-status: 0
[2017-08-21 19:31:22:60]  System.Err:

[2017-08-21 19:31:22:60]  System.Out:
dionisio
Mon Aug 21 19:31:21 BRT 2017


------------------------------------------------------------
```

Figura 5.4: Exemplo de dados registrados como resultado da execução de uma *Task*

- o nome da *Task* em execução;
- nome da *Action* em execução;
- o comando executado;
- o código de saída coletado após a execução de um comando;
- saída de erro coletado do *standard error*;
- saída de comando coletada do *standard out*.

Na próxima seção são apresentados detalhes de implementação do protótipo desenvolvido para esse trabalho.

5.4 Detalhes de implementação do protótipo iSRE

O protótipo do iSRE foi implementado utilizando a linguagem Java, versão 1.8. Com exceção das bibliotecas externas citadas nessa seção, toda a implementação fez uso das APIs nativas providas pela linguagem.

Para a implementação do serviço web *REST*, que escuta continuamente pela chegada de eventos, foi utilizada a biblioteca Jersey [28]. A biblioteca Jersey [28] implementa a especificação JAX-RS [27], que é a *API* referência oficial Java para criação de serviços web *REST*. O serviço recebe os dados referentes a um evento e envia uma resposta confirmando que o evento foi registrado ou não registrado em função de algum problema em seu conteúdo.

```
[2017-01-23 14:28:52:946]  2,Get hostname and date,hostname;date,0,,mica.dcc.ufmg.br
Tue Jan 23 14:28:52 BRST 2018
```

Figura 5.5: Log em uma linha com conteúdo associado a execução de uma *Task*

O formato padrão dos eventos enviados ao iSRE deve ser JSON [18]. O iSRE define uma classe chamada *Event*, uma *Java POJO (Plain Old Java Object)*, modelo padrão para classes que define somente um conjunto de variáveis e um número restrito de métodos, que possui uma série de variáveis que representam campos do conteúdo de um evento. Ao chegar ao iSRE, o conteúdo de eventos válidos, no formato JSON [18], formato recebido, é utilizado para uma instanciar um objeto *Event*. Por meio de uma conversão dos dados no formato JSON [18] para o formato da classe alvo, *Event*, o iSRE cria um objeto em memória que representa o conteúdo do evento recebido. Para efetuar essa conversão, é utilizada a biblioteca Jackson [11]. O objeto é então inserido em um fila para ser processado nos demais componentes da plataforma.

A biblioteca Jackson[11] também é utilizada para converter os arquivos de *Actions*, *Connections*, *Conversors* e *Matchers*, que são definidos pelo usuário no formato JSON[18], para classes Java *POJO* correspondentes. Cada uma dessas abstrações, *Action*, *Connection*, *Conversor* e *Matcher*, possui definida no iSRE uma classe Java *POJO* correspondente. Para cada arquivo de *Action*, *Connection*, *Conversor* e *Matcher* definido pelos usuários, o iSRE efetua a sua leitura ao ser inicializado, converte seu conteúdo e instância um objeto da classe Java POJO correspondente. Esses objetos são armazenados em um contêiner HashMap [25], que permite a recuperação em tempo constante. Para cada abstração de *Action*, *Connection*, *Conversor* e *Matcher* existe um contêiner HashMap [25] correspondente.

Para efetuar casamento de padrão, por exemplo, na análise do conteúdo dos eventos, o iSRE utiliza as funcionalidades providas pela classe nativa *Pattern* [26] para compilar a expressão regular em conjunto com as funcionalidades providas pela classe nativa *Matcher* [24], utilizada para interpretar expressão regular compilada e efetuar o casamento de padrão.

Para fins experimentais, o protótipo do iSRE estabelece conexões com objetos remotos para executar comandos, definidos nas *Tasks*, por meio do protocolo SSH. O iSRE utiliza as funcionalidades providas pela biblioteca JSCH [17]. Essa biblioteca implementa o protocolo SSH2[34] utilizando Java de forma nativa. A biblioteca JSCH[17] também provê as funcionalidades utilizadas para se efetuar a cópia de *scripts* em arquivos locais para componentes remotos em operações *script* definidas nas *Tasks*. A transferência dos arquivos é efetuada por meio do protocolo *SCP (Secure Copy)*.

O envio de emails, definidos em operações *email* nas *Tasks*, utiliza as funcionali-

dades providas pela *API* JavaMail para envio de emails. A hospedagem da aplicação é feita em um servidor Apache Tomcat[13].

5.5 Sumário

Nesse capítulo foram descritos alguns detalhes a respeito dos principais componentes do iSRE e a forma como estão implementados. São apresentados detalhes sobre os componentes que recebem e convertem eventos, analisam o conteúdo dos eventos e selecionam os procedimentos a serem executados, gerenciam a execução desses procedimentos e atualizam os logs que registram dados da execução. Além disso, apresentamos detalhes de implementação do protótipo, como a linguagem e bibliotecas utilizadas.

No próximo capítulo apresentamos a metodologia utilizada para realizar a avaliação experimental, seus objetivos e a descrição dos testes de validação funcional e de desempenho que foram planejados e executados. Também é apresentada uma descrição do ambiente de testes utilizado na avaliação experimental, demonstrando como esse ambiente foi configurado para acomodar os diferentes testes executados. Além disso, apresentamos os resultados obtidos com os testes executados.

Capítulo 6

Avaliação experimental

Esse capítulo apresenta a metodologia utilizada para realizar a avaliação experimental, seus objetivos e a descrição dos testes de validação funcional e de desempenho que foram planejados e executados nesse trabalho. Também é apresentada uma descrição do ambiente de testes utilizado na avaliação experimental, demonstrando como esse ambiente foi configurado para acomodar os diferentes testes executados. Por fim, apresentamos os resultados obtidos com os testes executados.

6.1 Metodologia

A metodologia utilizada para realizar a avaliação experimental do protótipo do iSRE é divida nas seguintes etapas:

1. definição dos objetivos a serem atingidos com a avaliação experimental;
2. descrição, execução e avaliação dos testes funcionais;
3. descrição, execução e avaliação dos testes não funcionais.

6.2 Objetivos

Os objetivos principais da realização da avaliação experimental do protótipo e arquitetura do iSRE são:

1. mostrar na prática como as principais funcionalidades oferecidas pelo iSRE podem ser utilizadas;

2. avaliar o comportamento do iSRE do ponto de vista funcional, ou seja, validar se as funcionalidades disponíveis no iSRE executam conforme esperado;

3. avaliar o comportamento do iSRE do ponto de vista não funcional, validando o desempenho do protótipo em resposta a execução de cargas de trabalho.

Nas próximas seções são apresentadas detalhes descritivos, de execução e de avaliação dos testes funcionais e testes não funcionais que constituem a avaliação experimental do protótipo proposto nesse trabalho.

6.3 Validação das funcionalidades

Nessa seção são apresentados detalhes sobre os testes que validam como o protótipo iSRE se comporta do ponto de vista funcional, os quais visam avaliar as seguintes funcionalidades: interpretação e execução de políticas de resposta a eventos definidas pelo usuário, recebimento e processamento de eventos, validação e alteração de estado em um componente remoto do sistema em rede, verificando assim se o protótipo implementado está alinhado as especificações e objetivos definidos no trabalho. Para execução dos testes funcionais foi elaborado um evento teste, descrito em detalhes na próxima subseção. As demais subseções descrevem a política de resposta ao evento, a configuração do ambiente de testes, a configuração dos testes funcionais e a apresentação dos resultados.

6.3.1 Evento: máquina não responde na rede

O cenário simulado para o evento pode ser descrito da seguinte forma: uma máquina que compõem um sistema em rede fica sem responder comandos probatórios por um tempo que excede o limite considerado adequado pelo time de suporte a rede, isso causa a geração de um evento que alerta a respeito de um potencial problema. Para validação do evento é esperado que ocorra uma validação para verificar se o servidor continua sem responder e caso continue, procedimentos de escalada são iniciados para que reparos sejam efetuados visando a remediação da causa do problema. Caso a máquina volte a responder é esperado que se estabeleça uma conexão com o servidor e que alguns dados básicos sejam coletados com o objetivo de se certificar que o servidor está operando conforme esperado.

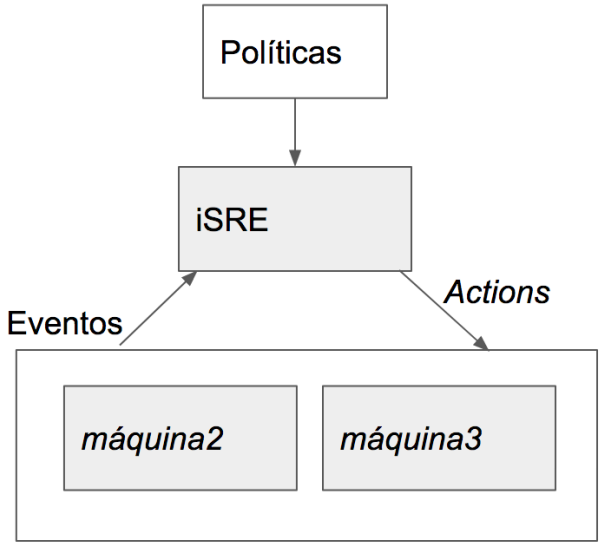

Figura 6.1: Estrutura ambiente de testes configurado para validação de funcionalidades

6.3.2 Configuração e descrição do ambiente de testes

O ambiente de testes configurado para execução dos testes funcionais é composto por três máquinas. Todas as máquinas executam o sistema operacional Linux e possuem suporte para execução de comandos *shell*. Além disso, possuem instalado um servidor *SSH* para receber conexões via *SSH*. A figura 6.1 ilustra como o ambiente de testes foi estruturado para acomodar os testes. A seguir são descritas as características funcionais de cada máquina disponível no ambiente de testes configurado:

- **máquina1:** hospedar o protótipo iSRE que recebe e processa os eventos;
- **máquina2:** máquina alvo dos eventos teste, que são enviado para o iSRE, hospedado na *máquina1*;
- **máquina3:** servidor de ponte para máquinas não acessíveis de modo direto através da *máquina1*;

6.3.3 Configuração da política de resposta a evento

Uma *Action* foi definida com o objetivo de validar se a máquina listada no evento teste continua sem responder na rede. Se a máquina continua sem responder, testes adicionais são executados e se ainda assim não houver sucesso na comunicação com o servidor afetado pelo problema, é enviado um email que detalha esse estado, juntamente de alguns dados coletados durante a execução da *Action*. Se a máquina responder, é efetuada uma conexão remota com a mesma e informações básicas são coletadas e

posteriormente enviadas por email para os interessados. Algumas das funcionalidades que são oferecidas pelo iSRE e que estão declaradas nas *Tasks* definidas na *Action* são:

- **variáveis de contexto:** a variável de contexto *hostname* é utilizada pelo *Matcher* para assinalar o endereço da máquina que não responde na rede, extraído do conteúdo do evento teste, e utilizado durante a execução da *Action*. Outras *variáveis de contexto* também são utilizadas na *Action* para se efetuar validação de controle de fluxo por meio das *Conditionals* e para se executar a passagem de parâmetros por referência para outras *Actions* invocadas em tempo de execução;

- **operação *local*:** utilizada para obter o *shell* da máquina que hospeda o iSRE, com objetivo de executar comandos *ping* para validar se é possível alcançar a máquina listada no evento teste, que conforme listado no evento, não responde comandos probatórios na rede;

- **reutilização de *Actions*:** referência para uma *Action* que agrupa um conjunto de comandos que podem ser reutilizados por outras *Actions*. É utilizada, por exemplo, uma *Action* que define *Tasks* que coletam informações básicas da máquina alvo de execução (*hostname*, data e hora do sistema, *uptime*);

- **operação *email*:** essa operação é utilizada para enviar email para uma lista de endereços declarados em uma variável da *Action*, e informa sobre o estado da execução da *Action* e das respectivas *Tasks*;

- ***conditional*:** diversas *Conditionals* são utilizadas na *Action* utilizada no teste. Um exemplo é o de uma *Conditional* que avalia a saída do comando *ping* e direciona o fluxo de execução da *Action* com base nesse conteúdo. Avalia-se por meio de casamento de padrão e expressões regulares a saída do comando *ping* executado utilizando a máquina que não responde na rede, listada no evento teste como alvo. O conteúdo de duas saídas possíveis do comando *ping*, o servidor está sendo alcançado ou não está sendo alcançado, fazem com que seja possível definir qual será a próxima *Task* à ser executada no fluxo de execução da *Action*;

- **operação *conexão*:** estabelece uma conexão com a máquina alvo listada no evento teste, se essa estiver alcançável. Também é utilizada para estabelecer conexão com a *máquina3*, que serve como ponte para máquinas não alcançáveis a partir da *máquina1*.

Para se acionar a execução da *Action*, configurada para os testes, foi configurado um *Matcher* que possui duas regras de casamento de padrão a serem satisfeitas em

tempo de execução quando o evento teste for recebido e analisado. Uma regra define que uma *string* deve estar presente no campo de descrição do evento. A *string* deve possuir o seguinte conteúdo: *server does not respond to ping probes*. A segunda regra estabelece que o campo *hostname* do evento teste deve possuir conteúdo composto por um ou mais caracteres alfanuméricos. Em tempo de execução, um *Extractor* obtém o conteúdo desse campo e o assinala para uma variável do contexto da *Action*.

Adicionalmente, foram criadas duas *Connections*. Uma *Connection* que armazena dados de acesso ao servidor *máquina2* foi criada e referenciada no *Matcher*. Outra *Connection*, que armazena dados de acesso ao servidor *máquina3* foi criada e declarada na *Action*. Essa *Connection* é utilizada em tempo de execução para se extrair os dados de conexão ao servidor ponte para um ambiente não acessível diretamente pelo servidor *máquina1*.

6.3.4 Configuração dos testes

Foram configurados e executados 5 tipos de teste para se avaliar alguns dos aspectos funcionais do iSRE, utilizando o evento previamente descrito nessa seção e duas variações. Também variamos o estado da *máquina2*. Para todos os testes, detalhados a seguir, um evento teste é gerado e a *máquina2* é listada como máquina alvo que não responde a comandos probatórios na rede. Cada teste foi executado mais de uma vez.

- **Teste 1:** a *máquina2* é desligada da rede e não responde aos comandos probatórios de forma proposital;
- **Teste 2:** a *máquina2* está respondendo a comandos probatórios, mas gera-se o evento com intuito de validar o comportamento funcional do iSRE e *Action* configurada;
- **Teste 3:** a *máquina2* é colocada em um ponto da rede no qual possui rota direta somente para a *máquina3*. O objetivo é simular situações onde o iSRE não possui rota direta para se comunicar com um componente remoto na rede e deve utilizar um componente intermediário para estabelecer conexão com o componente alvo;
- **Teste 4:** é gerado um evento em que o campo destinado aos dados de *hostname* não está preenchido de forma a satisfazer uma das regras de casamento de padrão definida no *Matcher* configurado para o teste, ou seja, o conteúdo do campo não possui um ou mais caracteres alfanuméricos. Dessa forma, simulamos uma situação em que o iSRE não deve acionar a *Action* configurada para o teste, pois nem todas as regras de casamento de padrão definidas no *Matcher* são satisfeitas;

- **Teste 5:** é gerado um evento aleatório, com conteúdo que não tem relação com o evento teste. A *Action* configurada para o teste não deve ser acionada, porque nem todas as regras de casamento de padrão definidas no *Matcher* são satisfeitas.

6.3.5 Resultados

A seguir são apresentados os resultados obtidos para cada um dos testes listados na subseção anterior.

- **Teste 1:** o evento é recebido e a *Action* disponível para tratar o evento teste é acionada para tratá-lo, uma vez que todas as regras de casamento de padrão listadas no *Matcher* configurado para os testes foram satisfeitas. Ao iniciar a execução, um comando *ping* é executado a partir da *máquina1*, utilizando as funcionalidades providas pela operação *local*, para validar se a *máquina2* continua em um estado não alcançável. A saída da execução desse comando é armazenada em uma variável. Ao constatar que a *máquina2* continua em um estado não alcançável, a *Action* executa *Tasks* para estabelecer conexão com a *máquina3*. A partir da *máquina3* é executado o comando *ping* que valida se a *máquina2* continua em um estado não alcançável. Constata-se que *máquina1* continua sem responder. Por fim, o fluxo de execução é direcionado para a execução de uma *Task*, que por meio da *operação email*, envia um email a uma lista de endereços de email informando que a *máquina2* continua em um estado não alcançável. Também são incluídos *logs* coletados durante a execução dos comandos *ping*. A execução leva em média 45 segundos;

- **Teste 2:** o evento é recebido e a *Action* disponível para tratar o evento teste é acionada para tratá-lo, uma vez que todas as regras de casamento de padrão listadas no *Matcher* configurado para os testes foram satisfeitas. Ao iniciar a execução, um comando *ping* é executado a partir da *máquina1*, utilizando as funcionalidades providas pela operação *local*, para validar se a *máquina2* continua em um estado não alcançável. A saída da execução desse comando é armazenada em uma variável. Ao constatar que a *máquina2* está um estado alcançável, o fluxo de execução é direcionado para executar *Tasks* que coletam informações básicas na *máquina2*. Estabelece-se uma conexão com a *máquina2*, onde são coletadas algumas informações básicas das configurações do sistema (*hostname*, *date*, *uptime* e configurações de rede com *ifconfig*). Os comandos para coleta de dados de configuração básica da máquina estão definidos em uma *Action* que é referenciada a partir da *Action* configurada para os testes, utilizando recursos

providos pelo iSRE para reuso de *Actions*. Por fim, um email é enviado para uma lista de endereços de email informando que o servidor está respondendo na rede novamente. No email, também são incluídas as informações básicas coletadas na *máquina2*. A execução leva em média 37 segundos.

- **Teste 3:** o evento é recebido e a *Action* disponível para tratar o evento teste é acionada para tratá-lo, uma vez que todas as regras de casamento de padrão listadas no *Matcher* configurado para os testes foram satisfeitas. Ao iniciar a execução, um comando *ping* é executado a partir da *máquina1*, utilizando as funcionalidades providas pela operação *local*, para validar se a *máquina2* continua em um estado não alcançável. A saída da execução desse comando é armazenada em uma variável. Ao constatar que a *máquina2* continua em um estado não alcançável, a *Action* executa *Tasks* para estabelecer conexão com a *máquina3*. A partir da *máquina3* é executado o comando *ping* que valida se a *máquina2* continua em um estado não alcançável. Ao constatar que a *máquina2* está sendo alcançada a partir da *máquina3*, o fluxo de execução inicia os procedimentos para se estabelecer conexão com a *máquina2* a partir da *máquina3*. Uma vez estabelecida a conexão, as informações básicas da *máquina2* são coletadas. Por fim, um email é enviado para uma lista de endereços de email informando que o servidor está respondendo na rede novamente. No email, também são incluídas as informações básicas coletadas na *máquina2*. A execução leva em média 47 segundos.

- **Teste 4:** o evento é recebido no iSRE e seu conteúdo é analisado de acordo com as regras de casamento de padrão definidas nos *Matchers*, visando identificar se existe uma *Action* a ser executada. Todos os *Matchers* são validados e verifica-se que não existe nenhum *Matcher* ao qual todas as regras de casamento de padrão são satisfeitas. Ao analisar o conteúdo do evento utilizando as regras de casamento de padrão definidas no *Matcher* configurado para o teste, o iSRE identifica que uma das duas regras listadas nesse *Matcher* não é satisfeita pois o campo *hostname* não possui um ou mais caracteres alfanuméricos. Sendo assim, nenhuma *Action* é acionada para em função da decorrência do evento;

- **Teste 5:** o evento é recebido no iSRE e seu conteúdo é analisado de acordo com as regras de casamento de padrão definidas nos *Matchers*, com o objetivo de se identificar se existe uma *Action* a ser executada. Todos os *Matchers* são validados e verifica-se que não existe nenhum *Matcher* ao qual todas as regras de casamento de padrão são satisfeitas. Ao analisar o conteúdo do evento utilizando

as regras de casamento de padrão definidas no *Matcher* configurado para o teste, o iSRE identifica que uma das regras de casamento de padrão não é satisfeita e imediatamente encerra a a análise para esse *Matcher*. Dessa forma, nenhuma *Action* é acionada para em função da decorrência do evento;

6.4 Avaliação de Desempenho

Nessa seção são apresentados testes que validam como o protótipo iSRE se comporta do ponto de vista de alguns aspectos não funcionais. São avaliadas as seguintes características:

- **escalabilidade:** capacidade do sistema em absorver uma quantidade crescente de carga sem demandar recursos adicionais;
- **confiabilidade:** habilidade do sistema em funcionar de forma esperada sob determinada condição de operação por um período específico de tempo;

A seguir são descritos os testes executados e os resultados obtidos.

6.4.1 Teste de carga

O teste de carga pode ser definido como o processo de se alocar demanda em um sistema de software com o objetivo de medir como esse sistema responde. No contexto dos testes conduzidos nesse trabalho, a demanda é constituída por eventos submetidos para registro e processamento no protótipo iSRE e a resposta é o tempo que o iSRE leva para registrá-los e processá-los.

Para execução dos testes, utilizamos três máquinas para enviar eventos ao protótipo do iSRE. O volume de eventos foi incrementado a cada iteração. As características dos eventos enviados ao protótipo são categorizadas da seguinte forma:

- ***tipo1*:** evento para o qual não existe no iSRE um *Matcher* em que todas as regras de casamento de padrão definidas são satisfeitas. Ou seja, o evento é recebido no iSRE e armazenado, nenhuma *Action* é acionada;
- ***tipo2*:** evento para o qual existe no iSRE um *Matcher* em que todas as regras de casamento de padrão são satisfeitas, com base em seu conteúdo, e uma *Action* é acionada e executada. Ou seja, o evento é recebido no iSRE, tem seu conteúdo armazenado e uma *Action* é acionada para execução. A *Action* possui dois comandos declarados, a serem executados na máquina alvo listada no evento.

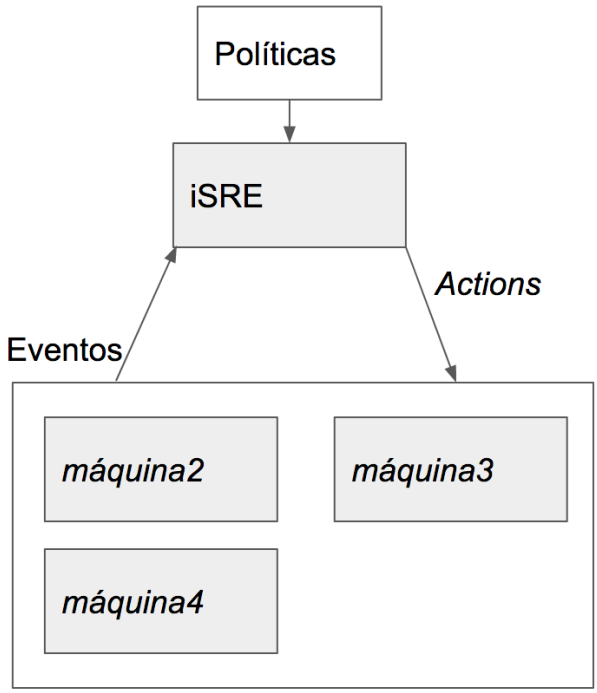

Figura 6.2: Estrutura ambiente de testes configurado para o teste de carga

Uma vez em execução, a *Action* estabelece conexão remota com a máquina alvo e executa os comandos *hostname* e *date*, conforme definido nas *Tasks*;

6.4.2 Configuração e descrição do ambiente de testes

O ambiente de testes é composto por quatro máquinas. Todas as máquinas executam o sistema operacional Linux e estão conectadas à mesma rede local. Além disso, possuem suporte para execução de comandos *shell* e têm instalado um servidor *SSH* para receber conexões via *SSH*. A figura 6.2 apresenta a estrutura de configuração do ambiente de testes para acomodar o teste de carga. A seguir detalhamos a função de cada uma das máquinas.

- ***máquina1:*** hospedar o protótipo iSRE que recebe e processa os eventos. A máquina possui 32 gigabytes de memória ram e uma cpu com 12 núcleos;
- ***máquina2:*** máquina utilizada para envio de eventos e como alvo da execução da *Action* de teste;
- ***máquina3:*** máquina utilizada para envio de eventos e como alvo da execução da *Action* de teste;

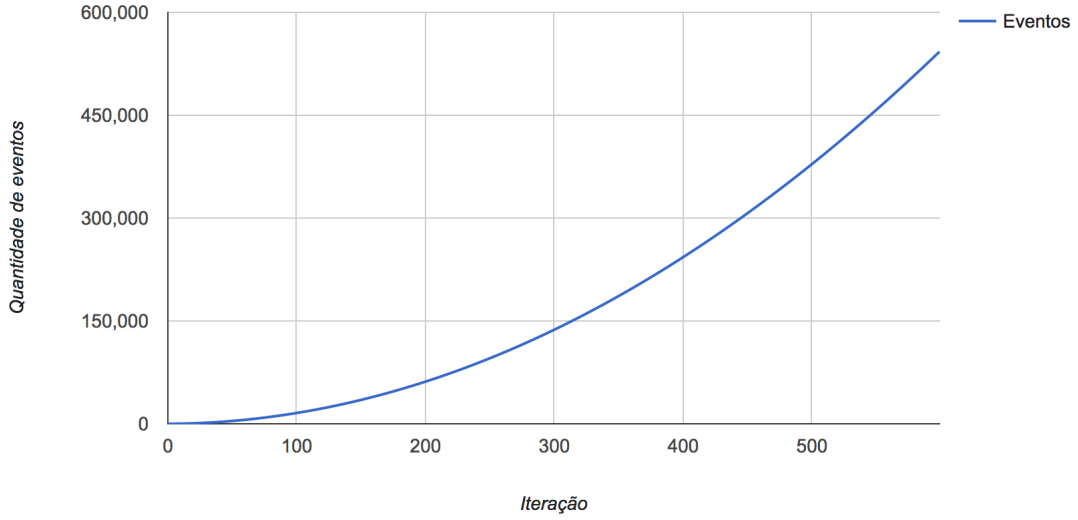

Figura 6.3: Evolução do número de eventos enviados para registro e processamento no iSRE

- ***máquina4:*** máquina utilizada para envio de eventos e como alvo da execução da *Action* de teste.

6.4.3 Configuração dos testes

Os testes são compostos por 600 iterações. Ao final de cada iteração, o número total de eventos a ser enviado durante a próxima iteração é incrementado em 1. Inicialmente, 2 eventos são enviados a partir de cada máquina. Na segunda iteração, são enviados 3 eventos de cada máquina. Na última iteração, são enviados 601 eventos de cada máquina.

O número total de eventos enviados a partir de cada máquina é de 180900, totalizando 542700 eventos enviados se somarmos os eventos enviados a partir de cada uma das três máquinas. O intervalo entre o fim de um iteração e o início de outra iteração é de 01 segundo. A figura 6.3 ilustra a evolução do número de eventos enviados para registro e processamento no iSRE como parte dos testes. A cada evento enviado, a probabilidade desse evento ser um evento *tipo2* é de 1,12%.

Ao executar os testes de carga, medimos o tempo de registro de cada evento, o tempo em processamento de cada evento e o consumo de CPU e memória do processo associado ao iSRE durante a execução dos testes. O tempo de registro de cada evento é medido utilizando-se a diferença entre o tempo de envio do evento e o tempo de recebimento da resposta do iSRE informando que o evento foi registrado. O tempo em processamento de cada evento é medido pela diferença entre o tempo em que o evento

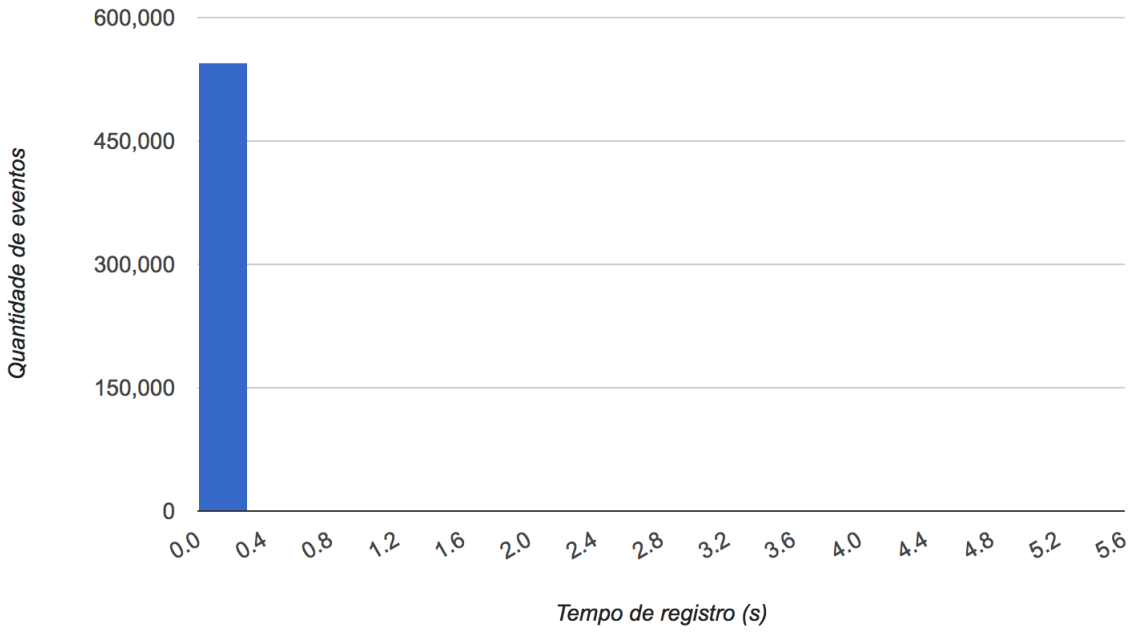

Figura 6.4: Tempo de registro de eventos no iSRE

é registrado no iSRE e o tempo em que esse evento tem seu processamento finalizado. O processamento consiste em operações como transformação dos dados do evento, análise de aplicabilidade de *Actions*, execução de *Actions* se aplicável, armazenamento dos dados e outras operações.

Por fim, efetuamos uma segunda bateria de testes, denominada *stress*, na qual enviamos um total de 15983645 eventos a partir das três máquinas do ambiente de testes. Não foi adicionado tempo de espera entre o envio de um evento e outro. O tempo de intervalo entre o envio de um evento e outro evento era dominado pela capacidade da máquina em enviar eventos de forma contínua, a rede estabelecer comunicação e o iSRE registrar o evento.

6.4.4 Resultados

Todos os 542700 eventos enviados ao iSRE foram registrados e processados com sucesso na plataforma. A figura 6.5 apresenta o histograma que mostra o tempo para registro de um evento na plataforma. Dos 542700 eventos registrados, apenas 11 não tiveram tempo de registro variando entre 0.0 e 0.4 segundos. Isso demonstra que o tempo de registro de eventos possui pouca variação mesmo quando aumentado o número de eventos por um período contínuo.

Ao analisarmos o tempo em que esses eventos permaneceram em processamento na plataforma, identificamos que a grande maioria, 524154 eventos, permanece em

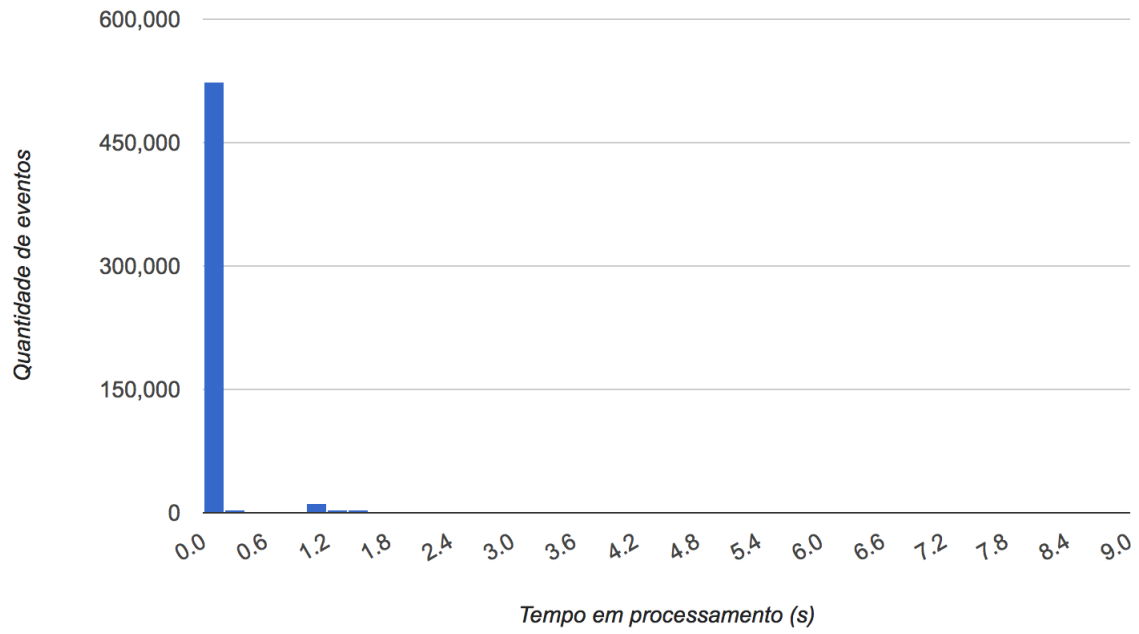

Figura 6.5: Tempo em processamento dos eventos registrados no iSRE

processamento entre 0.0 e 0.6 segundos. Esses dados são apresentados na figura 6.5. Uma outra faixa relevante dos dados comporta 14093 eventos, que ficaram em processamento entre 1.2 e 1.8 segundos. Nessa faixa de tempo de processamento observamos os eventos que possuem *Action* associada, as quais foram executadas com sucesso. Podemos afirmar que o iSRE é capaz de processar um número crescente de eventos, em sequência, sem que o tempo de processamento varie de forma considerável.

Durante a execução dos testes, foram medidos o consumo de CPU e memória do processo associado ao iSRE. A figura 6.6 apresenta a porcentagem de consumo referente a um núcleo de CPU e a porcentagem de consumo de memória da máquina. Podemos observar que o consumo de um núcleo de CPU sofreu pouca variação durante a execução, apresentando um pico de consumo de 135%, correspondente a um núcleo de CPU e 35% de um segundo núcleo. O consumo de memória subiu a medida que o teste evoluiu, porém de forma inferior a carga submetida.

Os resultados da bateria de testes denominada *stress* e exibidos nas figuras 6.7, 6.8 e 6.9 demonstram que mesmo aumentando consideravelmente o número de eventos enviados a quantidade de tempo para registro se mantém estável. A quantidade de tempo exigido para processamento se mantém estável para a grande maioria dos eventos, porém apresenta um quantidade considerável de eventos em que o tempo em que os eventos permanecem em processamento é muito maior do que no teste apresentado anteriormente. Como foram utilizadas somente 3 máquinas no teste e a quantidade de

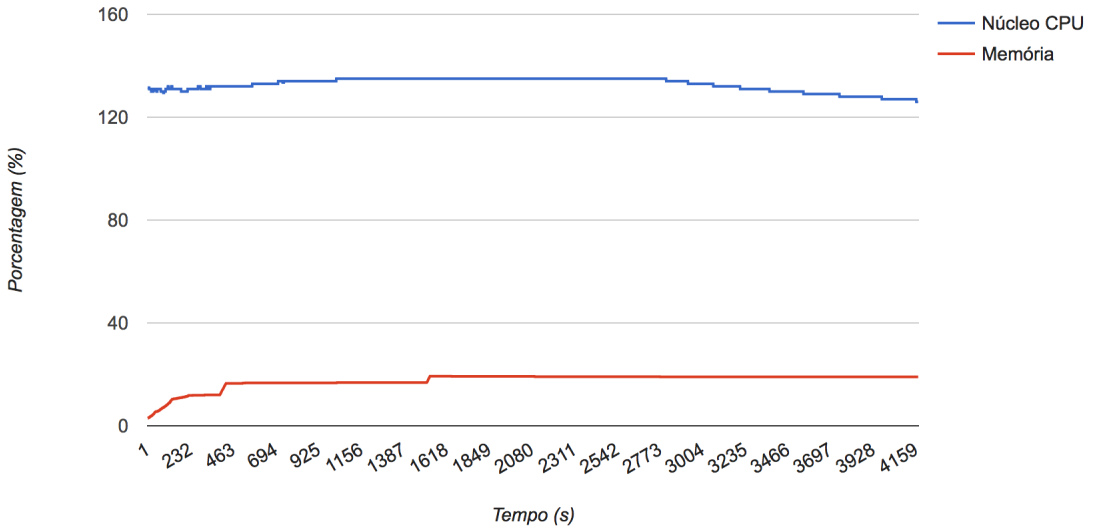

Figura 6.6: Consumo de núcleo de CPU e memória durante a execução dos testes

eventos enviado é grande, o tempo de espera de I/O domina o tempo de processamento dos eventos. O consumo de CPU diminui a medida do tempo, também em função da espera de I/O. O consumo de memória não cresce de forma proporcional ao número de eventos.

Com base nos dados gerados por meio dos testes de carga, podemos afirmar que o iSRE é capaz de absorver uma quantidade crescente de carga sem demandar recursos adicionais e funciona conforme esperado sem apresentar erros ou variações de desempenho consideráveis.

6.5 Sumário

Nesse capítulo apresentamos a metodologia utilizada para realizar a avaliação experimental, seus objetivos e a descrição dos testes de validação funcional e de desempenho que foram planejados e executados. Também é apresentada uma descrição do ambiente de testes utilizado durante a execução dos diferentes testes que compõem a avaliação experimental, demonstrando como esse ambiente foi personalizado para dar suporte a cada teste. Além disso, apresentamos os resultados obtidos com os testes executados.

No próximo capítulo são apresentados as considerações finais desse trabalho e uma discussão sobre potenciais oportunidades a serem explorados por meio de trabalhos futuros.

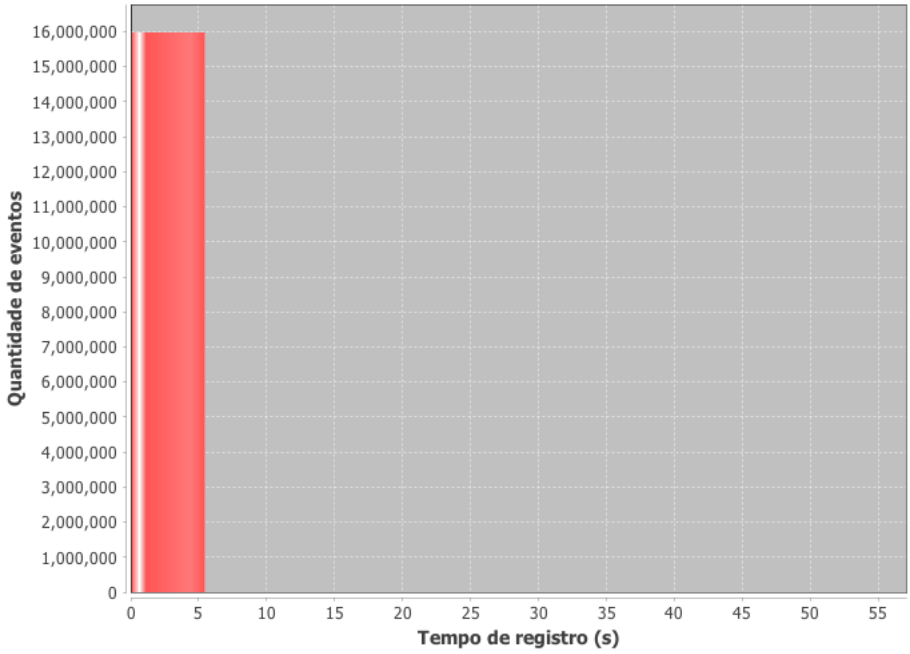

Figura 6.7: Tempo de registro de eventos no iSRE durante o teste *stress*

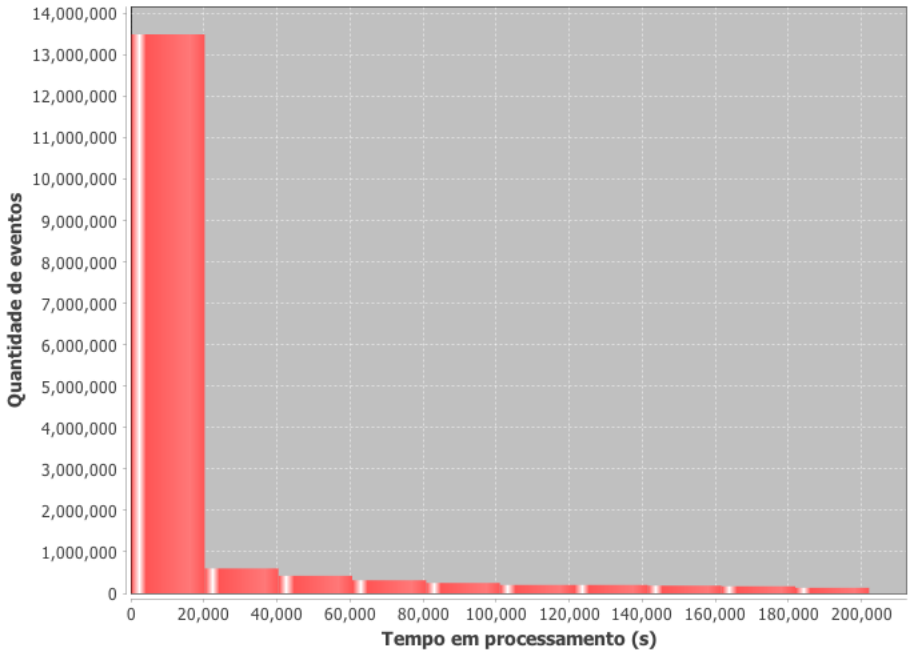

Figura 6.8: Tempo em processamento dos eventos registrados no iSRE durante o teste *stress*

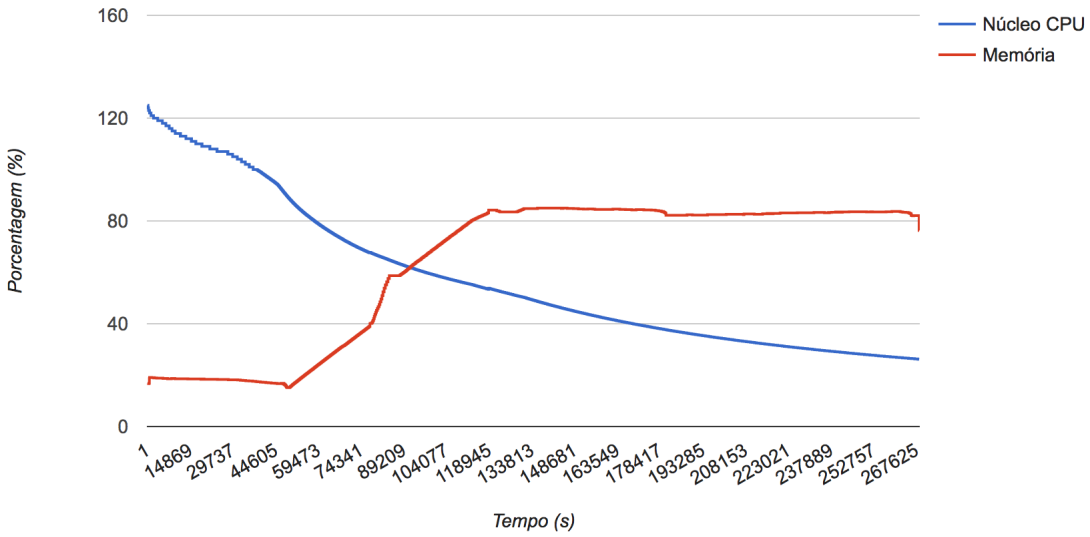

Figura 6.9: Consumo de núcleo de CPU e memória durante a execução do teste *stress*

Capítulo 7

Conclusão

Ao passo que o número de componentes de um sistema em rede cresce, ficam mais interconectados e diversificados, a operação, manutenção e evolução desse sistema se torna mais desafiadora. Problemas ocorridos em um ou mais componentes que estão em operação podem causar prejuízos materiais e imateriais às organizações. Por meio de atividades de gerenciamento, tarefas de prevenção e tratamento de problemas podem minimizar impactos causados por falhas ocorridas na execução dos componentes de um sistema em rede. Adicionar mais pessoas as atividades de operação, manutenção e evolução pode tornar os custos de gerenciamento proibitivos. Uma opção nesse contexto seria adotar ferramentas que viabilizem os princípios de computação autonômica e *RPA*, que adicionam capacidade de prevenção e atuação em problemas de forma automatizada e escalável. Porém, as ferramentas observadas na indústria e academia possuem algumas características que limitam a utilização de forma prática.

As ferramentas observadas se encaixam em um conjunto de instrumentos que apoiam atividades de operação, manutenção e evolução de sistemas em rede e algumas das limitações observadas são: capacidade restrita de interoperabilidade com outros sistemas de identificação de mudanças de estado em componentes de um sistema em rede; mecanismos incipientes de exploração dos dados e contexto do conteúdo dos eventos; funcionalidades que permitem o reuso de procedimentos são escassas; descentralização do controle sobre os procedimentos. Com essas limitações mapeadas, juntamente de requisitos derivados de observações, identificamos algumas oportunidades e desafios interessantes de serem explorados nesse trabalho.

Esse trabalho apresentou o projeto de uma plataforma de execução automatizada e controlada de procedimentos em um ou mais componentes de um sistema em rede, chamada iSRE, que prevê a interoperabilidade com diversas ferramentas de monitoração e predição de problemas, e com capacidade de escalar para suportar aumento nas

demandas de execução. O protótipo experimental do iSRE demonstra que pode ser utilizado como uma ferramenta que adiciona valor ao escopo de atividades de operação, manutenção e evolução dos componentes de um sistema em rede, por exemplo, permitindo aos usuários a definição de procedimentos que são executados de forma automática e controlada. Além disso, apresentamos uma avaliação qualitativa realizada junto a um grupo de profissionais a respeito da relevância do tema e do interesse em ferramentas e iniciativas que se propõem a suportar a execução automática e controlada de procedimentos de suporte a eventos em um ambiente de sistemas em rede, que permitiu confirmar o interesse no tema e validar alguns dos requisitos trabalhados na plataforma.

Um protótipo da plataforma foi implementado, e experimentos foram realizados considerando um ambiente de sistemas em rede em que eventos de mudança de estado que indicam potenciais problemas foram inseridos de forma proposital para validar se o protótipo implementado no trabalho atende de forma correta aos requisitos levantados. Os testes permitiram validar as funcionalidades e o desempenho da plataforma.

Os resultados derivados dos experimentos apresentados nos testes de validação funcional e de desempenho, efetuados utilizando o protótipo da plataforma, mostram que o iSRE consegue obter desempenho satisfatório ao que se propõe nos requisitos. Assim, podemos concluir que o iSRE pode ser uma opção viável em ambientes de sistemas em rede que demandam um elemento para automatizar a execução de processos de prevenção, diagnóstico, e remediação de problemas de tempo de execução nos componentes desse sistema.

7.1 Trabalhos Futuros

Algumas das oportunidades de trabalhos futuros se apresentam no sentido da exploração e aplicação de inteligência artificial em alguns dos componentes do iSRE. Por exemplo, para se analisar o conteúdo dos eventos em tempo de execução e escolher o procedimento mais adequado a ser executado, sem a necessidade de se basear em regras de casamento de padrão definidas pelos usuários.

Outro exemplo consiste em integrar a plataforma componentes de inteligência artificial com capacidade de entender o conteúdo dos eventos e desenvolver, com base nesse conteúdo e em objetivos de alto nível definidos pelos usuários, a política de resposta de maneira autônoma. Dessa forma, cria-se uma plataforma ainda mais alinhada aos princípios de computação autônomica ilustrados em [19].

Além disso, podem ser explorados trabalhos futuros na direção de se expandir ou

otimizar as funcionalidades oferecidas pela plataforma. Por exemplo, adicionando-se um número maior de funcionalidades que tornem mais produtiva a etapa de definição de políticas de resposta a eventos pelos usuários. Do ponto de vista de otimização de funcionalidades, a otimização de aspectos relacionadas a desempenho podem permitir níveis de escalabilidade ainda mais interessantes. No que diz respeito a alta disponibilidade, adicionar um gerenciador de fila de fila de eventos que apresente capacidade de armazenar os eventos de forma distribuída e tolerante a falhas.

Anexo A

Roteiro das Entrevistas

Dados Pessoais: Nome | Idade | Formação | Profissão

No contexto dessa pesquisa, vamos nos referir a incidentes/*tickets*/eventos/alertas/chamados como eventos

Bloco Perfil do Entrevistado

1) Quais são as principais atividades exercidas por você atualmente e qual a sua experiência profissional?

a - Tempo de atuação

2) No geral, quais são as características dos ambientes de TI que você possui contato, seja supervisionando os trabalhos de uma equipe ou os administrando de forma direta?

a - É composto por vários servidores que se comunicam em rede?

b - Possui alguma ferramenta de monitoração ou predição de eventos?

c - Possui alguma ferramenta de registro de eventos?

d - Qual quantidade média de eventos gerados mensalmente?

3) Quais os procedimentos adotados quando ocorrem eventos nesses ambientes?

Bloco Percepções e Experiências a Respeito do Tema

1) Você se preocupa em tornar a operação mais eficiente por meio da adoção de soluções que se propõem a automatizar a execução de tarefas no ambiente de TI?

2) Você acredita que existem procedimentos de suporte a eventos que poderiam ser executados de forma automática com o auxílio de uma ferramenta que se propõem a esse fim? Por exemplo, procedimentos para remediação de eventos de 1o e 2o nível.

3) Os ambientes de TI que você possui contato, seja supervisionando os trabalhos de uma equipe ou os administrando de forma direta, utilizam ou já utilizaram alguma fer-

ramenta que fornece mecanismos para tornar possível a execução de forma automática de procedimentos que efetuam suporte a eventos, por exemplo, que validam ou efetuam a remediação de eventos no ambiente de TI?

a - Descreva a experiência.

4) Quais são os requisitos que você considera importantes em uma ferramenta que se propõe a executar procedimentos de forma automática sobre eventos ocorridos no ambiente?

a - interoperabilidade

b - auditabilidade

c - facilidade de uso

d - controle

e - outro

5) Na sua opinião, quais são os principais desafios para se implementar uma solução que se propõe a automatizar a execução de procedimentos no ambiente de TI?

6) Você possui comentários adicionais sobre o tema dessa entrevista?

Anexo B

Termo de Consentimento de Participação

Título: Avaliação da percepção de profissionais de TI a respeito de soluções que se propõem a suportar a automatização de rotinas manuais típicas da operação de *datacenters*.
Data: outubro/2017
Instituição: Departamento de Ciência da Computação (DCC) / Universidade Federal de Minas Gerais
Pesquisadores Responsáveis: Prof. Renato A. Celso Ferreira (renato@dcc.ufmg.br) Achiles Caldeira Quintella Santana Junior (achiles.caldeira@dcc.ufmg.br)

Introdução: Este Termo de Consentimento contém informações sobre a pesquisa indicada acima. Para assegurar que você esteja informado sobre a sua participação nesta pesquisa, pedimos que leia este termo de Consentimento. Caso tenha alguma dúvida, não hesite em perguntar ao pesquisador responsável. Você também deverá assinar o termo do qual receberá uma cópia.

Objetivo da avaliação: O objetivo desta avaliação é identificar, do ponto de vista do profissional de TI com experiência em operações de datacenter, percepções e experiências a respeito da prospecção, planejamento, implementação e operação de soluções se propõem a suportar a automatização de rotinas manuais típicas da operação de datacenters, além de identificar potencial relevância do tema.

Informação geral sobre a pesquisa: A pesquisa é composta por uma entrevista, que contempla perguntas a respeito do tema e que devem ser respondidas pelo

entrevistado. As respostas são posteriormente analisadas pelo entrevistador.

Utilização dos dados coletados: Os dados coletados durante a avaliação serão utilizados para estudo o Método de Avaliação da Comunicabilidade (MAC). Quaisquer dados utilizados para publicação serão apresentados de forma a garantir o anonimato dos participantes da avaliação.

Privacidade: Informações que possam identificar os participantes da pesquisa não serão divulgadas. O seu nome não aparecerá em nenhum relatório. Caso deseje, poderá solicitar uma cópia dos dados gerados por você.

Se você decidir não participar na pesquisa: Você é livre para decidir, a qualquer momento, se quer participar ou não nesta pesquisa. Sua decisão não afetará sua vida estudantil e nem qualquer relacionamento com os avaliadores, professores ou a Instituição por trás desta.

Compensação: A participação nesta pesquisa é voluntária, e não será oferecida nenhuma remuneração aos seus participantes.

Se tiver algum problema ou se tiver outras perguntas: Se você tiver algum problema que pensa que pode estar relacionado com sua participação nesta pesquisa, ou se tiver qualquer pergunta sobre a pesquisa, poderá entrar em contato com os pesquisadores a qualquer momento pelo e-mail renato@dcc.ufmg.br; achiles.caldeira@dcc.ufmg.br.

Novas condições: Caso deseje, você pode especificar novas condições que devem ser atendidas para que você participe desta avaliação.

Consentimento Livre e Esclarecido (Acordo Voluntário) O documento mencionado acima descrevendo os benefícios, riscos e procedimentos da pesquisa "Avaliação da percepção de profissionais de TI a respeito de soluções que se propõem a suportar a automatização de rotinas manuais típicas da operação de datacenters." foi lido e explicado. Eu tive a oportunidade de fazer perguntas sobre a pesquisa, que foram respondidas satisfatoriamente. Eu estou de acordo em participar como voluntário.
Data:

Assinatura do participante:

Nome do participante:

Assinatura do pesquisador:

Nome do pesquisador:

Referências Bibliográficas

[1] Andrew S. Tanenbaum, M. v. S. (2007). *Distributed Systems: Principles and Paradigms*. CreateSpace Independent Publishing Platform; 2 edition.

[2] Armbrust, M.; Fox, A.; Griffith, R.; Joseph, A. D.; Katz, R.; Konwinski, A.; Lee, G.; Patterson, D.; Rabkin, A.; Stoica, I. et al. (2010). A view of cloud computing. *Communications of the ACM*, 53(4):50--58.

[3] Bigus, J. P.; Schlosnagle, D. A.; Pilgrim, J. R.; Mills III, W. N. & Diao, Y. (2002). Able: A toolkit for building multiagent autonomic systems. *IBM Systems Journal*, 41(3):350--371.

[4] Bouchenak, S.; Boyer, F.; Hagimont, D.; Krakowiak, S.; Mos, A.; De Palma, N.; Quema, V. & Stefani, J.-B. (2005). Architecture-based autonomous repair management: An application to j2ee clusters. Em *Reliable Distributed Systems, 2005. SRDS 2005. 24th IEEE Symposium on*, pp. 13--24. IEEE.

[5] Clemm, A. (2006). *Network management fundamentals*. Cisco Press.

[6] CollectD (2017). CollectD. `https://collectd.org`. Acessado em: 2017-05-01.

[7] Corsava, S. & Getov, V. (2003). Intelligent architecture for automatic resource allocation in computer clusters. Em *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, pp. 8--pp. IEEE.

[8] de Magalhães, J. P. F. (2013). *Self-healing Techniques for Web-based Applications*. Tese de doutorado, University of Coimbra.

[9] Enterprises, N. (2017). Nagios. `https://www.nagios.org`. Acessado em: 2017-06-01.

[10] Etsy (2017). StatsD. `https://github.com/etsy/statsd`. Acessado em: 2017-05-01.

[11] Faster XML (2017). Jackson. `https://github.com/FasterXML/jackson`. Acessado em: 2017-06-01.

[12] Fatema, K.; Emeakaroha, V. C.; Healy, P. D.; Morrison, J. P. & Lynn, T. (2014). A survey of cloud monitoring tools: Taxonomy, capabilities and objectives. *Journal of Parallel and Distributed Computing*, 74(10):2918--2933.

[13] Foundation, A. S. (2017). Apache Tomcat. `https://tomcat.apache.org`. Acessado em: 2017-06-01.

[14] GraphiteApp (2017). Graphite. `https://graphiteapp.org`. Acessado em: 2017-05-01.

[15] Horn, P. (2001). Autonomic computing: Ibm\'s perspective on the state of information technology.

[16] Huebscher, M. C. & McCann, J. A. (2008). A survey of autonomic computing—degrees, models, and applications. *ACM Computing Surveys (CSUR)*, 40(3):7.

[17] JCraft (2017). JSCH. `http://www.jcraft.com/jsch/`. Acessado em: 2017-06-01.

[18] JSON (2017). JSON. `http://www.json.org/json-pt.html`. Acessado em: 2017-06-01.

[19] Kephart, J. O. & Chess, D. M. (2003). The vision of autonomic computing. *Computer*, 36(1):41--50.

[20] Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558--565.

[21] Massie, M. L.; Chun, B. N. & Culler, D. E. (2004). The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing*, 30(7):817--840.

[22] Nicolaci-da Costa, A. M.; Leitão, C. F. & Romão-Dias, D. (2004). Como conhecer usuários através do método de explicitação do discurso subjacente (meds). *VI Simpósio Brasileiro sobre Fatores Humanos em Sistemas Computacionais, IHC*, pp. 47--56.

[23] Olups, R. (2010). *Zabbix 1.8 network monitoring*. Packt Publishing Ltd.

[24] Oracle (2017). Class Matcher. `https://docs.oracle.com/javase/8/docs/api/java/util/regex/Matcher.html`. Acessado em: 2017-05-01.

[25] Oracle Corporation (2017a). Class HashMap. https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html. Acessado em: 2017-05-01.

[26] Oracle Corporation (2017b). Class Pattern. https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html. Acessado em: 2017-05-01.

[27] Oracle Corporation (2017c). JAX-RS. https://jcp.org/aboutJava/communityprocess/final/jsr339/index.html. Acessado em: 2017-05-01.

[28] Oracle Corporation (2017d). JSCH. https://jersey.github.io. Acessado em: 2017-06-01.

[29] Services, A. W. (2017). Amazon CloudWatch. https://aws.amazon.com/pt/cloudwatch/. Acessado em: 2017-08-01.

[30] Silberschatz, A.; Galvin, P. B.; Gagne, G. & Silberschatz, A. (2012). *Operating System Concepts*, volume 9. Wiley.

[31] Valetto, G. & Kaiser, G. (2003). Using process technology to control and coordinate software adaptation. Em *Software Engineering, 2003. Proceedings. 25th International Conference on*, pp. 262--272. IEEE.

[32] W3C (2017). Web Services Architecture. https://www.w3.org/TR/ws-arch/. Acessado em: 2017-08-01.

[33] Willcocks, L. P.; Lacity, M. & Craig, A. (2015). The it function and robotic process automation.

[34] Ylonen, T. & Lonvick, C. (2006). The secure shell (ssh) protocol architecture.