SBC
Sociedade Brasileira de Computação

MINICURSOS DA ERSI

V ESCOLA REGIONAL DE SISTEMAS DE INFORMAÇÃO NOVA FRIBURGO - RJ

Organização:
Juliana Baptista dos Santos França
Tiago Cruz de França

Editora: Sociedade Brasileira de Computação

CEFET/RJ
campus Nova Friburgo

UFRRJ
UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO

Coordenadores

Juliana Baptista dos Santos França
Tiago Cruz de França

MINICURSOS DA ERSI

V ESCOLA REGIONAL DE SISTEMAS DE INFORMAÇÃO

Nova Friburgo - RJ

Realização

Sociedade Brasileira de Computação- SBC
Centro Federal de Educação Tecnológica Celso Suckow da Fonseca - CEFET/RJ
Universidade Federal Rural do Rio de Janeiro - UFRRJ
Universidade Federal do Rio de Janeiro - UFRJ
Prefeitura Municipal de Nova Friburgo - PMNF

Dados Internacionais de Catalogação na Publicação (CIP)

E612 Escola Regional de Sistemas de Informação (5. : 2018 : Nova Friburgo, RJ).

Minicursos da V Escola Regional de Sistemas de Informação [recurso eletrônico] : 24 a 26 de outubro de 2018, CEFET/RJ - Nova Friburgo / coordenadores Juliana Baptista dos Santos França, Tiago Cruz de França. – Rio de Janeiro : SBC, 2018.
1 recurso eletrônico. (171 p.)

Inclui bibliografia.
ISBN 978-85-7669-456-4 (e-book)

1. Tecnologia da informação. 2. Informática. 3. ERSI. I. França, Juliana Baptista dos Santos. II. França, Tiago Cruz de. III. Sociedade Brasileira de Computação. IV. Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ). V. Título.

CDD 004

Elaborado por Elaine Almeida – CRB-7/6465
Biblioteca – NCE - UFRJ

Prefácio

A Escola Regional de Sistemas de Informação do estado do Rio de Janeiro (ERSI-RJ) é um evento que reúne profissionais, professores e estudantes interessados em aprender e discutir problemas, soluções e conceitos relacionados a Sistemas de Informação. Os minicursos são atividades de curta duração (quatro horas) que fazem parte da programação da ERSI-RJ. Eles abordam temas relacionados a Sistemas de Informação com objetivo de proporcionar ao público da escola um ambiente de aprendizagem e discussão de tendências e desafios na área de Sistemas de Informação.

Este ano foram selecionados cinco minicursos entre onze propostas submetidas à ERSI-RJ. A seleção foi realizada por um comitê formado por seis avaliadores professores de Sistemas de Informação de diferentes instituições. As propostas foram avaliadas por todos os membros do comitê. Os critérios de seleção foram: relevância para o evento, expectativa de público, atualidade e conteúdo.

Os temas dos minicursos estão relacionados ao desenvolvimento de jogos digitais, a metodologia de pesquisa, aprendizado de máquina, otimização de operações em bancos de dados e testes de software. Os autores produziram o material de referência para o minicurso proposto. Esses documentos foram organizados em capítulos para compor este material que é a primeira edição de livro de minicurso da ERSI-RJ.

O primeiro minicurso (Capítulo 1), "Construção de Jogos Digitais Sérios para Processos de Serviços Públicos", adotou uma abordagem teórico-prática para apresentar conceitos de jogos digitais, jogos sérios e design de jogos, mostrando, na prática, como construir jogos digitais em um processo de serviço público real. O segundo minicurso (Capítulo 2), "Metodologia de Pesquisa de Estudo de Caso em Sistemas de Informação" apresentou e discutiu conceitos e exemplos dos benefícios da metodologia de estudo de casos para produzir conhecimento, validar e ajustar Sistemas de Informação em situações de uso real. O terceiro minicurso (Capítulo 3), "Introdução à Classificação Multirrótulo", apresentou conceitos e exemplos práticos da classificação automática de conteúdo por meio de características, exemplificando o uso prático da classificação multirrótulo em Sistemas de Informação. O quarto minicurso (Capítulo 4), "(Auto) Sintonia-fina em Sistemas de Bancos de Dados nas Organizações", mostrou como a (auto) sintonia fina agiliza o processamento de requisições de dados em bancos de dados no cenário de *Big Data,* por meio de conceitos e exemplos de aplicação prática. O quinto minicurso (Capítulo 5), "Testes de Desempenho de Software: Teoria e Prática" abordou como realizar testes de carga apresentando requisitos de desempenho, abordagens de planejamento de testes, consolidando terminologias e proporcionando experiências práticas de testes de desempenho.

Acreditamos que este material será útil em aulas de Sistemas de Informação; em discussões sobre novas abordagens de pesquisa suportando trabalhos atuais e futuros; e para apoiar a prática profissional. Parabenizamos e agradecemos aos autores dos minicursos. Agradecemos também ao Comitê de Seleção de Propostas pela dedicação e eficiência; ao Comitê Editorial pelo emprenho; e a CESI (Comissão Especial de Sistemas de Informação) da SBC pelo apoio para publicação deste livro.

Juliana Baptista dos Santos França e **Tiago Cruz de França** (UFRRJ)
Coordenadores de Minicurso da ERSI-RJ 2018
Rafael Elias de Lima Escalfoni (CEFET-RJ)
Coordenador Geral da ERSI e Coordenador Local de Minicursos da ERSI-RJ 2018

V Escola Regional de Sistemas de Informação do Estado do Rio de Janeiro

24 a 26 de Outubro de 2018

Nova Friburgo – RJ – Brasil

MINICURSOS

Promoção

Sociedade Brasileira de Computação – SBC

Coordenação da Escola Regional de Sistemas de Informação do Estado do Rio de Janeiro 2018

Rafael Elias de Lima Escalfoni – CEFET/RJ
Bruno Policarpo Toledo Freitas – CEFET/RJ

Coordenação de Painéis e Palestras

Eliezer Dutra Gonçalves – CEFET/RJ
Raimundo José Macário da Costa – UFRRJ

Coordenação de Sessões Técnicas e Pôsteres

Claudio Miceli de Farias – NCE/UFRJ
Marco André Abud Kappel – CEFET/RJ

Coordenação de Minicursos

Tiago Cruz de França – UFRRJ
Juliana França - UFRRJ

Coordenação da Feira de Robótica e Automação

Luís Cláudio Batista da Silva – CEFET/RJ
Fabrício Barros Gonçalves – IFF

Coordenação da Hackathon

Álvaro Ernesto Robles Rincón – NCE/UFRJ
Emanuele Jorge – IFRJ

Comitê de Organização

Rafaela Moreira Oliveira - CEFET/RJ
Gisela Bochner - CEFET/RJ
André Queiroz Ferreira de Mello - CEFET/RJ
Edvar Fernandes Batista – CEFET/RJ

Comitê de Programa

Alana Morais - IESP
Alessandro Copetti - UFF
Alexandre Sena - UERJ
Aline Morais - IESP
André Luiz de Castro Leal - UFRRJ
Angelica Dias - UFRJ
Áquila Ditzz - IFF
Bernardo Peralva - UERJ
Bruno Freitas - CEFET/RJ
Bruno Missi Xavier - IFES
Carlos Eduardo Pantoja - CEFET/RJ
Cláudia Cappelli - SBC
Claudio de Farias - UFRJ
Daniel França - UFPB
Daniel Paiva - UFF
Diego Brandão - CEFET/RJ
Elias S. Gonçalves - IME
Eliezer Dutra - UNIRIO
Emanuele Jorge - IFRJ
Fabiana Mendes - UnB
Fabrício Barros - IFF
Fábio Silveira Vidal - IFTO
Fernanda Baião - SBC
Flavia Santoro - SBC
Geiza Hamazaki - PUC-Rio
Geraldo Xexéo - UFRJ
Gizelle Vianna - UFRRJ
Henrique Prado de Sá Sousa - UFRJ
Hugo Cesar Carneiro - UFRJ
Isabel Cafezeiro - UFF
José Viterbo - UFF

José Ricardo Cereja - UNIRIO
Juliana França - UFRJ
Laci Mary Manhaes - UFF
Luis Orleans - UFRRJ
Luis Silva - CEFET/RJ
Marco Kappel - CEFET/RJ
Mônica Silva - UFRJ
Nilton Rizzo - UFRRJ
Rafael Costa - UFRJ
Rafael Escalfoni - CEFET/RJ
Rafael Lima de Carvalho - UFTO
Raimundo Macário Costa - UFRRJ
Renata Araújo - SBC
Ricardo Bernardo - IFES
Robson Silva - UFRRJ
Rodrigo Monteiro - UFF
Sergio Manuel Serra - UFRRJ
Tiago Cruz de França - UFRRJ

Comitê de Seleção de Propostas de Minicursos

Alessandro Cerqueira – CRF/RJ
Claudio Miceli de Farias - UFRJ
Henrique Prado de Sá Souza - UFRRJ
Juliana França - UFRRJ
Rafael Elias Escalfoni - CEFET-RJ
Tiago Cruz de França - UFRRJ

Comitê Editorial

Tiago Cruz de França – UFRRJ
Juliana França – UFRRJ
Rafael Elias Escalfoni – CEFET-RJ
Claudio Miceli de Farias - UFRJ
Henrique Prado de Sá Souza - UFRRJ

Sumário

Construção de Jogos Digitais Sérios para Processos de Serviços Públicos…….. 1

Tadeu Moreira de Classe (UNIRIO), Renata Mendes de Araújo (SBC) e Geraldo Bonorino Xexéo (UFRJ)

Metodologia de Pesquisa de Estudo de Caso em Sistemas de Informação……... 41

Nadja Piedade de Antonio (UNIRIO), Marcelo Fornazin (UFF), Renata Mendes de Araujo (SBC)

Introdução à Classificação Multirrótulo……..……... 68

Eduardo Corrêa Gonçalves (IBGE e ENCE/IBGE)

(Auto) Sintonia-fina em Sistemas de Bancos de Dados nas Organizações…… 111

Ana Carolina Brito de Almeida (UERJ e TRF2) e Sérgio Lifschitz (PUC-Rio)

Testes de Desempenho de Software: Teoria e Prática……... 139

Thiago Silva de Souza (SERPRO e UNIGRANRIO)

Chapter 1

Construção de Jogos Digitais Sérios para Processos de Serviços Públicos

Tadeu Moreira de Classe, Renata Mendes de Araujo, Geraldo Bonorino Xexéo

Abstract

The purpose of this course is to enable students to build a digital serious game based on a real public service process using the Construct 2 tool for game development. The course presents digital and serious games concepts, as well as business process concepts and process model elements, and the opportunities and challenges of designing these games. The game design process is based on a method of digital game design using public services process, which will be presented to the students.

Resumo

O objetivo deste minicurso é possibilitar que os alunos construam um jogo digital sério baseado em um processo de serviço público real, usando a ferramenta de criação de Jogos Construct 2. O curso apresenta os conceitos de jogos digitais e jogos sérios, as possibilidades e desafios do design destes jogos, bem como sucintamente os conceitos de processos de negócio e seus elementos constituintes. A construção destes tipos de jogos se baseia em um método de design de jogos digitais baseados em processos de serviços públicos, o qual também será apresentado ao público, a fim de contextualizar a etapa de codificação e desenvolvimento do jogo, foco deste minicurso.

1.1. Introdução

As tecnologias digitais estão em constante desenvolvimento, sendo cada vez mais presente na vida das pessoas e organizações. Seguindo tal tendência, as instituições, públicas ou não, precisam acompanhar esta evolução de modo a garantir a competitividade de seus negócios e/ou oferecer melhores serviços. Com o aumento do uso de tecnologias sociais, oportunidades inovadoras de diálogo entre organizações e seus clientes se apresentam, visando à melhoria de processos, ampliando a participação de pessoas tanto interna como externamente às organizações, na contribuição com tais melhorias [Pflanzl e Vossen 2014].

Neste sentido, há uma crescente necessidade de transparência e visibilidade dos processos organizacionais para uma maior interação entre as pessoas e as organizações. Para alcançar este objetivo, as pessoas antes de tudo precisam compreender e aprender sobre os processos organizacionais, sobretudo os de prestação de serviços. O uso de jogos digitais surge como uma alternativa em potencial a este ideal de aproximação, uma vez que jogos possuem a característica de engajamento de seus jogadores e têm sido utilizados como soluções para aprendizado em diversas áreas [Classe et al. 2016].

Em se tratando de contextos públicos, os jogos digitais podem representar uma nova maneira dos cidadãos compreenderem e aprenderem sobre os diversos processos de serviços prestados por entidades públicas, conscientizando-se sobre eles.

Este minicurso tem como foco os processos de prestação de serviços públicos e no design de jogos digitais baseados nestes processos. Neste curso os alunos irão construir um jogo digital sério, com uma única fase do gênero aventura. Este jogo irá se basear em um processo de prestação de serviço público real, o processo de Emissão do Cartão SUS[1], e terá como objetivo explicar os detalhes para que o cidadão possa solicitar e usufruir deste serviço. Para a construção do jogo será usada a ferramenta *Construct2*, sendo demonstrado o passo-a-passo para os alunos construírem o jogo digital proposto. Todos os materiais (assets, imagens, áudios, cenários etc.) necessários para a construção do game serão fornecidos. Ao final do minicurso os alunos saberão conceitos fundamentais sobre: i) prestação de serviços públicos; ii) elementos de modelos de processo, iii) conceitos e etapas para o design de jogos digitais, iv) atores envolvidos em design de jogos, v) principais *engines* de desenvolvimento de jogos digitais; e vi) uma visão do método de design de jogos digitais baseados em processos de serviços públicos, proposto por Classe, Araujo e Xexéo (2018).

As seções deste capítulo estão organizadas para a melhor compreensão das abordagens teóricas e práticas do curso. A Seção 1.2 aborda o referencial teórico do curso trazendo temas como democracia digital, modelagem de processos de negócio e jogos digitais. A seção 1.3 aborda o método de design de jogos baseados em processos de serviços públicos, apresentados as suas etapas. A seção 1.4, demonstra a criação de um jogo para o processo real de emissão do Cartão SUS. E finalmente, a seção 5 apresenta a conclusão do curso.

1.2. Referencial Teórico

Nesta seção são apresentadas as bases teóricas que motivaram este estudo, compreendendo as premissas e conceitos básicos envolvidos na proposta de design de jogos digitais baseados em processos de serviços públicos.

1.2.1. Democracia Digital

A democracia digital (também conhecida na literatura como democracia eletrônica ou ainda e-democracia) entende a internet como agente de transformação dos processos políticos e de prestação de serviços públicos tradicionais [Vedel 2006]. Silva [2005] define o conceito de democracia digital como "o conjunto de discursos, teorizações e experimentações que empregam Tecnologias de Informação e Comunicação (TICs) para mediar as relações políticas, tendo em vista as possibilidades de participação

[1] http://portalms.saude.gov.br/acoes-e-programas/cartao-nacional-de-saude

democrática nos sistemas políticos contemporâneos". Sendo esta última a definição que consideramos para este trabalho [Araujo e Magdaleno 2015].

É inegável que o uso de TICs ajuda na promoção do engajamento de cidadãos, na construção de políticas e na participação social, embora devamos reconhecer que a tecnologia é apenas um meio para viabilizar a solução, e não a solução por completo. Sendo assim, para a implementação da democracia digital, barreiras culturais, organizacionais, constitucionais, políticas e também tecnológicas devem ser transpostas [Kneuer 2016][Santos 2014].

Dentre um dos objetivos mais básicos dos governos se encontra a entrega e prestação de serviços públicos para a sociedade [Bertot et al. 2016]. Entretanto, isso é uma tarefa extremamente desafiadora devido às crescentes desigualdades sociais e econômicas, à diversidade social, ao quantitativo populacional, à carência de recursos, à dependência de cenários políticos, dentre outras [Bertot et al. 2016]. Devido a estes desafios, é necessário compreender que as agências prestadoras de serviços públicos possuem um papel complexo na medida em que precisam balancear a eficiência de seus processos com os interesses públicos [Daniels 2016].

As mudanças nas formas de prestação dos serviços ocasionadas pelos ideais de democracia digital, são um ponto crítico para o fornecimento e entrega de serviços públicos, pois os cidadãos demandam maior agilidade e transparência em sua prestação [Brown et al. 2017]. A falta de recursos, de gerenciamento adequado, e a forma como as informações sobre a utilização dos recursos públicos são passados para o cidadão, influencia diretamente na forma e na qualidade que os serviços são entregues à sociedade [Winters et al. 2014].

1.2.2. Prestação de Serviços Públicos

Grönlund [2010] defenda que a função do governo não é somente a prestação de serviços, e que este inclui outras características de relacionamento com o cidadão, como confiança, equidade e outros aspectos que não pertencem somente ao quadro de serviços. Os serviços públicos são produzidos por instituições nacionais, estaduais e locais, com o objetivo de serem entregues aos cidadãos, organizações e entidades em suas jurisdições [Bertot et al., 2016]. O fornecimento de serviços públicos pelos governos pode ser considerado como uma obrigação moral, prevista nos direitos humanos tanto quanto o direito à água, comida, energia, segurança e saúde [Bertot et al., 2016].

Porém, a prestação de serviços públicos para a sociedade é extremamente desafiadora [Tavares et al., 2016]. As crescentes desigualdades sociais e econômicas, a diversidade social, dentre outras, torna mais importante do que nunca a prestação contínua dos serviços públicos essenciais a todos os cidadãos, isto é, independentemente dos seus status na sociedade, o que contribui significativamente como complexidade de sua prestação com eficiência [Bertot et al., 2016][Tavares et al., 2016]. Devido a estes desafios é necessário compreender que as agências prestadoras de serviços públicos possuem um papel mais complexo do que as organizações privadas, pois necessitam balancear a eficiência de seus processos com os interesses públicos [Daniels, 2016]. A prestação de serviços públicos requer várias formas de inovação, sendo necessário que funcionem juntas as agências governamentais, não governamentais, particulares, universidades, cidadãos e outros atores envolvidos em todo o processo. Isso traz o

serviço para perto do seu usuário, de maneira que estes mesmo usuários possam aprender com os serviços [Tavares et al. 2016].

Embora atualmente haja este ideal de melhoria e transparência na prestação de serviços públicos, Winters et al. [2014] e Tavares et al. [2016] questionam, o porquê de os governos falham em prover os serviços básicos, com agilidade e qualidade para o cidadão. Os autores argumentam que a falta de recursos é um fator que contribui bastante para esta falha, e quando estes recursos existem, existem problemas na forma com que são geridos pelas instituições públicas. Esta falta de gerenciamento adequado, é associado também a falta ou à forma como as informações sobre a utilização dos recursos públicos são passados para o cidadão, devido a inexistência de meios eficazes para que o cidadão possa monitorar o governo. Essa falta de transparência e gestão de recursos públicos se associa à corrupção, que influencia diretamente na forma e na qualidade que os serviços são entregues à sociedade.

1.2.3. Gestão de Processos de Negócio

Independentemente do tipo da organização, todas elas possuem um conjunto de atividades que se relacionam no intuito de desenvolver um produto ou serviço para determinado cliente. Estas atividades interligadas compõem os processos, os quais ocorrem em diferentes níveis organizacionais - desde as atividades operacionais até as de níveis estratégicos do negócio. Sendo assim, não existe produto ou serviço oferecido por uma organização sem que para isso exista um processo relacionado a eles [Souza e Medeiros, 2008]. Neste sentido, Sharp e Mcdermott [2009] definem que um processo de negócio são tarefas de trabalhos inter-relacionadas, começadas a partir de uma resposta a algum evento, para que seja atingido um resultado específico ao cliente e outros envolvidos com o processo.

Os processos são executados pelas empresas visando entregar um produto ou serviço que agregue valor a seus clientes. A maneira com que os processos são definidos e executados influenciam diretamente na qualidade do produto ou serviço a ser entregue [Dumas et al. 2013]. Existem vários tipos de processos de negócios, esses tipos são vinculados ao ramo de cada organização, podendo ser tanto processos de negócios privados ou públicos. Os processos de negócio privados são aqueles internos às organizações, podendo fazer parte de processos estratégicos (processos onde são decididos os objetivos da organização, políticas, mudanças no planejamento e utilização de recursos), processos gerenciais (processos onde os gerentes garantem que recursos são obtidos e usados com eficiência e efetividade para atingir os objetivos organizacionais), ou processos operacionais (processos onde tarefas específicas são utilizadas com eficiência e eficácia no dia-a-dia organizacional). Já os processos de negócio públicos (também conhecidos como: processos de negócio colaborativos), são aqueles que envolvem organizações (ou pessoas) externas à empresa, seu relacionamento e a forma que interagem [Ko, 2009].

Business Process Management (BPM), ou em português, Gerenciamento de Processos de Negócio (GPN), é a arte e ciência de analisar como o trabalho dentro de uma organização é realizado para assegurar resultados positivos e oportunidades de melhorias nas suas atividades [Dumas et al. 2013]. Sendo assim, a GPN busca revisar e orientar a realização dos trabalhos executados em uma organização, definindo oportunidades de melhorias na execução das tarefas e no alcance aos objetivos, como redução de custos, agilidade de execução, ou melhoria na qualidade de produtos ou

serviços. Além disso, a área reúne princípios, métodos e ferramentas para projetar, analisar, executar e monitorar os processos de negócio [Dumas et al. 2013].

Uma característica constantemente associada à GPN é o seu ciclo de vida [Dumas et al. 2013], o qual descreve as fases que apoiam a gestão de processos de negócio. O ciclo de vida de GPN envolve as seguintes fases (Figura 1) de acordo com Dumas et al. [2013]: **Identificação de Processos; Descoberta de Processo; Análise do Processo; Redesenho de Processo; Implementação do Processo; e Monitoramento e Controle do Processo**.

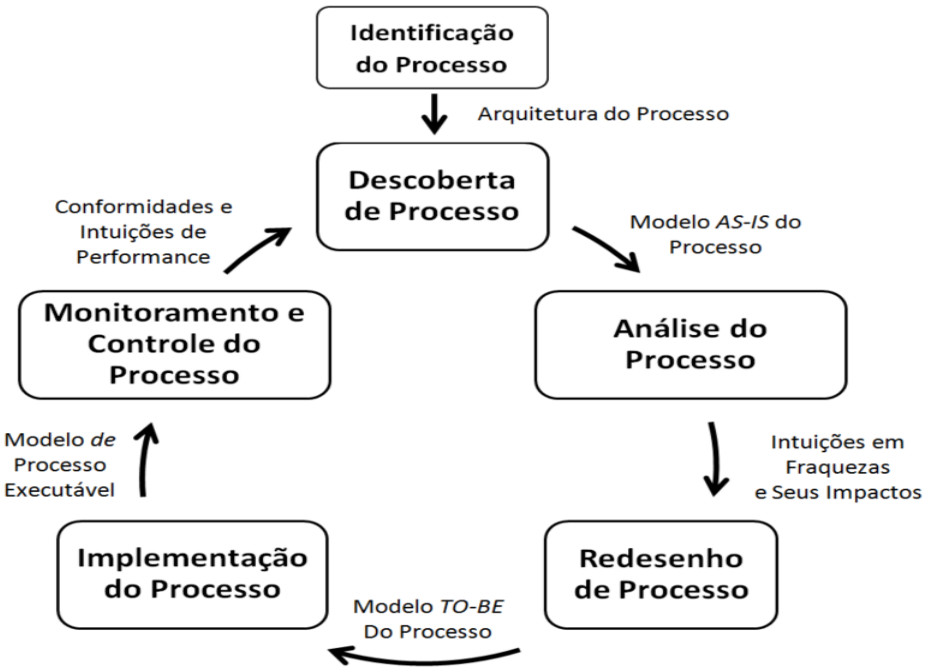


Figura 1 - Ciclo de Vida GPN [Traduzido de Dumas et al., 2013]

1.2.3.1 Modelagem de Processos de Negócio

A modelagem de processos de negócio está presente dentro das fases do ciclo de vida do BPM, no qual está presente a formalização de um modelo do processo de negócio, seus objetivos, métricas, fluxos, dados, integrações e relacionamentos entre áreas [Sobreira Neto, 2009]. Um modelo de processo de negócio é uma representação gráfica que simplifica o entendimento relacionado à execução sequencial das atividades que compõem o processo organizacional. Para representar estes modelos, existem linguagens de modelagem de processos. Dumas et al. [2013] discorrem que o propósito do modelo deve considerar quem deve utilizá-lo, a fim de abstrair certos elementos e objetos a serem modelados. Sendo assim, uma linguagem de modelagem de processos tem por finalidade a representação dos elementos destes processos.

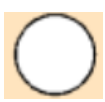
Os especialistas recorrem ao modelo de processo para fornecer uma compreensão de como o mesmo funcionam, como se relaciona com os componentes organizacionais, seus objetivos e regras, no intuito de que eles possam ser executados de forma mais simples e eficiente o possível, permitindo, assim sua análise e melhoramento [Aguilar-Saven, 2004]. O propósito do modelo é considerar quem deve compreendê-lo, a fim de abstrair certos elementos a serem modelados [Dumas et al, 2012].

Um modelo de processo é transcrito pode meio de alguma linguagem, está podendo ser uma linguagem natural falada ou escrita, ou um conjunto de elementos específicos que associados podem fornecer significados para quem conhece tais significados. Uma linguagem de modelagem de processos tem como foco a representação dos elementos de um processo e seu funcionamento. Essas linguagens focam nas mudanças contínuas dos objetos do processo, ou seja, em um modelo pode ser mostrado como as atividades se modificam no decorrer da execução e com as interações entres os atores do processo ocorrem [Reijers et al., 2013].

Dentre as várias linguagens de modelagem de processos de negócio a *Business Process Modeling and Notation* (BPMN) é uma das mais utilizadas, fornecendo um padrão entre o design de processos de negócios e a implantação destes processos. Este modelo representa práticas da comunidade de modelagem de processos de negócio, formalizando notações e semânticas sobre diagramações de processos. [OMG, 2013].

Atualmente em sua versão 2.0.2, o BPMN é reconhecido como um padrão internacional, compreendendo: a formalização de semântica para todos os seus elementos; a definição de mecanismos tanto para modelos de processos quanto para extensões gráficas; o uso de composição de eventos e suas relações; a redefinição das interações humanas; e a definição de modelos de coreografia. Tal padronização permite que o BPMN possa ser mapeado para mais de uma plataforma, fornecendo o padrão intercambiável para linguagens de modelagem de processos, com simplicidade de compreensão de seus elementos (Tabela 1) [OMG, 2013].

Tabela 1 – Principais Elementos BPMN

Título	Elemento	Descrição
Tarefa		Tarefa representa uma ação que pode ser executada por uma pessoa ou sistema dentro do fluxo do processo.
Sub-Processo		Sub-Processo representa uma abstração de um conjunto lógico (sequência) de atividades com um propósito específico.
Gateway Exclusivo		Gateway exclusivo fornece seguimento a uma condição exclusiva, no qual apenas um dos caminhos será seguido de acordo com uma condição testada.
Gateway Paralelo		Gateway paralelo divide o fluxo em dois ou mais, sendo executados todos os caminhos paralelamente.
Gateway Inclusivo		Gateway Inclusivo dá segmento ao fluxo por uma condição inclusiva, ou seja, pode haver uma combinação de um ou mais caminhos a serem seguidos dependendo da condição verificada.
Evento Inicial		Os eventos iniciais marcam pontos de início de um processo. Um processo pode apresentar diferentes pontos de início.

Evento Final		Eventos finais marcam onde o fluxo do processo terminar. Um processo pode apresentar diferentes pontos finais.
Evento Intermediário		Eventos intermediários indicam um ponto no fluxo do processo em que é planejado o acontecimento de algum evento, podendo mudar a sequência de ações.
Piscinas		Uma piscina é um contêiner de um processo de negócio. É permitida apenas uma piscina em um processo, no qual seu nome representa o mesmo. Uma piscina que não revela seu processo recebe o nome de "*pool black box*".
Raias		As raias são subdivisões da piscina, que podem ser usadas para representar um papel ou uma área organizacional responsável por uma tarefa no processo.
Objeto de Dados		Objeto de dados são informações cujas representações sejam necessárias e importantes para a compreensão do fluxo do processo.
Repositório de Dados		Repositório de dados representam informações de qualquer espécie, podendo ser bancos de dados, sistemas de informações e etc.
Fluxo Sequencial		Representa o fluxo, a sequência em que as atividades são executadas no processo.
Fluxo de Mensagem		Representa um fluxo de mensagens e é usado para mostrar a comunicação entre duas entidades no processo.
Associação		Associa artefatos aos elementos do fluxo do processo.

1.2.4. Jogos Digitais

Diversos autores definem o que são jogos. Crawford [2003] define que jogos devem ser interativos, possuir objetivos, permitir a competição e que um jogador possa atacar o outro de alguma maneira. Salen e Zimmerman [2003] possuem a definição que jogos são sistemas onde os jogadores estão engajados em um conflito artificial, regidos por regras que possam gerar um resultado quantificável. De acordo com Adams e Rollings [2007], jogos são atividades reais onde os jogadores tentam alcançar objetivos, guiando-se por regras pré-estabelecidas de maneira voluntária. Segundo Juul [2009], os jogos devem possuir regras fixas, resultados variados e valorizados, consequências negociáveis e ligações entre o jogador e os resultados.

Nesta linha, é proposta a seguinte definição a ser utilizada neste trabalho: "Jogos são atividades voluntárias com significância, altamente imersivas, onde os jogadores

estão engajados em conflitos em busca de seus objetivos, modificando interativamente de maneira quantificável um sistema artificial através de decisões e ações, sendo que todo este processo é regido por regras, o que normalmente resulta em diversão e entretenimento sobre adversários ou desafios" [Xexéo et al., 2017].

Em se tratando de jogos digitais, Schuytema [2008] define que jogos eletrônicos são atividades lúdicas formada por ações e decisões que geram uma condição final, sendo que as decisões são comandadas e limitadas por um conjunto de regras, as quais são regidas por um programa de computador. Complementarmente Marston [2015] dizem que jogos digitais são programas interativos que permitem um ou mais jogadores se engajarem em um propósito de entretenimento, através de dispositivos como computadores, videogames, aparelhos de TV e telefones, por exemplo [Ilomäki e Kankaanranta, 2009].

Desta maneira, para este trabalho, entendemos como definição de jogos digitais: "Jogos Digitais são atividade voluntárias com significâncias, altamente imersivas, onde os jogadores são engajados em conflitos em busca de seus objetivos, modificando interativamente de maneira quantificável um sistema artificial através de decisões e ações, sendo que todo este processo é regido por regras, controladas por programas de computador, executados por dispositivos digitais como computadores, videogames, aparelhos de TV e telefones, resultando normalmente em diversão e entretenimento".

Em geral, os jogos servem ao propósito de entretenimento, porém eles possuem um grande potencial para dar suporte à socialização, à educação e treinamentos [Michael e Chen, 2005]. Neste sentido, oportunidades que utilizam jogos e/ou seus elementos surgem, como a gamification (gamificação ou ludificação no Brasil), que consiste no uso de elementos e mecânicas de jogos em contextos que anteriormente não eram "jogáveis" [Deterding et al., 2011]; e os serious game (jogos sérios), jogos desenvolvidos para contextos sérios [Abt, 1970].

1.2.4.1 Jogos Sérios

Jogos sérios não é um tema relativamente novo, uma das suas primeiras menções foi por Abt [1970], o qual é considerado o primeiro autor a definir o termo. Segundo ele, jogos sérios são jogos em contextos sérios, no qual existe um compromisso com o caráter educacional e de treinamento, e não destinados simplesmente para o entretenimento. Neste sentido Michael e Chen [2005] em seu livro reafirmam o conceito de Abt ao escrever que um jogo sério tem como objetivo primário a educação (em seus vários formatos, sendo treinamento, conscientização, dentre outros), além do entretenimento. Ou seja, em sua definição mais simples, jogos sérios implicam em dizer que entretenimento e diversão não são o foco destes jogos, o que não quer dizer que estas características não possam estar presentes, onde nestes jogos é necessário conter elementos lúdicos, com o propósito de tratar problemas do mundo real Rocha e Araujo [2013].

Jogos sérios são jogos que usam seu meio artístico para transmitir mensagens, ensinar lições ou fornecer alguma experiência. Desta maneira, o termo "sério", não implica que o jogo seja maçante, mas sim o de refletir o propósito a que o jogo é criado [Michael e Chen, 2005]. Eles são ferramentas inovadoras amplamente conhecidas por seu potencial de apoiar a aprendizagem [Romero et al., 2015], podendo ser usados não

somente para fins puramente educacionais, onde podem contribuir para obter objetivos definidos [Susi et al., 2007].

Nos últimos anos, a literatura sobre jogos sérios apresenta crescimento significativo, pois seu estudo tem se demonstrado promissor para as técnicas de educação e treinamento, principalmente em se tratando das atuais gerações. Existem várias classificações sobre os diversos tipos de jogos sérios, de acordo com o propósito de cada um deles. De acordo com Connolly et al. [2012], é possível classificar os jogos sérios pelos seus propósitos, mídias, tecnologias utilizadas, gênero, área, impactos esperados, resultados habilidades necessárias ou conquistadas e pelo tipo de mudança comportamental. Entretanto, de acordo com [Alves, 2013], ainda não existe uma classificação formal amplamente aceita pela comunidade, embora ele destaque em seu trabalho algumas das classificações mais conhecidas (Tabela 2).

Tabela 2 – Classificação de Jogos Sérios

Tipo de Jogos Sérios	Descrição
Advergames	Jogos utilizados na promoção de alguma marca, produto, organização ou ponto de vista.
Edutainment	Jogos projetados com objetivo educacionais e também entretenimento.
Game-based Learning	Jogos com objetivos de aprendizagem, projetados de forma a equilibrar o componente lúdico e didático.
Newsgames	Jogos jornalísticos destinados a reportar acontecimentos recentes ou envio de comentários editoriais sobre acontecimentos.
Training and Simulation Games	Jogos de simulação que tentam abordar as atividades da vida real com o maior grau de exatidão possível. Normalmente utilizados para a aquisição ou exercício de habilidades.
Persuasive games	Jogos que persuadem os jogadores por meio da jogabilidade, normalmente projetados para mudar atitudes e comportamentos dos jogadores por meio de persuasão.
Organizational dynamic	Jogos projetados para promover o desenvolvimento pessoal e a formação do caráter.
Games for Health	Jogos utilizados para educação em saúde, simulações médicas, terapias e reabilitação de pacientes.
Art games	Jogos que expressam ideias artísticas ou arte produzidas por meio de jogos digitais.
Militainment	Jogos utilizados para fins militares por meio de simulações de operações com o alto grau de precisão.

1.2.4.2 Design de Jogos Digitais

Em se tratando do desenvolvimento de jogos digitais, é necessário pensar um pouco além de somente programação, pois o mesmo envolve também um grande trabalho criativo alinhados às complexas implementações de software. É necessário que exista um bom balanceamento entre arte e programação, do contrário o jogo pode não atingir os objetivos esperados. Por exemplo, um jogo com gráficos exuberantes que contenha

regras desconexas pode não oferecer uma boa experiência ao jogador, o qual irá abandoná-lo [Mastrocola, 2012].

Desta maneira, por serem programas de computador, é comum que desenvolvedores de jogos digitais usem métodos tradicionais de engenharia de software para sua construção. Contudo, as tradicionais metodologias de desenvolvimento de sistemas precisam ser adaptadas para o design jogos digitais, utilizando métodos que possam auxiliar a etapas de desenvolvimento do projeto do jogo, sua concepção até a sua finalização.

Pensando nisso, alguns autores como Zimmerman [2003], Adams e Rollings [2007] e outros, escreveram sobre suas experiências e seus processos e métodos que contribuíram para a concepção de jogos, seu desenvolvimento e meios de avaliação ao longo de todo o processo. Zimmerman [2003] propõe um ciclo interativo para o desenvolvimento de jogos, afirmando que "[...] o design interativo é uma metodologia baseada no processo cíclico de prototipação, análise e refino de um produto ao longo do processo. ". Desta maneira, a cada ciclo, o design anterior do jogo pode ser usado para sua melhoria e até mesmo para a concepção de novos jogos.

Na indústria de desenvolvimento de software o método de Schell [2009] é bem aceito. Ele subdivide os componentes de design em quatro grandes categorias (Figura 2), os quais devem se apresentar de forma balanceada no jogo. Estes quatro elementos básicos são: estética (*aesthetics*), mecânicas (*mechanics*), narrativa (*story*) e tecnologia (*technology*). Neste modelo, o autor descreve como **Estética** os elementos que dão aparência sensorial ao jogo (imagens, sons e etc.), fortemente relacionada à experiência do jogo, imergindo o jogador em seu ambiente. As **Mecânicas** são os procedimentos, as regras do jogo, os objetivos do jogo, o que um jogador pode ou não fazer e o que acontece quando ele atinge ou não um objetivo. As **Narrativas** são sequências de eventos, no qual se desdobram o jogo. É neste elemento em que podem ser apresentados os personagens e o contexto que o jogador interage com os elementos. Neste sentido, as mecânicas ajudam a contar a história e a estética ajuda a reforçar as ideias da história. Finalmente, a **tecnologia**, é o meio no qual a história é contada, implementando as mecânicas do jogo e criando a estética. A tecnologia pode ser criada, uma vez que a mesma não exista.

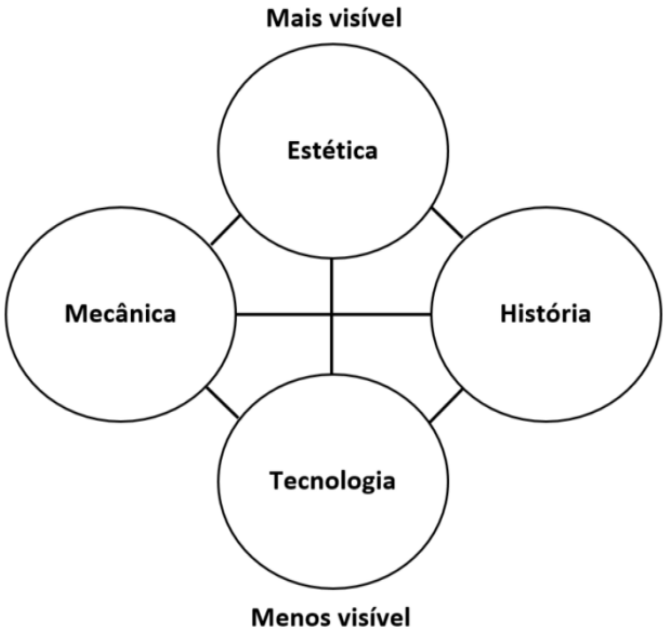


Figura 2 - Elementos de Design de Jogos [Schell, 2009]

1.3. Método de Design de Jogos Digitais Baseados em Processos de Serviços Públicos

Diferentemente dos métodos tradicionais de design de jogos digitas ou de jogos sérios, jogos baseados em processos de negócio (inclusive os processos de prestação de serviços públicos) precisam usar como fonte primária de inspiração os processos organizacionais e seus modelos. Portanto o método de design de jogos digitais baseados em modelos de processos de negócio (Figura 3), parte da existência de um modelo de processo de negócio para o design de jogos digitais.

O método é iterativo, ou seja, permite o retorno a etapas anteriores no caso de os designers dos jogos sentirem a necessidade de rever alguma informação de etapas já executadas. O método conta com as etapas de: i) estudo do contexto do processo; ii) mapeamento de elementos do modelo do processo para elementos de design do jogo; iii) projeto do jogo; iv) desenvolvimento e prototipação do jogo; v) validação com equipe de design; vi) validação com executores do processo; vii) validação com público alvo; e, viii) empacotamento e entrega do jogo. Estas etapas serão explicadas nas seções a seguir.

1.3.1 Estudo de Contexto

O método de design tem como entrada um modelo de um processo de negócio do processo que se deseja transformar em jogo. Este modelo de processo precisa ser compreendido pelos designers do jogo. Nesta etapa, o foco não é compreender todas as informações do processo, mas especificamente informações qualitativas como o contexto, valores, abordagens sociais, por exemplo, que não estão expressas em um modelo gráfico do processo. Além disso, esta etapa objetiva, também, discutir e organizar as informações sobre o contexto do processo de negócio, como ele é executado, qual são as percepções dos seus usuários ao solicitá-los, qual sua relevância, performance e qualidade. Estas informações são muito importantes para que os

designers consigam pensar em informações do jogo como ambiente, comportamento dos personagens, e outras informações relevantes.

Neste sentido, questões como: “O que é o processo?”, “Por que o processo existe?”, “Como o processo é executado?”, “Quem são os clientes do processo?”, “Como os clientes experenciam o processo?”, “Quais são as principais tarefas que precisam ser compreendidas no processo?”. São agrupadas em três categorias: **i) contexto organizacional**, a fim de entender critérios organizacionais (valores e missão, estratégia organizacional, clientes e etc.); **ii) contexto do processo**, a fim de compreender os detalhes do processo (motivação, objetivos, desafios, principais atividades, atores e etc.); e, **iii) contexto do usuário**, que investiga a experiência do usuário sobre o processo (sentimentos, frustrações, pensamentos e etc.).

Todas estas informações devem ser registradas em um documento (documento de contexto), o qual é útil para a contextualização do projeto do jogo, além de ajudar a compreensão do processo, permitindo que os designers do jogo entenderem particularidades do processo. Posteriormente, este documento também será útil para pensar em situações de fluxos de jogo, como histórias e fluxo temporal do jogo.

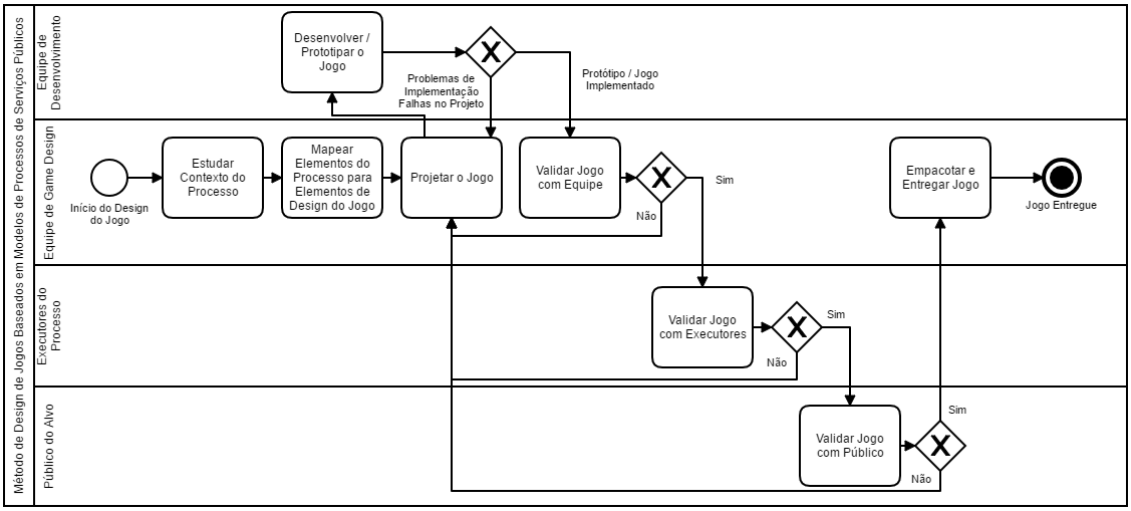


Figura 3 – Método de Design de Jogos Digitais Baseados em Modelos de Processos de Negócio

1.3.2 Mapeamento de Elementos

O mapeamento de elemento é a etapa do design do jogo que permite, a partir de um modelo de processo de negócio (considerando modelos gráficos e documentais), e um gênero de jogo digital, ser possível associar elementos (elementos do modelo do processo e elementos do gênero do jogo) a partir de seu significado, realizando o cruzamento de informações entre eles, que permite construir um documento de mapeamento de elementos (Anexo 1). A partir deste documento de mapeamento é gerado um documento de design de jogos (GDD – *game design document*), o qual é um documento de requisitos necessário para auxiliar a equipe de design do jogo na construção de jogos sérios de processos de negócio.

1.3.3 Projeto do Jogo

A etapa de projeto de jogo é inspirada na visão de design de jogos de Schell [2009]. Nesta etapa, o objetivo principal é melhorar a versão inicial do GDD gerado na etapa anterior com criatividade, no qual devem ser pensados elementos como: a temática do jogo, público, história e mecânicas, por exemplo. Portanto, apesar de o método propor a sistematização do design de jogos baseados em processos de negócio, é necessário utilizar a criatividade dos designers de jogos. Um jogo baseado em processo de negócio deve estar entre o determinismo de um processo de negócio, e a ludicidade de um jogo. Não deve ser somente uma simulação do processo, e nem somente um jogo de entretimento.

Para melhorar o documento de design a equipe de game design precisa refletir sobre: **i) definição do público alvo** (escolha do público do jogo); **ii) definição do gênero do jogo** (qual gênero de jogo mapeado pode representar melhor o processo a ser transformado em jogo); **iii) temática do jogo** (seleção do tema do jogo); **iv) personagens** (personagens relacionados aos atores do processo e suas características); **v) narrativa** (histórias, enredos, que deem suporte ao processo de negócio); **vi) mecânicas e dinâmicas** (ações e respostas às ações dentro do mundo do jogo); e **vii) tecnologia** (*engine* ou ambiente de criação do jogo).

1.3.4 Prototipação e Desenvolvimento

Esta etapa recebe como entrada o documento de design atualizado na etapa anterior. Aqui o objetivo é desenvolvimento, a codificação do jogo de acordo com os requisitos destacados do documento de design de jogos.

1.3.5 Avaliação com Equipe de Design

A avaliação com a equipe de design consiste em avaliar as versões dos jogos geradas pelo método de design do ponto de vista do balanceamento de elementos do jogo. Nesta etapa é necessário averiguar se todos os requisitos presentes do GDD foram devidamente implementados. Além disso, esta etapa é responsável por verificar o balanceamento entre os elementos do processo e elementos lúdicos no jogo.

1.3.6 Avaliação com Executores do Processo

Esta etapa é necessária pois os executores são as pessoas que têm o conhecimento sobre como o processo é executado. Portanto, eles devem ser capazes de reconhecer o seu contexto organizacional no mundo do jogo, por meio dos ambientes, regras, objetivos e aspectos organizacionais colocados no design.

1.3.7 Avaliação com Público Alvo

Esta é uma importante etapa no método de design pois é nela que se verifica se o jogo desenvolvido cumpre o propósito de fornecer a compreensão do processo pelos jogadores. Nesta etapa espera-se avaliar o jogo produzido pela ótica de qualidade do jogo, a qual investiga aspectos como clareza de objetivos, interação, desafios, imersão e etc.; além de verificar também se houve a compreensão dos valores, regras, etapas, dificuldades e desafios da execução do processo.

1.3.8 Empacotamento e Entrega

A última etapa do design do jogo compreende a publicação e entrega do jogo. Isso significa que o objetivo é juntar todos os recursos para execução do jogo em um único pacote (ajuda, jogo, sistemas de comunicação e etc.).

1.4. Construção de Jogo Digital Baseado em Serviço Público

Como já mencionado este minicurso tem como objetivo é a construção de um jogo digital para um processo de serviço público, portanto o ponto de partida será um GDD (*documento de game design*) previamente gerado na etapa de projeto de jogos (etapa 3 do método de design), ou seja, as etapas de estudo de contexto, mapeamento de elementos e projeto de jogos já foram executadas anteriormente. O foco desta seção é colocar em prática os requisitos descritos em documento de game design (disponível em: https://bit.ly/2QGClV1) para a construção do jogo digital que ele especifica (etapa 4 do método de design).

O processo de serviço público escolhido para a construção do jogo é o "Processo de Emissão do Cartão SUS", uma vez que é necessário que todo o cidadão brasileiro possa ter acesso à sua emissão, para a realização de procedimentos de saúde na rede do SUS. Além disso, este processo é pequeno o bastante para ser trabalhado no tempo do minicurso.

Como *engine* (ferramenta) de desenvolvimento de jogos será usada a plataforma de desenvolvimento de jogos *Sierra Construict 2*, a qual permite que todos os níveis de pessoas com pouco conhecimento em desenvolvimento de software consigam construir um jogo digital, uma vez que não apresenta sintaxes de linguagens de programação. As imagens e sons do jogo não serão construídos no minicurso, uma vez artes audiovisuais não está no foco deste curso, e, portanto, os mesmos estão disponíveis no link: https://bit.ly/2xk2wJ2.

1.4.1. Processo de Emissão do Cartão SUS

O SUS (Sistema Único de Saúde) foi instituído no Brasil por ocasião da promulgação da Constituição da República Federativa do Brasil em 1988 (Artigo 196) como forma de efetivar o mandamento constitucional do direito à saúde como um "direito de todos" e "dever do Estado", passando a oferecer a todo cidadão brasileiro acesso integral, universal e gratuito a serviços de saúde [Fiocruz, 2017]. O SUS traz benefícios a aproximadamente 180 milhões de brasileiros, realizando anualmente cerca de 2,8 milhões de atendimentos, desde procedimentos ambulatoriais simples a atendimentos de alta complexidade como transporte de órgãos, tendo como princípios básicos que o regem: universalidade do acesso à saúde, equidade dos serviços, integralidade, descentralização, regionalização, hierarquização e a participação social [Teixeira, 2011].

Devido à quantidade de brasileiros beneficiados pelo SUS, para que possam ter direito aos benefícios do sistema de saúde, todo cidadão tem direito ao Cartão Nacional de Saúde, ou como é conhecido, Cartão SUS [Portal Brasil, 2012]. Esta foi uma formalização instaurada para a identificação única do usuário e contribuir com a sua organização. O Cartão é um instrumento obrigatório do cidadão que possibilita a vinculação dos procedimentos executados no âmbito do SUS ao seu usuário, profissionais e a unidade de saúde onde foram realizados os procedimentos, sem este

documento, em várias instituições vinculadas ao sistema de saúde, os procedimentos não são realizados.

Existe um processo gratuito para solicitar a sua emissão (Figura 4). Atualmente o processo é descrito por etapas a serem seguidas. Devido à regionalização dos serviços de saúde, algumas etapas podem sofrer mudanças de um município para outro. O processo para realizar o seu cadastramento começa com a descoberta em sua localidade do local onde se deve comparecer para a confecção do cartão. Em seguida, é necessário se dirigir a esta unidade de atendimento de saúde com os documentos, carteira de identidade, CPF, comprovante de residência e, se possível, a carteira de vacinação. Estes documentos devem ser apresentados a um atendente, o qual incluirá seus dados no sistema, irá imprimi-lo, e poderá sair do local pronto para utilizá-la. Em algumas localidades, existe outro processo para a emissão do cartão, este realizado pelo Agente Comunitário de Saúde (ACS). Este servidor público é um agente intermediário entre o cidadão e o serviço de saúde, e devido a sua proximidade com ambas as partes, é possível solicitar ao mesmo que emita o cartão SUS. Outro meio de requisição do cartão é o cidadão realizar um pré-cadastro no Portal da Saúde do Cidadão, sendo necessário, o comparecimento à alguma unidade de saúde para a sua validação.

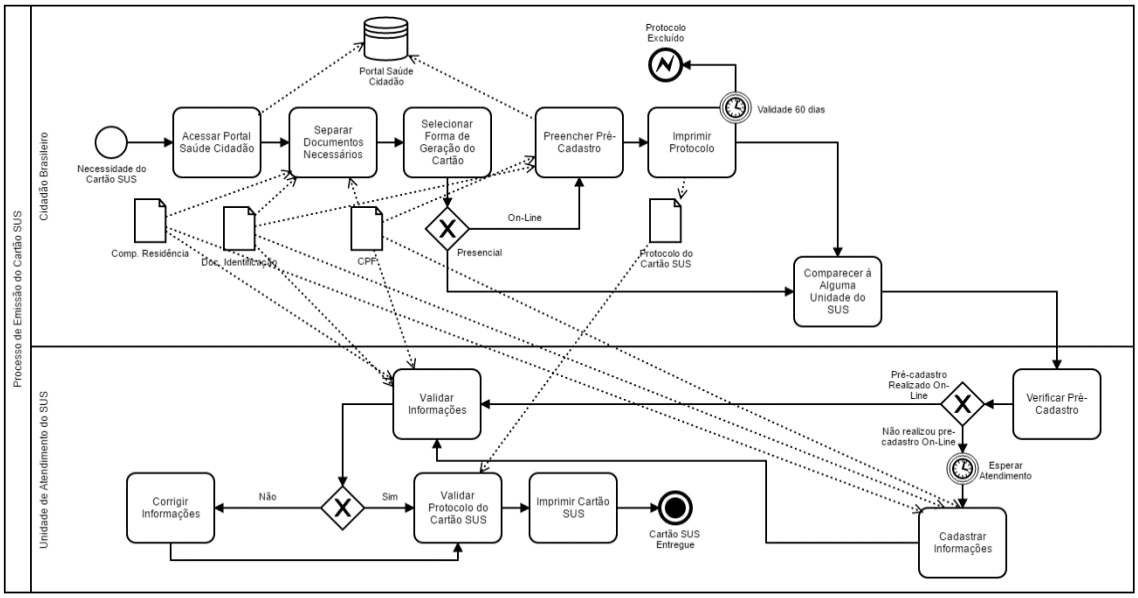


Figura 4 – Processo de Emissão do Cartão SUS

1.4.2. Plataforma Construct 2

Construct 2 é um programa que o deixa criar jogos de computador em HTML5 sem ter qualquer experiência em programação utilizando um intuitivo, ***'drag and drop'*** ambiente de desenvolvimento. A maioria das ferramentas dos programas pode ser usada a partir da interface gráfica sem ter que escrever uma única linha de código. A plataforma tem como foco a criação de jogos 2D, e vem com recursos que a torna fácil, incluindo um sistema físico que causa os itens no jogo serem governados pela lei da gravidade, como também bits gráficos e de som como os *sprites*, fundos e efeitos de som. Além disso, é tão simples como adicionar qualquer arquivo de mídia de uma aplicação exterior.

Basicamente a plataforma é organizada em 2 ambientes. O primeiro ambiente consiste no "desenho" do jogo, chamada ***Layout***, enquanto o segundo ambiente foca-se os eventos do jogo, ou seja, nos comportamentos de cada item inserido no layout. Esta segunda área recebe o nome de ***Event Sheet*** (Figura 5).

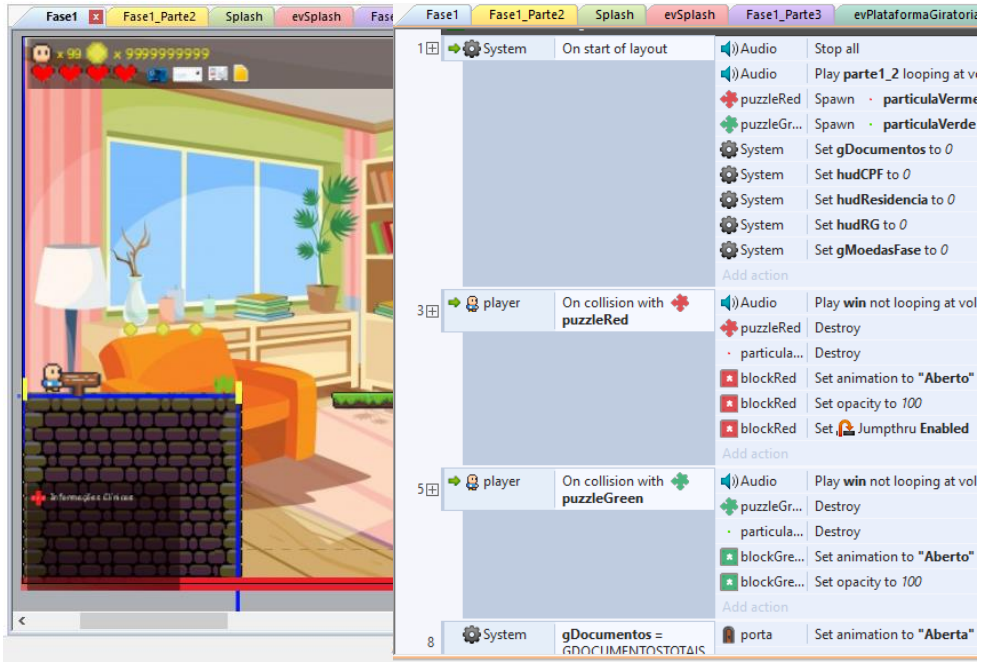


Figura 5 – Construct 2 (Layout e Event Sheet)

Outra característica importante da plataforma Construct 2 é que ela apresenta as áreas de: "***Projects***", responsável por exibir todos os projetos e sua organização de pasta; e "***Properties***", importante pois exibe as principais propriedades de algum item selecionada na área de *layout* ou na área de *projects* (Figura 6).

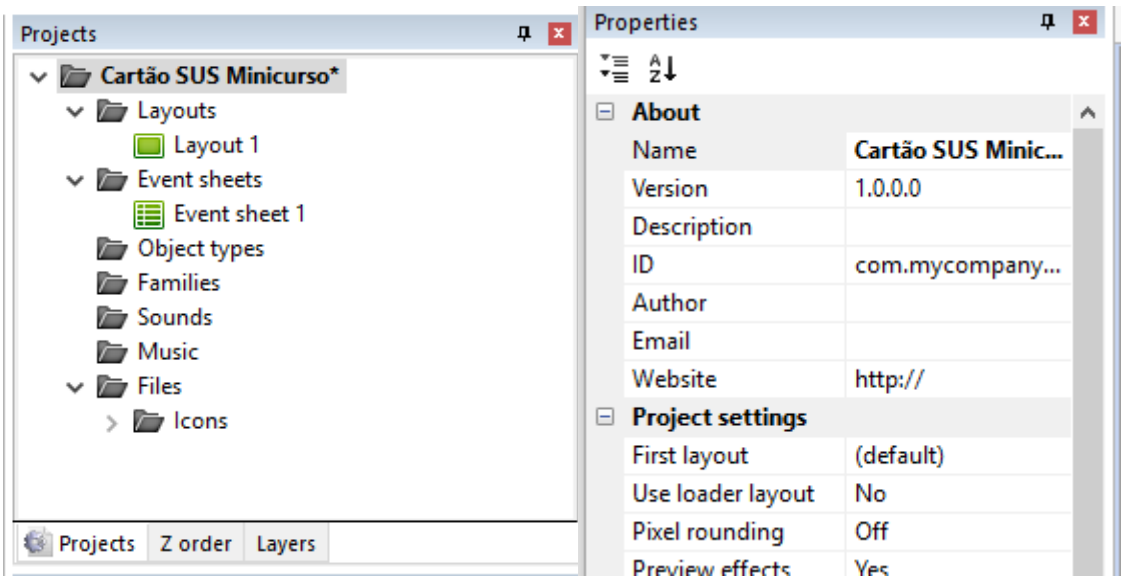


Figura 6 – Construct 2 (Project e Properties)

A filosofia de Construct 2 é uma da simplicidade e de um ambiente visual intuitivo. Quando arrasta os seus *sprites* (imagens) e os larga na sua posição do ecrã

(tela, fase etc.), outra parte do programa irá os fazer interagir com outros objetos de acordo com o tipo de objetivo. Ela é uma ferramenta interessante para qualquer um que quer começar a criar jogos digitais, mas que não tem qualquer habilidade em programação. As suas fáceis ferramentas e documentação extensiva tornam o numa escolha interessante.

1.4.3. Configurando o Projeto

Inicialmente deve-se criar um novo projeto no Construct 2, dê preferência para criar um "***New empty project***", o qual deverá criar uma área de *layout* em branco. Para atender ao mundo e à fase descrita no GDD (seção ***Scenarios (Places)***) é preciso realizar a configuração inicial do projeto de acordo com a Tabela 3 (A primeira coluna informa qual é o item selecionado; a segunda coluna indica o nome da propriedade; a terceira coluna indica qual é o valor esperado para à propriedade).

Tabela 3 – Configuração do Projeto

Item / Elemento / Sprite	Propriedade	Valor
New Project	Name	Cartão SUS Minicurso
	Windows Size	320, 300
	Loader style	Progress Bar & Logo
Layout 1	Name	Fase 1
	Layout Size	2400, 320
	Margins	160, 120
Event Sheet 1	Name	evFase1

1.4.4. Criando a Fase do Jogo

Segundo a seção ***Scenarios (Places)*** do GDD, a fase deste jogo será um ambiente em plataforma no qual o jogador deverá pular sobre obstáculos, coletar itens até conseguir chegar no posto de atendimento do SUS. Isso é descrito pelo processo de emissão do cartão SUS, portanto esta fase descreverá descrever os passos de coleta de documentação até sua entrega ao posto de atendimento para emissão do cartão SUS.

A Construct 2 permite trabalhar o layout como camadas, assim como os editores de imagens no mercado. Se selecionarmos o Layout "Fase 1", dentro da área de Projects existe uma aba chamada "***Layers***". Essas layers são as camadas de interação de um layout, ou seja, o que será exibido para o usuário em tempo de jogo. É comum ao construir jogos, fazê-lo em camadas de acordo com a sua funcionalidade.

Por exemplo, considerando o jogo que estamos construindo do cartão SUS, vamos trabalhar com a fase em 5 camadas: Fundo, Mapa, Principal, HUD, Mensagens (caso necessário). O fundo será apenas a imagem de fundo que a fase irá apresentar; o mapa é a imagem com as plataformas fixas do mundo; a principal será incluíra personagens, inimigos, itens e qualquer outro elemento de interação com o jogador; HUD são as informações de ***feedback*** dos jogadores como vidas, pontos e etc.; e. Mensagens, será telas de mensagem ou pausa do jogo. Portanto, crie estas layers para a Fase 1, lembrando de seguir a ordem tal como descrita aqui neste parágrafo, pois as camadas sobrepõem umas às outras (Figura 7).

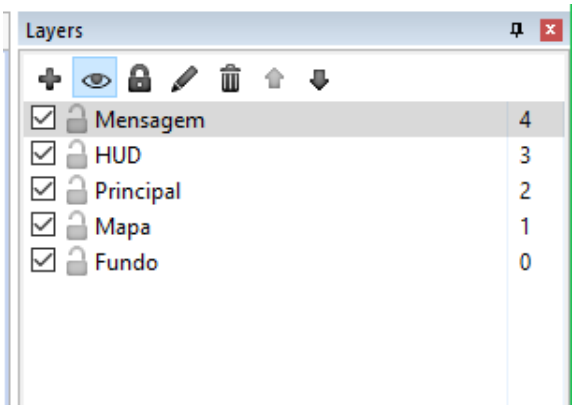


Figura 7 – Camadas da Fase 1

Já foi mencionado que o foco deste curso não é a criação artística de imagens, portanto, as imagens e sons do jogo já estão desenvolvidos de disponíveis para uso. Para inserir uma imagem como o **fundo da fase** ou o **mundo** do jogo e demais sprites (elementos gráficos) é necessário incluir um novo Sprite no level que deseja utilizar.

Para a criação do fundo da fase selecione a camada "Fundo". Na área de layout clique com o botão direito do mouse na parte de design e selecione a opção "***insert new object***". Irá aparecer uma tela de seleção com vários itens, nesta tela selecione "***Sprite***" e coloque em "***Name when inserted***" como "Fundo" (Figura 8). Atenção, quando um Sprite é inserido no projeto ele irá aparecer dentro de "***Object Types***" na área Projects. Quando você altera um objeto diretamente na área Projects você estará modificando todas as instâncias deles inclusas nas áreas de layout. Portanto, se quiser mudar propriedades de um objeto para um *layout* específico, é necessário selecionar este objeto dentro da área de layout, somente assim você irá alterar especificamente àquele objeto.

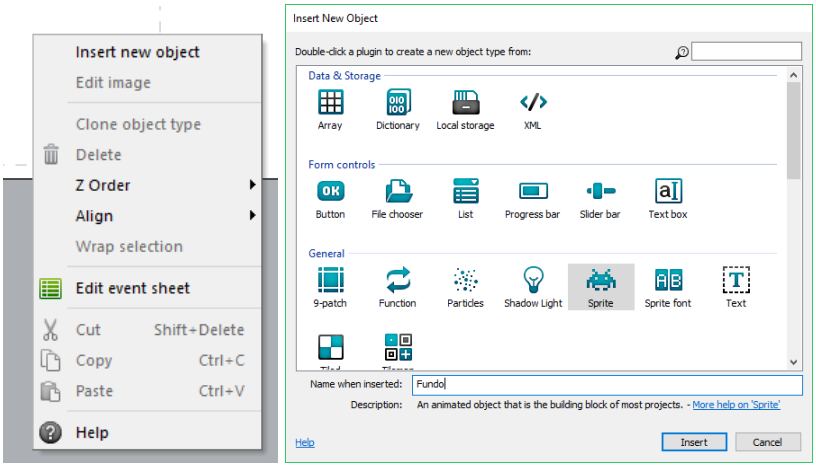


Figura 8 – Inclusão do Sprite "Fundo"

Após incluir esse Sprite a tela de edição de imagens será exibida (Figura 9). Esta tela apresenta um pequeno editor de imagens, mas ela também está preparada para trabalhar com animações (movimentos de objetos do jogo), ponto de colisão (pontos necessários de impacto entre objetos do jogo), pontos de origens (pontos sobre qual um objeto é construído ou criado) e outras funcionalidades.

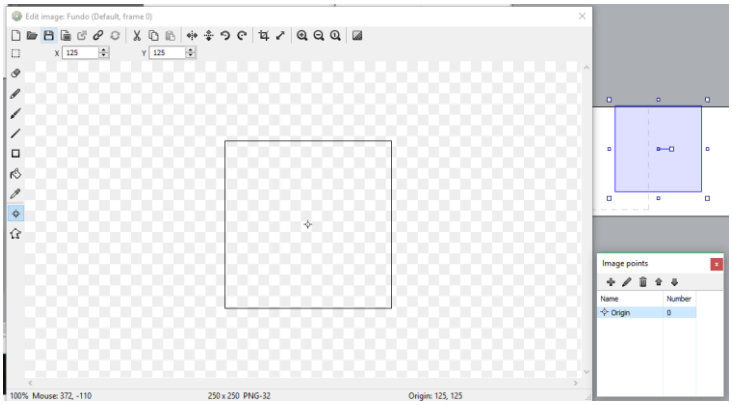


Figura 9 – Inclusão do Sprite "Fundo" – Edição de Sprite

O fundo se encontra dentro da pasta "***assets/fundos***" no link de recursos de utilização para o jogo. Para incluir o mundo abra nesta tela de edição de Sprite o arquivo "fundo1" para que a imagem seja carregada. Outra alteração importante neste Sprite é seu ponto de origem para que ele seja melhor posicionado na tela. Portanto na tela de "***imagem point***" clique com botão direito sobre *"**origin**", "**quick assign**", "**top-left**"*. Isso fará com que o ponto de origem desta imagem seja em cima e a esquerda. É possível fechar a tela de edição, e a imagem aparecerá no layout. Para que ela seja posicionada corretamente, selecione o Sprite "Fundo" e altere sua propriedade position para **0,0**. Para finalizar, ajuste somente o tamanho da imagem com a altura do layout. A largura não precisa, pois veremos mais à frente a opção de ***Parallax***.

Para incluir o mundo do jogo faremos o mesmo processo descrito acima para a inclusão do fundo, porém deve-se selecionar a camada de "Mapa" na Fase 1. Desta forma, insira um novo Sprite na Fase 1 com o nome de "**Mapa**". A imagem para o mapa está dentro de **"*assets/fundos***", selecione o arquivo "mapa". Selecione o ponto de origem como "***top-left***" e feche a tela de edição. Com o objeto de mapa selecionado, altere sua propriedade de "position" para **0,0**. Perceba que neste momento o mapa do jogo se sobrepôs ao fundo da fase. Neste momento, se rodarmos o jogo, a fase já tem uma aparência de jogo.

Uma propriedade muito importante ao cria um fundo de uma fase é o ***Parallax***. Esta propriedade permite dar velocidades diferentes de movimento para as camadas de um projeto.

Para finalizar esta seção, insira um objeto ***Keyboard*** no projeto. Este objeto é responsável pelo uso do teclado como "***joystick***" do jogo. Para inclui-lo é muito similar a um Sprite, porém, o que varia aqui é que deve ser selecionado o objeto "***keyboard***". Este objeto uma vez inserido, funciona para todas as telas e fases do jogo.

1.4.5. Criando o Chão do Jogo

Assim como o fundo e o mapa do jogo foram inseridos na Fase 1 o chão deve ser feito. Porém, para isso, serão usados blocos simples de Sprites, ou seja, sem imagens. Como assim sem imagens? Para criar chão de fases de jogos de plataformas não é necessário criar imagens, desde que já exista um mapa com o chão fixo desenhado. O que é o caso deste projeto.

Selecione a camada "Principal" da mesma forma que foram inclusos o fundo e o mapa, vamos criar um Spirte chamado "**chao**". Este Sprite não tem uma imagem, mas

para que ele não fique transparente, coloque uma cor sólida. Essa cor sólida é bem útil, até mesmo para o programador identificar onde estão os objetivos de chão da fase. No exemplo foi utilizada a cor azul para o chão. Neste momento, o projeto deve ser semelhante à Figura 10. Agora é necessário modelar chão da fase. Copie e cole o objeto chão no decorrer da fase, sempre os modelando (esticando, girar, diminuindo o objeto) de acordo com as plataformas do mapa (Figura 11).

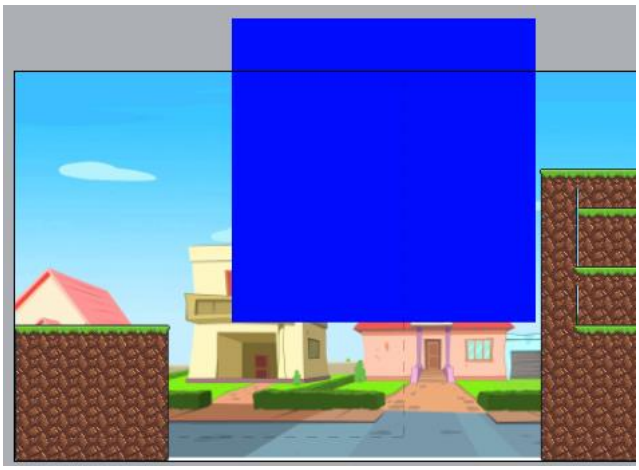


Figura 10 – Inclusão do Sprite de Chão

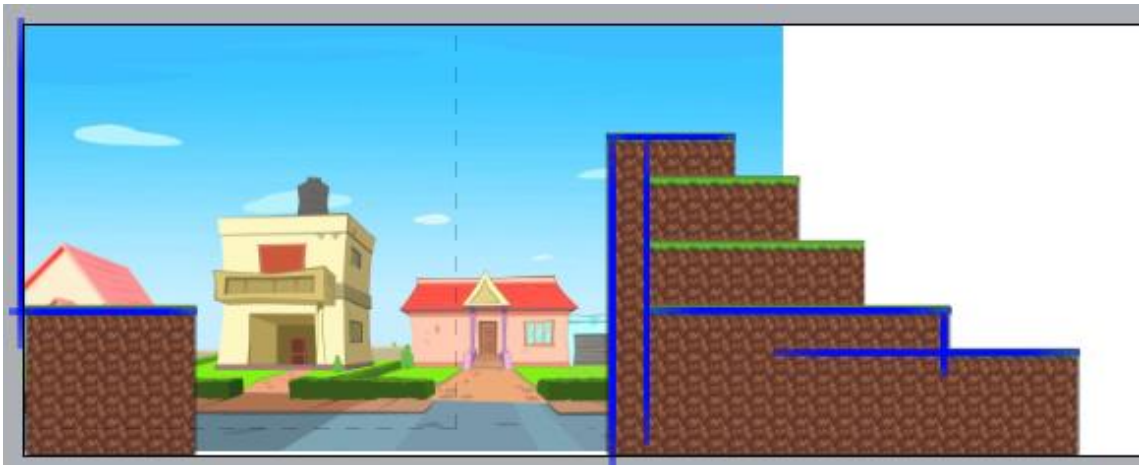


Figura 11 – Exemplo de Modelagem do Chão da Fase

Após a modelagem dos sprites de chão é preciso fazer com que eles tenham comportamento de chão. Para isso, selecione o objeto chão dentro de "***objetct types***". Em suas propriedades, identifique "**Behaviors**". Isso é uma facilidade da ferramenta Construct que assume que todo objeto tem comportamentos no jogo, e estes comportamento são adicionados aos objetos para que eles executem ações ou tenham determinadas características no jogo. Como este jogo é uma plataforma, e o chão deve possuir o comportamento de ser sólido, clique em behaviors. Será exibida uma tela para seleção de comportamento. Portanto, adicione um behavior chamado "***Solid***" (Figura 12).

Porém repare que não é bonito para o jogo que esses retângulos azuis fiquem sendo exibidor em tempo de jogo. Para escondê-los basta selecionar trocar sua propriedade "***Initial Visibility***" para "***Invisible***". O chão da fase está pronto.

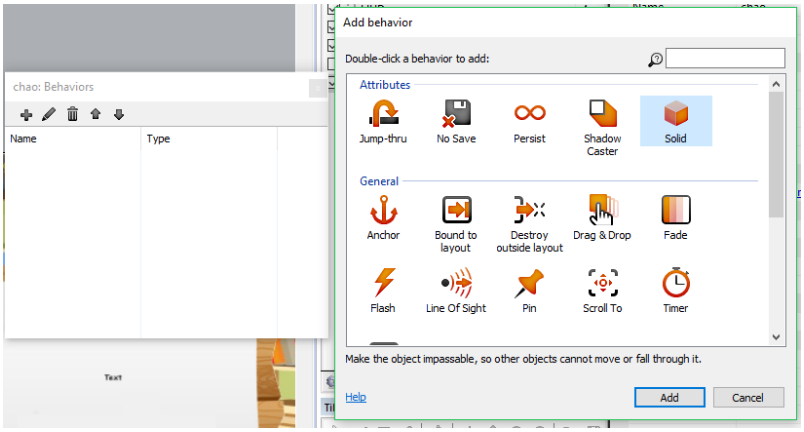


Figura 12 – Behavior Solid para o Chão.

1.4.6. Criando o Personagem Jogador

Um dos itens mais importantes de um jogo é o jogador. E neste caso, temos um personagem que deve andar pela fase, saltar e coletar itens para conseguir o seu cartão SUS. O personagem é uma seção específica do GDD, e neste exemplo suas definições podem ser observadas na seção ***Player(s)*** do documento de design. Como usual, da mesma forma que inserir sprites nas seções anteriores, vamos neste momento, inserir um novo Sprite na camada Principal, chamado "**jogador"**.

Neste primeiro momento apenas dê uma cor para ele. No exemplo optamos pelo jogador na cor vermelha (assim como coloquei a cor azul para o fundo). Coloque também a propriedade "***Size***" do jogador com o valor de **20,20**. E posicione o quadrado vermelho que representa o jogador no início da fase (Figura 13).

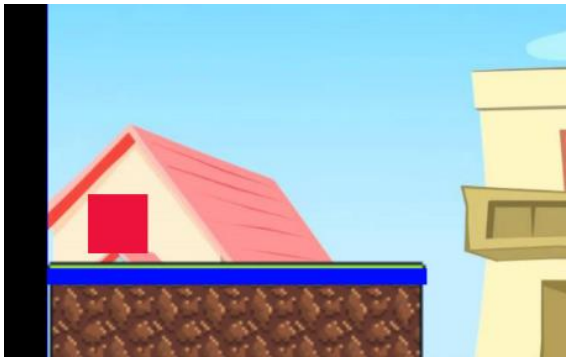


Figura 13 – Inclusão do Jogador na Fase.

Se rodarmos o jogo é possível perceber que são apenas imagens imóveis. Para que ele se mova como um personagem de jogo de plataforma, o que é necessário fazer é adicionar um "***behavior***" chamado "***Platform***" para o objeto jogador. Esse comportamento implementa automaticamente os comandos de um jogo de plataforma no objeto que o possui. Se rodar o jogo agora, vai perceber que o quadrado do jogador responde aos comandos de seta do teclado do computador. Como o comportamento de plataforma foi implementado automaticamente, algumas propriedades precisam ser modificadas para que o gameplay seja melhor aproveitado, como por exemplo: alterar força da gravidade, distância de salto e outros (Tabela 4).

Tabela 4 – Configurações do Comportamento do Jogador

Item / Elemento / Sprite	Propriedade	Valor
Jogador	Max Speed	100
	Jump Strength	400

Outro behavior que deve ser inserido para o jogador é o chamado "***ScrollTo***", pois do jeito que está no momento, a câmera do jogo não acompanhará o jogador pela fase. Após inserir este comportamento, a câmera do jogo irá acompanhar o jogador por ponde ele for na fase.

1.4.6.1. Definindo a Imagem do Personagem

Para os sprites anteriores foram inseridas imagens fixas ou alguma cor para sua identificação, mas em se tratando de personagens, é necessário movimento. A Construct 2 permite trabalhar com movimento, ou no termo correto, "Animações" de várias imagens separadas, ou de um arquivo único com várias imagens, os conhecidos "***tilesets***".

Portanto, para incluir imagem para o jogador é necessário abrir a tela de edição do Sprite do jogador. Dê um duplo clique no objeto do jogador para isso. Perceba que toda vez que você abre essa tela de edição do Sprite, surge duas telas menores: "***Animations***" e "***Animation Frames***". Estas telas são responsáveis pela animação de sprites. A primeira, é responsável por um delimitar animações, por exemplo, andar, saltar, parar e etc.; enquanto a segunda representa cada uma das imagens destas animações. No nosso exemplo criaremos as Animations: Andar, Parado, Pulando, Abaixar e Morte (Figura 14). Em cada uma delas iremos colocar a animação necessária.

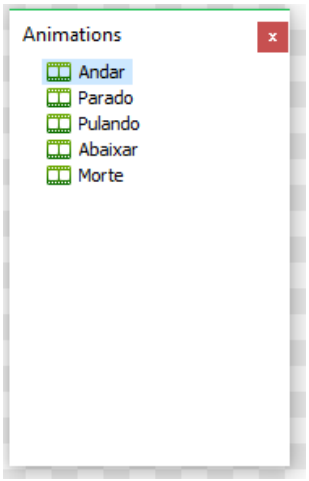


Figura 14 – Animações do Jogador.

Para criar a animação de andar, selecione a animação "Andar" e na tela de "***Animation Frame***", clique com botão direito, então em "***Import Frames***", "***From Sprite Strip...***". As imagens do jogador estão dentro de "***assets/jogador***", carregue o arquivo "***jogador***". A ferramenta perguntará quantas linhas e quantas colunas a imagem é composta, o que significa quantas imagens de movimento contém dentro deste único arquivo (Figura 15). Nesta imagem devemos colocar 6 verticais e 5 horizontais.

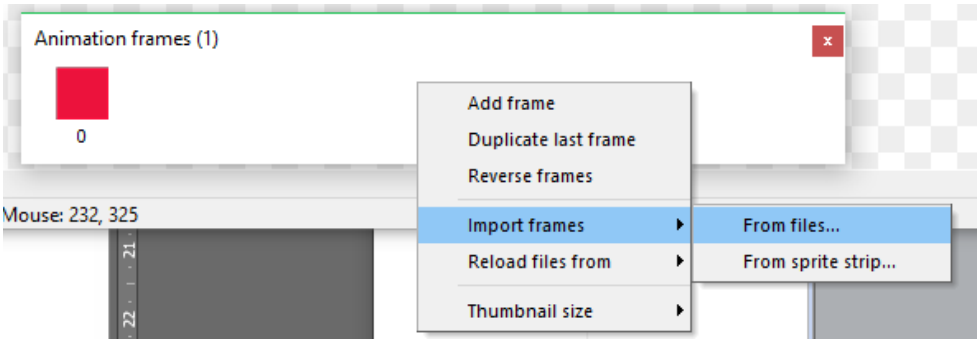


Figura 15 – Animações do Jogador Carregando TileSets.

Ao carregar o arquivo todas imagens do personagem irão aparecer. Como estamos interessados apenas no "Andar" do jogador. Devemos excluir o quadrado vermelho, e deixar apenas as imagens de 6 a 11, deletando as demais. Cada animação criada também possui propriedades. Portanto, altere a propriedade "***Loop***" de "Andar" para "***Yes***". Agora é só repetir estes passos para as demais animações. Com isso, ao executar o jogo, o jogador já aparece com um personagem.

Para finalizar em cada uma das animações, coloque a "***Origem Point***" de origem do jogador como "***Botton***".

1.4.6.2. Movimentando as Animações Corretamente

O jogador está se movimentando, já é possível ver animações do personagem, porém se formos detalhistas perceberemos que apesar das teclas de movimento de jogos de plataforma estarem funcionando, as animações do personagem não respeitaram isso. Por exemplo, ele continua virado para frente quando apertamos esquerda, ou mesmo quando está parado, parece que está em uma esteira.

Para que estas animações respeitem os movimentos do jogador é necessário programar estes eventos. Criaremos então uma nova ***Event Sheet*** com o nome de "***evJogador***". Para que este evento *sheet* seja usado na fase é necessário referencia-lo dentro da Event Sheet "***evFase1***". Clique com botão direito e depois em "***Include Event Sheet***", selecionando a "evJogador" que criamos.

Para programar os eventos do jogado, vamos a "***evJogador***". O primeiro evento que vamos programar os movimentos (Figura 16):

- "Pular": se o jogador estiver pulando OU se o jogador estiver caindo ele deverá estar com a animação de pulo;
- **SENÃO** "Parar": se o jogador não estiver pressionado direita OU se o jogador não estiver pressionado esquerda E se o jogador está sobre alguma plataforma (Sólida) ele ficará parado.
- **SENÃO** "Andando":
  - Se o jogador estiver pressionado a seta para direita ele está andando para direita.
  - **SENÃO**: Se o jogador estiver pressionado a seta para esquerda ele está andando para esquerda.

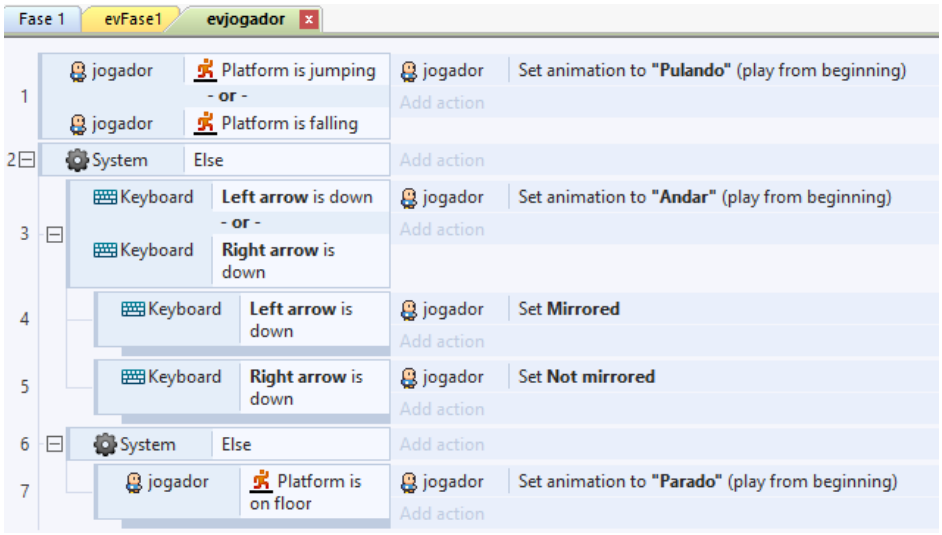


Figura 16 – Programação dos Movimentos e Animações do Jogador.

1.4.6.3. "Morte" do Jogador

Todo jogo deve ter critérios de sucesso e insucesso. De acordo com o GDD, na seção de ***Success – Game Overs e Fails – Game Overs*** são descritos os principais meios de ganhar ou perder o jogo. De acordo com o processo, a emissão do cartão SUS é encerrada quando o cidadão consegue emitir o seu cartão ou quando ele não consegue emitir o cartão. Desta maneira o jogo seguirá o critério de vitória de conseguir emitir o cartão SUS. Porém como derrota, os game designers pensaram nos jogos clássicos de plataforma que existem chances para realizar objetivos. Desta maneira, o jogador deverá ter critérios de derrota em alguns pontos.

O primeiro deles se baseia na quantidade de chances para reiniciar o objetivo do jogo, as famosas "vidas". Para criar vidas, partiremos de uma variável inicial chamada "***gVidas***". Para cria-la vá na "***evFase1***" clique com o botão direito do mouse e depois em "***Add global variáble***". Em suas configurações deixe-a como "***Numéric***" e adicione 5 como "***Initial Value***".

O outro critério se baseia em perder muita energia. A energia é configurar como uma variável do próprio jogador, no qual, ao atingir o valor de Zero, o mesmo perderá uma chance. Para criar uma variável de objeto, selecione o objeto de jogador e clique na sua propriedade "***Instance Variable***". Inclua a variável "***Energia***", marcando o tipo como "***Numéric***" e valor inicial igual a 3. Ou seja, o jogador terá 3 chances de sofrer danos antes de perder uma chance de jogo (Este critério será explicado na seção 1.4.10, ao criar inimigos).

1.4.6.3.1. Área de Morte

Para perder uma chance direta, foi definido que o jogador ao cair de uma plataforma para um vão "sem fundo" o mesmo irá perder uma chance. Para realizar isso, devemos colocar um Sprite conhecido como "área de morte" por toda a extensão da fase na borda inferior. Quando o jogador colidir com essa área ele perderá uma chance. Desta forma crie o Spirte "***AreaMorte***", ele não possui imagens, mas no exemplo optamos pela cor vermelha sólida para identifica-la com maior facilidade. Posicione-a na parte inferior da fase, esticando-a por toda extensão (Figura 17).

Figura 17 – Área de Morte (Estendida por toda a borda inferior da fase).

Posicionado este elemento, o que resta é apenas programar o evento de morte. Portanto no *event sheet* "***evFase1***", iremos definir a seguinte lógica: quando o jogador colidir com a área de morte, então perderá uma chance, destruirá o jogador, e reiniciará a fase. A lógica básica é essa, entretanto, para chamar a atenção a este evento, a animação também será alterada para "morte" e o jogador perderá o controle do personagem, até que a fase seja reiniciada. Além disso, é preciso verificar se não foi zerado o número de chances, que em caso positivo o jogo é encerrado por falta de chances. A Figura 18 mostra todos os eventos para esta etapa do jogo.

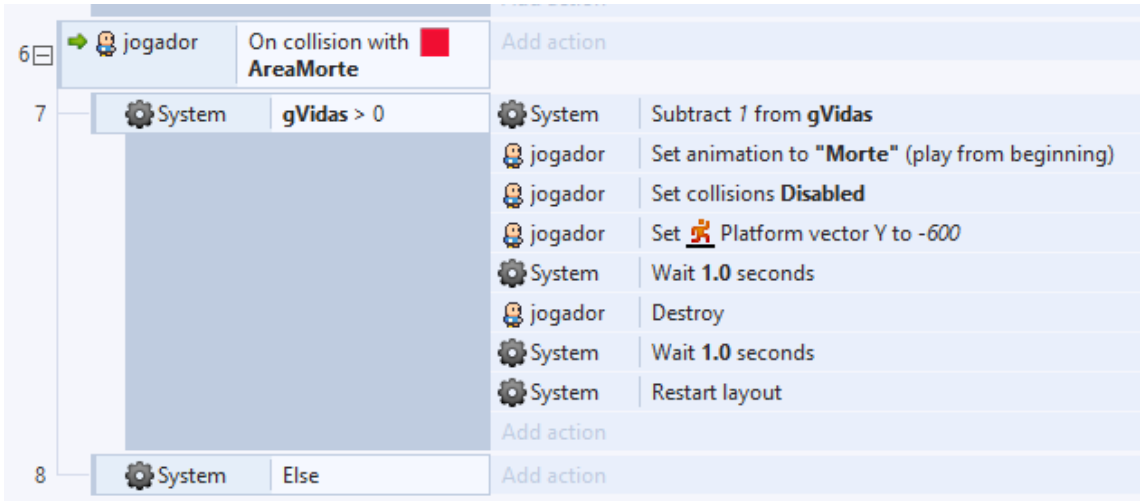


Figura 18 – Eventos da Área de Morte.

1.4.7. Criação de Plataformas Móveis

De acordo as informações da seção ***Scenarios (Places)*** no documento de design de jogo existem plataformas móveis a serem inclusas por toda a fase para possibilitar o jogador ultrapassar grandes distâncias, tanto em altura quanto em comprimento. A inclusão de plataformas móveis pode ser feita de uma maneira bem simples, e para isso, devemos criar um Sprite novo que terá o ***behavior*** "***Solid***", para que o personagem consiga subir neste objeto, mas também deverá ter o comportamento de "***Sine***". O behavior Sine permite movimentar elementos nas direções vertical ou horizontal, obedecendo a magnitude (distância) do movimento e também a período, que consiste no tempo que o objeto com o comportamento demora para percorrer toda a sua magnitude de movimento.

Desta forma, para criar as plataformas no jogo crie um novo Sprite com o nome de "***Plataforma***". Não é necessário ainda colocar uma imagem, no exemplo optamos por uma cor sólida amarela para diferenciar o Spirte. Inclua também os dois behaviors

mencionados "***Solid***" e "***Sine***". Para ilustrar no início da fase, foram colocadas duas plataformas (conforme diagrama da seção ***Scenários (Places)*** do GDD) (Figura 19).

Figura 19 – Configuração das Plataformas Móveis.

Para a plataforma móvel A, a propriedade "***Movement***" do behavior ***Sine*** deverá ser "***Horizontal***" fazendo com ela se movimente na horizontal. Enquanto a plataforma B, a propriedade "***Movement***" será "***Vertical***", fazendo a plataforma subir e descer. Neste momento ao executar o jogo, verá que a plataforma já segue os padrões de movimento especificados.

Para concluir esta seção, basta incluir as plataformas restantes no decorrer da fase, de acordo com a ilustração da Seção ***Scenários (Places)*** do documento de design do jogo.

Inserido todos as plataformas da fase, devemos então configurar sua aparência. Dê um duplo clique no objeto plataforma e abara a imagem "plataforma" dentro de "***assets/itens***" nos arquivos de construção do jogo. Ao realizar isso, suas plataformas terão a mesma aparência do cenário.

1.4.7.1. Movimentando o Cenário Junto com o Jogador

Neste ponto do jogo, já é possível perceber que a medida que o jogador caminha na fase, o cenário de fundo o acompanha, porém chega em determinado ponto que o jogador ultrapassa o cenário e o fundo fica totalmente branco. O que fazer? Aumentar a imagem de fundo? Isso pode ocasionar distorções no jogo.

Para solucionar isso, existe uma propriedade chamada ***Parallax***. Esta propriedade permite definir velocidades diferentes para as diversas camadas de um projeto de jogo no Construct 2. Por exemplo, se quisermos uma camada em que seus objetos sempre fiquem no mesmo local da câmera do jogo, devemos configurar um ***Parallax*** estático, ou seja, **0,0**. No nosso exemplo queremos que o fundo se movimente em uma velocidade menor que o movimento da camada do jogador. Como o jogador está situado na camada Principal, e a imagem de fundo do jogo na camada Fundo, devemos configurar um Parallax menor para ela. Com a camada "Fundo" selecionada, configure sua propriedade ***Parallax*** para **10,10**. Ao executar o jogo, é possível perceber que o fundo do jogo segue uma velocidade um pouco menor do que o jogador na fase, permitindo que todo o conteúdo da imagem seja preenchido no decorrer da fase.

1.4.8. Criação de Parede e Chão Atravessáveis

Em vários jogos do gênero aventura em plataforma, é comum se deparar com algumas plataformas que são passíveis de serem transportadas de baixo para cima apenas saltando. Estas são as plataformas atravessáveis e no Construct 2, para criarmos tal mecanismo basta configurar em um objeto o behavior "***Jump-Thru***". Esse comportamento permite que um chão seja atravessável por um pulo do personagem. Ele tem um comportamento de sólido, mas é atravessável.

Para criar este tipo de objeto, insira um novo Sprite no jogo e dê a ele o nome de "***PlataformaAtravessavel***", configurando também seu behavior com o comportamento de "***Jump-thru***". A Figura 20 mostra o ponto da fase onde este objeto será configurado (de acordo com o GDD). Repare que nestes pontos o jogador irá atravessar o "chão" saltando. Foi selecionado estes pontos, pois segundo o GDD do jogo, aqui acontecerá um puzzle, onde veremos nas próximas seções, que a área mais inferior irá conter um documento do cartão SUS e um inimigo que só poderá ser derrotado usando raciocínio e habilidades.

Configurada as plataformas atravessáveis, é possível executar o jogo e testá-las. É possível perceber que estes pontos ainda estão aparentes. Para que eles fiquem escondidos configure sua propriedade "***Initial Visibility***" como "***Invisible***".

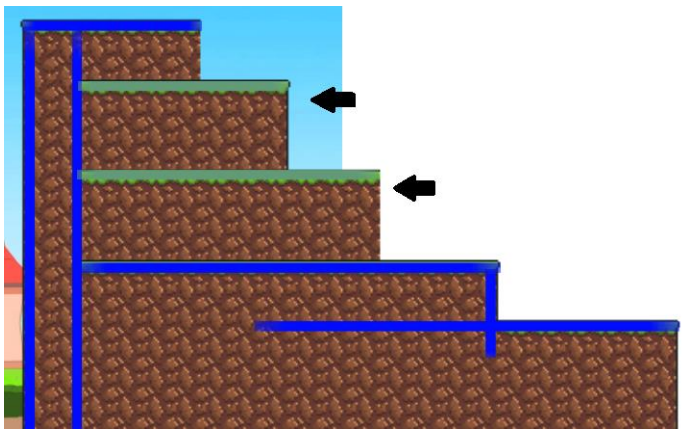


Figura 20 – Configurando Locais de Plataformas Atravessáveis.

1.4.9. Criação de Itens Coletáveis

Os itens de coleta do jogo estão descritos na seção ***Itens and Objects*** do GDD. No qual são detalhados as suas funcionalidades e localizações no decorrer do jogo. Assim como qualquer outro objeto do jogo, estes também são sprites que devem ser adicionados à fase. A Tabela 5 exibe a localização dos itens, seu nome, imagem e funcionalidade.

Tabela 5 – Itens da Fase

Item	Localização	Assets	Funcionalidade

 RG	Início da fase, dentro em baixo da primeira plataforma atravessável	“assets/itens/rg.png”	Item de desbloqueio de acesso ao posto de atendimento do SUS.
 CPF	Metade da fase, na plataforma superior.	“assets/itens/cpf.png”	Item de desbloqueio de acesso ao posto de atendimento do SUS.
 Comp. Residência	Final da fase na plataforma inferior.	“assets/itens/residencia.png”	Item de desbloqueio de acesso ao posto de atendimento do SUS.
 Cartão SUS	Meio da fase na plataforma superior.	“assets/itens/cartaosus.png”	Item de final de jogo. Só é obtido após acesso no posto de atendimento do SUS com toda a documentação.
 Computador	Início da Fase	“assets/itens/computador.png”	Informa ao jogador qual seu objetivo no jogo.

Para o objeto do Cartão SUS as propriedades de colisão e visibilidade serão desabilitadas (***Colisions* = *Disabled; Initial Visibility* = *Invisible***), pois o mesmo só poderá ser coletado pelo jogador após o mesmo visitar o posto de atendimento do SUS, de acordo com a tarefa do modelo de processo “Comparecer a Alguma Unidade do SUS” que tem como regra a coleta dos documentos para a validação e cadastro no SUS.

Outro item bastante comum em jogos de plataforma são as moedas, as quais contam como pontuação no jogo. Para incluir moedas crie um novo Sprite chamado “moeda”. Sua imagem é uma animação, que dará a impressão que ela está girando. Portanto, assim como foram feitas as animações do jogador, vamos criar a animação para a moeda. Com a animação default selecionado vá no “***Animation Frames***”, clique com botão direito do mouse, “***import frames/from strip...***”, marque 4 horizontais e 1 vertical. E para finalizar altere a propriedade Loop da animação para “***Yes***”. Agora cabe a você espalhar as moedas pela fase, onde desejar.

1.4.9.1. Contagem de Itens e Moedas

O sentido de ter itens espalhados pela fase, é permitir que sua coleta possa refletir em benefícios ao jogador. No caso dos itens de documentos, eles são necessários para permitir o acesso do jogador ao posto de atendimento do SUS para a emissão do seu

cartão. Para isso, o jogo deve permitir armazenar (ou minimamente) contar a quantidade destes itens coletados.

Realizar essa contagem, deve-se criar variáveis na ***Event Sheet*** que consigam contabilizar a coleta. Portanto, vá a ***Event Sheet "evFase1"***. Para criar variáveis clique com o botão direito do mouse e depois em "Add global variável". Inclua as variáveis chamadas "***gDocumentos***" e "***gMoedas***" todas do tipo "***Number***".

Para contabilizar as moedas e documentos devemos trabalhar com colisões de objetos, ou seja, quando o jogador encostar em um item, algo deverá ser executado. No contexto de documentos e moedas, a lógica executada será: jogador colidiu com algum destes itens, então, destrua o item, e contabilize mais um para o jogador de acordo como mostrado na Figura 21. A única diferença entre os itens será a do cartão SUS, que ao colidir com ele, o jogador finalizará o jogo.

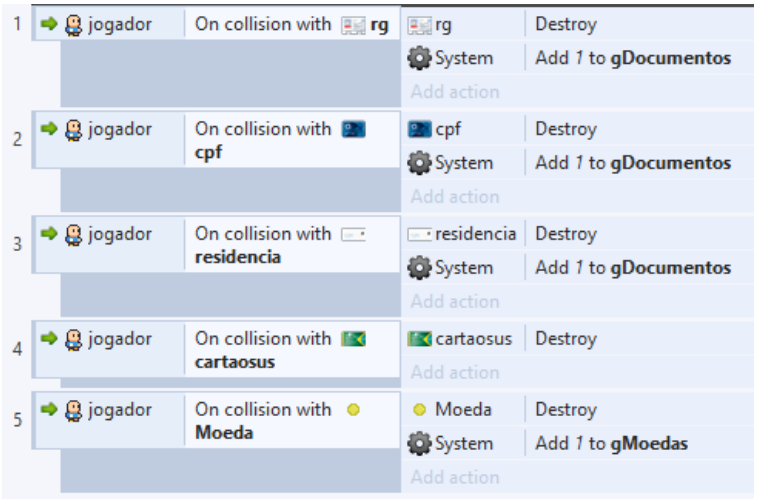


Figura 21– Eventos de Coleta de Itens para o Personagem.

Ao perder uma chance é reiniciada a fase do jogo, entretanto, do jeito que está no momento, as variáveis de moedas e de documentos não estão sendo reiniciadas. Por isso é necessário programar o evento de início do Layout para zerar tais variáveis (Figura 21).

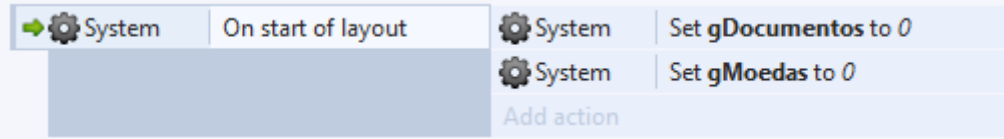


Figura 22– Eventos de Reinicio da Fase.

1.4.10. Criação dos Inimigos

Para este jogo, os designers definiram alguns inimigos que representam doenças, as quais, sem o jogador possuir o cartão SUS, o mesmo não poderá ter acesso a rede pública de saúde para trata-las. Para isso, serão criados como inimigos, as cobras (necessidade de soro antiofídico), os mosquitos (vacinas contra a dengue) e as abelhas (choque anafilático).

Para isso vamos incluir novos Sprites, um chamado "***cobra***", o "***mosquito***" e a "***abelha***". Estes sprites estão localizados dentro de "***assets/inimigos***". Assim como a

animações de personagem, que foi construída em 1.4.6.1, estes inimigos possuem imagens de movimento que devem ser inseridas na animação "***Default***".

O jogador precisa evitar a todo o custo encostar nestes inimigos, pois há modos de derrota-los, o mesmo sofrerá danos até perder as chances de concluir o objetivo. Os inimigos devem ser colocados na fase de acordo com o modelo da seção "***Scenarios (Places)***" do GDD

1.4.10.1. Cobras (Inteligência Artificial)

As cobras apenas ficam rastejando lentamente de um lado para o outro em determinado ponto da fase. Para programar tal comportamento para as cobras, são colocados Sprites limitadores, ou seja, serão colocados sprites que servirão de limitadores de movimentos paras as cobras. Quando uma cobra bater nestes limitadores, ela então dará meia-volta e rastejará para o outro lado. Estes limitadores receberão o nome de "***ColisorInimigo***", deverá ser invisível para os jogadores, funcionando apenas para colisão com as cobras. Para que a cobra se movimente, devemos configurar um behavior "***Platform***" nela, entretanto a propriedade "***Default Controls***" deverá esta como "***No***", "***Max Speed = 5***" e "***Acceleration = 50***". Além disso, para saber a direção do movimento da cobra, incluiremos uma variável de objeto nela chamada "***paraDireita***" (Figura 23).

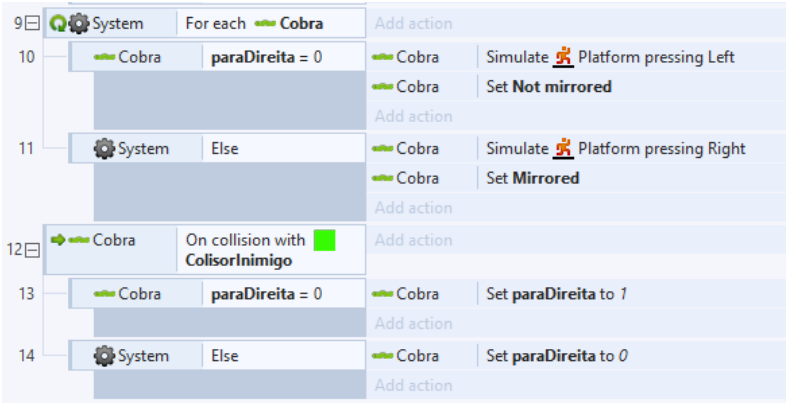


Figura 23– Eventos de Inteligência Artificial para Cobras.

1.4.10.2. Mosquitos (Inteligência Artificial)

Os mosquitos somente ficam voando em movimento aleatórios e em bando. Para simular este comportamento, devemos colocar nos mosquitos dois behaviors do tipo "***Sine***", bastando colocar diferentes propriedades de Magnitude e Amplitude de seus movimentos. Não é preciso programar nenhum evento uma vez que o comportamento da *Sine* já consegue fazer a simulação do voo do mosquito.

1.4.10.3. Abelhas (Inteligência Artificial)

As abelhas são os inimigos mais complicados, pois ao avistar o jogador ela irá prossegui-lo onde quer que ele esteja. Para este comportamento acontecer é necessário incluir na abelha o behavior "***Pathfinding***" o qual cria um caminho de um objeto para outro. Como estamos trabalhando com um jogo em que a maioria das imagens são em 16bits, é necessário ajustar a propriedade "***Cell size***" do behavior ***pathfinding*** para 16, no intuito de dar maior fluência ao movimento da abelha e "***Rotate Object = No***", para

não permitir que o objeto fique girando na tela. A programação da inteligência artificial da abelha pode ser observada na Figura 24.

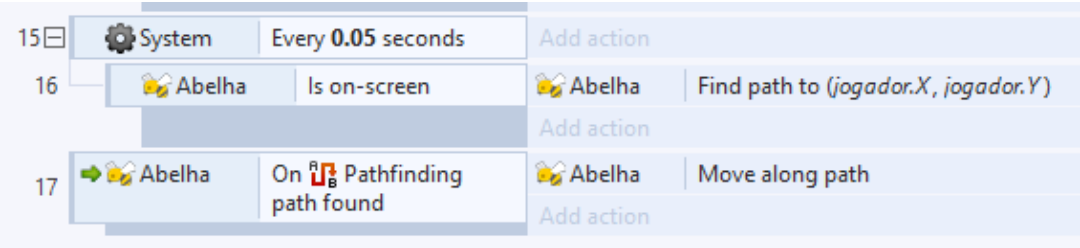


Figura 24– Eventos de Inteligência Artificial para Abelhas.

1.4.10.4. Colisão de Inimigos com o Jogador

Como mencionado anteriormente, quando o jogador encostar nos inimigos o mesmo sofrerá dano, e para isso foi criado uma variável de objeto para armazenar a energia do jogador. É comum em jogos de plataforma, quando o jogador sofre dano, o mesmo ficar alguns segundos invencível. Para cumprir este comportamento de invencibilidade momentânea, selecione o objeto do jogador e inclua nele uma nova variável de objeto chamada "invencível" do tipo numérico. Além disso, para mostrar a invencibilidade, ele ficará piscando. O comportamento de piscar precisa da inclusão do behavior "***Flash***".

Portanto, colidir com um inimigo o jogador irá sofrer um dano, mas ficará alguns segundos (3 segundos) invencível, sem poder sofrer danos novamente. Toda a lógica destes eventos pode ser observada na Figura 25.

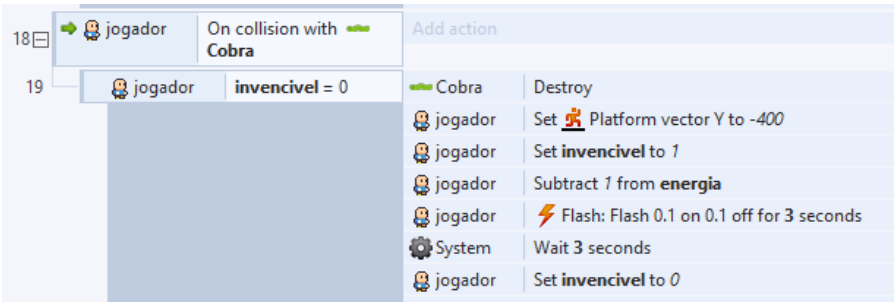


Figura 25– Eventos de Colisão com Inimigos (Todos inimigos devem ter os mesmos comandos).

1.4.11. Morte por Falta de Energia

Com os ataques dos inimigos o jogador deverá perder uma chance ao final de sua energia. Para isso basta verificar a energia do mesmo a cada ciclo de processamento do jogo. A Figura 26 exibe os comandos para realizar a funcionalidade de perda de chance por energia.

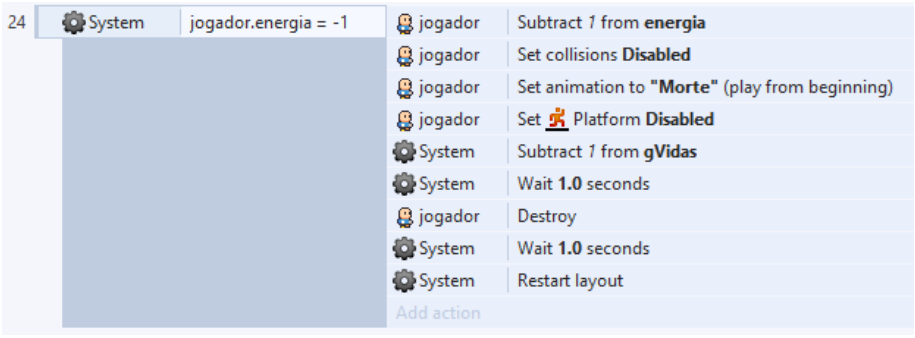


Figura 26– Eventos de Perda de Chance por Término de Energia.

1.4.12. Morte por Falta de Energia

Os HUD são itens de interface com o jogador que fornecem feedback para ele compreender o que está acontecendo com o jogo. No início do projeto foi criado uma camada somente para inclusão destes elementos. Portanto, para que os mesmos fiquem sempre expostos para o jogador na tela é necessário tomar cuidado com o ***Parallax*** desta camada, pois a mesma não pode se movimentar. Para fazer isso, coloque sua propriedade Parallax em 0,0, garantindo que o conteúdo fique estático. Além disso, é necessário tomar cuidado com a posição dos elementos na tela, pois se eles não fazerem parte da câmera do jogo eles irão desaparecer na medida que o jogador se movimenta.

Iremos então selecionar a camada HUD, e incluir os objetos de acordo com a Tabela 6.

Tabela 6 – Componentes de HUD

Objeto	Nome	Propriedade	Valor
Text	txtVidas	Text	Vidas: 5
Text	txtDocumento	Text	Documentos: 0/3
Text	txtMoedas	Text	Moedas: 0
		Horizontal Align	Right
Progress Bar	pgbEnergia	Maximum	3

Após incluir os componentes é necessário programar seu comportamento. Seus valores devem ser ajustados de acordo com cada ciclo do jogo, atrelados às variáveis de controle (Figura 27).

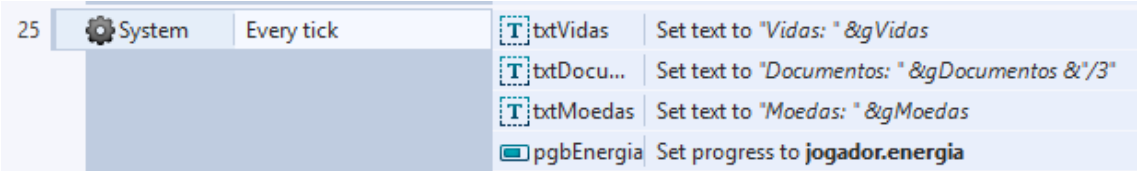


Figura 27 – Eventos de Configuração dos Objetos de HUD

1.4.13. Mensagens de Final de Jogo

Este jogo por se basear no processo de emissão do Cartão SUS brasileiro, algumas informações sobre o processo, curiosidades e contexto sobre o qual ele é prestado

precisam ser transmitidas para o jogador. Estas informações permitem o jogador a compreender como o processo é executado, suas dificuldades, motivos de sua criação, prestador e etc.

Neste jogo existem alguns itens durante a fase, os computadores, que dão estas informações aos jogadores, quando estes os acionam. Para exibir as mensagens neste jogo, no início do projeto foi criada a camada "Mensagem" exatamente para transmitir informações ao jogador. Portanto, para exibi-las, selecione tal camada e altere a suas propriedades ***Parallax*** para ***0,0***, pois queremos a mensagem estática para o jogador; e "***Initial Visibility = Invisible***" para que ela não esteja visível ao jogador em um primeiro momento.

Ainda nesta camada, incluiremos um novo Spirte com o nome de "***fundoMensagem***", no qual colocaremos nele uma cor branca sólida e alteraremos a propriedade "***Opacity = 70***". Expanda-o na tela de forma que fique parecendo uma caixa de mensagem de um software. Dentro do "***fundoMensagem***", coloque um objeto de Texto com o nome de "***txtMensagem***", e também o expanda de maneira que cubra o fundo branco da caixa. Neste momento você também pode alterar tamanho de fonte, para melhor comportar os conteúdos das mensagens.

Com isso pronto agora basta configurar os computadores para exibir as mensagens. Para isso, selecione um objeto computador e inclua uma variável de objeto com o nome de "mensagem" do tipo texto. Ao todo, de acordo com o GDD teremos 3 computadores espalhados pela fase no qual suas variáveis de mensagem devem conter os seguintes conteúdos:

1. "Prezado cidadão, para realizar qualquer atendimento na rede do SUS, você precisa do Cartão SUS. Sem ele nenhum hospital ou posto da rede pública poderá atendê-lo! A emissão do Cartão SUS é Gratuita. Você só precisa comparecer a qualquer posto do SUS com seu CPF, Documento de Identificação e Comprovante de Residência. Portanto tenha cuidado para não sofrer nenhum ataque de animais, quedas ou qualquer problema de saúde enquanto não emitir seu cartão. PORÉM CUIDADO: Existem animais pela fase que podem te machucar, como você não tem o Cartão SUS ainda, não poderá ser socorrido pela rede pública de saúde. NÃO ENCOSTE NELES.";
2. "Durante todo a fase estão espalhados itens e documentos que são úteis para a conclusão dos objetivos. Para isso você deve visitar as áreas da fase, ir ao final dela e então retornar ao início. Somente assim conseguirá concluir seus objetivos."
3. "Prezado cidadão, para emitir o cartão SUS você poderá realizar um pré-cadastro dos documentos no Portal "Saúde do Cidadão". Ao comparecer no posto de atendimento do SUS com o protocolo do pré cadastro, a emissão do seu Cartão SUS é mais rápida, pois é preciso somente validar os dados dos documentos. Não fazendo o pré-cadastro a emissão é mais demorada, pois é necessário, também, cadastrar todas as suas informações no sistema do SUS."

Para exibir tais mensagens, basta colidir com o computador. E para fechar as mensagens pressione a tecla "***Escape***" (ESC) no teclado (Figura 28).

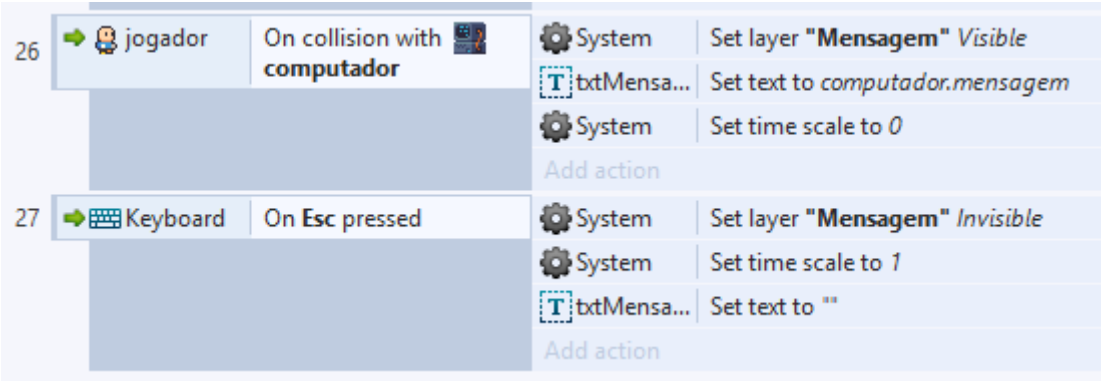


Figura 28 – Exibir e Fechar Mensagens de Informação.

1.4.14. Criação dos Finais da Fase

Assim como foi definido para a morte do jogador o final da fase irá ocorrer de acordo com as regras do processo de negócio, ou seja, o jogo termina com sucesso quando o jogador conseguir emitir o cartão SUS. O GDD define que deverá ter ao final da fase um Sprite que remete ao posto de atendimento do SUS onde o cartão será confeccionado. Desta forma, inclua um novo Sprite de nome "***PostoSUS***", com a imagem "hospital" que está em "***assets/itens***".

Haverá duas ações associadas a este Sprite. A primeira, caso todos os documentos não estejam coletados, será exibida uma mensagem para buscar os documentos pela fase e somente depois voltar ao posto de atendimento. A Segunda, habilitará a colisão e exibirá o objeto de cartão SUS, então uma mensagem sobre a localização do documento será mostrada para o jogador (Figura 29).

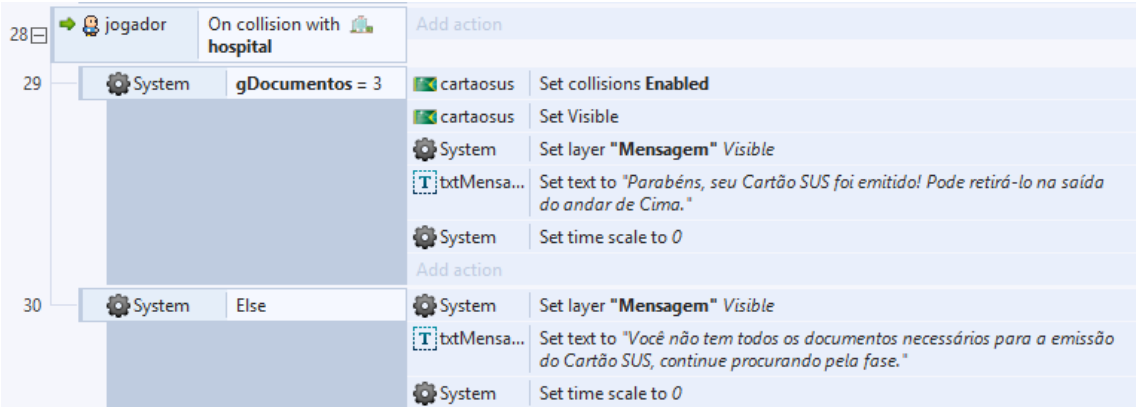


Figura 29 – Eventos do Hospital para Emissão do Cartão SUS

Para concluir o processo, basta o usuário coletar o cartão SUS no qual será exibida a mensagem de Sucesso do Final de Jogo. Ajuste o evento da colisão do cartão SUS seguindo o mostrado na Figura 30.

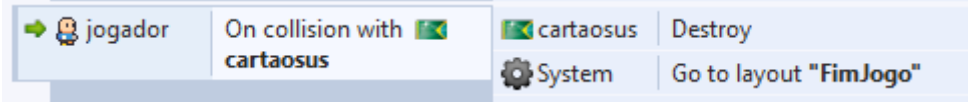


Figura 30 – Conclusão do Evento de Colisão do Cartão SUS

1.4.15. Finalização do Jogo

Para finalizarmos o jogo alguns detalhes precisam ser feitos, como: inclusão de uma tela para o game over e sons.

1.4.15.1. Tela de Game Over

Para incluir a tela de game over inclua um novo Layout e dê a ele o nome de ***"FimJogo"***. Ajuste seu tamanho para ficar do tamanho da câmera do jogo. Coloque um objeto de texto alinhado ao centro de nome "***txtGameOver***". Quando um layout novo é criado, também é gerado um novo event sheet, renomeie para "***evGameOver***".

A cada ciclo de processamento do jogo verifique se as vidas chegaram a zero, conforme a Figura 31.

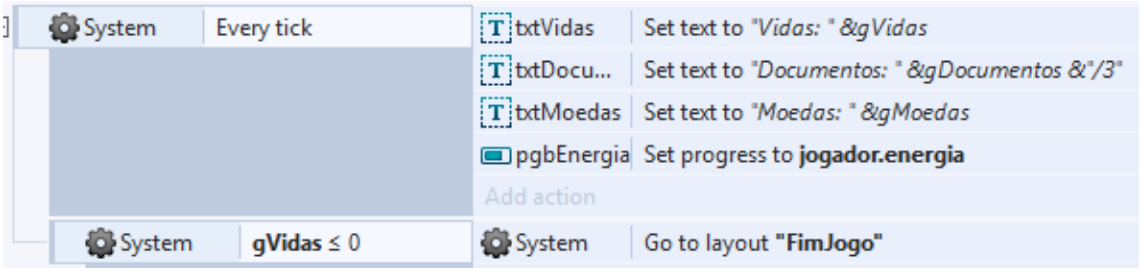


Figura 31 – Game Over por Falta de Chances

Dentro da ***Event Sheet*** "***evGameOver***" para registrar as mensagens de falhas e de sucesso corretas para o jogador coloque os eventos mostrados na Figura 32.

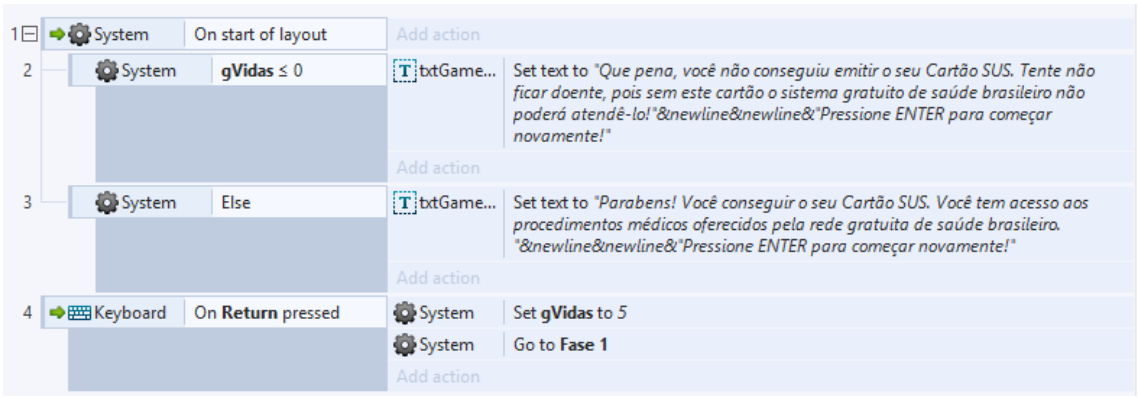


Figura 32 – Eventos da Tela de Game Over

1.4.15.2. Sons do Jogo

Um jogo de plataforma sempre remete a sons de 16 bits. Portanto devemos incluir sons no jogo. Para incluir sons no Construct é preciso incluir no projeto um objeto de áudio, e para isso, basta clicar com o botão direito do mouse em qualquer layout e inserir um novo objeto de "***Audio***". Este objeto uma vez incluso serve para todo o projeto.

Existe uma pasta em Projects chamada "***Music***" e se clicarmos nela com o botão direito, a opção "***import music***" irá surgir. Todos os áudios que vamos importar está dentro da pasta "***assets/sons***". Uma vez importados ao projeto, basta o mesmo ser utilizado nos seus eventos. Por exemplo: i) em todo evento de colisão ou morte chamaremos o evento de executar som "***dano2***" (Figura 33); ii) colisão com moedas,

"***coin***"; iii) saltar/pular, "***new_jump***"; iv) pegar o cartão sus, "***victory***"; e v) música de fundo que fica executando durante a fase, "***parte1_2***" (Figura 34).

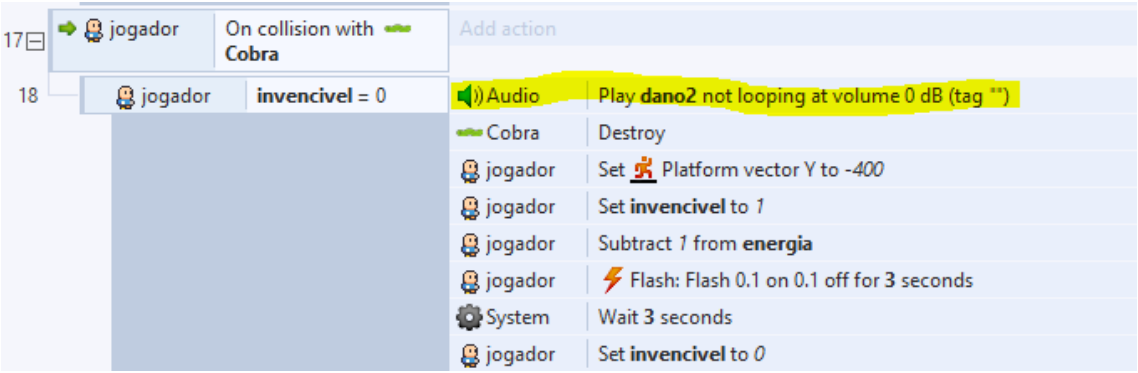


Figura 33 – Som de Tomada de Dano nas Colisões com Inimigo

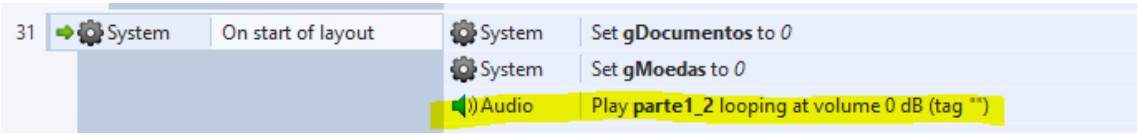


Figura 34 – Música da Fase

1.5. Conclusões

Este minicurso teve como objetivo principal a construção de um jogo digital baseado em um processo de serviço público especificamente a partir do processo de Emissão do Cartão SUS, o qual permite que todo brasileiro tenha acesso aos procedimentos médicos da rede pública de saúde.

As construções destes jogos visam auxiliar na compreensão dos cidadãos sobre a necessidade de determinados serviços públicos, não somente suas regras e etapas, mas também, transmitir aos jogadores, quem são os prestadores do serviço, o motivo da criação do processo, as dificuldades de sua entrega aos seus clientes, os valores embutidos neles e outros aspectos. Portanto, ao consumir estes jogos, o jogador é convidado a refletir sobre as instituições públicas e a forma com que elas realizam seu trabalho.

O método de design de jogos baseados em processo proposto por Classe et al. [2016] visa permitir que os designers de jogos possam construir seus jogos pensando nos contextos organizacionais, de prestação do serviço e na percepção do cidadão. Desta forma, considerando este método, este curso buscou demonstrar a execução da quarta etapa do método de design, que consistiu do desenvolvimento (programação) de um jogo digital para a compreensão do cidadão, a partir dos documentos originados pela equipe de design de jogo.

As avaliações acerca da compreensão do processo pelos jogadores, também é prevista pelo método de design, entretanto, não é o foco deste curso. Existem algumas escalas e métricas que podem ser utilizadas para realizar a avaliação do jogo. Estas abordagens focam em averiguar a percepção da qualidade do jogo sob a ótica do jogador, mas também, servem para verificar o grau de conhecimento adquirido por eles com o jogo. O que queríamos transmitir com este curso é que, por meio de um processo de negócio de prestação de serviços públicos, é possível obter elementos deste processo

através de seus modelos, construir uma especificação de requisitos de jogos e, com esta especifica, o jogo pode ser construído.

Espera-se que com este curso, os participantes consigam refletir sobre a necessidade de que a sociedade precisa compreender os serviços públicos, e que os jogos digitais podem ser uma maneira de transmitir tal compreensão. Pois acreditamos que ao compreenderem um processo, as pessoas possam refletir sobre suas necessidades, pensando inclusive em possíveis melhorias, as quais podem ser compartilhadas com as organizações.

Referências

Abt, C.C. (1970). “Serious Games”. Viking Press, New York.

Adams, E., Rollings, A. (2007). "Fundamentals of game design". Prentice Hall.

Aguilas-Saven, R.S. (2004). "Business process modelling: Review and framework". In: International Journal of production economics, v.90(2), pp. 129-149.

Alves, E. (2013). "Jogos Sérios para Ensino de Engenharia de Software". Dissertação de Mestrado, Faculdade de Engenharia da Universidade do Porto, Portugal. Disponível em: <https://repositorio-aberto.up.pt/bitstream/10216/68502/2/53127.pdf>. Acesso em: 22 de dezembro de 2016.

Araujo, R.M., Magdaleno, A.M. (2015). “Social BPM: Processos de Negócio, Colaboração e Tecnologia Social”, In Simpósio Brasileiro de Sistemas de Informação (SBSI).

Bertot, J; Estevez, E.; Janowski, T. (2016). “Universal and contextualized public services: Digital public service innovation framework”. In: Government Information Quarterly, v.33(2), pp. 211-222.

Brown, A.W.; Fisheden, J.; Thompson, M. (2014). “Revolutionising Digital Public Service Delivery: A UK Government Perspective”. Available at: <http://www.blogs.jbs.cam.ac.uk/markthompson/wp-content/uploads/2014/02/Digital-Public-Service-Delivery.pdf>. Access in: may 03 2017.

Classe, T., R. Araujo, R. (2016). “Jogos Digitais Para Participação Cidadã em Processos de Prestação de Serviços Públicos”, In Workshop de Teses e Dissertações SBSI (WTDSI).

Classe, T.; Araujo, R.M.; Xexéo, G.B. (2018). “Jogos Digitais Baseados em Processos de Prestação de Serviços Públicos: Um Estudo Exploratório”, International Journal of Game Studies (Ludica), v.2(2), pp. 26-56.

Connolly, T.M,; Boyle, E.A.; Macarthur, E.; Hainey, T.; Boyle, J.M. (2012). "A systematic literature review of empirical evidence on computer games and serious games". In: Computers & Education, v.59(2), pp. 661-686.

Crawford, C. (2003). "Chris Crawford on game design". New Riders.

Daniels, A. (2016). "Quality in Public Service Delivery". In: International Journal of Civil Service Reform and Practice, v.1(2).

Deterding, S.; Dixon, D.; Khaled, R.; Nacke, L. (2011). "From game design elements to gamefulness". In: Proceedings of the 15th International Academic MindTrek Conference on Envisioning Future Media Environments - MindTrek '11, pp. 9–11.

Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A. (2013). "Fundamentals of business process management", Springer, Berlin.

Fiocruz. (2017). "SUS: O que é?". Pense SUS. Disponível em: <http://pensesus.fiocruz.br/sus>. Acessado em: 08 de maio de 2017.

Grölund, A. (2010). "Ten years of E-Government: The "End of History" and New Beginning". In: 9TH IFIP WG 8.5 INTERNATIONAL CONFERENCE, EGOV 2010. Lausanne, Switzerland: Springer.

Ilomäki L.; Kankaanranta, M. (2009). "The information and communication technology (ICT) competence of the young". In: Handbook of research on new media literacy at the K-12 level: Issues and challenges, pp.101-118.

Juul, J. (2009). "A Casual Revolution: Reinventing Video Games and Their Players". The MIT Press.

Kneuer, M. (2016). "E-democracy: A new challenge for measuring democracy". In: International Political Science Review, v.37(5), pp. 666-678.

Ko, R.K. (2009). "A computer scientist's introductory guide to business process management (BPM)". In. ACM Crossroads, v.15(4), pp. 4.

Marston, H. R. (2015). "Gamification: Applications for Health Promotion". In: Handbook of Research on Holistic Perspectives in Gamification for Clinical Practice. p.78.

Mastrocola, V.M. (2012). "Ludificador: um guia de referências para o game designer brasileiro". São Paulo: Independente.

Michael, D., Chen, S. (2005). "Serious Games - Games that Educate, Train, and Inform.". Thomson Course Technology PTR, Boston.

OMG. (2013) "Business ProcessModelandNotation (BPMN)". OMG. Disponível em: <http://www.omg.org/spec/BPMN/2.0.2/PDF>. Acessoem: 10 de novembro de 2016.

Pflanzal, N.; Vossen, G. (2014). "Challenges of Social Business Process Management". In 47th Hawaii International Conference on System Science, pp. 3868-3877.

Portal Brasil. (2012). "Saiba Como Solicitar Seu Cartão SUS. Cidadania e Justiça. Disponível em: <http://www.brasil.gov.br/cidadania-e-justica/2012/03/saiba-como-solicitaro-seu-cartao-sus>. Acessado em: 08 de Maio de 2017.

Reijers, H.A., Slaats, T., Stahl, C. (2013). "Declarative modeling--An academic dream or the future for BPM?". Business Process Management, Springer, pp. 307-322.

Rocha, R.V., Araújo, R.B. (2013). "Metodologia de Design de Jogos Sérios para Treinamento: Ciclo de vida de criação, desenvolvimento e produção". In: XII Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames 2013), pp. 1-10.

Romero, M., Usart, M., Ott, M. (2015). "Can serious games contribute to developing and sustaining 21st century skills?". In: Games and Culture, v.10(2), pp. 148-177.

Salen, K., Zimmerman, E. (2003). "Rules of play: Game design fundamentals". Cambridge, Mass.: MIT Press.

Santos, P.M. (2014). “Framework de Apoio à Democracia Eletrônica em Portais de Governo Com Base nas Práticas de Gestão do Conhecimento”. Doctoral Thesis, Universidade Federal de Santa Catariza, Programa de Pós-Graduação em Engenharia e Gestão do Conhecimento.

Schell, J. (2009). “The Art of Game Design A Book of Lenses”. Burlington, USA: Morgan Kaufmann Publishers & Elsevier.

Schuytema , P. (2008). “Design de games: uma abordagem prática”. São Paulo: Cengage Learning, pp. 447.

Sharp, A., McDermott, P. (2008). "Workflow modeling: tools for process improvement and applications development". Artech House.

Silva, S. P. (2005). “Graus de participação democrática no uso da Internet pelos Governos das capitais brasileiras”. In Opinião Pública, v. XI(2), pp. 450-468.

Sobreira Neto, F. (2009). "Gerenci amento de Processos de Negócio – BPM segundo a Gestão Empresarial e a Tecnologia da Informação: uma Revisão Conceitual". In: XXXIII Encontro da ANPAD, São Paulo.

Souza, M.V., Medeiros, J.V. (2008). "Afinal, o que é business process management (BPM)? Um novo conceito para um novo contexto". In. Revista Eletrônica de Sistema de Informação, v.7.

Susi, T., Johannesson, M., Backlund, P. (2007). "Serious Games : An Overview". Sweden: InstitutionenFörKommunikationOch Information, pp. 28.

Tavares, A.; Soares, D.; Estevez, E. (2016). "Electronic Governance for Context-Specific Public Service Delivery: a Survey of the Literature". In: 9th International Conference on Theory and Practice of Electronic Governance, pp. 135-138.

Teixeira, C. (2011). "Os Princípios do Sistema Único de Saúde". Conferências Municipal e Estadual de Saúde, Salvador. Disponível em: <http://www.saude.ba.gov.br/pdf/OS_PRINCIPIOS_DO_SUS.pdf>. Acessado em: 08 de maio de 2017.

Vedel, T. (2006). “The idea of electronic democracy: Origins, visions and questions”. In: Parliamentary Affairs, v.59(2), pp. 226-235.

Winters, M.S.; Karim, A.G.; Martawardaya, B. (2014). “Public Service Provision under Conditions of Insufficient Citizen Demand: Insights from the Urban Sanitation Sector in Indonesia”. In: World development, v.60, pp. 31-42.

Xexéo, G.; Carmo, A.; Acioli, A.; Taucei, B.; Dipolitto, C.; Mangeli, E.; Kritz, J.; Costa, L.F.C.; Areas, M.; Monclar, R.; Garrot, R.; Classe, T.; Azevedo, V. (2017). "O Que São Jogos: Uma Introdução ao Objeto de Estudo do Ludes". Relatório Técnico do Programa de Engenharia de Sistemas e Computação, n. 5/2017 (ES-752/17), (COPPE/UFRJ). Disponível em: <http://www.cos.ufrj.br/index.php/ptBR/publicacoes-pesquisa/details/15/2766>. Acessoem 11 de maio de 2017.

Autores

Tadeu Moreira de Classe é Doutorando no Programa de Pós-Graduação em Informática da UNIRIO. Mestre em Ciência da Computação na Universidade Federal de Juiz de Fora (PGCC / UFJF). Graduado no Curso Bacharelado em Sistemas de Informações do Centro de Ensino Superior de Juiz de Fora (CES / JF). Professor universitário com mais de 4 anos de experiência na área de Sistemas de Informação e Analista de Sistemas com mais de 8 anos experiência. Membro do Grupo de Pesquisa e Inovação em CiberDemocracia (CIBERDEM / UNIRIO) e Laboratório de Ludologia, Engenharia e Simulação (COPPE / UFRJ). Tópicos de pesquisa são: Sistemas de Informação, Democracia Eletrônica e Jogos Digitais. Detalhes em http://lattes.cnpq.br/4540774422689570.

Renata Araujo é Doutora em Engenharia de Sistemas e Computação. Seus tópicos de pesquisa são: Sistemas de Informação, Democracia e Governança Digital, Gestão de Processos de Negócio e Gestão da Inovação. Atualmente ocupa a Diretoria de Educação da SBC (2018-2019). Renata é uma das editoras do livro Pesquisa e Inovação, da editora PUBL!T Soluções Editoriais. Renata é bolsista de Produtividade em Desenvolvimento Tecnológico e Extensão Inovadora do CNPq, Brasil processo no 305060/2016-3. Detalhes em http://lattes.cnpq.br/3589012014320121.

Geraldo Bonorino Xexéo tem Graduação em Engenharia Eletrônica pelo Instituto Militar de Engenharia (1988) e Doutorado em Engenharia de Sistemas e Computação pelo Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia (1994). Desde 1995 é professor da Universidade Federal do Rio de Janeiro. Tem experiência na área de Ciência da Computação, com ênfase em Banco de Dados e Engenharia de Software. Suas linhas de pesquisa atuais incluem Sistemas Peer-to-Peer, Busca, Recuperação e Extração da Informação, Qualidade de Dados e Lógica Fuzzy. Participou de projetos de consultoria em empresas públicas e privadas. Detalhes em http://lattes.cnpq.br/4783565791787812.

Capítulo 2

Metodologia de Pesquisa de Estudo de Caso em Sistemas de Informação

Nadja Piedade de Antonio, Marcelo Fornazin, Renata Mendes de Araujo

Abstract

The objective of this course is to instruct students and professionals on how to perform research using case study methodology. The course will address the following topics: what case study methodology is, how to accomplish it, how this methodology differs from other research methodologies, what is its importance and practice, and how it can be used to produce knowledge and to validate or adjust information systems in real-use situations.

Resumo

O objetivo deste minicurso é capacitar os alunos e profissionais a realizarem pesquisas utilizando a metodologia de estudo de caso. O minicurso irá abordar tópicos, tais como: o que é a metodologia de Estudo de Caso, como realizá-la, o que esta metodologia difere das outras metodologias de pesquisa, qual a sua importância e prática, e como esta pode ser utilizada para produzir conhecimento e assim validar ou ajustar os Sistemas de Informação para situações de uso reais.

2.1. Introdução

Atualmente cada vez mais percebemos a complexidade em se desenvolver e gerir Sistemas de Informação (SI), com todas as nuances e particularidades que emergem na interação dos SI com seus contextos de produção e uso. Sendo assim, como profissionais e estudantes de SI, cabem algumas reflexões na área, tanto de pesquisa como de prática, tais como: Será um SI apenas um artefato tecnológico? Como pensamos o contexto de uso dos SI? Buscamos aprender com o fracasso de um SI e

buscar entender esse fenômeno como um todo? Somos executores de demandas de desenvolvimento de sistemas ou buscamos entender a complexidade existente nos SI?

A literatura já demonstra o consenso de que os estudos de SI devem se dar de forma interdisciplinar, envolvendo conhecimentos sobre tecnologias, pessoas e organizações (Boscarioli, Araujo e Maciel, 2017; Hirschheim e Klein, 2011; Laudon e Laudon, 2014). Contudo, como estamos considerando os conhecimentos organizacionais e sociais ao pensarmos os SI? Estamos levando esses conhecimentos em conta ou focamos apenas no artefato tecnológico?

No âmbito nacional, estamos ainda distantes da interdisciplinaridade nos estudos de SI. Araújo, Fornazin e Pimentel (2017) observaram que, em meio aos artigos publicados na Revista Brasileira de Sistemas de Informação (iSys), há um desafio de reconhecer o potencial científico decorrente de uma abertura a paradigmas epistemológicos que compreendam a complexidade e a multiplicidade do mundo real, essencial para o estudo dos SI. Ou seja, é importante problematizar o contexto sociotécnico dos SI, pois caso contrário, conforme nos diz Orlikowski (2001), permaneceremos observadores passivos das transformações tecno-sociais que ocorrem ao nosso redor. A Sociedade Brasileira de Computação também coloca a complexidade dos SI como um desafio que a comunidade deve encarar (Boscarioli, Araujo e Maciel, 2017).

A complexidade dos SI em seus contextos também deve ser compreendida em processos de inovação tecnológica. Araujo e Chueri (2017) argumentam que, quando consideramos o processo de inovação, é necessário observar contextos organizacionais, sociais ou de mercado para identificação de problemas e oportunidades, bem como exige a análise do impacto do produto inovador em uso.

Assim, é fundamental elaborarmos pesquisas por meio de paradigmas metodológicos que considerem o contexto de uso dos SI. Para tanto, precisamos lançar mão de outros métodos de investigação científica que permitam compreender o SI em seu contexto, sem transportá-lo para o espaço controlado dos laboratórios e simuladores, revelando assim a complexidade interdisciplinar do SI.

Neste capítulo buscamos endereçar a questão sobre como pesquisar Sistemas de Informação em seus contextos de produção e uso. Para isso, apresentamos conceitos e técnicas do método Estudo de Casos (EC) objetivo de fomentar o uso desse método em pesquisas e em atividades profissionais na área de SI. Ao longo deste capítulo apresentamos o método de EC em seus detalhes conceituais e práticos como forma de se sistematizar as técnicas para análise de um SI em seu contexto de uso.

O capítulo está configurado da seguinte forma: além desta Introdução, na Seção 2 apresentamos o que é o EC e suas diferenças com outros casos e porque realizar EC. Na Seção 3, trataremos do projeto de pesquisa de EC, tipos existentes e os critérios para validar o projeto de pesquisa. Na Seção 4, abordaremos as características que o pesquisador precisa ter para se realizar uma coleta de dados eficaz e os tipos de evidências de coleta de dados. Na Seção 5, serão apresentadas técnicas para análise das evidências coletadas e como devem ser reportados os casos e o que devemos ter em mente ao elaborar o relatório de EC. Na Seção 6 apresentaremos elementos para um EC "exemplar" e concluiremos o capítulo.

2.2. O que é o Estudo de Caso?

O Estudo de Caso é um método de pesquisa adequado para situações em que é difícil se estabelecer um delineamento claro entre fenômeno estudado e seu contexto, de modo que não é possível investigar o fenômeno fora de seu ambiente prático (Yin, 2001). Por exemplo, ao se estudar como um sistema cresceu em escalabilidade tecnológica e social dentro de uma organização, não é possível separar o SI da organização, de seus atores e práticas. Quando se deseja estudar o impacto de um SI em um contexto social também não é possível separá-lo das características específicas deste contexto.

Por meio do EC podem ser realizadas entrevistas, observação direta e análise de documentação e artefatos, de modo a explicar o fenômeno da presença de um SI em um determinado contexto. Outras situações similares, tais como: mudança organizacional, transferência de tecnologia e resistência ao uso de SI, podem requerer análises complexas do SI em seu contexto de uso.

De acordo com Benbasat, Goldstein e Mead (1987) a pesquisa através de EC é particularmente apropriada para certos tipos de problemas: aqueles em que a pesquisa e a teoria estão em seus estágios iniciais de formação e onde serão estudados problemas que são baseados na prática, onde as experiências dos atores são importantes e o contexto de ação é crítico.

Seguem algumas características de um EC (Benbasat, Goldstein e Mead, 1987; Yin, 2001):

- O fenômeno é examinado em um ambiente natural;
- Baseia-se em várias fontes de evidências e os dados são recolhidos por múltiplos meios;
- São examinados diversas entidades ou atores: pessoas, grupos ou organizações e os resultados dependem dos um olhar integrativo do pesquisador;
- O pesquisador deve ter uma atitude positiva quanto a exploração a fim de se obter bom resultado;
- Enfrenta uma situação tecnicamente única em que haverá mais variáveis de interesse do que pontos de dados;
- Precisam ser desenvolvidas proposições teóricas para conduzir a coleta e a análise dos dados.

Os EC, quando comparados a outros métodos, apresentam peculiaridades e diferenças em seus pressupostos e critérios de investigação, conforme sintetizado na Tabela 2.1.

Em relação aos métodos experimentais, dominantes nas pesquisas de Computação, os EC se diferenciam na forma de elaboração da pergunta de pesquisa, nos procedimentos e coleta e análise de dados e nos critérios de validação. Os métodos experimentais, sejam eles experimento, teste de laboratório ou simulação, pressupõem a construção do artefato tecnológico no ambiente controlado do laboratório e em seguida a submissão do artefato a sucessivos testes de modo a se obter medidas quantitativas

que demonstrem a diferenciação ou avanço tecnológico do artefato em relação aos resultados previamente demonstrados na literatura.

A diferença mais importante do Estudo de Caso para os Experimentos, é a formulação de perguntas de pesquisa do tipo "COMO" e "POR QUE?", as quais não podem ser respondidas apenas por medidas quantitativas, mas necessitam de uma explicação contextual, por exemplo, que descreva o SI em sua trajetória cronológica e dinâmica social. A fim de se buscar uma explicação para estas perguntas e considerando que o estudo de SI envolve tecnologia, pessoas e organizações não podemos levá-lo ao laboratório, pois assim, iremos retirar o SI de seu contexto e separá-lo de seus usuários, desmantelando a complexidade que as questões do tipo "COMO" ou "POR QUÊ?" requerem para serem respondidas. Desse modo, faz-se necessário ir até o ambiente de uso do SI para coleta de dados que subsidiarão a formulação de um resposta às perguntas relacionadas ao fenômeno em estudo. Nesse sentido, são empregadas técnicas de coleta de dados baseadas em entrevistas abertas, observação direta e análise de documentação e artefatos de modo a explicar o fenômeno em estudo.

Tabela 2.1 - Diferentes estrategias de pesquisa (Yin, 2001).

estratégia	forma da questão de pesquisa	exige controle sobre eventos comportamentais?	focaliza acontecimentos contemporâneos?
experimento	como, por que	sim	sim
levantamento	quem, o que, onde, quantos, quanto	não	sim
análise de arquivos	quem, o que, onde, quantos, quanto	não	sim/não
pesquisa histórica	como, por que	não	não
estudo de caso	como, por que	não	sim

Como exemplo de EC em SI, podemos citar o estudo de Lapointe e Rivard (2005), que por meio da análise de três casos referentes a implementação de um SI de Prontuário Eletrônico em hospitais descreveram a dinâmica temporal e organizacional da resistência a Sistemas de Informação. As autoras formularam um Modelo de Resistência a Implementação de SI, o qual explica como a resistência a SI emerge, progride e culmina nas organizações.

Outro exemplo é o EC de Fornazin e Joia (2014) que buscaram descrever como o sucesso ou fracasso de um SI não depende somente das características do artefato tecnológico, mas ocorreu também por meio de negociações entre os atores envolvidos no processo de implantação do SI. Os autores, ao observar o processo de implantação do

SI em seu contexto, observaram que o sucesso de um SI é um processo de construção coletiva que se desenrola ao longo do tempo.

Em resumo, no EC o pesquisador precisa estar inserido no contexto do SI a ser estudado a fim de conseguir observar o fenômeno na sua forma natural e a estratégia de pesquisa a ser escolhida está intrinsecamente ligada ao tipo de objetivo que se pretende alcançar e comunicar com a sua pesquisa.

2.2.1. Estudos de Caso, Casos de Ensino e Casos de Sucesso

Nas nossas atividades nos deparamos com diversos casos, por exemplo, nas atividades de ensino utilizamos casos de ensino, empresas apresentam seus produtos por meio de casos de sucesso. Contudo, o método de Estudo de Casos se diferencia dos casos de ensino e dos casos de sucesso mencionados por alguns elementos.

Os casos de ensino são utilizados como estratégia metodológica para exemplificar situações práticas em que uma determinada técnica, método ou tecnologia foi aplicada por uma organização. Assim, por meio dos casos de ensino os estudantes podem compreender como um conceito abstrato é aplicado em um contexto prático. Nesse ponto, os casos de ensino se aproximam das pesquisas baseadas em EC. Contudo, os casos de ensino não possuem o objetivo de de produzir novos conhecimentos ou analisar novas técnicas em seus contextos prática. Isto é, os casos de ensino objetivam disseminar conhecimento, enquanto as pesquisas de EC têm por objetivo produzir novos conhecimentos.

O outro tipo de caso, o caso de sucesso, é aquele utilizado por empresas para ilustrarem seus produtos e tecnologias em situações práticas. Nesse caso, as empresas relatam um caso de uma determinada tecnologia em um contexto prático de modo a evidenciar os potenciais benefícios da tecnologia e mostrar sua aplicabilidade a possíveis clientes. Assim como o caso de ensino, o caso de sucesso se aproxima das pesquisas de EC ao relatar uma situação de uso de SI em seu contexto prático. Contudo, os casos de sucesso não passaram pelo processo de validação científica, por exemplo, revisão por pares e debates em bancas e congressos, nem necessariamente se baseiam na literatura ou nas técnicas de coleta e análise de dados aceitas pela comunidade científica. Assim, uma pesquisa baseada em EC, diferente de um caso de sucesso, deve observar critérios de qualidade e rigor científico, dialogar com a literatura prévia e se submeter ao escrutínio da comunidade científica. Além disso, no Estudo de Caso, o pesquisador deve buscar um distanciamento crítico (Walsham, 2006) para analisar o caso sem ser capturado por ele e assim apresentar suas teorizações de forma independente de outros interesses.

2.2.2. Porque fazer Estudos de Caso em Sistemas de Informação?

O método de Estudos de Caso contribui para nos ajudar a produzir conhecimento sobre os fenômenos organizacionais, das pessoas e da tecnologia, ou seja, em tudo que compõe um SI de uma forma totalmente diferente das outras metodologias de pesquisa. A necessidade pelos EC surge do desejo de se compreender fenômenos sociais complexos. Em resumo, o EC permite uma investigação para se preservar as características holísticas e significativas dos eventos dos SI.

Os EC não são um método exclusivo da área de Sistemas de Informação. Pelo contrário, a área de SI vem se apropriando do método de EC na medida em que estamos despertando nossa consciência para a necessidade de se estudar os SI em seus contextos práticos de uso. Yin (2001) nos mostra que os EC são usados para pesquisas em psicologia, sociologia, ciência política, antropologia, entre outros, e nas áreas voltadas à prática, como planejamento urbano, administração, política pública, trabalho social e educação. Além disso, na computação, a área de Engenharia de Software Experimental vem advogando sobre a importância dos métodos qualitativos e dos Estudos de Caso.

Nas ciências sociais, os EC muitas vezes se focam a compreender o contexto de interesse da pesquisa, deixando a problematização da tecnologia para um segundo plano, tratando os artefatos tecnológicos como caixas pretas ou ferramentas prontas. Já na Computação, os EC são utilizados majoritariamente para se obter acesso ao contexto de uso de um software em maior profundidade do que experimentos ou levantamentos poderiam prover, sem necessariamente problematizar o contexto de uso do software.

Entretanto, os EC em SI se diferenciam das duas abordagens anteriormente citadas (Ciências Sociais e Engenharia de Software) pois se dedica a estudar a relação entre o SI e seu contexto de uso, problematizando ambos. Nesse ponto, nos baseamos na definição de Orlikowski e Iacono (2001), na qual o SI é visto em conjunto com o seu contexto por meio de interações dinâmicas entre pessoas e tecnologias, em que os SI moldam práticas organizacionais, ao mesmo tempo que são moldados por elas. Os SI criaram e foram criados por novas práticas, por exemplo, comunidades de software livre, comércio eletrônico, redes sociais, novas formas de trabalho e de participação política. Além disso, novas questões emergiram, tais como desafios de propriedade intelectual e privacidade. É justamente nesse ponto de relação entre os SI e seus contextos de uso que os pesquisadores de SI podem explorar ao limite suas habilidades de problematizar ao mesmo tempo a tecnologia e seu contexto.

Do ponto de vista da pesquisa científica, Pozzebon e Freitas (1998) trazem a necessidade de investigar o relacionamento entre os SI através de aspectos humanos (cognitivos, psicológicos, sociais, culturais) e aspectos técnicos (ergonomia, projeto), que devem ser levados em conta não de forma isolada, mas integrada, por uma abordagem sociotécnica. Isto nos remete à exploração de diversos métodos de pesquisa. O rigor científico que se espera atingir na área de SI enquanto disciplina científica sugere a não restrição a uma única abordagem, notadamente a quantitativa, mas que se busque explorar uma variedade de métodos, sobretudo qualitativos, como o EC. Além disso, podemos criar teorias a partir de EC, como por exemplo: Teoria da Resistência em SI, Paradoxo de SI e Transferência de SI.

Do ponto de vista profissional, a possibilidade de realizar EC pode ampliar a prática de desenvolvimento e implantação de SI no sentido de permitir ao profissional ou as organizações a compreenderem SI em seu contexto, respondendo a perguntas importantes, em geral negligenciadas, a respeito de **como** e **porque** SI são utilizados, que significados estabelecem organizacional ou socialmente, que impactos ou resistências geram com seu uso. Todos estes aspectos, quando não estudados pelas organizações, tem o potencial de gerar uma série de problemas comuns relacionados ao

desenvolvimento e gestão de SI, como: aplicações que não cuprem seus objetivos esperados quando implantados, desvios e contornos de utilização, gastos de recursos com ajustes e manutenções, descontinuidade, entre tantos outros. Veremos na seção seguinte os tipos de Projeto de Pesquisa de EC e os critérios de validade de pesquisa para o EC.

Um último ponto a ser discutido nessa seção de justificativa é o posicionamento epistemológico do Estudo de Caso. Nas ciências e nas pesquisas de SI internacionais, há um consenso de que existem duas grandes abordagens para realização dos EC. Por um lado há os EC positivistas, os quais buscam produzir relatos objetivos a partir dos casos, apresentando modelos e variáveis que possam posteriormente ser testados por métodos quantitativos. Entre os autores que trabalham na abordagem positivista estão Benbasat, Paré, Yin etc. Uma outra abordagem, define-se como interpretativa, o qual entende que o SI são produzidos de maneira subjetiva a partir das relações entre as pessoas e desse modo não são um objeto estritamente técnico, mas um fenômeno a ser interpretado. Nos EC interpretativas, não se busca uma análise objetiva do caso, mas um interpretação narrativa. Os autores que advogam em favor dos EC interpretativos são, por exemplo, Walsham (2006) e Orlikowski e Iacono (2001). Contudo, na comunidade Brasileira de Computação há pouco conhecimento sobre pesquisa interpretativa. Assim, para promover um maior diálogo com a comunidade científica e um entendimento dos EC mais próximo das pesquisas experimentais, neste capítulo apresentamos conceitos e técnicas de EC positivistas, mas também trazemos alguns elementos de EC interpretativos.

2.3. Projeto da Pesquisa de Estudo de Caso

Conforme Yin (2001) explica, um projeto de pesquisa constitui-se de um plano de ação para sair de um ponto e chegar a outro. Teremos um conjunto de perguntas a serem respondidas e um conjunto de conclusões, que serão as respostas das questões. A fim de começarmos a pensar no projeto de pesquisa, devemos que ter em mente alguns pontos chave, tais como: quais questões estudar, quais dados são relevantes, quais dados coletar e como analisar os resultados.

Segundo Yin (2001), temos cinco elementos essenciais para se elaborar o projeto do EC, a saber: as questões do estudo; propostas de estudo; unidades de análise; a lógica que une os dados às propostas e, por fim, os critérios para se interpretar as descobertas.

Primeiramente, as **questões** de EC são aquelas já citadas de formato: "COMO" e "POR QUE" a fim de se buscar entender o fenômeno do SI na sua totalidade. Devemos definir, com clareza, a natureza das suas questões de estudo nesse sentido. Por exemplo, Lapointe e Rivard (2005) em seu estudo formularam a seguinte pergunta: “Como se deu o processo de implementação do software de Prontuário Eletrônico e o seu uso ou não uso pelos médicos do hospital?”. A partir dessa pergunta, podemos notar que as autoras buscaram compreender a dinâmica temporal de um SI no contexto hospitalar, levando em consideração um grupo de atores que interagiam com o SI (os médicos).

Quanto às **propostas de estudo**, estas destinam atenção a algo que deve ser examinado dentro do escopo do estudo. São importantes para manter o pesquisador ou o analista do EC dentro do limite da sua pesquisa. Pode ser que não haja propostas de estudo dentro de um projeto de pesquisa. Isso acontece nos tipos de estudos

exploratórios, onde o tópico da pesquisa é o tema da exploração. São chamados de estudo de caso exploratórios. Trazendo mais uma vez o EC de Lapointe e Rivard (2005), a proposta de estudo é estudar o processo de implementação do SI de Prontuário Eletrônico e as resistências do uso deste SI pelos médicos do hospital. Um processo de implantação de SI é complexo e pode durar meses ou anos, ou seja, não é possível repetir esse processo em laboratório, fora de seu contexto.

No item **unidade de análise**, deve ser definido o que está sendo estudado no EC. Uma unidade de análise pode ser a organização em que o SI está sendo utilizado, o processo de implantação do SI, um grupo de pessoas que usa do SI, um grupo de organizações que trocam informações por meio de SI. A unidade de análise está relacionada à maneira como as questões do caso foram definidas. Caso a questão de pesquisa seja voltada à interação individual de pessoas com o SI, a unidade de análise pode ser definida em nível mais próximo dos usuários, em que serão coletadas informações detalhadas sobre as pessoas, suas impressões sobre o SI e também sobre o sistemas e seus detalhes. Por outro lado, se questão de pesquisa se dedicar a investigar o SI de maneira mais geral na organização (por exemplo, mudanças na estratégia e na estrutura organizacional), a unidade de análise pode ser definida como a organização ou o SI, assim, serão coletadas informações mais gerais sobre os grupos de usuários, a estrutura organizacional, as funções do SI de maneira mais ampla. A título de exemplo no EC de Lapointe e Rivard (2005), a unidade primária de análise é o SI de Prontuário Eletrônico. Assim, foi possível analisar as impressões a respeito do SI ao longo do tempo, por diferentes grupos e em diferentes hospitais. Tomando o SI de Prontuário Eletrônico como unidade de análise foi possível analisar o SI e seu contexto por diferentes ângulos.

Após isso, devemos ligar as informações coletadas às propostas do estudo. **Ligar os dados às propostas** pode ser feito de várias maneiras. Uma das maneiras é utilizar a “adequação ao padrão", (Campbell, 1975), por meio da qual várias partes da mesma informação do mesmo caso podem ser relacionadas à mesma proposição teórica. Esta adequação pode ser com efeito (mostrando que existe uma relação) ou sem efeitos (não há correlação). No EC de Lapointe e Rivard (2005), a proposta de estudo é se analisar se na implementação do Prontuário Eletrônico houve resistências no uso deste SI. Nos EC que foram realizados, mostrou que houve resistências no uso: cinco anos após a introdução do sistema, apenas o primeiro módulo ainda estava em uso. O segundo módulo retirados após grandes conflitos, primeiro entre os enfermeiros e os médicos e depois entre os médicos e a administração. Houve uma apatia e falta de interesses no uso do Prontuário Eletrônico. O sistema foi utilizado em menos da metade da sua capacidade, e havia e não havia planos para expandir o seu uso. Desta forma houve uma relação entre a proposta e as informações coletadas no EC.

Quanto aos **critérios** não há uma maneira precisa de se estabelecer os critérios para a interpretação dessas descobertas. No EC de Lapointe e Rivard (2005), quanto aos critérios para interpretação do EC, um dos critérios utilizados foi a análise de caso cruzado, realizada usando duas táticas para melhorar a probabilidade de capturar novas descobertas entre os dados (Eisenhardt, 1989). Em primeiro lugar, foram selecionadas categorias para identificar padrões em cada dimensão da estrutura. Segundo, os casos

foram comparados em pares para identificar as sutis semelhanças e diferenças entre eles.

Por fim, para embasar o projeto de pesquisa é importante termos alguma teoria que iremos utilizar de embasamento para o nosso EC. Assim, o projeto completo de pesquisa fornecerá uma direção forte ao determinar quais dados devem ser coletados e as estratégias de análise desses dados. Exemplo: Para estudos de SI, temos a Teoria da Sociomaterialidade (Wanda Orlikowski) que pode ser utilizada para problematizar um SI através de uma abordagem diferenciada, sociotécnica, de uma forma conjunta, com todos os seus elementos e atores. Nada impede que utilizemos teorias das Ciências Sociais, Filosofia ou Ciências Políticas para problematizar o nosso contexto.

2.3.1. Tipos de Projeto de Estudo de Caso

De acordo com (Yin, 2001), antes da coleta de dados é importante definir o tipo de projeto do EC de modo a organizar os procedimentos de campo. A Tabela 2.2. apresenta um diagrama com os tipos de projetos de EC, quais sejam: caso único holístico; caso único com múltiplas unidades de análise; casos múltiplos (holísticos); e casos múltiplos com múltiplas unidades de análise.

Tabela 2.2. Tipos básicos de Projetos para o Estudo de Caso. Adaptada de Yin (2001).

	Projeto de Caso Único	**Projeto de Casos Múltiplos**
Holísticos - unidade única de análise	Remontando a Rede de Atores na implantação de um Sistema de Informação em Saúde (Fornazin e Joia, 2014)	Um Modelo Multinível de Resistência à Implementação de TI (Lapointe e Rivard, 2005)
Incorporados - unidades múltiplas de análise	Estudo de caso único com multiplas unidades de análise	Estudo de casos múltiplos com multiplas unidades de análise

Um caso único será usado quando for testar uma teoria. Um dos exemplos citados foi o EC de Fornazin e Joia (2014) que utiliza a Teoria Ator-Rede para embasar a análise e atuação dos atores na implantação do SI em um hospital. O caso de Fornazin e Joia (2014) é um caso holístico. Uma das características do caso holístico é que as questões iniciais do estudo podem apresentar uma orientação, mas, à medida que o estudo avança, pode surgir uma orientação diferente, e as evidências começam a se voltar para questões diferentes. De acordo com Orlikowski e Baroudi (2001), o pesquisador não apenas descreve um fenômeno nas palavras e dos atores, mas presume-se promulgar a realidade social que se está estudando.

O projeto de caso único ainda pode ser incorporado com múltiplas unidades de análise. Nesse tipo, o caso pode ter mais de uma unidade de análise dentro do mesmo contexto. Assim, as diferentes unidades que permitem análises sobre diferentes questões que complementam e permite uma melhor compreensão do caso.

Por sua vez, o projeto de caso múltiplo, podem conter várias análises de unidades incorporadas ou unidades de processo. Trouxemos mais uma vez o EC de Lapointe e Rivard (2005) que estuda três casos diferentes para identificar a resistência de um Prontuário Eletrônico. O tipo de projeto de caso é múltiplo quando o mesmo estudo possui mais de um caso. Assim, são analisados os vários casos individualmente e os resultados são posteriormente comparados de modo a se identificar abstrações teóricas. A condução de um EC múltiplos pode exigir tempo e recursos além daqueles que um pesquisador possui.

Projetos de caso múltiplos podem constituir-se em um estudo de caso holístico ou incorporado. A diferença entre os dois projetos é o tipo de fenômeno que está sendo estudado. Este tipo de projeto apesar de ser usado, são mais caros e consome mais tempo para serem realizados. Qualquer utilização de projetos de casos múltiplos deve seguir uma lógica de replicação, e não de amostragem, e o pesquisador deve escolher cada caso cuidadosamente. Os casos individuais, dentro de um projeto de estudo de casos múltiplos, podem ser qualquer um dos dois tipos de projetos. Veremos a seguir alguns dos critérios de validade de pesquisa para o nosso EC.

2.3.2. Critérios de validade de pesquisa do Estudo de Caso

Durante toda a condução de uma pesquisa de Estudo de Caso, precisamos nos atentar a critérios de qualidade da pesquisa. Com isso, a pesquisa de EC pode assegurar sua validade e demonstrar o conhecimento produzido para a comunidade científica. Diferente dos métodos experimentais e de levantamento que se utilizam de técnicas estatísticas para assegurar sua validade, nos EC os critérios são qualitativos e discursivos.

Assim, apresentamos alguns critérios de aferição da qualidade de um EC. Robert Yin (2001), conforme apresentado na Tabela 2.3. sugere quatro critérios para verificar a qualidade de um EC, quais sejam: validade do constructo, validade interna, validade externa, confiabilidade (Yin, 2001).

- **Validade do constructo:** estabelecer medidas operacionais corretas para os conceitos que estão no estudo. Este ponto é especialmente crucial no estudo de caso. Yin (2001) nos sugere “selecionar os tipos específicos de mudanças que devem ser estudadas em relação aos objetivos originais do estudo de caso e demonstrar que as medidas selecionadas dessas mudanças realmente refletem os tipos específicos de mudanças que foram selecionadas.”
- **Validade interna** (apenas para estudos explanatórios ou causais): estabelecer uma relação causal, por meio da qual são mostradas certas condições que levem a outras condições.
- **Validade externa:** estabelecer o domínio ao qual as descobertas de um estudo podem ser generalizadas.
- **Confiabilidade:** demonstrar que as operações de um estudo - como os procedimentos de coleta de dados - podem ser repetidas, apresentando os mesmos resultados.

Esses quatro critérios pressupõem que o estudo seja conduzido de modo a produzir resultado objetivos, tais como variáveis e relações causais entre elas.

Tabela 2.3. - Critérios de Validade para o Estudo de Caso propostos por Yin (2001).

testes	tática do estudo de caso	fase da pesquisa na qual a tática deve ser aplicada
validade do constructo	– utiliza fontes múltiplas de evidências – estabelece encadeamento de evidências – o rascunho do relatório estudo de caso é revisado por informantes-chave	coleta de dados coleta de dados composição
validade interna	– faz adequação ao padrão – faz construção da explanação – faz análise de séries temporais	análise de dados análise de dados análise de dados
validade externa	– utiliza lógica de replicação em estudos de casos múltiplos	projeto de pesquisa
confiabilidade	– utiliza protocolo de estudo de caso – desenvolve banco de dados para o estudo de caso	coleta de dados coleta de dados

Já pesquisadores da linha interpretativa nos oferecem critérios que permitem assegurar a validade de um estudo mais dialógico, em que o conhecimento é produzido a partir da interação do SI com o seu contexto. Entre esses critérios estão: serendipidade (Eisenhardt, 1989); autenticidade e plausibilidade (Golden-Biddle e Locke 1993); círculo hermenêutico, da contextualização, da interação entre pesquisadores e sujeitos, da abstração e generalização, do raciocínio dialógico, das múltiplas interpretações e da suspeita (Klein e Meyers, 1999). A Tabela 2.4. apresenta uma síntese dos diferentes critérios para validação de estudos de caso.

Tabela 2.4. - Critérios de Validade para o Estudo de Caso baseados em autores da linha interpretativa.

Critérios de Validade	Descrição	Fase da Pesquisa
serendipidade	**abarcar o desconhecido: acolher as novas questões que surgem ao longo da pesquisa**	**coleta e análise de dados**
autenticidade	**verifica se pesquisador esteve no campo de pesquisa**	**coleta de dados**
plausibilidade	**verifica se a história faz sentido**	**análise de dados**
criticidade	**motiva os leitores para re-examinar pressupostos que fundamentam o Estudo de Caso**	**coleta e análise de dados**
círculo hermenêutico	**fazer sentido a leitura do todo para as partes e das partes para o todo**	**análise de dados**
contextualização	**fundo histórico e social do Estudo de Caso para se entender o contexto**	**projeto de pesquisa**
interação entre pesquisadores e sujeitos	**reflexão crítica em como os dados da pesquisa são construídos entre interação pesquisador e sujeito**	**coleta e análise de dados**
abstração e generalização	**interpretação dos dados através de uma teoria**	**análise de dados**
raciocínio dialógico	**sensibilidade para dar transparência nos preconceitos iniciais do pesquisador confrontando com os resultados da pesquisa**	**análise de dados**
múltiplas interpretações	**diferenciar as possíveis diferenças de interpretações entre participantes**	**análise de dados**
suspeita	**sensibilidade para identificar distorções nas narrativas dos participantes**	**análise de dados**

Segue o detalhamento sobre os critérios de validade de pesquisa do EC.

- Serendipidade: tem relação com as descobertas que não são o cerne do EC;
- Autenticidade: interação com o material empírico, o pesquisador de fato deve estar em campo para coletar as informações. (Pozzebon, 2009).
- Plausibilidade: sensibilidade para realizar a interpretação da história do EC na análise dos dados;

- Círculo Hermenêutico: fazer com que as informações dentro do EC façam sentido entre todas elas.
- Contextualização: este critério é de extrema importância. A contextualização do EC, onde ocorre, em quais circunstâncias, o fundo histórico, social ou econômico, se a organização é pública ou privada, se há governo envolvido na construção deste SI, dentre outros.
- Interação entre Pesquisador e Sujeito: como o pesquisador interage com os entrevistados, buscando sempre ter uma visão crítica não deixando que seus preconceitos surjam no momento de registrar a entrevista, para não enviesar a pesquisa.
- Abstração e generalização: neste critério, agregamos com a visão de Walsham (1995), que argumenta que existem quatro tipos de generalizações de estudos de caso interpretativos: o desenvolvimento de conceitos, a geração de teoria, o desenho de implicações específicas, e a contribuição de *insigh*s. A teoria desempenha um papel crucial na pesquisa interpretativa.
- Raciocínio Dialógico: este princípio exige que o investigador confronte seus preconceitos que guiaram o projeto original da pesquisa com os dados que emergem através do processo de pesquisa. O ponto mais fundamental é que o pesquisador deve fazer a base intelectual histórica da pesquisa tão transparente quanto possível para o leitor e si mesmo.
- Múltiplas interpretações: este princípio requer que o pesquisador examine as influências que o contexto social tem sobre as ações em estudo ao procurar e documentar vários pontos de vista, juntamente com as razões para eles. A análise de razões podem incluir buscando compreender os conflitos relacionados ao poder, economia ou valores. Além disso, o pesquisador deve enfrentar as contradições potencialmente inerentes aos vários pontos de vista uns com os outros, e rever a sua entendimento em conformidade. (Klein and Myers, 1999)
- Suspeita: o pesquisador deve estar sempre com o pensamento crítico atentos para descoberta de falsos preconceito na pesquisa.

Com estes critérios de validade em mente, vamos agora analisar na próxima seção como se dá a coleta de dados em campo.

2.4. Coleta de Dados em campo

A coleta de dados em campo não é um procedimento que segue uma rotina específica. Segundo Yin (2001), a metodologia de EC exige mais do pesquisador no que se refere ao seu intelecto, ego e emoções em relação a outras metodologias de pesquisa. Isto se dá porque no EC deve haver uma contínua interação entre as questões teóricas que estão sendo estudadas e os dados que estão sendo coletados, a fim de que se possa realizar um EC com qualidade.

Para que possamos realizar uma boa pesquisa, algumas qualidades são necessárias no pesquisador. Primeiramente o pesquisador deve ter a capacidade de fazer boas perguntas, ser um bom ouvinte e interpretar as respostas sem se deixar ser enganado por suas próprias ideologias e preconceitos. Além disso, o pesquisador deve ser adaptável e flexível, de forma que as situações recentemente encontradas possam ser

vistas como oportunidades, não ameaças. Assim, o pesquisador pode conhecer melhor o fenômeno que está sendo estudado e fazer escolhas mais consistentes com as questões e os rumos da pesquisa. Por fim, o pesquisador deve buscar um distanciamento crítico que permita a ele interagir com o contexto sem no entanto virar um advogado dos grupos envolvidos no caso.

2.4.1. Fazendo perguntas e ouvindo as respostas

Uma mente indagadora é um importante pré-requisito durante todo processo da coleta de dados, mesmo quando não se estiver em campo. A coleta de dados segue um plano formal, mas as informações específicas que podem se tornar relevantes a um estudo de caso não são previsíveis imediatamente. À medida que você realiza um trabalho de campo, você deve constantemente se perguntar por que os eventos ocorreram ou estão ocorrendo.

Se você é do tipo de pessoa para quem uma resposta tentadora já leva a uma quantidade enorme de novas questões, e se essas questões eventualmente se juntam a algum estudo significativo sobre como e por que o mundo funciona desta maneira, é provável que você seja um bom entrevistador.

Ser um bom ouvinte significa ser capaz de assimilar um número enorme de novas informações sem pontos de vista tendenciosos. À medida que um entrevistado relata um incidente, o bom ouvinte escuta as palavras exatas utilizadas, captura o humor e os componentes afetivos e compreende o contexto a partir do qual o entrevistado está percebendo o mundo. É importante observar o que está nas entrelinhas, o que está sendo dito sem ser dito explicitamente.

Fazendo um paralelo com a pesquisa interpretativa, quanto a fazer perguntas e ser um bom ouvinte na coleta de dados, o princípio da interação entre os pesquisadores e os sujeitos (Klein e Myers, 1999), exige de nós pesquisadores uma reflexão crítica sobre como os "dados" de pesquisa foram construídos socialmente através da interação entre os pesquisadores e os participantes. A compreensão da pesquisa melhora a medida em que vamos nos tornando mais conscientes dos dados que estão sendo coletados na pesquisa e vamos nos apropriando disso como conhecimento do fenômeno de pesquisa.

2.4.2. Ser adaptável e flexível e conhecer o fenômeno em estudo

Em geral, raramente os estudos de caso terminarão exatamente como foram planejados. Inevitavelmente, você terá que fazer pequenas, quando não grandes, alterações, que variam da necessidade de tomar uma direção inesperada (uma alteração potencialmente pequena) à necessidade de identificar um novo "caso" para um estudo (alteração potencialmente grande).

O pesquisador habilidoso deve lembrar do propósito inicial da investigação, mas aí, se ocorrerem eventos imprevistos, ele provavelmente desejará alterar os procedimentos ou os planos. Isto faz parte, pois os problemas podem ir mudando e irem aparecendo novas questões na coleta de dados, que devem ser incorporadas ao estudo de caso. Ser adaptável e flexível faz parte para ir conduzindo o estudo de caso incluindo as alterações que forem aparecendo ao longo do caminho investigativo. Assim, a

necessidade de equilibrar a adaptatividade com rigor - mas não com rigidez - não pode receber uma ênfase demasiada.

Os pesquisadores devem entender as questões teóricas e políticas, pois é preciso fazer julgamentos e demonstrar conhecimento durante a fase de coleta de dados. Sem uma noção muito clara das questões em discussão, pode se deixar passar pistas importantes e não se saberia identificar uma mudança no curso do estudo quando ele fosse aceitável ou mesmo desejável. O ponto-chave é que na coleta de dados para um estudo de caso devemos ser capazes de interpretar as informações como estão sendo coletadas e saber imediatamente, por exemplo, se as diversas fontes de informação se contradizem ou se complementam.

Este ponto se coaduna mais uma vez com dois princípios da pesquisa interpretativa (Klein e Myers, 1999): o primeiro é o princípio da contextualização, onde se requer que o pesquisador possua uma reflexão crítica do contexto social e histórico do cenário de pesquisa, de modo que o público pretendido possa ver como a investigação emergiu.

Outro princípio é o princípio fundamental do círculo hermenêutico. Este princípio sugere que todo entendimento humano é alcançado pela interação entre considerar o significado interdependente das partes e do todo que elas formam. Ou seja, entender como cada parte da coleta de dados interage com as outras partes da pesquisa, formando todo o fenômeno, trazendo a visão sistêmica do fenômeno para coleta de dados também é fundamental para que seja realizada uma boa pesquisa.

2.4.3. Buscar um distanciamento crítico

Todas as condições precedentes serão invalidadas se o pesquisador procurar utilizar o EC apenas para comprovar uma posição preconcebida. Os pesquisadores de EC geralmente estão propensos a esse problema porque eles devem compreender as questões e agir com discrição. O pesquisador deve estar aberto aos resultados da pesquisa, mesmo se não for o resultado que este estava imaginando de início com o EC.

Podemos trazer mais uma vez outro princípio da pesquisa interpretativa: o princípio do raciocínio dialógico. (Klein e Myers, 1999), nos mostram que o pesquisador deve ter sensibilidade a possíveis contradições entre os preconceitos teóricos que orientam o desenho da pesquisa e as descobertas reais ("a história que os dados contam") com os subsequentes ciclos de revisão a cada coleta de dados.

2.4.4. Protocolo de Estudo de Caso

Após realizarmos a coleta de dados, trataremos do protocolo de EC. Um protocolo de EC pode ajudar a aumentar a confiabilidade da pesquisa e ajuda a orientar o pesquisador. Este documento deve conter algumas seções, tais como:

- **Visão geral do projeto do EC**. Uma boa visão geral mostrará ao leitor que esteja familiarizado com o tópico geral da investigação o objetivo do EC e o cenário no qual ele ocorrerá.
- **Procedimentos de campo** (acesso aos locais do estudo de caso e fontes gerais de informações) coletados de pessoas e instituições existentes. Assim, em um

EC, o pesquisador deve aprender a integrar acontecimentos do mundo real às necessidades do plano traçado para a coleta de dados. A natureza da entrevista é muito mais aberta, e o entrevistado pode não cooperar integralmente ao responder às questões. Devemos assim, nos ater aos seguintes pontos na coleta de dados: obter acesso a organizações ou a entrevistados-chave; possuir materiais suficientes enquanto estiver no campo - tais como computador e bloco de notas, um local calmo e preestabelecido para tomar notas em particular e preparar-se para acontecimentos inesperados.

- **Questões do EC**. As questões são para nós, pesquisadores, nos atermos ao cerne da pesquisa. São lembretes que você deverá utilizar para lembrar das informações que precisam ser coletadas e por qual motivo. Devemos nos lembrar sempre de permitir que o entrevistado se expresse de forma livre, tendo perspicácia para deixar a pessoa contar a sua história, abarcando os elementos inesperados fazendo fluir a serendipidade na investigação. Muitas vezes, os entrevistados ficam tocados ao relembrar eventos que participaram na construção dos SI, relação com atores e devemos ter consideração, empatia e paciência neste momento a fim de extrairmos a informação da melhor forma possível.
- **Guia para o relatório do EC.** Resumo, formato de narrativa e especificação de quaisquer informações bibliográficas e outras documentações.

Este protocolo nos ajudará a guiar o nosso EC e será um guia para o relatório final.

2.4.5. Tipos de evidências de coleta de dados

É importante que o pesquisador utilize várias fontes de evidências ao realizar o seu estudo, conforme Yin (2001). O uso de várias fontes de evidências nos EC permite que o pesquisador dedique-se a uma ampla diversidade de questões históricas, comportamentais e de atitudes. A Tabela 1.1.5. nos mostra de forma resumida as evidências que poderemos utilizar a fim de construirmos o nosso EC.

Tabela 2.5. Fontes de evidências de coleta de dados (Yin, 2001).

FONTE DE EVIDÊNCIAS	PONTOS FORTES	PONTOS FRACOS
Documentação	•estável – pode ser revisada inúmeras vezes •discreta – não foi criada como resultado do estudo de caso •exata – contém nomes, referências e detalhes exatos de um evento •ampla cobertura – longo espaço de tempo, muitos eventos e muitos ambientes distintos	•capacidade de recuperação – pode ser baixa •seletividade tendenciosa, se a coleta não estiver completa •relato de visões tendenciosas – reflete as idéias preconcebidas (desconhecidas) do autor •acesso – pode ser deliberadamente negado
Registros em arquivos	•*[Os mesmos mencionados para documentação]* •precisos e quantitativos	•*[Os mesmos mencionados para documentação]* •acessibilidade aos locais graças a razões particulares
Entrevistas	•direcionadas – enfocam diretamente o tópico do estudo de caso •perceptivas – fornecem inferências causais percebidas	•visão tendenciosa devido a questões mal-elaboradas •respostas tendenciosas •ocorrem imprecisões devido à memória fraca do entrevistado •reflexibilidade – o entrevistado dá ao entrevistador o que ele quer ouvir
Observações diretas	•realidade – tratam de acontecimentos em tempo real •contextuais – tratam do contexto do evento	•consomem muito tempo •seletividade – salvo ampla cobertura •reflexibilidade – o acontecimento pode ocorrer de forma diferenciada porque está sendo observado •custo – horas necessárias pelos observadores humanos
Observação participante	•*[Os mesmos mencionados para observação direta]* •perceptiva em relação a comportamentos e razões interpessoais	•*[Os mesmos mencionados para observação direta]* •visão tendenciosa devido à manipulação dos eventos por parte do pesquisador
Artefatos físicos	•capacidade de percepção em relação a aspectos culturais •capacidade de percepção em relação a operações técnicas	•seletividade •disponibilidade

Conforme visto acima, as evidências para um EC podem vir de seis fontes distintas: documentos, registros em arquivo, entrevistas, observação direta, observação participante e artefatos físicos (Yin, 2001). Trazendo a realidade de EC em SI,

entendemos que as fontes relevantes são: documentos, registros em arquivos e entrevistas.

Quanto aos **documentos** relativos ao EC, podem ser buscados dentro da organização que aquele SI está inserido, bem como por estudos já realizados de outros pesquisadores, trazendo estatísticas e o contexto histórico à época que aquele SI foi criado. Devemos ter em mente como pesquisadores e profissionais o princípio da contextualização, trazidos por (Klein e Myers, 1999), ao analisar estas informações. Estas informações podem nos ajudar a elaborar uma contextualização histórica, econômica e social do SI a ser estudado, nos posicionando no tempo e alinhando as informações que vão vir a ser coletadas através de outras fontes.

Sobre os **registros em arquivos**, tais como dados de um SI a arquitetura utilizada, número de usuários do sistema, dados estatísticos oriundos de levantamentos, dentre outros, podem ser útil se combinados a outras evidências ao se montar o estudo de caso.

Apesar de termos diversas possibilidades e fontes de informação, sem dúvida uma das mais importantes fontes de informação para um EC são as **entrevistas**. Como a história aconteceu de fato, muitas vezes não se tem registros em documentos, ou levantamentos estatísticos. A história está na mente das pessoas que vivenciaram aquele fenômeno que estamos estudando! Somente realizando entrevistas que poderemos ter acesso a essas informações preciosas para nossa pesquisa.

Segundo Yin (2001), as entrevistas para o EC é comum que sejam conduzidas de forma espontânea. As entrevistas espontâneas permitem que você tanto indague os entrevistados e peça a opinião deles sobre determinados eventos. Em algumas situações, você pode até mesmo pedir que o respondente apresente suas próprias interpretações de certos acontecimentos e pode usar essas proposições como base para uma nova pesquisa. Informantes-chave são sempre fundamentais para o sucesso de um estudo de caso. Estas pessoas não apenas fornecem ao pesquisador do estudo percepções e interpretações sobre um assunto, como também podem sugerir outras pessoas-chave para se fazer entrevistas sobre o mesmo estudo de caso, que é a chamada técnica *snowball*.

Quanto ao registro da entrevista esse é um ponto importante a ser citado. Muitos entrevistados não ficam confortáveis com o uso de gravadores, sendo assim, devemos buscar fazer notas, verificando termos chaves e importantes que foram utilizados pelo entrevistado, que podem traduzir alguma teoria que podemos identificar, corroborar e usar na nossa pesquisa.

Uma boa sugestão é tão logo termine a entrevista, as notas sejam passadas a limpo e o registro seja validado, por escrito, com o entrevistado. Muitas vezes ainda há possibilidade de correção de detalhes que talvez o pesquisador possa não ter captado naquele momento da entrevista. Além de ficar um registro assertivo ainda temos a anuência do entrevistado do que foi dito e o registro pode ser inserido no banco de dados de documentos que citamos acima.

Muitas vezes, a ‘história a ser contada’, os meandros de um SI, não estão registrados em documentos. Por isso, é muitíssimo importante extrair a informação através de entrevistas, buscando várias pessoas como referência e fazendo com que as

informações sejam corroboradas por outras pessoas da mesma organização ou de outras organizações que possuem relação com aquele SI que está sendo estudado.

O que se deve ter em mente, após termos as evidências conosco dos dados que foram coletados, é desenvolver linhas convergentes de investigação. Assim, qualquer descoberta ou conclusão em um EC provavelmente será muito mais convincente e acurada se se basear em várias fontes distintas de informação, obedecendo a um estilo corroborativo de pesquisa. Podemos utilizar no EC em SI a triangulação de fontes de dados para corroborar a nossa pesquisa. Com a triangulação, podemos utilizar as várias fontes de evidências coletadas que fornecem essencialmente várias avaliações do mesmo fenômeno, conforme Figura 1.1.1 a seguir.

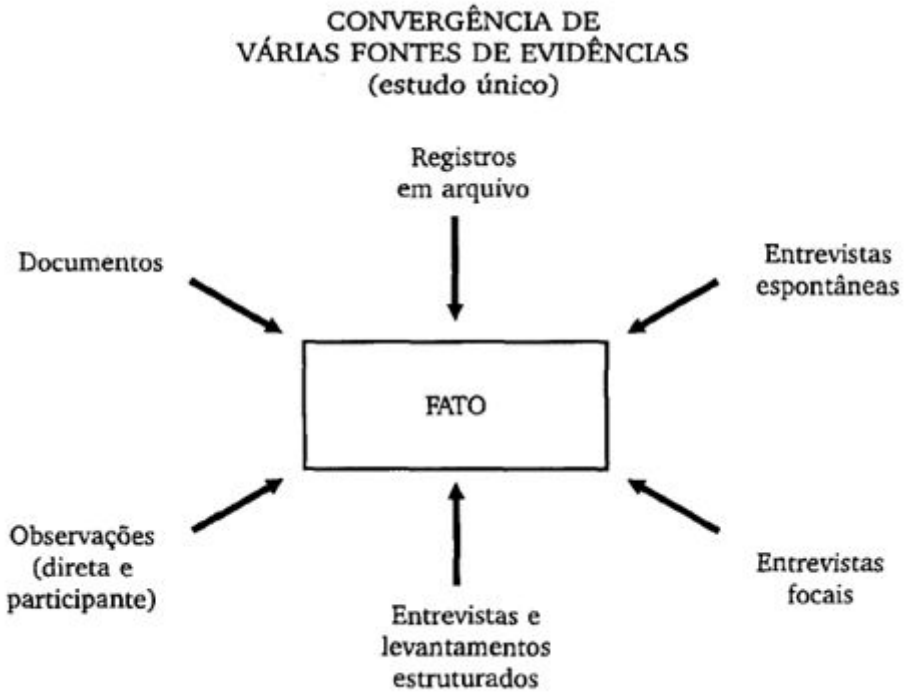


Figura 2.1 Convergência das fontes de evidências em estudo único (Yin, 2001).

2.5. Analisando e Reportando o Estudos de Caso

Yin (2001) nos traz algumas opções de como se analisar evidências de um EC. Trazendo para o EC de um SI, uma das estratégias que podemos utilizar é a **proposição teórica** na qual irá se basear a análise dos dados do EC. Proposições teóricas sobre relações causais - respostas a questões do tipo "**como**" e "**por que"** - podem ser muito úteis para orientar a análise do estudo de caso dessa maneira.

Um outro método de análise seria a análise de **acontecimentos cronológicos**. A sequência cronológica permite que o pesquisador pesquise os eventos ao longo do tempo. A disposição dos eventos em uma linha cronológica permite que o pesquisador

determine os eventos causais ao longo do tempo, uma vez que a sequência básica de uma causa e seu efeito não pode ser temporalmente invertida.

Devemos nos certificar como pesquisadores de que a nossa análise é de alta qualidade. Sua análise deve deixar claro que ela se baseou em todas as evidências relevantes. Além disso, devemos nos dedicar aos aspectos mais significativos do seu EC. Deve-se buscar conhecer profundamente o objeto do seu Estudo de Caso.

Lembrando que estas técnicas não são triviais de serem utilizadas e devem ir sendo adaptadas a cada estudo de caso. A seguir, veremos como reportar os EC realizados.

A fase de reportar os casos estudados é uma das mais complicadas de se conduzir ao realizar EC. É importante que seja realizado em partes o quanto antes o Estudo de Caso (exemplo: a bibliografia, seção metodológica), em vez de esperar até o final do processo de análise dos dados para começar a escrever.

Examinaremos nesta seção alguns itens importantes, que referem-se a composição do EC e que irão refletir no relatório, tais como: o público a que os EC se destinam; as variedades de composição do EC; as estruturas ilustrativas para as composições do EC; os procedimentos a serem adotados ao realizar um relatório de EC e as especulações sobre as características de um EC.

2.5.1. O público para quem o estudo de caso se destina

Quanto ao público, este pode ser os colegas da mesma área de atuação profissional e no nosso caso também comunicar e divulgar a metodologia de estudo de caso na comunidade de pesquisa de SI.

Cada público possui necessidades distintas, e nenhum relatório em especial atenderá às demandas de todos os públicos simultaneamente. Para os colegas de profissão, o mais importante é, provavelmente, a relação entre o EC, suas descobertas e as teorias já existentes. Para a comunidade de pesquisa em SI, trata-se de divulgar o fenômeno e mostrar que é possível, sim, realizar pesquisa de estudo de caso em SI e trazer contribuições científicas a área.

2.5.2. Estruturas ilustrativas para composição do Estudo de Caso

Quanto às estruturas ilustrativas, Yin (2001) nos sugere as seguintes estruturas: estruturas analíticas lineares; estruturas cronológicas; estruturas de construção da teoria; eza" e estruturas não-sequenciais.

As estruturas analíticas lineares é a abordagem-padrão ao elaborar um relatório de pesquisas. A sequência de subtópicos inclui o tema ou o problema que está sendo estudado, uma revisão da literatura importante já existente, os métodos utilizados, as descobertas feitas a partir dos dados coletados e analisados e as conclusões e implicações feitas a partir das descobertas.

Uma vez que os EC tratam, em geral, de eventos ao longo do tempo, uma segunda abordagem é apresentar as evidências para o EC em ordem cronológica. Aqui, a sequência dos capítulos ou das seções deve obedecer às fases iniciais, intermediárias e finais da história de um caso. Essa tática pode servir a um objetivo muito importante ao

realizar EC, já que podem ocorrer sequências causais linearmente ao longo do tempo de pesquisa.

Uma terceira abordagem é a de construção de teoria, onde a lógica dependerá do tópico ou da teoria específica, mas cada capítulo ou seção deve desenredar uma nova parte do argumento teórico que está sendo feito.

Por fim, a quarta abordagem trata das estruturas não-sequenciais são aquela em que a ordem de seções ou capítulos não possui uma importância em especial. Estudos de caso descritivos sobre organizações frequentemente apresentam essa mesma característica. A ordem em particular que esses capítulos ou seções são apresentados não é importante e pode ser classificada como uma abordagem não-sequencial.

2.5.3. Procedimentos ao fazer um relatório de Estudos de Caso

Yin (2001) nos mostra três características importantes dos EC para se elaborar um bom relatório. O primeiro procedimento a ser adotado é começar a redigir o relatório logo no início do processo analítico. Praticamente desde o início da pesquisa, é possível se fazer a minuta de certas seções do relatório, e ela deve prosseguir mesmo antes de a coleta e de a análise dos dados terem sido concluídas. Por exemplo, depois que a literatura existente já tiver sido revisada e que o estudo de caso estiver projetado, já é possível se fazer o rascunho de duas seções do relatório do estudo de caso: a bibliografia e as seções em que é apresentada a metodologia. Também é possível se rascunhar a seção metodológica nesse estágio porque os procedimentos principais para a coleta e a análise de dados devem ter feito parte do projeto do estudo de caso. Seja como parte do texto, seja como apêndice, no entanto, pode-se e deve-se fazer o rascunho da seção metodológica neste estágio inicial. Você se lembrará dos procedimentos metodológicos que utilizou com maior precisão durante esse momento crítico.

O segundo procedimento refere-se a deixar no anonimato os entrevistados, preservando a sua identidade e mantendo a ética da pesquisa.

O terceiro procedimento se refere à revisão do protocolo do estudo de caso. Isto pode revelar a qualidade total do estudo. O procedimento que se deve adotar é fazer com que a minuta do relatório seja revisada, não apenas pelos colegas do pesquisador mas também pelos participantes e informantes do caso. Se os comentários forem excepcionalmente úteis, o pesquisador pode até desejar publicá-los como parte de todo o estudo de caso. Uma ótima maneira de aumentar a qualidade dos estudos de caso e garantir a validade do constructo é fazer com que as minutas do caso sejam revisadas pelas pessoas que foram objeto do estudo.

Desta forma faremos uma comunicação mais assertiva do nosso Estudo de Caso a comunidade a no âmbito profissional ao qual estamos inseridos, se for o caso.

2.6. Um estudo de caso “exemplar” - principais elementos

O Estudo de Caso exemplar vai além dos procedimentos metodológicos já mencionados. Mesmo se você, como pesquisador de Estudo de Caso, seguir a maioria das técnicas básicas - utilizando um protocolo de Estudo de Caso, mantendo um encadeamento de evidências, estabelecendo um banco de dados para o Estudo de Caso,

e assim por diante - ainda assim você pode não ter produzido um Estudo de Caso exemplar. Devemos reproduzir no nosso estudo de caso as percepções sobre os processos humanos e sociais que estão inerentes no estudo de um SI.

Vamos agora descrever cinco características gerais de um EC exemplar e relacioná-los aos critérios de validade verificados na seção 2.3.2.

2.6.1. O Estudo de Caso deve ser significativo

Antes de selecionar um EC, você deve descrever, em detalhes, a contribuição que se fará com o estudo pretendido e avaliar se este terá significância ou não. O EC exemplar será aquele em que: o caso ou os casos individuais não forem usuais e de interesse público e as questões subjacentes forem de importância nacional, tanto em termos teóricos quanto em termos políticos ou práticos. Está caracteristica se relaciona com o critério de criticidade, onde devemos motivar os leitores a examinar a fundamentação do nosso estudo.

2.6.2. O estudo de caso deve ser "completo"

Para os estudos de caso, a completude pode ser caracterizada de pelo menos três maneiras. Primeiro, o caso completo é aquele em que os limites do caso - isto é, a distinção entre o fenômeno que está sendo estudado e seu contexto recebem uma atenção explícita. A melhor maneira de se fazer tal coisa é demonstrar, ou através de argumentos lógicos ou da apresentação de evidências, que, à medida que se alcança a periferia analítica, as informações serão de relevância cada vez menor para o estudo de caso. Essa verificação dos limites pode ocorrer durante as etapas analítica e de exposição dos estudos de caso.

Uma segunda forma envolve a coleta de evidências. O estudo de caso completo deve demonstrar, de maneira convincente, que o pesquisador despendeu esforços exaustivos ao coletar as evidências relevantes. O objetivo geral, no entanto, é convencer o leitor de que pouquíssimas evidências relevantes permaneceram intocadas pelo pesquisador, dados os limites do estudo de caso. Isso não significa que o pesquisador deve coletar, literalmente, todas as evidências disponíveis - uma tarefa impossível -, mas que as partes importantes receberam total atenção.

Uma terceira maneira diz respeito à ausência de certos artefatos. O responsável deve projetar um estudo de caso que pode ser concluído dentro desses limites, em vez de atingi-los ou possivelmente estendê-los. Não obstante, são estas as condições sob as quais provavelmente será realizado um estudo de caso exemplar.

Está caracteristica de completude se relaciona com o princípio do círculo hermenêutico, onde as partes do EC se relacionam com o todo e o todo se relaciona com as partes estudadas.

1.6.3. O estudo de caso deve considerar perspectivas alternativas

Um pré-requisito fundamental a todos que ensinam a prática dos EC, por exemplo, é que sejam capazes de apresentar o ponto de vista de todos os participantes principais do caso (Stein, 1952).

Muitas vezes, se um pesquisador descreve um EC a um ouvinte muito crítico, o ouvinte imediatamente dará uma interpretação alternativa dos fatos do caso. Sob tais circunstâncias, o pesquisador provavelmente ficará na defensiva e irá argumentar que a interpretação original era a única importante ou era a interpretação correta. Na verdade, o EC exemplar antecipa essas alternativas óbvias, até defende seus posicionamentos da maneira mais veemente possível e mostra - empiricamente - a base segundo a qual tais alternativas podem vir a ser rejeitadas.

Esta característica se relaciona com o critério de validade de pesquisa múltiplas interpretações, onde todos os entrevistados com seus diferentes pontos de vista devem ser contemplados no EC.

2.6.4. O estudo de caso deve apresentar evidências suficientes

O EC exemplar é aquele que, apresenta as evidências mais convincentes, para que o leitor possa fazer um julgamento independente em relação ao mérito da análise. As evidências devem ser apresentadas de forma neutra, tanto com dados de sustentação quanto com dados de contestação. O leitor, dessa forma, deve ser capaz de concluir, de forma independente, se uma determinada interpretação é válida.

Um outro objetivo é apresentar evidências suficientes para obter a confiança do leitor de que o pesquisador conhece o assunto com o qual está lidando. Ao realizar um estudo de campo, por exemplo, as evidências apresentadas devem convencer o leitor de que o pesquisador realmente esteve no campo, trabalhou com afinco enquanto esteve lá e mergulhou por inteiro nas questões do caso.

Existe um objetivo paralelo nos estudos de casos múltiplos; o pesquisador deve mostrar ao leitor que todos os casos únicos foram tratados de forma justa e que todas as conclusões cruzadas não foram influenciadas por terem recebido atenção indevida de uma ou de algumas das séries de casos.

Finalmente, a exposição de evidências adequadas deve vir acompanhada por alguma indicação de que o pesquisador esteve atento à validade das evidências - mantendo o seu encadeamento, por exemplo.

Esta característica se relaciona com o critério de validade de pesquisa validade interna, confiabilidade e autenticidade.

2.6.5. O estudo de caso deve ser elaborado de uma maneira atraente

Uma última característica global do EC tem a ver com a elaboração do relatório do estudo. O relatório deve ser atraente. Isto significa que o pesquisador deve escrevê-los em um estilo claro, e que incite o leitor a continuar lendo. Um bom manuscrito é aquele que "seduz" os olhos do leitor.

Engajamento, instigação e sedução - essas são características incomuns dos EC. Produzir um EC como esse exige que o pesquisador seja entusiástico em relação à investigação e deseje transmitir amplamente os resultados obtidos. Um entusiasmo como esse deve permear a investigação inteira e conduzir, de fato, a um estudo de caso exemplar. Esta característica se relaciona com o critério de validade de pesquisa plausibilidade e contextualização.

2.7. Conclusão

Neste capítulo, apresentamos tópicos gerais para o planejamento e condução de EC para atividades de pesquisa ou mesmo para a prática profissional, pois a análise de SI em contextos organizacionais e sociais é fundamental para o conhecimento das organizações e para o entendimento dos impactos dos SI na sociedade contemporânea.

Concluímos que, apesar de desafiadora, a Metodologia de Estudo de Caso a ser utilizada em SI é muito rica, e pode revelar diversos elementos que emergem no contexto do SI que não seriam revelados através de outra metodologia de pesquisa. Desta forma, conseguiremos estudar o fenômeno dos SI de uma forma mais multidisciplinar, utilizando outros paradigmas e abarcando todos os atores envolvidos na construção de um SI.

Convidamos a comunidade de pesquisa e prática de SI a praticar esta metodologia e compartilhar suas descobertas, ampliando o conhecimento da área e fortalecendo nossa visão sistêmica e multidisciplinar em SI.

Referências

Araujo, R. M., Chueri, L. O. V. Pesquisa e Inovação: Visões e Interseções. 1. ed. Rio de Janeiro: Publit Soluções Editoriais. v. 1. 296p. (2017)

Benbasat, I. Goldstein, D. K. e Mead, M. The Case Research Strategy in Studies of Information Systems. MIS Quarterly, Vol. 11, No. 3, pp. 369-386 (1987). http://www.jstor.org/stable/248684. Acesso em 12/08/2018.

Boscarioli, C., Araujo, R. M. e Maciel, R. S. I GranDSI-BR: Grand Research Challenges in Information Systems in Brazil 2016-2026. 1. ed. Porto Alegre: Sociedade Brasileira de Computação, v.1. 184p. (2017) http://www2.sbc.org.br/ce-si/arquivos/GranDSI-BR_Ebook-Final.pdf . Acesso em 20/09/2018.

Dube, L. Pare, G. MIS Quarterly, Vol. 27, No.4, p.597 (2003). Rigor in Information Systems Positivist Case Research: Current Practices, Trends, and Recommendations.

Diniz, E. H. Petrini, M. Barbosa, A. F. Christopoulos, T. F. Santos, H. M. Abordagens Epistemológicas em Pesquisas Qualitativas: Além do Positivismo nas Pesquisas na Área de Sistemas de Informação. EnANPAD (2006).

Eisenhardt, K.M. Building Theories from Case Study. The Academy of Management Review, Vol. 14, No. 4, pp. 532-550 (1989). http://www.jstor.org/stable/258557. Acesso em 12/08/2018.

Eisenhardt, K.M. e Graebner, M. Theory Building From Cases: Opportunities and challenges. Academy of Management Journal, Vol. 50, No. 1, pp. 25-32 (2007).

Hirschheim, R.; Klein, H. K. Tracing the History of the Information Systems Field. In: Currie, W.; Galliers, R. D. (Ed.). *The Oxford Handbook of Management Information Systems*. Oxford: Oxford University Press, pp. 16-62.(2011).

Klein, H. K. e Myers, M. D. A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems. MIS Quarterly, Vol. 23, No. 1, pp. 67-93 (1999). http://www.jstor.org/stable/249410 . Acesso em 12/08/2018.

Laudon, K. C. e Laudon, J. P. *Sistemas De Informação Gerenciais*. São Paulo: Pearson Education (2014).

Lee, A. S. A Scientific Methodology for MIS Case Studies. MIS Quarterly, Vol. 13, No. 1, pp. 33-50 (1989). http://www.jstor.org/stable/248698 . Acesso em 12/08/2018.

Orlikowski W. J. e Iacono, S. J. Research Commentary: Desperately Seeking the "IT" in IT Research—A Call to Theorizing the IT Artifact. Information Systems Research, Vol 12, No. 2, pp.121-134 (2001).

Orlikowski W. e Baroudi J. Studying Information Technology in Organizations. Information Systems Research. (1991).

Pozzebon, M. Conducting and Evaluating Critical Interpretative Research: Examining Criteria as a Key Component in Building a Research Tradition. (2004).

Recker, J. Scientific Research in Information Systems – A Beginner's Guide. Springer. (2013).

Walsham, G. Interpretive case studies in IS research: nature and method. European Journal of Information Systems, 4,pp.74-81 (1995).

Walsham, G. Doing interpretive research. European Journal of Information Systems, 15,pp.320-330 (2006).

Yin, Robert K. Estudo de caso: Planejamento e Métodos. trad. Daniel Grassi - 2.ed. - Porto Alegre : Bookman, (2001).

Autores

Nadja Piedade de Antonio - - http://lattes.cnpq.br/3769571781611695

Mestranda do Programa de Pós-Graduação em Informática da UNIRIO fazendo pesquisa em Sistema de Informação em Gestão Social. Possui Bacharelado em Ciências Econômicas pela Universidade Federal Fluminense e pós-graduação *lato sensu* em Engenharia de Produção, com ênfase em Serviços pelo Instituto Nacional de Tecnologia. Atua desde 2007 na área de desenvolvimento de sistemas da Caixa Econômica Federal e desde 2012 atuando como Coordenadora de Projetos de TI, possuindo experiência em gestão no desenvolvimento de sistemas sociais.

Marcelo Fornazin - http://lattes.cnpq.br/0396928965160154

Professor Adjunto no Instituto de Computação da Universidade Federal Fluminense (UFF). Doutor em Administração pela EBAPE/FGV, possui Bacharelado e Mestrado em Ciência da Computação pela UNESP. Integrante do Grupo Temático Informação, Saúde e População da Associação Brasileira de Saúde Coletiva (GTISP/Abrasco), atua também no Programa de Pós-Graduação em Informática da Unirio (PPGI-Unirio) e no Programa de Pós-Graduação em Ciência da Informação IBICT-UFRJ. Tem experiência na área de Ciência da Computação e Tecnologia da Informação, com ênfase em Gestão de Tecnologia da Informação, Governo Eletrônico e Computação Social.

Renata Araujo - http://lattes.cnpq.br/3589012014320121

Doutora em Engenharia de Sistemas e Computação, possui mais de 20 anos de experiência em ensino, pesquisa e extensão na área de Sistemas de Informação, em diversos temas: Democracia e Governança Digital, Gestão de Processos de Negócio, Sistemas Colaborativos, Gerência de Projetos e Gestão da Inovação. Atualmente ocupa a Diretoria de Educação da SBC (2018-2019). Bolsista de Produtividade em Desenvolvimento Tecnológico e Extensão Inovadora do CNPq, Brasil processo no 305060/2016-3.

Capítulo 3

Introdução à Classificação Multirrótulo

Eduardo Corrêa Gonçalves

Abstract

Multi-label classification (MLC) is the task of assigning multiple class labels to an object based on the features that describe the object. There are many important and modern applications of MLC, such as text categorization (associating documents to various subjects) and semantic scene classification (categorizing images into concepts). This chapter is intended to those who wish to work with MLC in information systems. We present the basic approaches for the construction and evaluation of multi-label classifiers as well as examples of software libraries that support the development of MLC applications in Java and R.

Resumo

Classificação multirrótulo (CMR) pode ser definida como a tarefa de associar múltiplos rótulos de classe para um objeto com base nas características que o descrevem. Existem muitas aplicações modernas e importantes para a tarefa, tais como a categorização de textos (associar documentos texto a tópicos) e a classificação semântica de cenas (associar imagens a conceitos). Este capítulo é direcionado a pessoas que querem trabalhar com classificação multirrótulo em sistemas de informação. Ele apresenta as abordagens básicas para a construção e avaliação de classificadores multirrótulo e também exemplos de bibliotecas que oferecem suporte para o desenvolvimento de aplicações de CMR nas linguagens Java e R.

3.1. Introdução

Aprendizado de máquina (*machine learning*) é a linha de pesquisa que investiga como computadores podem aprender a executar tarefas baseando-se na utilização de dados [Han et al., 2011]. Dentre os diferentes tipos de tarefas que podem ser realizadas, a classificação é provavelmente a mais conhecida e utilizada. Seu objetivo é bastante simples: treinar um programa de computador para que o mesmo seja capaz de atribuir

automaticamente classes (ou "rótulos de classe") para objetos cujas classes sejam desconhecidas.

Para que o conceito fique claro, será apresentado um exemplo. Considere um programa que receba como entrada a fotografia do rosto de uma pessoa e que seja capaz de determinar automaticamente se o rosto pertence a um adulto ou a uma criança. Veja que o objetivo do programa é associar uma classe ("Adulto" ou "Criança") a um objeto (a fotografia de um rosto). Desta forma, trata-se de um programa que realiza a tarefa de classificação. Nos dias atuais, técnicas de classificação vêm sendo exploradas tanto pela comunidade acadêmica como pelas empresas para resolver diversos problemas importantes. Alguns deles são tradicionais e bastante conhecidos, como a detecção de *spam* (identificar um e-mail como "Spam" ou "Normal"), a detecção de fraudes (identificar se uma transação de cartão de crédito é "Fraudulenta" ou "Genuína") e a análise de crédito (classificar clientes que solicitaram um empréstimo bancário de acordo com os riscos de crédito "Baixo", "Médio" ou "Alto"). Já outras aplicações são mais recentes e menos conhecidas, entre elas a genômica funcional (determinar as funções biológicas de genes e proteínas) e a categorização automática de músicas (associar canções a estilos musicais).

Na maioria dos problemas de classificação, cada objeto deve ser associado a um e somente um rótulo pertencente a um conjunto predeterminado de rótulos de classe. Estes problemas são conhecidos como problemas de classificação tradicional ou monorrótulo (*single-label classification*) [Flach, 2012; Han et al., 2011; Marsland, 2015; Witten et al., 2016]. Por exemplo, no problema de classificação de *spam*, um e-mail recebido deve ser classificado como "Spam" ou "Normal", mas nunca com ambos os rótulos. Analogamente, no problema da análise de crédito, um candidato a empréstimo pode ser classificado em apenas um dos rótulos de classe possíveis ("Baixo", "Médio" ou "Alto"), jamais podendo ser classificado em dois ou três deles ao mesmo tempo.

Entretanto, nem todos os problemas reais de classificação são do tipo monorrótulo. Um dos exemplos é a categorização automática de músicas, onde o objetivo é associar canções a gêneros musicais. Neste problema, tem-se, por exemplo, que muitas músicas compostas pelo grupo brasileiro Novos Baianos podem ser classificadas como pertencentes aos gêneros "Rock", "MPB" e "Samba" ao mesmo tempo. De maneira similar, um grande número de composições do músico Tom Jobim consiste em uma mistura de dois gêneros: "Jazz" e "Bossa Nova". Por esta razão, o problema da categorização de músicas representa um problema de classificação multirrótulo (CMR – *multi-label classification*) [Gibaja and Ventura, 2015; Gonçalves et al., 2018; Tsoumakas et al., 2010; Zhang and Zhou, 2014], em que os objetos podem ser associados a múltiplos rótulos de classe. Ao longo dos últimos anos, diversas outras aplicações modernas e importantes de CMR surgiram. A listagem a seguir inclui apenas algumas delas, como forma de motivação para o leitor.

- Classificação de artigos de jornais ou portais de notícias em um número fixo categorias, como "Esporte", "Cultura", "Política", etc. [Tsoumakas et al., 2014];
- Genômica funcional: determinar as múltiplas funções biológicas de genes e proteínas [Diplaris et al., 2005; Elisseeff, and Weston, 2001; Huang et al., 2015];

- Classificação semântica de cenas: consiste em identificar os diferentes componentes de uma cena da natureza, como “Montanhas”, “Mar”, “Árvores”, etc. [Boutell et al., 2004];
- Classificação de músicas [Trohidis et al., 2011] ou textos curtos [Almeida et al., 2018] em um conjunto de emoções (ex.: “Triste”, “Relaxante”, “Alegre”, etc.);
- Predição do efeito colateral de drogas, ou seja, prever o conjunto de reações adversas que novas drogas podem causar [Zhang et al., 2015];
- Classificação de textos de laudos médicos em classes de diagnóstico [Cherman and Monard, 2009; Pestian et al., 2007]
- Categorização dos tipos de falhas de execução de programas a partir da análise de *crash reports* [Xia et al., 2014];
- Atribuição automática dos gêneros de filmes (ex.: “Drama”, “Romance”, “Ação”, etc.) em função do texto contendo o seu resumo [Ururahy et al., 2018];
- Sugestão de *tags*: processo de atribuir pequenas informações textuais – as chamadas *tags* ou palavras-chave – a objetos multimídia como vídeos e imagens [Xiao et al., 2016].

Este capítulo, embora de maneira resumida, tem por objetivo apresentar um panorama geral da área de classificação multirrótulo, focando nos aspectos essenciais para aqueles que desejam incorporar esta tecnologia em seus próprios aplicativos. O texto está divido em duas partes. A primeira, denominada “Classificação – uma Visão Geral”, apresenta a teoria necessária para habilitar o leitor a entender como funciona um sistema de classificação. Nesta primeira parte, o foco é a classificação monorrótulo. A segunda e principal parte do curso, “Classificação Multirrótulo”, introduz os principais conceitos sobre CMR, cobrindo os seguintes tópicos: abordagens básicas para a construção de classificadores multirrótulo, propriedades das bases de dados multirrótulo e medidas de avaliação de desempenho para CMR. O capítulo também aborda a classificação na prática, através da apresentação de exemplos de *scripts* em Java e R que descrevem os passos básicos para a criação e utilização classificadores nestas linguagens.

3.2. Classificação – uma Visão Geral

3.2.1. Como Construir um Classificador?

Para que seja possível construir um classificador, “apenas” duas coisas são necessárias:

1. Uma boa base de dados de treinamento (ou base de dados rotulada). Esta base deve conter objetos pré-classificados, de acordo com um número finito de classes pré-definidas.
2. Um bom algoritmo de classificação. Alguns exemplos: naïve Bayes, k-NN, SVM, entre outros.

Nesta seção, o problema da classificação será introduzido de forma prática, a partir de exemplos que envolvem a classificação monorrótulo (é importante observar que o entendimento dos conceitos básicos de classificação monorrótulo representa um pré-requisito para quem deseja trabalhar com classificação multirrótulo). O texto está dividido da seguinte forma. Inicialmente, as bases de dados de treinamento e os algoritmos de classificação são respectivamente cobertos nas Subseções 3.2.2 e 3.2.3. Em seguida, a Subseção 3.2.4 aborda as técnicas básicas para avaliação da qualidade de classificadores. Por fim, a Subseção 3.2.5 dedica-se à classificação em sistemas de informação, apresentando exemplos de programas que utilizam bibliotecas de aprendizado de máquina nas linguagens Java e R.

3.2.2. Bases de Dados de Treinamento

A Figura 3.1 apresenta um exemplo de base de dados de treinamento. Considere que o objetivo seja utilizá-la para possibilitar a criação de um classificador monorrótulo capaz de determinar se a foto do rosto de uma pessoa é de um “Adulto” ou de uma “Criança”.

X	Y
x_1: 	y_1: Adulto
x_2: 	y_2: Criança
x_3: 	y_3: Criança
x_4: 	y_4: Adulto
...	...
x_N: 	y_N: Adulto

Figura 3.1. Exemplo de base de dados de treinamento para classificação monorrótulo. Ela pode ser utilizada para treinar um classificador capaz de determinar se a foto de um rosto é de um “Adulto” ou de uma “Criança”

Em uma base de treinamento existem duas categorias de atributos, *X* e *Y*:

- *X*: conjunto de atributos preditivos. São os atributos que descrevem as características (*features*) dos objetos, podendo ser tanto numéricos como discretos. Neste exemplo, a descrição de cada objeto da base de dados é dada pela foto de seu rosto. É importante deixar claro que em uma base de treinamento real para classificação de imagens, cada foto (dado bruto) precisa ser

transformada em um vetor numérico (dado estruturado). Este vetor armazena um conjunto de medições que caracteriza as propriedades da imagem [Chen et al., 2018].

- *Y*: atributo especial, denominado atributo classe. Trata-se do atributo alvo da classificação. No exemplo apresentado, cada $y_i \in$ {"Adulto", "Criança"} (conjunto de rótulos de classe pré-definido). O atributo classe é sempre do tipo discreto.

Observe que cada objeto[1] da base rotulada é representado por um par (*x*,*y*), onde *x* é um vetor contendo valores para os atributos do conjunto *X* e *y* é um dos valores possíveis (pré-determinados) de *Y*. A criação de um classificador é realizada através do processamento da base por um algoritmo de classificação, que se encarregará de "aprender" a mapear as características dos objetos (*X*) em rótulos de classes (*Y*).

A seguir, apresenta-se a definição formal para o problema de classificação monorrótulo:

Definição 1 *(Classificação Monorrótulo). Seja* $X = \{X_1, \ldots, X_d\}$ *um conjunto de* d *atributos preditivos e* $L = \{l_1, \ldots, l_q\}$ *um conjunto de* q *rótulos de classe, onde* $q \geq 2$. *Considere uma base de dados de treinamento D composta por N objetos do tipo* $\{(x_1,y_1), (x_2,y_2), \ldots, (x_N,y_N)\}$. *Nesta base de dados, cada* x_i *corresponde a um vetor* $\{x_1, \ldots, x_d\}$ *que armazena valores para os* d *atributos preditivos do conjunto* X *e cada* $y_i \in$ L *corresponde a um único rótulo de classe. O objetivo da tarefa de classificação monorrótulo é, a partir de* D*, realizar o aprendizado de uma função* f *(classificador ou modelo de classificação) que, dado um objeto não rotulado* t = (x,?)*, seja capaz de prever de forma efetiva o seu rótulo de classe* y.

Quando $q = 2$, o problema é chamado de problema de classificação binário. Este é o caso da detecção de *spam*, onde $L =$ {"Spam", "Normal"} e também da detecção de fraudes, em que $L =$ {"Fraudulenta", "Genuína"}. Por outro lado, se $q > 2$, o problema é chamado de problema de classificação multiclasse. Este é o caso da classificação do risco de crédito, onde $L =$ {"Baixo", "Médio", "Alto"}.

3.2.3. Algoritmos de Classificação

A construção de classificadores precisos e eficientes é considerada um dos grandes desafios na área de aprendizado de máquina. Por este motivo, foram desenvolvidas diversas técnicas para a execução desta tarefa, como naïve Bayes, k-NN, SVM, redes neurais e árvores de decisão, entre outros [Han et al., 2011; Witten et al., 2016].

Todas estas técnicas trabalham em duas etapas: treinamento (ou aprendizado) e classificação. A etapa de treinamento é aquela em que o modelo de classificação é

[1] Na literatura sobre classificação, as palavras "objeto", "instância", "tupla", "observação", "exemplo", "*data point*" e "registro" podem ser utilizadas como sinônimos.

construído a partir da base de dados rotulada. Já a classificação corresponde à etapa em que o modelo criado é utilizado para classificar novos objetos.

Esta seção apresenta um breve resumo sobre alguns dos principais algoritmos de classificação monorrótulo. Ênfase especial é dada ao algoritmo naïve Bayes (NB) que, a despeito de sua simplicidade, é apontado na literatura como um dos mais eficazes para problemas de classificação de texto [Witten et al., 2016] – a área de aplicação dominante e motivadora do surgimento da CMR [Tsoumakas et al., 2010].

Naïve Bayes (NB)

NB é um dos mais simples, populares e eficientes algoritmos de classificação. Trata-se de um classificador estatístico fundamentado em diversos conceitos da Teoria da Probabilidade, como probabilidade condicional, independência condicional, regra da multiplicação, distribuição conjunta de probabilidades e, especialmente, em uma importante fórmula conhecida como Fórmula de Bayes.

Esta seção explica, de forma resumida, o funcionamento do algoritmo a partir da descrição dos principais passos por ele empregados nas etapas de treinamento e classificação. Conforme introduzido na subseção anterior, a tarefa de classificação possui como objetivo associar objetos de classe desconhecida a um conjunto pré-definido de classes. Os algoritmos de classificação extraem os modelos classificadores a partir de bases de dados rotuladas, onde cada objeto é representado por um par (x,y). A base de dados de filmes apresentada na Tabela 3.1 é um exemplo de base rotulada. Considere que ela foi montada com o objetivo de viabilizar a construção de um classificador monorrótulo capaz de classificar o gênero de um filme como Romance (‘Sim’ ou ‘Não’) em função da ocorrência das palavras “love”, “people” e “relationship” em seu resumo.

Tabela 3.1. Base de dados de treinamento para criar classificador que determina se um filme é do gênero “Romance” em função da ocorrência das palavras “love”, “people” e “relationship” em seu resumo.

love	*people*	*relationship*	**Romance (classe)**
1	1	0	Sim
1	0	0	Não
1	0	1	Sim
0	0	1	Sim
1	1	1	Não
1	1	0	Sim
0	0	0	Não
0	1	0	Não
1	1	1	Sim
0	1	1	Não
1	0	1	Sim
1	1	0	Não
0	0	0	Não
1	1	1	Sim
1	0	1	Sim

A base de dados possui as seguintes características:

- Os atributos “love”, “people” e “relationship” descrevem as propriedades dos resumos (ou seja, eles formam o conjunto *X*). Estes atributos indicam se as respectivas palavras ocorrem ou não no resumo do filme.
- O atributo “Romance” é o atributo classe (*Y*). Ele registra o valor ‘Sim’ quando o filme é do gênero Romance e ‘Não’ caso contrário.

A base de dados possui informações sobre 15 filmes. Observe que o primeiro filme possui as palavras “love” e “people” em seu resumo, mas não possui “relationship”. Veja ainda que este filme está rotulado como Romance (Romance = ‘Sim’). Já o segundo filme possui a palavra “love”, não possui “people” e “relationship” e não é um Romance (Romance = ‘Não’). E assim por diante. A seguir, são descritas as etapas de treinamento (construção do modelo classificador) e classificação do algoritmo NB utilizando a base de filmes como exemplo.

A etapa de treinamento consiste basicamente em computar e armazenar em memória uma tabela de probabilidades condicionais que resume o conjunto de dados de treinamento. O modelo de classificação gerado pelo NB corresponde exatamente a esta tabela em memória. A Tabela 3.2 apresenta um exemplo que corresponde ao modelo que seria gerado para a base de dados de resumos de filmes da Tabela 3.1.

Tabela 3.2. Modelo de classificação gerado pelo naïve Bayes a partir da análise de base de dados de resumos de filmes.

Romance (classe)	***love***		***people***		***relationship***	
	0	**1**	**0**	**1**	**0**	**1**
Não 7/15 (46,67%)	4/7 (57,14%)	3/7 (42,86%)	3/7 (42,86%)	4/7 (57,14%)	5/7 (71,43%)	2/7 (28,57%)
Sim 8/15 (53,33%)	1/8 (12,50%)	7/8 (87,50%)	4/8 (50,00%)	4/8 (50,00%)	2/8 (25,00%)	6/8 (75,00%)

No modelo da Tabela 3.2, a primeira coluna apresenta as probabilidades “a priori” de um filme pertencer ao gênero Romance ou não (para Romance=‘Não’ o valor é 46,67% e para Romance=‘Sim’ o valor é 53,33%). Já as colunas 2 a 7 apresentam as probabilidades condicionais dos valores dos atributos preditivos “love”, “people” e “relationship” dados os dois rótulos possíveis da classe Romance. Considere, por exemplo, o valor 4/7 localizado na primeira linha e segunda coluna da tabela. Ele indica que existem 7 filmes que não são do gênero Romance (Romance=‘Não’) na base de treinamento e que 4 deles (57,14%) não possuem a palavra “love” em seu resumo (love=0). De maneira análoga, na célula localizada logo abaixo, o valor 1/8 indica que apenas 1 dos 8 filmes (12,50%) do gênero Romance (Romance=‘Sim’) não possuem a palavra “love” em seu resumo (love=0).

Uma vez criado o modelo de classificação, o NB está apto para realizar a classificação de novos objetos. Para tal, o algoritmo faz uso de uma adaptação Fórmula de Bayes, apresentada na Equação 1:

$$P(Y \mid X) = \frac{P(X \mid Y) \times P(Y)}{P(X)} \quad (1)$$

Suponha que o objetivo seja classificar um novo filme *t*, com as seguintes características: *t* = (*love=1, people=0, relationship=1, Romance = ?*). A questão é estimar se *t* é ou não um filme do gênero Romance utilizando a Equação 1. Para classificar *t*, o algoritmo NB faz o seguinte: ele observa as caraterísticas de *t* e, de acordo com estas, busca os valores de probabilidade correspondentes no modelo de classificação. Estes valores são então "plugados" na fórmula de Bayes para que seja possível realizar o cálculo da estimativa para os dois rótulos de classe possíveis (Romance='Não' e Romance='Sim'). A seguir, é mostrado como o cálculo dessas estimativas é realizado, considerando o novo objeto *t* e o modelo de classificação da Tabela 3.2:

- Estimativa 'Não':

P(Romance='Não' | *t*) = P(love=1| Romance='Não') ×
P(people=0| Romance='Não') ×
P(relationship=1| Romance='Não') ×
P(Romance='Não')

P (Romance='Não' | *t*) = 0,4286 x 0,4286 x 0,2857 x 0,4667 = **0,0245**

- Estimativa 'Sim':

P(Romance='Sim' | *t*) = P(love=1| Romance='Sim') ×
P(people=0| Romance='Sim') ×
P(relationship=1| Romance='Sim') ×
P(Romance='Sim')

P (Romance='Sim' | *t*) = 0,8750 x 0,5000 x 0,7500 x 0,5333 = **0,1750**

O resultado indica que há uma maior chance de o filme ser um Romance (pertencer à classe 'Sim'). Para que seja possível analisar o resultado de uma forma mais interessante, os valores calculados podem ser convertidos para probabilidades através da normalização da soma para 1. Isto é feito da seguinte forma:

- P('Não') = 0,0245 ÷ (0,0245 + 0,1750) = 12,28%
- P('Sim') = 0,1750 ÷ (0,0245 + 0,1750) = 87,72%

Desta forma, o rótulo de classe 'Sim' é atribuído ao novo objeto *t*. Observe que para realizar o cálculo não foi preciso utilizar o denominador da Fórmula de Bayes –

P(X) – uma vez que o seu valor seria o mesmo tanto para o cálculo da estimativa da classe 'Não' como da classe 'Sim'.

O algoritmo NB possui a palavra *naïve* (ingênuo) em seu nome porque se baseia em duas suposições:

1. Todos os atributos da base de dados são igualmente importantes.
2. Todos os atributos preditivos são condicionalmente independentes em relação ao atributo classe.

São exatamente estas suposições que fazem com que seja possível realizar a classificação com o uso de uma adaptação simples da Fórmula de Bayes.

Outros Algoritmos de Classificação Monorrótulo

Conforme mencionado anteriormente, além do NB, existem muitos outros algoritmos de classificação. O texto a seguir apresenta um breve resumo sobre quatro deles: *k-Nearest Neighbors* (k-NN), C4.5, *Sequential Minimal Optimization* (SMO) e *Multi-Layer Perceptron* (MLP). Explicações detalhadas sobre estes algoritmos e também sobre o NB podem ser obtidas em [Han et al., 2011; Witten et al., 2016].

- k-NN: este algoritmo executa a tarefa de classificação de um novo objeto *t* em dois passos: primeiro, o algoritmo encontra os *k* objetos da base de treinamento que sejam mais similares a *t*. Tipicamente, a similaridade é computada em termos de alguma medida de distância, como a distância Euclidiana ou a distância de Jaccard. Em seguida, *t* é classificado com a classe mais comum entre os *k* objetos.
- C4.5: constrói um modelo de classificação na forma de uma árvore de decisão. Em uma estrutura deste tipo, cada nó interno corresponde a um atributo preditivo (ex.: "love"), cada ramo representa um teste sobre um nó pai (ex.: o valor de love é 0?) e cada folha consiste em um rótulo de classe. Um novo objeto é classificado através da aplicação de sucessivos testes que encontrarão um caminho na árvore, desde o nó raiz até um nó folha. Para construir a árvore, o algoritmo C4.5 adota uma abordagem recursiva e gulosa que emprega a medida da entropia como critério de seleção de atributos. A cada passo, esta medida é aplicada para definir qual atributo melhor particiona os objetos da base de treinamento nas classes distintas.
- SMO: trata-se de um algoritmo para treinar classificadores SVM (*support vector machines*). Na técnica SVM, os dados de treino são separados em duas classes através da busca por uma fronteira de decisão que maximiza a distância entre os objetos de treino destas duas classes.
- MLP: algoritmo utilizado para treinar redes neurais. Uma rede neural MLP é uma estrutura formada por um conjunto de nós (denominados neurônios artificiais) e arestas estruturados em três camadas: de entrada, oculta (*hidden*) e de saída. Cada aresta possui um peso associado, cujo valor inicial é tipicamente determinado de forma aleatória, com base na distribuição normal. A técnica MLP emprega um processo de treinamento denominado *backpropagation* para

determinar o melhor conjunto de pesos para a rede. Primeiro, para cada objeto de treino, os valores de classe são determinados com base nos pesos correntes. Os erros de classificação são então computados e os pesos associados a cada aresta são atualizados. O processo é repetido até que um critério especificado para minimização do erro seja alcançado.

3.2.3. Como Avaliar a Qualidade de um Classificador?

Para criar um classificador, basta ter em mãos um bom algoritmo de classificação e uma boa base de dados de treinamento. Entretanto, para colocar o classificador em produção é preciso estimar a sua eficácia, isto é, estimar se ele terá um bom desempenho para classificar novos objetos. O princípio básico empregado na avaliação de classificadores consiste em utilizar um conjunto separado de dados de teste formado por objetos que não estiveram envolvidos no processo de treinamento do classificador. Esse conjunto também deverá conter objetos rotulados, ou seja, deve ser formado por pares (x,y).

O método *holdout* [Han et al., 2011] é o mais simples dentre os que empregam o princípio descrito acima. Na abordagem *holdout*, a base de dados rotulada é dividida de forma aleatória em dois conjuntos independentes: conjunto de treinamento e conjunto de teste. Tipicamente, dois terços dos dados são alocados para treino e o terço restante é alocado para teste (por exemplo, considerando a base de resumos de filmes apresentada na Tabela 3.1, tem-se que 10 objetos seriam alocados para o conjunto de treino e os 5 restantes para o conjunto de teste). Preferencialmente, a divisão deve ser feita de modo a garantir que cada classe seja adequadamente representada tanto no conjunto de treinamento como no de teste (processo conhecido como estratificação). O conjunto de treinamento é então usado para a construção do classificador, cuja acurácia é estimada com o conjunto de teste. A acurácia corresponde à porcentagem de objetos de teste corretamente classificados pelo modelo. A Figura 3.2 resume o processo *holdout*.

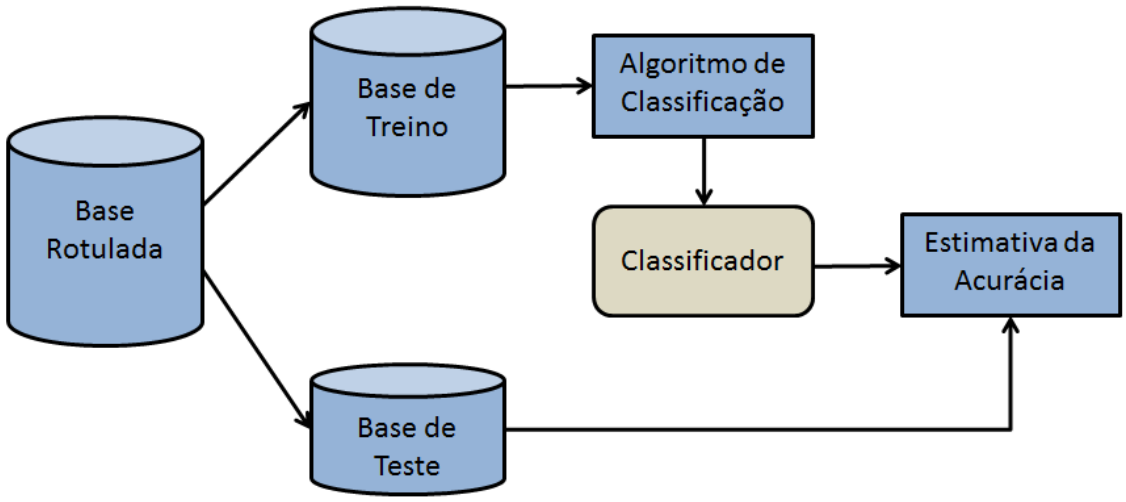


Figura 3.2. Estimando a acurácia de um classificador com o método *holdout*

O método *holdout* não é o mais utilizado na prática, pois existem técnicas capazes de obter estimativas mais confiáveis para o desempenho preditivo de um

classificador. Uma delas é a validação cruzada (*cross-validation*) [Han et al., 2011; Witten et al., 2016]. Neste método, a base rotulada é dividida em *k* partições D_1, D_2,..., D_k de tamanho aproximadamente igual (na Figura 3.3, apresenta-se um exemplo onde *k=10*, um dos valores mais comumente utilizados). Após a divisão, são realizadas *k* rodadas de treino e teste. A cada iteração *i*, a partição D_i é reservada para teste e as partições restantes são utilizadas para treinar o modelo. Considerando o exemplo onde *k*=10, na primeira rodada as partições D_2, D_3,... D_{10} seriam utilizadas para treinamento e a partição D_1 para teste. Na segunda rodada, as partições D_1,D_3... D_9 seriam utilizadas para treinamento e a partição D_2 para o teste. E assim por diante, até a décima rodada. A acurácia final estimada para o modelo será igual à acurácia média das 10 rodadas. Sendo assim, ao contrário do que ocorre com o método *holdout*, na validação cruzada a estimativa da acurácia não é feita "apostando-se todas as fichas" no resultado obtido sobre uma única base de teste, o que constitui uma importante vantagem.

É importante comentar que além da validação cruzada, existem outras técnicas para estimar a qualidade de classificadores, dentre as quais a validação cruzada repetida, *leave-one-out* e *bootstrap* [Witten et al., 2016].

Figura 3.3. Base rotulada dividida em 10 partições (*folds*). Na primeira rodada do método de validação cruzada, as partições D_2, D_3,..., D_{10} são utilizadas para treinar o modelo enquanto a partição D_1 é utilizada para testá-lo.

Matriz de Confusão

A matriz de confusão (MC) é uma das mais úteis ferramentas para a avaliação da qualidade de classificadores. Sua finalidade é armazenar todos os diferentes tipos de erros e acertos realizados pelo classificador ao processar um conjunto de objetos de teste. Uma MC é uma matriz quadrada *q* x *q*, onde *q* representa o número de rótulos de classe envolvidos no problema. Cada célula c_{ij} denota o número de objetos de teste que o classificador associou à classe *i* e que, de fato, pertencem à classe *j*. Desta forma, as células da diagonal principal sempre irão conter o número de objetos corretamente classificados pelo modelo. No exemplo da Figura 3.4, apresenta-se o formato da MC que seria produzida após o processo de teste do classificador de resumos de filmes (nesse problema, *q*=2).

		Rótulos Preditos	
	Classes	Romance='Não'	Romance='Sim'
Rótulos Reais	Romance='Não'	VN	FP
	Romance='Sim'	FN	VP

Figura 3.4. Matriz de confusão gerada após um processo de teste do classificador de resumos de filmes.

Na Figura 3.4, a célula $MC_{1,1}$ armazena o total de objetos da base de teste cujo rótulo era 'Não' e que foram corretamente classificados como 'Não', e por isso são chamados Verdadeiro Negativos (VN). Em $MC_{1,2}$ está representado o total de objetos de teste cujo rótulo era 'Não' e que foram incorretamente classificados como 'Sim', e que por isso são chamados Falso Positivos (FP). O valor em $MC_{2,1}$ indica o número de objetos de teste cujo rótulo era 'Sim' e que foram incorretamente classificados como 'Não', sendo então chamados de Falso Negativos (FN). Por fim, em $MC_{2,2}$ está armazenado o número de objetos de teste cujo rótulo era 'Sim' e que foram corretamente classificados como 'Sim', chamados de Verdadeiro Positivos (VP)

A partir da MC é possível calcular diversas medidas de desempenho [Flach, 2012; Han et al., 2011; Japkowicz and Shah, 2011]. Uma delas é a própria Acurácia, mencionada no início desta subseção, que computa a proporção de objetos de teste corretamente classificados. Sua fórmula é apresentada na Equação 2.

$$Acurácia = (VP + VN) \div (VP + FP + FN + VN) \quad (2)$$

A Acurácia é a medida de desempenho de classificadores mais empregada. Entretanto, não é a mais indicada em situações onde a base de dados é desbalanceada, isto é, quando existem muito mais objetos da classe 'Não' do que da classe 'Sim', ou vice-versa. Este é o caso típico de aplicações como a classificação de fraudes, onde normalmente vão existir muito mais objetos com a classe 'Não' (transações que não são fraudes, ou seja, transações genuínas) do que com a classe 'Sim' (transações fraudulentas). A Figura 3.5 mostra um exemplo de matriz de confusão hipotética que poderia ter sido gerada a partir de uma base de dados contendo 1000 objetos de teste em um problema de classificação de fraudes.

		Rótulos Preditos	
	Classes	Fraude='Não'	Fraude='Sim'
Rótulos Reais	Fraude='Não'	900	90
	Fraude='Sim'	4	6

Figura 3.5. Matriz de confusão gerada após um processo de teste de uma base de dados desbalanceada (classificador de fraudes).

A partir do conteúdo apresentado na MC da Figura 3.5, é possível saber que a base de teste possui 990 objetos com a classe Fraude= 'Não' (soma das colunas da primeira linha da matriz) e apenas 10 objetos com a classe Fraude='Sim' (soma das colunas da segunda linha). A Acurácia do classificador pode ser obtida com o uso da Equação 2:

$$Acurácia = (VP + VN) \div (VN + FP + FN + VP) = (906 / 1000) = 90{,}60\%$$

Este valor de Acurácia de 90,60% parece indicar que o desempenho preditivo do classificador na base de teste foi satisfatório. No entanto, o resultado é ilusório. Ao

observar a matriz, é trivial perceber que o classificador teve uma boa taxa de acertos para o rótulo ‘Não’, mas seu desempenho foi bem inferior para classe ‘Sim’ (algo que seria péssimo para um sistema real, pois ele não foi capaz de identificar grande parte das transações fraudulentas). Por este motivo, é interessante usar a Acurácia em conjunto com métricas capazes de avaliar o desempenho do classificador em cada rótulo de classe. Dois exemplos de medidas deste tipo são a Especificidade (também conhecida como *true negative rate*) e a Sensibilidade (também conhecida como *true positive rate*), respectivamente apresentadas nas Equações 3 e 4.

$$Especificidade = VN \div (VN + FP) \quad (3)$$

$$Sensibilidade = VP \div (VP + FN) \quad (4)$$

Com a Especificidade, torna-se possível avaliar o desempenho do classificador com relação aos verdadeiros negativos. No exemplo da Figura 3.5:

$$Especificidade = VN \div (VN + FP) = 900 \div (900 + 90) = 90{,}90\%.$$

Por sua vez, a Sensibilidade torna possível avaliar o desempenho do classificador com relação aos verdadeiros positivos:

$$Sensibilidade = VP \div (VP + FN) = 6 \div (6 + 4) = 60{,}00\%.$$

Os resultados demonstram que embora o classificador tenha tido um bom desempenho na classificação de casos que não são fraudes (90,90%), seu desempenho foi muito inferior para classificar casos de fraudes (60,00%). Além disso, o seu valor alto de Acurácia de 90,60% foi determinado unicamente pelo fato de a base de teste ter muito mais objetos da classe ‘Não’ (onde o classificador se sai bem) do que da classe ‘Sim’ (onde o desempenho do classificador é insatisfatório).

Além da Especificidade e Sensibilidade, existem diversas métricas que podem ser calculadas a partir dos dados de uma matriz de confusão. Os trabalhos de Flach (2012) e Japkowicz and Shah (2011) são indicados para aqueles que desejam se aprofundar no tema. Adicionalmente, para obter um exemplo de utilização da MC em um problema de classificação monorrótulo multiclasse, consulte [da Silva et al., 2015].

3.2.4. Classificação na Prática

Esta subseção apresenta a receita padrão para incorporar processos de classificação monorrótulo aos seus próprios programas. Os exemplos envolvem bibliotecas disponibilizadas para as linguagens Java e R. Estas duas linguagens foram escolhidas pelo fato de serem bastante diferentes entre si: enquanto Java é uma linguagem compilada e de propósito geral, R é uma linguagem interpretada e de propósito específico (serve basicamente para análise de dados). Os exemplos apresentados demonstram como executar as principais etapas de um processo de classificação: importação da base de dados rotulada, treinamento do modelo de classificação e classificação de novos objetos. Embora os exemplos envolvam o algoritmo NB, é

importante observar que, de uma forma geral, o mesmo modelo pode ser seguido com qualquer outro algoritmo (k-NN , C4.5, SMO, MLP, etc.).

Java (Weka API)

Weka [Frank et al., 2016; Witten et al., 2016] é uma popular ferramenta de aprendizado de máquina e mineração de dados *open-source* desenvolvida em Java. Esta ferramenta fornece uma API bastante poderosa e flexível que permite a integração de suas classes a qualquer tipo de sistema Java.

A Weka API trabalha preferencialmente com arquivos de entrada no formato ARFF (*Attribute Relation File Format*). Este formato corresponde a um arquivo texto contendo a base de dados propriamente dita, precedida por um pequeno cabeçalho que fornece informações a respeito de seus atributos. A Figura 3.6 apresenta a versão ARFF da base de dados de filmes apresentada na Tabela 3.1.

```
@relation filmes
@attribute love {0, 1}
@attribute people {0, 1}
@attribute relationship {0, 1}
@attribute Romance {Nao,Sim}

@data
1,1,0,Sim
1,0,0,Nao
1,0,1,Sim
0,0,1,Sim
1,1,1,Nao
1,1,0,Sim
0,0,0,Nao
0,1,0,Nao
1,1,1,Sim
0,1,1,Nao
1,0,1,Sim
1,1,0,Nao
0,0,0,Nao
1,1,1,Sim
1,0,1,Sim
```

Figura 3.6. Base de dados "filmes_treino.ARFF": versão ARFF da base de dados de filmes apresentada na Tabela 3.1.

O cabeçalho de um arquivo ARFF deve conter um apelido para o mesmo (*tag* @relation), seguido da especificação de cada atributo (*tag* @attribute). Para atributos categóricos é preciso especificar as categorias que ele pode assumir entre chaves (ex.: {0, 1}), enquanto que para os numéricos, utiliza-se a palavra "numeric". No exemplo da Figura 3.5, os atributos são "love", "people", "relationship" e "Romance" (este último, o atributo classe). Os dados propriamente ditos são precedidos por uma *tag* @data. Cada objeto é estruturado em uma linha do arquivo, com os valores separados por vírgula.

A Weka API possui uma classe denominada "NaiveBayes" que oferece uma implementação do algoritmo NB. A seguir será apresentado um exemplo que mostra o código mínimo para treinar um classificador NB e utilizá-lo para classificar dois novos

objetos (objetos de classe desconhecida). Os novos objetos foram estruturados em um arquivo ARFF denominado "filmes_novos.ARFF", que é apresentado na Figura 3.7. No arquivo de novos objetos, um ponto de interrogação ("?") é utilizado para expressar que o valor da classe "Romance" é desconhecido (trata-se da sintaxe adotada pela Weka API). Observe que as bases de dados "filmes_treino.ARFF" (que será usada para treinar o modelo) e "filmes_novos.ARFF" (base com os novos objetos) são compatíveis, isto é, elas possuem os mesmos atributos, dos mesmos tipos e declarados na mesma ordem. Isto costuma ser um pré-requisito exigido pelas bibliotecas para classificação em qualquer linguagem, como Java, R, Python, etc.

```
@relation filmes
@attribute love {0, 1}
@attribute people {0, 1}
@attribute relationship {0, 1}
@attribute Romance {Nao,Sim}

@data
1,0,1,?
0,1,0,?
```

Figura 3.7. Base de dados "filmes_novos.ARFF": contém dois novos objetos (objetos de classe desconhecida)

O código da Figura 3.8, apresenta um programa-exemplo que executa os dois passos necessários para a execução da tarefa de classificação com o algoritmo NB: (i) construção do modelo classificador e (ii) aplicação da Fórmula de Bayes para classificar novos objetos. O exemplo assume que tanto o programa Java, como as bases de dados e a biblioteca "weka.jar" (Weka API) estão localizados na mesma pasta, chamada "c:\cmr". Também é assumido que o código será compilado e executado em uma máquina que possua a JDK (*Java SE Development Kit*) versão 8 ou superior instalada. Informações detalhadas sobre como obter e configurar a Weka API podem ser obtidas em [Gonçalves, 2013].

Para compilar e executar o exemplo, salve o programa com o nome "NaiveBayesUsoBasico.java". Supondo que ele seja salvo dentro da mesma pasta onde estão localizados os arquivos "weka.jar" e as bases de dados ARFF, é possível realizar a compilação utilizando o comando abaixo (obs.: no ambiente Linux, é necessário trocar o ponto-e-vírgula ";" por dois pontos ":"):

```
> javac -cp .;weka.jar NaiveBayesUsoBasico.java
```

Para executar o programa, o processo é análogo, bastando digitar a linha de comando apresentada a seguir (o resultado da execução é mostrado na Figura 3.9):

```
> java -cp .;weka.jar NaiveBayesUsoBasico
```

```
import weka.classifiers.bayes.NaiveBayes;
import weka.core.DenseInstance;
import weka.core.Instance;
import weka.core.Instances;
import weka.core.converters.ConverterUtils.DataSource;

public class NaiveBayesUsoBasico {

public static void main(String[] args) throws Exception {

  // (i) Construção do Classificador

  // (i.1) importação da base de dados de treinamento
  DataSource source = new DataSource("c:\\cmr\\filmes_treino.arff");
  Instances D = source.getDataSet();

  // (i.2) espeficicação do atributo classe
  if (D.classIndex() == -1) {
      D.setClassIndex(D.numAttributes() - 1);
  }

  // (i.3) Treinamento do Modelo de Classificação
  //       (gera a "tabelona" de probabilidades condicionais)

  //basta instanciar a classe "NaiveBayes" e chamar o método
  //"buildClassifier" passando a base de treinamento como entrada
  NaiveBayes nbc = new NaiveBayes();
  nbc.buildClassifier(D);

  // (ii) Classificação dos Novos Objetos

  //(ii.1) importação da base de novos objetos
  DataSource source2 = new DataSource("c:\\cmr\\filmes_novos.arff");
  Instances novosObjetos = source2.getDataSet();

  //(ii.2) loop que classifica todos os novos objetos
  for (int i=0; i < novosObjetos.numInstances(); i++) {

      //(ii.2.1) recupera o objeto
      Instance t = novosObjetos.instance(i);

      //(ii.2.2) classifica esse objeto
      double probs[] = nbc.distributionForInstance(t);

      //(ii.2.3) imprime os resultado da classificação
      System.out.println("- Classificação do Novo Objeto " + i + ":");
      System.out.println("\tProbababilidade de Romance = 'Nao':" +
                                                        probs[0]);
      System.out.println("\tProbababilidade de Romance = 'Sim':" +
                                                        probs[1]);
      }
  }
}
```

Figura 3.8. NaiveBayesUsoBasico.java: programa que utiliza a Weka API para implementar um classificador monorrótulo NB em Java.

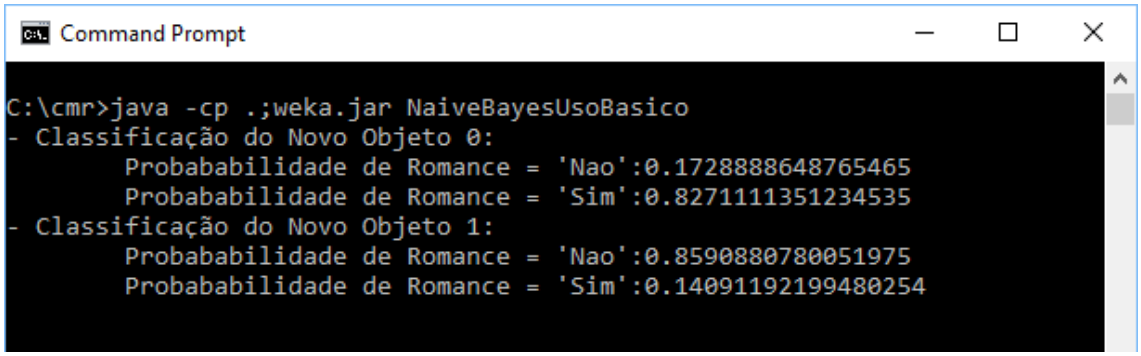


Figura 3.9. Resultado da execução do programa Java

A seguir apresenta-se uma breve explicação sobre o código. Observe que o programa-exemplo está dividido em duas seções: (i) Construção do Classificador e (ii) Classificação dos Novos Objetos. A seção (i) começa com a importação da base de dados de treinamento "filmes_treino.arff" para a memória através dos comandos mostrados na seção de código (i.1). Na Weka API, a importação de bases de dados ARFF é sempre realizada de forma padrão, com o uso das classes "DataSource", "Instances" e "DenseInstances" (ou "SparseInstances"). É importante comentar que a classe "Instances" é uma das mais importantes classes da Weka. Ela representa o conjunto de instâncias da base de dados que está armazenado em memória, oferecendo diversos métodos para a manipulação das mesmas. Consulte o JavaDoc da Weka [Weka, 2018a] e também as referências [Gonçalves, 2013] e [Weka, 2018b] para maiores detalhes.

Na seção (i.2), encontra-se o comando: "D.setClassIndex(D.numAttributes-1);". Este comando serve para informar a Weka que o último atributo da base de dados *D*, isto é, o atributo "Romance" deve ser tratado como atributo classe.

Fechando a primeira parte, a seção (i.3) é a responsável por instanciar a classe "NaiveBayes" e comandar a criação do classificador. Em outras palavras: fazer a Weka API criar a tabela de probabilidades condicionais em memória. Observe que apenas duas linhas de código são necessárias para realizar uma tarefa relativamente complexa. Para instanciar a classe "NaiveBayes" não é preciso passar nenhum parâmetro (basta fazer "new NaiveBayes()"). E para construir o classificador é preciso fazer uma chamada ao método "buildClassifier" passando a base de dados de treinamento como entrada.

A seção (ii) mostra um exemplo de código para classificação dos novos objetos. Em (ii.1) e (ii.2) encontra-se o código necessário para, respectivamente, importar a base de dados de novos objetos ("filmes_novos.ARFF") e depois percorrê-la em um laço. Dentro do laço, cada novo objeto é recuperado individualmente, sendo armazenado em *t*. Em (ii.2.2) está a parte onde a classificação NB é, de fato, realizada. É preciso uma única linha de código para executar a ação: "double probs[] = nbc.distributionForInstance(t);". Nessa linha, o método "distributionForInstance" da classe "NaiveBayes" é chamado passando como entrada o novo objeto *t*. O método "distributionForInstance" contém uma implementação da Fórmula de Bayes que efetua o cálculo das probabilidades associadas aos rótulos de classe. Como resposta, o método "distributionForInstance" retorna um vetor do tipo double, em que cada índice conterá a

probabilidade associada para um dos rótulos de classe. No exemplo, uma variável double [] chamada "probs" foi declarada para receber o retorno do método. O índice probs[0] armazenará a probabilidade computada para Romance='Não' e o índice probs[1] para Romance= 'Sim' (isto ocorre porque no cabeçalho do arquivo ARFF, o atributo classe Romance foi declarado com os valores {Nao,Sim}, ou seja, o 'Não' foi especificado antes do 'Sim').

Fechando o programa, a seção (ii.2.3) simplesmente imprime as estimativas associadas à cada classe, para os dois novos objetos. Observando a Figura 3.8, veja que para o primeiro objeto t_1 = (*love=1, people=0, relationship=1, Romance = ?*), o valor Romance ='Não' foi estimado com probabilidade de 17,29%, enquanto Romance='Sim' com probabilidade de 82,71%. Perceba que esses valores estão diferentes daqueles obtidos através do cálculo mostrado Subseção 3.2.3. Essa divergência ocorre apenas porque a implementação do NB presente na Weka é um pouco mais sofisticada e utiliza, de forma embutida, uma técnica conhecida como correção laplaciana [Han et al., 2011], que adiciona uma constante em todos os valores de probabilidade condicional mantidos na tabela de probabilidade com o intuito de evitar a presença de células contendo probabilidades condicionais com o valor zero.

Embora o exemplo tenha envolvido um programa muito simples, é importante deixar claro que o modelo básico para utilizar a Weka dentro de qualquer sistema será sempre similar. Primeiro será preciso importar a base de dados de treinamento, depois construir o classificador para, em seguida, aplicá-lo sobre novos objetos. Evidentemente, em um projeto real você precisará lidar com algumas questões adicionais. Em especial, será preciso executar um procedimento de teste do modelo de classificação antes de colocá-lo em produção usando o método *holdout* ou a validação cruzada para que seja possível computar os valores de Acurácia, Especificidade, Sensibilidade e outras métricas. Informações sobre a forma de estimar o desempenho preditivo de classificadores através da Weka API podem ser obtidas em [Weka, 2018b].

R

R [R, 2017] é um dos softwares *open-source* para análise de dados mais utilizados em todo o mundo. Existem diversos pacotes de classificação monorrótulo disponibilizados para este ambiente. Esta seção apresenta um exemplo de classificação com a técnica NB que utiliza o pacote "e1071" [Meyer, 2018]. Para obter detalhes sobre como instalar o ambiente R e qualquer um de seus pacotes, consulte [Lander, 2017].

O ambiente R suporta diferentes formatos de arquivo de entrada, sendo CSV um dos mais utilizados. Desta forma, para executar o exemplo a ser apresentado, é preciso converter as bases de dados apresentadas nas Figuras 3.6 e 3.7 do formato ARFF para o formato CSV. Para tal, basta retirar todo o conteúdo compreendido entre as *tags* @relation e @data e acrescentar uma linha de cabeçalho simples, com os nomes dos atributos separados por vírgula (love, people, relationship, Romance). Em seguida, os arquivos deverão ser salvos com a extensão CSV.

O código da Figura 3.10, apresenta um programa R para a classificação. Os passos são os mesmos que foram implementados no programa Java: (i) Construção do Classificador e (ii) Classificação dos Novos Objetos. Entretanto, observe que o código R

é bem menor, algo justificável tendo em vista que R é uma linguagem específica para análise de dados, enquanto Java é uma linguagem de propósito geral. O programa assume que o ambiente R possui o pacote "e1071" instalado e que os arquivos CSV estão localizados na pasta "c:\cmr".

```
#carrega o package "e1071"
library(e1071);

# (i) Construção do Classificador

#(i.1) importação da base de dados de treinamento para um data frame
D <- read.table("c:\\cmr\\filmes_treino.csv",header=TRUE,sep=",")

#(i.2) Treinamento do Modelo de Classificação
#      deve-se utilizar a função naiveBayes do pacote "e1071",
#      indicando o atributo classe ("Romance")
#      e a base de treinamento como parâmetros
nbc <- naiveBayes(Romance ~ ., data = D)

#visualização da tabela de probabilidades:
nbc

# (ii) Classificação dos Novos Objetos

#(ii.1) importação da base de novos objetos
novos <- read.table("c:\\cmr\\filmes_novos.csv",header=TRUE,sep=",")

#(ii.2) classificação dos novos objetos
probs = predict(nbc,novos[,-1], type="raw")

probs #mostra as estimativas para cada classe

preds = predict(nbc,novos[,-1])
preds #mostra apenas os rótulos classe preditos
```

Figura 3.10. NaiveBayesUsoBasico.R: programa que utiliza o pacote "e1071" para implementar um classificador monorrótulo NB em R.

A seção (i.1) começa com a importação da base de treinamento "filmes_treino.csv" para um *data frame* R, utilizando o método "read.table". A seção (i.2) mostra o código responsável por treinar o modelo de classificação com o uso do método "naiveBayes" do pacote "e1071". Observe que dois parâmetros são passados para o método:

- Romance ~ .: indica que o "Romance" é o atributo classe e que os demais atributos (representados por um ponto, ".") serão utilizados como atributos preditivos (essa é a notação, conhecida como "formula", é comumente utilizada na linguagem R para indicar a variável *Y* e as variáveis de *X* em diferentes pacotes para classificação e regressão).
- data = D: indica o *data frame* que armazena a base de dados de treinamento.

A seção (ii) contém o código para classificar novos objetos utilizando o modelo de classificação construído. Inicialmente, em (ii.1) importa-se a base de dados com os novos objetos para o *data frame* "novos". Então, em (ii.2) a classificação é realizada utilizando o método "predict" do pacote "e1071". Foram especificados três parâmetros:

- nbc: modelo de classificação treinado.
- novos[,-1]: *data frame* com os novos objetos. A especificação [,-1] indica quais são os atributos preditivos (todos menos o último).
- type= "raw": o método irá computar as estimativas de probabilidade para cada classe. Se for omitido, os apenas o rótulo de classe previsto é retornado.

Como resposta o método "prediction" retorna um vetor de reais, em que cada índice conterá a probabilidade associada para um dos rótulos de classe (Figura 3.11).

```
> probs = predict(nbc,novos[,-1], type="raw")
> probs #mostra as estimativas de cada classe

           Nao       Sim
[1,] 0.2234392 0.7765608
[2,] 0.7447740 0.2552260

> preds = predict(nbc,novos[,-1])
> preds #mostra apenas os rótulos de classe preditos

[1] Sim Nao
Levels: Nao Sim
```

Figura 3.11. Resultado da execução do programa R

A linguagem R também oferece pacotes para implementar o processo de teste do classificador através das técnicas *holdout*, validação cruzada e outras abordagens. Um dos mais utilizados para este fim é o pacote 'caret' [Kuhn, 2018].

3.3. Classificação Multirrótulo (CMR)

Esta seção dedica-se à apresentação dos conceitos fundamentais sobre CMR. O texto está estruturado da seguinte forma. A Subseção 3.3.1 introduz o problema da CMR, discutindo suas características gerais e seus principais desafios. Em seguida, a Subseção 3.3.2 apresenta as abordagens básicas para a construção de classificadores multirrótulo. A Subseção 3.3.3 discute as propriedades das bases de dados multirrótulo. A Subseção 3.3.4 introduz as principais medidas utilizadas para estimar a qualidade de classificadores multirrótulo. Por fim, a Subseção 3.3.5 cobre a CMR na prática, através da apresentação de *scripts* desenvolvidos em Java e R.

3.3.1. Como Construir um Classificador Multirrótulo?

Assim como ocorre com a classificação monorrótulo, duas coisas são necessárias para que seja possível construir um classificador multirrótulo:

1. Uma boa base de dados de treinamento.
2. Uma boa técnica de CMR.

Entretanto, em problemas de CMR, cada objeto pode estar associado a uma ou mais classes. Um exemplo de base de dados multirrótulo é apresentado na Figura 3.12. Suponha que o objetivo seja utilizá-la para possibilitar a criação de um classificador multirrótulo para categorização de músicas. Neste exemplo, tem-se que $L = \{$*"Metal", "Jazz", "Bossa", "Pop"*$\}$ representa um conjunto não-disjunto de rótulos de classe (gêneros musicais). Considere que cada objeto *i* é um par (x_i, Y_i), onde x_i armazena um número arbitrário de valores de atributos preditivos, enquanto $Yi \subseteq L$ armazena um subconjunto de rótulos de classe.

X	Metal	Jazz	Bossa	Pop
x_1	•			•
x_2		•	•	
x_3		•		
x_4	•			
x_5		•	•	•

Figura 3.12. Base de dados de treinamento de um classificador multirrótulo para a categorização de músicas

No exemplo da Figura 3.12, observe que o primeiro objeto (música) está associado a dois gêneros: "Metal" e "Pop". O segundo objeto também está associado a dois gêneros: "Jazz" e "Bossa". O terceiro objeto possui apenas um gênero ("Jazz") e o quarto também apenas um gênero ("Metal"). Por fim, o quinto objeto possui três gêneros ("Jazz", "Bossa" e "Pop"). A seguir apresenta-se a definição formal para o problema da classificação multirrótulo:

Definição 2 *(Classificação Multirrótulo). Seja* $X = \{X_1, \ldots, X_d\}$ *um conjunto de* d *atributos preditivos e* $L = \{l_1, \ldots, l_d\}$ *um conjunto de* q *rótulos de classe, onde* $q \geq 2$. *Considere uma base de dados de treinamento D composta por N objetos do tipo* $\{(x_1,Y_1), (x_2,Y_2), \ldots, (x_N,Y_N)\}$. *Nesta base de dados, cada* x_i *corresponde a um vetor* $\{x_1, \ldots, x_d\}$ *que armazena valores para os* d *atributos preditivos do conjunto* X *e cada* $Y_i \subseteq L$ *corresponde a um subconjunto de rótulos de classe. O objetivo da tarefa de classificação multirrótulo é, a partir de* D, *realizar o aprendizado de uma função* h *(classificador) que, dado um objeto não rotulado* t = (x,?), *seja capaz de prever de forma efetiva o seu labelset Y (conjunto de rótulos de classe).*

Os problemas de classificação multirrótulo costumam ser muito mais desafiadores do que os de classificação monorrótulo. Isto se deve as seguintes razões [Gonçalves et al., 2018]:

- As aplicações CMR normalmente precisam lidar com um número enorme de combinações de rótulos. Considerando um problema que envolva *q* rótulos de classe distintos, o tamanho do espaço de resultados em um problema CMR é 2^q, enquanto em um problema de classificação monorrótulo é de apenas *q*.
- As aplicações típicas de CMR referem-se a problemas modernos de *big data*, onde o dado a ser processado costuma ser semi-estruturado ou não estruturado: dados multimídia, dados biológicos, etc. Bases de dados reais para CMR costumam ser muito volumosas, tanto em número de atributos como no de objetos.
- Na CMR, é muito comum a existência da correlação entre rótulos de classe. Por exemplo: é improvável que uma canção seja classificada ao mesmo tempo com os rótulos "Heavy Metal" e "Jazz" dado que estes dois estilos musicais possuem uma forte correlação negativa. De maneira análoga, a probabilidade de uma canção ser rotulada como "Pop" torna-se mais forte caso ela tenha sido rotulada como "Dance" e "R&B". Consequentemente, é intuitivo esperar que algoritmos capazes de capturar e modelar as correlações entre rótulos sejam mais acurados. De fato, explorar a dependência entre rótulos tem sido tratado como uma das principais linhas de pesquisa na área de CMR [Boutell et al., 2004; Gibaja and Ventura, 2015; Gonçalves et al., 2013; Hüllermeier et al., 2008; Read et al., 2011; Zhang and Zhou, 2014].
- Em problemas monorrótulo, a classificação de um novo objeto pode estar apenas correta ou errada. Por exemplo, se um classificador monorrótulo classifica um e-mail normal como *spam*, esta situação claramente representa um erro. De maneira oposta, em uma aplicação CMR, os resultados podem estar parcialmente corretos, isto é, o modelo pode classificar corretamente alguns rótulos, mas pode errar outros. Por exemplo, se os rótulos reais de uma música são "Pop" e "Jazz" e o classificador rotulou essa música como "Pop" e "Bossa", este resultado está parcialmente correto. Devido a este fato, a avaliação de classificadores multirrótulo precisa empregar métricas diferentes das utilizadas na avaliação de classificadores monorrótulo.

3.3.2. Técnicas de Classificação Multirrótulo

Nos últimos anos diversas estratégias distintas foram propostas na literatura para a construção de classificadores multirrótulo [Gibaja and Ventura, 2014; Gonçalves et al., 2018; Tsoumakas et al., 2010]. Estes métodos podem ser divididos em duas categorias gerais: Adaptação de Algoritmo (AA) e Transformação do Problema (TP). Esta seção discute ambas as categorias com ênfase dada à segunda, uma vez que os métodos de TP costumam ser mais utilizados na prática.

Os métodos da família AA estendem ou adaptam um algoritmo de classificação monorrótulo existente para que ele possa executar a tarefa de CMR. Por exemplo, o

trabalho de Zhang and Zhou (2007) introduz o método Multi-label kNN (ML-kNN), que é derivado de dois algoritmos monorrótulo: k-NN e NB. Nesta abordagem, o processo de classificação multirrótulo de um novo objeto *t* é realizado em duas etapas. Primeiro, os *k* vizinhos mais próximos de *t* na base de treinamento são identificados. Então, para cada rótulo de classe l_j presente no *labelset* destes *k* vizinhos, a Fórmula de Bayes é empregada para estimar se *t* deverá ou não ser rotulado com l_j. Outros exemplos de métodos da família AA são apresentados e discutidos em [Gibaja and Ventura, 2014].

Os métodos da categoria TP trabalham transformando o problema de CMR original em um ou mais problemas de classificação monorrótulo. Isto possibilita com que qualquer algoritmo de classificação monorrótulo (como NB, k-NN, C4.5, etc.) possa ser diretamente aplicado através do mapeamento das predições monorrótulo para predições multirrótulo. Os métodos de transformação do problema são flexíveis, uma vez que permitem a abstração do algoritmo de classificação base (monorrótulo). Esta é uma vantagem importante, uma vez que diferentes algoritmos monorrótulo são mais ou menos efetivos de acordo com os diferentes domínios de aplicação.

Existem algumas estratégias distintas para executar a transformação de um problema multirrótulo em um ou mais problemas monorrótulo [Gibaja and Ventura, 2015; Tsoumakas et al., 2010]. Entretanto, três delas são principais, de onde as demais foram derivadas: *Label Powerset* (LP), *Ranking by Pairwise Comparison* (RPC) e *Binary Relevance* (BR). A seguir, estas três estratégias são explicadas com ajuda da base de dados para categorização de músicas ilustrada na Figura 3.12.

Label Powerset (LP)

A abordagem LP [Boutell et al., 2004; Tsoumakas et al., 2010] − também referenciada na literatura como "Label Combination" e "Label Creation" − reduz o problema multirrótulo a um único problema monorrótulo do tipo multiclasse. Isto é realizado através da definição de um novo atributo classe composto cujos valores correspondem a todas as combinações distintas de rótulos de classe presentes na base de treinamento. A Figura 3.13 ilustra o processo de transformação LP da base de dados de músicas originalmente apresentada na Figura 3.12.

X	Metal	Jazz	Bossa	Pop
x_1	•			•
x_2		•	•	
x_3		•		
x_4	•			
x_5		•	•	•

X	Y
x_1	Metal-Pop
x_2	Jazz-Bossa
x_3	Jazz
x_4	Metal
x_5	Jazz-Bossa-Pop

Figura 3.13. Transformação LP da base de dados de categorização de músicas

Observe que os dois rótulos associados ao primeiro objeto da base de dados original ("Metal" e "Pop") foram combinados para criar uma nova classe composta chamada "Metal-Pop" na base de dados transformada. De maneira análoga, as novas

classes monorrótulo “Jazz-Bossa” e “Jazz-Bossa-Pop” foram criadas, respectivamente, a partir dos dois rótulos associados ao segundo objeto e aos três rótulos associados ao quinto objeto. É interessante observar que as transformações afetam apenas os rótulos, isto é, os atributos preditivos são mantidos em sua forma original (de fato, esta é uma característica comum a todos os métodos da família TP).

Uma vez que a transformação tenha sido aplicada à base de dados original, a criação de um modelo LP pode ser feita de maneira trivial: basta treinar um classificador monorrótulo multiclasse sobre a base transformada. A classificação de uma nova instância também é trivial: o modelo LP simplesmente gera como saída uma classe composta que, de fato, corresponde a um conjunto de rótulos no problema original.

O método LP é simples e oferece a vantagem de implicitamente levar a correlação entre rótulos em consideração. Entretanto, possui dois problemas principais. Primeiro, ele não é capaz de prever *labelsets* que não estejam presentes na base de treino original. Por exemplo, o modelo LP gerado a partir da base de dados transformada da Figura 3.12 jamais será capaz de classificar uma nova música como “Jazz-Pop” ou somente como “Bossa” porque essas combinações não existem na base de dados multirrótulo original. Segundo, e mais importante, a transformação LP pode gerar um número exponencial de classes compostas, algumas delas associadas a um número reduzido de objetos. O número máximo de classes compostas é dado por $min(N, 2^q)$, onde N corresponde ao número de objetos da base de treino e q representa o número de rótulos. Embora na prática o número real de combinações seja usualmente bem menor do que o limite superior possível, ele é normalmente muito maior do que q (como será mostrado na Subseção 3.3.3). Desta forma, o método LP costuma ser inviável para muitos problemas reais.

Ranking by Pairwise Comparision (RPC)

A abordagem RPC [Hüllermeier et al., 2008] transforma a base de dados multirrótulo original em diversas bases de dados monorrótulo. Cada base de dados derivada é associada a um par distinto de rótulos $\{l_i, l_j\}$, $1 \leq i < j \leq q$ e deve conter os objetos da base de treinamento D que estejam associados ao rótulo l_i ou l_j, mas nunca associados a ambos. A Figura 3.14 apresenta o resultado da transformação RPC da base de dados de categorização de músicas. Veja que seis bases de dados binárias são produzidas como resultado da transformação.

Observe que a primeira base de dados binária se refere ao par de rótulos “Metal” e “Jazz”. Todos os objetos da base de dados podem estar associados a um dos seguintes rótulos de classe: “Metal(1)-Jazz(0)” ou “Metal(0)-Jazz(1)”. Nesta base binária, o primeiro e o quarto objetos possuem a classe “Metal(1)-Jazz(0)” porque na base de dados multirrótulo original, os respectivos objetos estão rotulados como “Metal” e não estão como “Jazz”. De maneira análoga, o segundo, terceiro e quinto objetos da base binária possuem a classe “Metal(0)-Jazz(1)” porque, na base de dados multirrótulo original estão associados ao rótulo “Jazz”, mas não estão associados ao rótulo “Metal”.

A construção de um modelo RPC consiste no treinamento de um classificador monorrótulo binário para cada uma das bases de dados derivadas. Portanto, cada classificador é treinado para um par de rótulos $\{l_i, l_j\}$, formando uma fronteira de

decisão entre ambos. Ao contrário do que ocorre no método LP, a etapa de classificação do RPC não pode ser executada diretamente. No método RPC, um novo objeto *t* a ser classificado deve ser primeiro submetido a todos os classificadores binários. O conjunto de rótulos final predito para *t* será obtido através da aplicação de uma função de limiar capaz de separar os rótulos com maior número de vitórias daqueles com menor número.

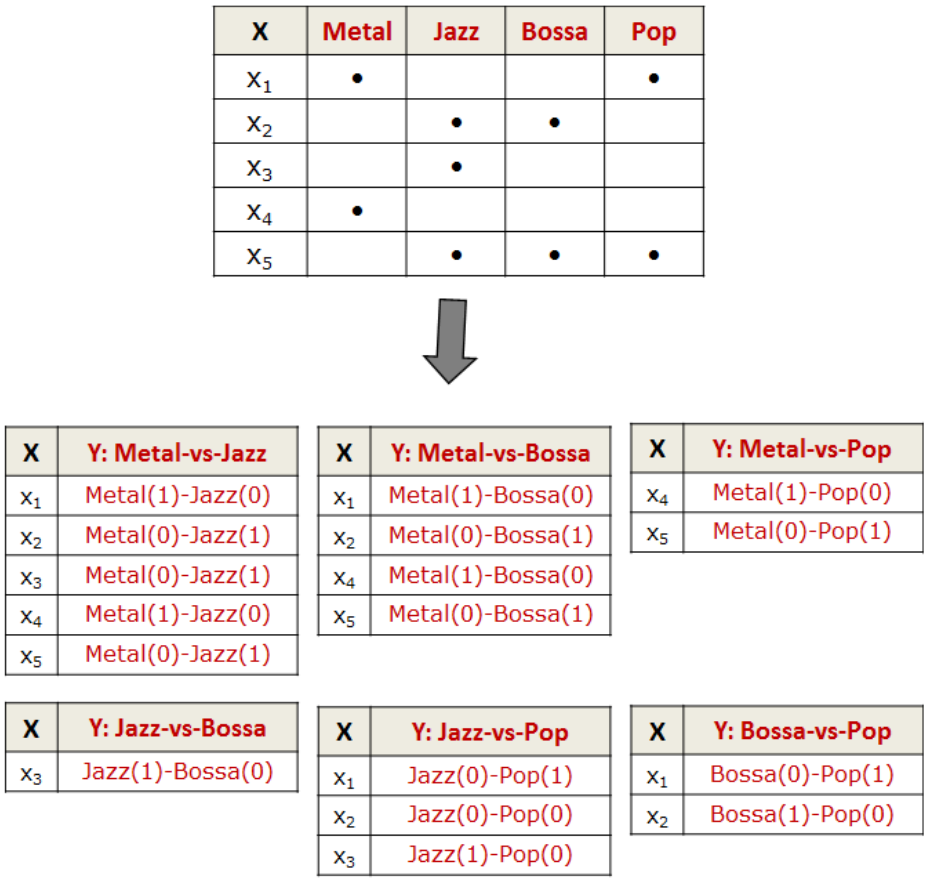


Figura 3.14. Transformação RPC da base de dados de categorização de músicas

O método RPC possui a vantagem de conseguir prever combinações de rótulos que não estão presentes na base de dados original, sendo ainda capaz de incorporar implicitamente as correlações entre rótulos no processo de construção do modelo de classificação. Entretanto, ele possui duas desvantagens importantes: primeiro o seu processo de classificação é dependente de uma função de limiar que, na prática, precisa ser calibrada de acordo com as diferentes bases de dados. Segundo, ele pode atingir complexidade quadrática em termos de espaço e tempo, uma vez que, no pior caso, um total de $q(q-1) / 2$ classificadores binários precisarão ser treinados e mantidos em memória. Todos eles precisam ser consultados no momento da classificação. Devido a este fato, o RPC acaba sendo intratável em alguns problemas reais onde q é alto.

Binary Relevance (BR)

BR [Godbole and Sarawagi, 2004; Tsoumakas et al., 2010] é o método de classificação multirrótulo mais conhecido e adotado. Nesta abordagem, a base de dados multirrótulo original é decomposta em *q* bases de dados binárias, uma para cada rótulo distinto. A Figura 3.15 apresenta o resultado da transformação BR da base de dados de categorização de músicas.

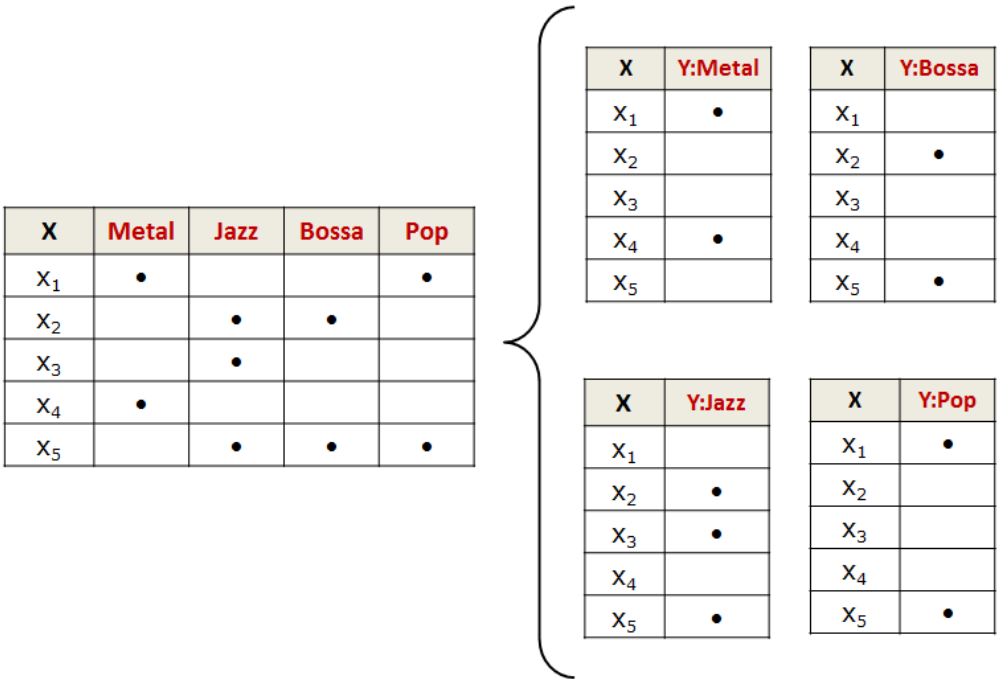


Figura 3.15. Transformação BR da base de dados de categorização de músicas

A indução de um modelo BR consiste no treinamento de um classificador binário para cada base de dados derivada. Consequentemente, no exemplo da categorização de músicas, quatro modelos binários independentes precisariam ser treinados, sendo cada um deles exclusivamente responsável pela classificação de um dos gêneros.

Tendo o modelo BR sido treinado, o processo de classificação é bastante simples: o conjunto de rótulos de um novo objeto é predito através da simples combinação das saídas produzidas por cada um dos classificadores binários. Este processo é ilustrado no exemplo da Figura 3.16. Ela mostra o processo de classificação de um novo objeto *t* (a música "Garota de Ipanema") considerando um modelo BR hipotético treinado utilizando o grupo de bases de dados derivadas apresentado na Figura 3.14. Nesta figura, f_1, f_2, f_3 e f_4 representam, respectivamente, os classificadores binários treinados para classificar os gêneros "Metal", "Jazz", "Bossa" e "Pop", enquanto *x* representa o conjunto de atributos preditivos que descrevem *t*.

t = "*Garota de Ipanema*" – Tom Jobim & Frank Sinatra

Classificadores	Classificações
f_1: x → {Metal, ~Metal}	~Metal
f_2: x → {Jazz, ~Jazz}	Jazz
f_3: x → {Bossa, ~Bossa}	~Bossa
f_4: x → {Pop, ~Pop}	Pop
Conjunto de Rótulos Predito	***{Jazz, Pop}*** $\subseteq L$

Figura 3.16. Classificação da música "Garota de Ipanema" por um modelo BR

Observe que para realizar a classificação multirrótulo do novo objeto *t*, o modelo BR agrega as predições de todos os classificadores binários independentes. No exemplo, "Metal" e "Bossa" foram preditos como não-relevantes, enquanto "Jazz" e "Pop" foram considerados relevantes. Desta forma, o *labelset* {"Jazz", "Pop"} será associado a *t*.

A estratégia BR oferece vantagens importantes. Primeiro, assim como a abordagem LP, trata-se de um método simples e intuitivo. Segundo, assim como a abordagem RPC, é capaz de prever conjuntos de rótulos que não estão presentes na base de dados multirrótulo original. Terceiro, ao contrário do LP e RPC, o método BR possui complexidade linear em *q*. Entretanto uma desvantagem óbvia e importante se encontra no fato de que um classificador BR ignora inteiramente as possíveis correlações entre os rótulos de classe, uma vez que os classificadores binários tomam decisões independentes. Apesar disso, o método costuma apresentar desempenho satisfatório, sendo muitas vezes capaz de superar os resultados obtidos por técnicas bem mais sofisticadas, como foi demonstrado em [Madjarov et al., 2012].

Como comentário final, é importante mencionar que a partir dos métodos LP, RPC e BR, diversos métodos derivados foram propostos na literatura. Algumas destas variações alcançaram grande destaque, como, por exemplo, *Classifier Chains* [Read et al., 2011], *Hierarchy of Multi-Label Classifiers* [Tsoumakas et al., 2008] e *Random k-Labelsets* [Tsoumakas et al., 2007].

3.3.3. Propriedades das Bases de Dados Multirrótulo

O desempenho dos diferentes métodos de CMR pode ser afetado pelas características da base de dados de treinamento. Esta subseção apresenta um estudo das propriedades que caracterizam e definem a complexidade de uma base de dados multirrótulo. Para facilitar a discussão, são apresentados exemplos obtidos a partir de bases de dados *benchmark* comumente utilizadas em experimentos de classificação multirrótulo (estas bases contêm dados reais de diferentes domínios de aplicação). A relação a seguir apresenta o nome de cada base, seguido de seu domínio de aplicação e uma breve descrição de suas características. A maioria delas está disponibilizada no *Web site* da ferramenta Mulan[2] e maiores informações sobre as mesmas podem ser obtidas consultando-se as referências indicadas no texto.

[2] http://mulan.sourceforge.net/datasets-mlc.html

- Emotions [Throdis et al., 2008] (classificação de músicas em emoções): classificação de músicas em 6 tipos de emoções: “sad-lonely”, “angry-aggressive”, “amazed-surprised”, “relaxing-calm”, “quiet-still” e “happy-pleased”. A base de dados possui 593 músicas diferentes descritas por 72 atributos preditivos numéricos.
- Scene [Boutell et al., 2004] (classificação semântica de cenas): classificação de imagens em 6 diferentes contextos: “beach”, “sunset”, “field”, “fall-foliage”, “mountain” e “urban”. A base de dados é formada por 2407 imagens descritas por 294 atributos preditivos numéricos.
- Flags [Gonçalves et al., 2013] (geografia): base de dados simples contendo informações sobre 194 países e suas bandeiras nacionais. Cada objeto é descrito por 19 atributos discretos e numéricos (como, por exemplo, área em km^2, idioma e religião predominante). A tarefa de CMR consiste em prever as cores presentes nas bandeiras dos países (“red”, “green”, “blue”, “yellow”, “white”, “black” e “orange”).
- Yeast [Elisseeff and Weston, 2001] (genômica funcional): esta é uma base de dados biológica onde genes de levedura podem estar associados a até 14 categorias funcionais. Possui 2417 objetos descritos por 103 atributos numéricos que representam expressões de *microarray* e perfis filogenéticos dos genes da levedura.
- POF-16 [Gonçalves et al. 2006] (pesquisa social): contém dados de 904 famílias entrevistadas pela Pesquisa de Orçamentos Familiares. Cada família é caracterizada por apenas 3 atributos: renda mensal, número de membros e cidade de residência. A tarefa CMR consiste em prever a coleção de produtos adquiridos por cada família em sua última visita a um supermercado.
- Birds [Briggs et al. 2013] (classificação de áudio e biologia): classificação das espécies de pássaros presentes em áudios de 10 segundos. Contém 645 objetos, 260 atributos preditivos e 19 rótulos de classe (espécies de pássaros).
- Genbase [Diplaris et al., 2005] (genômica funcional): classificação de proteínas em 27 funções. A base mantém dados de 662 cadeias de proteínas caracterizadas por um vetor binário de comprimento 1186.
- Medical [Pestian et al., 2007] (categorização de texto e diagnose médica): classificação de relatórios clínicos em 45 códigos distintos que representam diferentes doenças ou condições clínicas. Mantém informação sobre 978 relatórios, cada um associado a um paciente. Cada relatório é descrito por 1149 atributos binários que indicam se um termo (palavra) específico está presente ou ausente do texto (formato similar ao da base de dados de resumos de filmes, apresentada na Tabela 3.1).
- Enron [Klimt and Yang, 2004] (categorização de texto): classificação de e-mails em 54 categorias. A base contém 1702 mensagens de e-mail trocadas entre empregados da Enron Corporation que foram tornadas públicas durante uma investigação. Cada e-mail é descrito por 1001 atributos binários que indicam se um termo específico está presente ou ausente.

- Cal500 [Turnbull et al. 2008] (categorização de músicas): classificação de músicas em 174 diferentes tipos de conceitos, que, entre outros, podem representar gêneros musicais (ex.: “Soul”, “Jazz”, “Pop”, etc.) e a presença de certos instrumentos (ex.: “Synthetizer”, “Piano”, “Saxophone”, etc.). A base contém 502 músicas populares gravadas nos últimos 50 anos. Cada música é descrita por 68 atributos numéricos.

A Tabela 3.3 apresenta uma coleção de propriedades que podem ser utilizadas para indicar, de uma forma simplificada, a complexidade associada a cada uma das bases de dados *benchmark* recém-apresentadas. A primeira coluna informa o nome de cada base, enquanto a segunda, terceira e quarta colunas (N, d e q) apresentam, respectivamente, o número de objetos, atributos preditivos e rótulos de classe. Observe que as bases de dados estão apresentadas em ordem crescente de quantidade de rótulos. A quinta coluna indica a complexidade da base de dados, como proposta em [Read et al., 2011], que é obtida pelo produto $N \times d \times q$. Entretanto, é importante observar que métodos baseados na transformação do problema multirrótulo em diversos problemas monorrótulo, como o RPC e o BR, podem ser mais afetados pelo tamanho de q (parâmetro que, de fato, irá definir o número de classificadores binários que precisarão ser treinados) do que pela complexidade propriamente dita.

Tabela 3.3. Lista de bases de dados multirrótulo benchmark e as suas estatísticas com respeito à complexidade.

Base de Dados	*N*	*d*	*q*	*Complexidade*
Emotions	593	72	6	256.176
Scene	2.407	294	6	4.245.948
Flags	194	19	7	25.802
Yeast	2.417	103	14	3.485.314
POF-16	904	3	16	43.392
Birds	645	260	19	3.186.300
Genbase	662	1.186	27	21.198.564
Medical	978	1.449	45	63.770.490
Enron	1.702	1.001	53	90.296.206
Cal500	502	68	174	5.939.664

A Tabela 3.4 apresenta uma série de indicadores utilizados para fornecer uma razoável indicação das “propriedades multirrótulo” de uma base de dados [Charte and Charte, 2015]. A primeira coluna informa o nome da base, enquanto a segunda e terceira reproduzem, respectivamente o número de objetos (N) e rótulos (q). Os valores na quarta coluna (*LCard*) apresentam a cardinalidade associada a cada base de dados, que corresponde ao número médio de rótulos por objeto. O menor valor, próximo a 1,0 é da base de dados “Birds”, onde a maioria dos objetos está associada a apenas um rótulo. Por outro lado, o maior valor é superior a 26,0 na base de dados “Cal500”. Complementarmente, a quinta coluna (*LDens*) fornece o valor da densidade, que corresponde a *LCard* / q. Estas duas métricas foram introduzidas em [Tsoumakas et al., 2010]. A sexta coluna (*NC*) apresenta o número de combinações de rótulos distintas presentes em cada base. A sétima coluna (*NU*) apresenta o número de combinações

únicas, isto é, o número de combinações de rótulos que possuem frequência igual a 1 em cada base de dados. Estes dois índices foram propostos em [Read et al., 2011]. O presente trabalho introduz dois novos índices: O primeiro é *NU/NC* (coluna 8) que informa a proporção de *labelsets* únicos em relação ao número total de *labelsets* distintos. O segundo é o número de pares RPC (*NP*), apresentado na última coluna. Este índice indica o número total de pares rótulos de classe distintos $\{l_i, l_j\}$, $1 \leq i < j \leq q$ que estão associados a pelo menos um objeto da base de dados, considerando que o objeto deve possuir o rótulo l_i mas não o l_j e vice-versa. Em outras palavras: este índice fornece o número de classificadores binários que precisariam que precisariam ser treinados para construir um classificador multirrótulo através do método RPC.

Tabela 3.4. Lista de bases de dados multirrótulo benchmark e as suas estatísticas com respeito às propriedades multirrótulo.

Base de Dados	*N*	*q*	*LCard*	*LDens*	*NC*	*NU*	*NU/NC*	*NP*
Emotions	593	6	1,87	0,31	27	4	0,15	15
Scene	2.407	6	1,07	0,18	15	3	0,20	15
Flags	194	7	3,39	0,49	54	24	0,44	21
Yeast	2.417	14	4,24	0,30	198	77	0,43	91
POF-16	904	16	3,17	0,20	475	358	0,75	120
Birds	645	19	1,01	0,05	133	73	0,55	171
Genbase	662	27	1,25	0,05	32	10	0,31	350
Medical	978	45	1,25	0,03	94	33	0,35	984
Enron	1.702	53	3,38	0,06	753	573	0,76	1.378
Cal500	502	174	26,04	0,15	502	502	1,00	15.028

Conforme mencionado no início desta subseção, as propriedades de cada base de dados exercerão influência no desempenho de cada um dos diferentes métodos de CMR. Por exemplo, considere o método LP que trata cada combinação distinta de rótulos de classe existente na base de treinamento como uma nova classe monorrótulo composta. As informações apresentadas na Tabela 3.4 demonstram que este método é inviável para a maioria das bases de dados. Isto se deve a dois motivos: primeiro, o valor da propriedade *NC* (número de combinações distintas de rótulos) é superior a 30 na maioria das bases, o que pode ser considerado um número de rótulos de classes consideravelmente alto para um problema de classificação multiclasse. Segundo, observe que muitas bases possuem um número alto de combinações que estão associadas a apenas um objeto (propriedade *NU*). No caso das bases "POF-16", "Birds", "Enron" e "Cal500", onde a proporção *NU/NC* é superior a 50%, o que significa que entre o total de combinações de rótulos distintas, mais de 50% estão associadas a um único objeto da base de treinamento. Portanto, o método LP teria que lidar com um grande número de classes compostas que ocorrem apenas uma vez na base resultante da transformação. No caso extremo da "Cal500", o número de combinações distintas de rótulos é igual ao número de objetos (*NC = N*), e, desta forma, o método LC teria que lidar com apenas um objeto por classe composta.

As informações da Tabela 3.4 também possibilitam uma comparação entre os métodos RPC (onde um classificador binário deve ser treinado para cada par de rótulos de interesse) e BR (onde um classificador binário é treinado para cada rótulo do

problema) em termos de complexidade de tempo e espaço. Neste sentido, a propriedade *NP* revela que o número de classificadores binários que precisariam ser treinados e mantidos em memória para o método RPC é igual ou muito próximo ao limite superior de *q(q-1) / 2* para todas as bases de dados. Por outro lado, o método BR requer o treinamento de apenas *q* classificadores.

Em resumo: o método BR costuma ser computacionalmente viável para qualquer problema real de classificação. Por este motivo, tornou-se muito difundido e utilizado na prática, representando ainda o principal método *benchmark* na área de CMR (isto é: novos métodos propostos costumam ser comparados com o BR, tanto em termos de eficácia como de eficiência).

3.3.4. Medidas de Desempenho para Classificadores Multirrótulo

Nos últimos anos, diversas medidas de desempenho preditivo especialmente projetadas para CMR foram propostas na literatura. As plataformas Mulan [Tsoumakas et al., 2011] e Meka [Read et al., 2016], por exemplo, disponibilizam mais de vinte diferentes métricas aos seus usuários. Esta subseção introduz e compara algumas das mais comumente utilizadas. Nos exemplos e definições, a seguinte notação foi utilizada:

- n: número de objetos de teste;
- q: número de rótulos;
- Y_i: labelset real do objeto i da base de teste.
- Z_i: labelset predito para o objeto i da base de teste.

A medida mais trivial para avaliar o desempenho de métodos CMR é o Casamento Exato (*exact match*), definida na Equação 5. Esta medida determina a proporção de objetos que tiveram todos os seus rótulos preditos de maneira correta na base de teste. Considere que *I(VERDADEIRO) = 1* e *I(FALSO) = 0*.

$$CE = \frac{1}{n}\sum_{i=1}^{n} I\left(Y_i = Z_i\right) \tag{5}$$

Apesar de fornecer uma informação de grande importância, o Casamento Exato é uma medida considerada rigorosa demais para ser empregada sozinha. Isto porque, em problemas de CMR o resultado de uma classificação pode frequentemente estar parcialmente correto. Portanto, torna-se necessário empregar outros índices em processos de avaliação de classificadores multirrótulo. Um dos mais simples é a Acurácia Multirrótulo (*multi-label accuracy*), apresentada na Equação 6. Esta medida pode ser vista como uma versão "mais leve" do Casamento Exato. Ela provê informação sobre a proporção de rótulos classificados de maneira correta, levando em consideração resultados parcialmente corretos.

$$AC = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|} \quad (6)$$

A Tabela 3.5 ilustra o uso da Acurácia Multirrótulo em um problema hipotético envolvendo 6 exemplos (objetos de teste) e 4 rótulos de classe [Gonçalves et al., 2013]. Observe que a medida é simples e intuitiva: quanto maior o seu valor, melhor a classificação. Entretanto, os casos ilustrados nos exemplos E_5 e E_6 evidenciam uma desvantagem associada a esta medida: ela não é muito severa para penalizar predições binárias erradas (sejam falsos positivos ou falsos negativos).

Tabela 3.5. Ilustração da Acurácia Multirrótulo para 6 objetos de teste.

	Y_i	Z_i	*Acurácia Multirrótulo*	*Comentários*
E_1	l_1, l_2, l_3, l_4	l_1, l_2, l_3, l_4	1,00	Classificação perfeita
E_2	l_1	l_2, l_3, l_4	0,00	Pior caso
E_3	l_1	l_2	0,00	Pior caso
E_4	l_1, l_3	l_1, l_2	0,33	
E_5	l_1	l_1, l_2	0,50	
E_6	l_1, l_2	l_1, l_2, l_3, l_4	0,50	Número maior de falsos positivos em relação à E_5, mas o valor continua sendo 0,50

Para lidar com esse problema, é possível empregar a medida conhecida como Hamming Loss, definida na Equação 7. Esta medida informa o número médio de predições binárias incorretas por objeto de teste. A expressão $|Y_i \Delta Z_i|$ representa a diferença simétrica entre Y_i e Z_i, a qual é equivalente à operação booleana XOR. Quanto menor o valor do Hamming Loss, melhor o desempenho. O uso da medida Hamming Loss em um novo problema com 6 exemplos e quatro rótulos de classe é ilustrado na Tabela 3.6 [Gonçalves et al., 2013].

$$\mathrm{HL} = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \Delta Z_i|}{q} \quad (7)$$

Tabela 3.6. Ilustração do Hamming Loss para 6 objetos de teste.

	Y_i	Z_i	*Acurácia Multirrótulo*	*Comentários*
E_1	l_1, l_2, l_3, l_4	l_1, l_2, l_3, l_4	0,00	Classificação perfeita
E_2	l_1	l_2, l_3, l_4	1,00	Pior caso
E_3	l_1	l_2	0,50	
E_4	l_1, l_3	l_1, l_2	0,50	Esta classificação é melhor do que a de E_3, mas o valor continua sendo 0,50
E_5	l_1	l_1, l_2	0,25	
E_6	l_1, l_2, l_3	l_1, l_2, l_3, l_4	0,25	Número maior de predições corretas em relação à E_5, mas o valor continua sendo 0,25

Observe que a mais importante vantagem do Hamming Loss sobre a Acurácia Multirrótulo é o fato de que ele é mais adequado para penalizar predições binárias incorretas. Contudo, em muitas situações práticas essa característica pode se tornar uma desvantagem. Por exemplo, considere os quatro últimos exemplos da Tabela 3.6. Eles mostram que, considerando um par de exemplos com o mesmo número de predições binárias incorretas (como E_3 e E_4 ou E_5 e E_6), a medida de Hamming Loss não diferencia o caso com um número maior de classificações corretas do caso que possui um menor número.

É importante enfatizar que, a despeito de suas idiossincrasias, todas as três medidas (Casamento Exato, Acurácia Multirrótulo e Hamming Loss) são importantes, uma vez que fornecem informações complementares sobre o desempenho de um classificador multirrótulo. Outro importante aspecto relacionado a estas três medidas é elas são computadas levando em conta todos os rótulos reais e previstos de um exemplo (objeto de teste). Por este motivo, são chamadas de medidas baseadas em exemplo (*example-based measures*).

Entretanto, em certas situações, pode também ser interessante avaliar o classificador em cada rótulo de classe separadamente. Isto é: criar uma matriz de confusão para cada rótulo e computar a Sensibilidade, Especificidade e outras medidas para cada *label*. Desta forma, torna-se possível verificar se o desempenho do classificador está equilibrado entre os vários rótulos.

3.3.5. Classificação Multirrótulo na Prática

Esta subseção apresenta exemplos de *scripts* para CMR desenvolvidos em Java e R. Nestes exemplos, um classificador BR que utiliza o NB como algoritmo base (BR + NB) é criado a partir de uma base de treinamento multirrótulo. Em seguida, o modelo BR + NB é utilizado para classificar uma base de novos objetos. O objetivo principal é dar uma noção sobre como a CMR funciona na prática.

Java (Weka API)

As plataformas Mulan [Tsoumakas et al., 2011] e Meka [Read et al., 2016] são muito referenciadas na literatura de CMR. Ambas são ferramentas *open source*, criadas no topo da Weka API e voltadas para a elaboração de pesquisas científicas: elas foram projetadas de modo a facilitar o desenvolvimento de protótipos e a implementação de experimentos de avaliação de classificadores. Além disso, representam ótima fonte de consulta para aqueles que desejam aprender aspectos avançados de programação relacionados à Weka API. No entanto, nem a Mulan e nem a Weka foram construídas com o propósito de serem integradas a aplicativos desenvolvidos em Java. Sendo assim, nesta seção apresenta-se um exemplo que depende apenas da Weka API para ser executado e que, com poucas modificações, poderá ser integrado diretamente a qualquer sistema desenvolvido em Java (é importante observar que o código apresentado tem por base a implementação do método BR disponível na Mulan).

O exemplo utiliza a base de dados “Flags” [Gonçalves et al., 2013], descrita na Subseção 3.3.3. Esta base pode ser obtida no formato ARFF a partir do Web site da ferramenta Mulan. Ela armazena informações sobre países, que são caracterizados por

19 atributos preditivos. A tarefa de CMR consiste em prever as cores presentes nas bandeiras dos países, onde L = {“red”, “green”, “blue”, “yellow”, “white”, “black”, “orange”}. No exemplo, serão utilizadas as partições de treino para criar o classificador (“flags-train.ARFF”) e de teste para classificar novos objetos (“flags-test.ARFF”) – ambas as partições são fornecidas no arquivo de *download* da base de dados. O cabeçalho da base “flags-train.ARFF” e as seus três primeiros objetos são apresentados na Figura 3.17. Observe que os sete últimos atributos representam os rótulos de classe[3].

```
@relation flags_ml

@attribute landmass {1,2,3,4,5,6}
@attribute zone {1,2,3,4}
@attribute area numeric
@attribute population numeric
@attribute language {1,2,3,4,5,6,7,8,9,10}
@attribute religion {0,1,2,3,4,5,6,7}
@attribute bars numeric
@attribute stripes numeric
@attribute colours numeric
@attribute circles numeric
@attribute crosses numeric
@attribute saltires numeric
@attribute quarters numeric
@attribute sunstars numeric
@attribute crescent {0,1}
@attribute triangle {0,1}
@attribute icon {0,1}
@attribute animate {0,1}
@attribute text {0,1}
@attribute red {0,1}
@attribute green {0,1}
@attribute blue {0,1}
@attribute yellow {0,1}
@attribute white {0,1}
@attribute black {0,1}
@attribute orange {0,1}

@data
4,1,164,7,8,2,0,0,2,1,0,0,0,1,1,0,0,0,0,1,0,0,0,1,0,0
4,4,111,1,10,5,0,11,3,0,0,0,1,1,0,0,0,0,0,1,0,1,0,1,0,0
1,4,0,0,1,1,0,0,3,1,0,0,0,7,0,1,0,1,0,1,1,0,1,0,0,0
...
```

Figura 3.17. Base de dados “flags-train.ARFF”

A seguir, apresenta-se um programa-exemplo que treina um modelo de classificação BR + NB utilizando a base de dados apresentada na Figura 3.17. Em seguida, o modelo é utilizado para classificar objetos armazenados no arquivo “flags-test.ARFF”. Como o código é bem maior do que o do exemplo envolvendo a classificação monorrótulo, ele foi dividido em duas figuras: Figura 3.18 (etapa de treinamento) e Figura 3.19 (mostra a etapa de classificação).

[3]As bases de dados ARFF utilizadas pela ferramenta Mulan são sempre fornecidas em conjunto com um arquivo XML que indica quais atributos representam os rótulos de classe.

```
import weka.core.converters.ConverterUtils.DataSource;
import weka.core.DenseInstance;
import weka.core.Instance;
import weka.core.Instances;
import weka.core.Attribute;
import weka.classifiers.AbstractClassifier;
import weka.classifiers.Classifier;
import weka.classifiers.meta.FilteredClassifier;
import weka.filters.unsupervised.attribute.Remove;
import weka.classifiers.bayes.NaiveBayes;

public class BR {

public static void main(String[] args) throws Exception {

  // (i) Construção do Classificador BR
  //  (implementação baseada no código da ferramenta MULAN)

  // (i.1) importacao da base de dados de treinamento
  DataSource source = new DataSource("c:\\cmr\\flags-train.arff");
  Instances D = source.getDataSet();
  int q = 7;                    //número de rótulos
  int d = D.numAttributes()-q; //número de atributos preditivos

  //(i.2) cria um array h[] de classificadores NB (um para cada label)
  NaiveBayes baseClassifier = new NaiveBayes();
  FilteredClassifier[] h = new FilteredClassifier[q];
  for (int yj = 0; yj < q; yj++) {
    h[yj] = new FilteredClassifier();
    h[yj].setClassifier(AbstractClassifier.makeCopy(baseClassifier));
    }

  // (i.3) treina o modelo BR
  // Este laço treina um classificador binário para cada label
  for (int yj = 0; yj < q; yj++) {

    // define os labels que serão removidos (todos, exceto yj!)
    int l_index = d + yj;
    int [] labelsToRemove = new int[q - 1];
    int aux = 0;
    for (int k = 0; k < q; k++) {
      if (k != yj) {
        labelsToRemove[aux] = d + k;
        aux++;
      }
    } //--for k

    // treina o "filtered classifier" associado a yj
    Remove remFilter = new Remove();
    remFilter.setAttributeIndicesArray(labelsToRemove);
    remFilter.setInputFormat(D);
    remFilter.setInvertSelection(false);
    h[yj].setFilter(remFilter);
    D.setClassIndex(l_index);
    h[yj].buildClassifier(D);
  } // --for yj
```

Figura 3.18. BR.java (parte 1 – etapa de treinamento): programa que utiliza a Weka API para implementar um classificador multirrótulo BR + NB

```
  // (ii) Classificação dos Novos Objetos

  //(ii.1) importação da base de novos objetos
  DataSource source2 = new DataSource("c:\\cmr\\flags-test.arff");
  Instances T = source2.getDataSet();
  int n = T.numInstances();

  //(ii.2) loop que classifica todos os novos objetos
  for (int k = 0; k < n; k++) {

    Instance t = T.instance(k); //obtém o objeto corrente

    int Z[] = new int[q];  //vetor que armazenará as predições
                           // de cada label para o objeto corrente
    System.out.println();

    for (int yj = 0; yj < q; yj++) {

      double probs[] = h[yj].distributionForInstance(t);

      Z[yj] = (probs[0] > probs[1]) ? 0 : 1;

      System.out.print(Z[yj]); //imprime a predição para o label yj
    } // --for yj
  } // --for k

  }
}
```

Figura 3.19. BR.java (parte 2 – etapa de classificação): programa que utiliza a Weka API para implementar um classificador multirrótulo BR + NB

Para compilar o programa, é preciso juntar o código apresentado nas duas figuras e salvar o arquivo com o nome "BR.java". Mais uma vez, o exemplo assume que o código será criado e executado em uma máquina que possua a JDK versão 8 e superior e que o programa Java, as bases de dados e a biblioteca "weka.jar" (Weka API) estão localizados na mesma pasta (neste caso, pasta "c:\cmr").

A seguir apresenta-se uma breve explicação sobre o *script*, começando pela etapa de treinamento (Figura 3.18). Neste programa, o método BR é implementado nas seções de código (i.2) e (i.3). Na Seção (i.2) cria-se um vetor "h", do tipo *filtered classifier* (classificador baseado em filtros) com dimensão *q* (número de rótulos de classe). Cada índice de "h" irá referenciar um classificador binário a ser treinado para um *label* específico. Inicialmente, associa-se um classificador NB "vazio" a cada posição, ou seja, apenas instanciado, mas não treinado.

A Seção (i.3) é a maior e mais importante do código. Nesta seção, é implementado um laço com *q* iterações. Um vetor denominado "labelsToRemove[]" de dimensão *q-1* é criado. Este recebe os índices de todos os rótulos de classe que deverão ser removidos para a criação de um classificador binário específico. Por exemplo, a primeira iteração é responsável por criar o classificador binário para o primeiro rótulo de classe, "red". Sendo assim, nesta iteração, o vetor "labelsToRemove" recebe os índices de todos os outros rótulos ("green", "blue", "yellow", "white", "black" e "orange"). Esse vetor é então utilizado em um filtro do tipo "Remove" (um filtro é um recurso da Weka

API para transformar bases de dados em memória) para possibilitar com que, no primeiro elemento "h" seja treinado um classificador binário para o label "red". Mais especificamente, o filtro remove todos os outros rótulos, exceto "red" e treina o classificador considerando apenas os 19 atributos preditivos da base "flags-train.ARFF". Nas iterações seguintes, o mesmo processo é realizado para os labels restantes. Para maiores informações sobre o uso de filtros na Weka, consulte [Weka, 2018a; Weka, 2018b].

Terminado o treinamento, o processo de classificação de novos objetos com o modelo BR + NB é trivial, como mostra a Figura 3.19. Para cada novo objeto, basta implementar um laço com *q* iterações que acionará cada classificador binário. A classificação multirrótulo final é obtida através da união das saídas geradas por todos os *q* classificadores binários. Os resultados da classificação multirrótulo dos 65 objetos da base "flags-test.ARFF" são apresentados na tela.

R

As Figuras 3.20 e 3.21 apresentam o *script* que reproduz na linguagem R os mesmos passos executados pelo programa Java recém-apresentado. Para utilizá-lo, é necessário converter as bases "flags-train.ARFF" e "flags-teste.ARFF" para o formato CSV. O programa assume que os arquivos CSV estão localizados na pasta "c:\cmr".

```
#carrega o package "e1071"
library(e1071);

# (i) Construção do Classificador BR

# (i.1) importacao da base de dados de treinamento
D <- read.table("c:\\cmr\\flags-train.csv",header=TRUE,sep=",")


#(i.2) cria uma lista h[] de classificadores NB (um para cada label)
h = list()
q = 7
d = ncol(D) - q

for (yj in 1:q) {
  #converte o atributo classe para "factor"
  D[,ncol(D)-q+yj] = factor(D[,ncol(D)-q+yj])

  #monta a formula para treinar o classificador binário yj
  aux = paste(names(D[ncol(D)-q+yj]),'~',names(D[1]))
  for (x in 2:d) {
    aux = paste(aux,"+",names(D[x]))
  }
  print(aux) #apenas para visualizar a formula na tela...

  f = as.formula(aux)
  h[[length(h)+1]] <- naiveBayes(f, data=D)
}
```

Figura 3.20. BR.R (parte 1 – etapa de treinamento): *script* R que implementa um classificador multirrótulo BR + NB

```
# (ii) Classificação dos Novos Objetos


#(ii.1) importação da base de novos objetos
novos <- read.table("c:\\cmr\\flags-test.csv",header=TRUE,sep=",")


#(ii.2) classificação dos novos objetos
preds = list()
for (yj in 1:q) {
  preds[[length(preds)+1]] <- predict(h[[yj]],novos[,1:d])
}


#(ii.3) imprime as classificações para todos os novos objetos
n = nrow(novos)
for (i in 1:n) {
  Z=""
  for (yj in 1:q) {
    Z = paste(Z,preds[[yj]][i])
  }

  print(Z)
}
```

Figura 3.21. BR.R (parte 2 – etapa de classificação): *script* R que implementa um classificador multirrótulo BR + NB

A ideia utilizada no *script* R é basicamente a mesma utilizada no programa Java. Na etapa de treinamento (Figura 3.20), cria-se uma lista de classificadores binários chamada "h" de tamanho *q*. Cada posição da lista é associada a um classificador NB treinado para um rótulo de classe específico (na posição 1, classificador para a classe "red", na posição 2 para a classe "green" e assim por diante). Em seguida, na etapa de classificação (Figura 3.21), os novos objetos são importados. A classificação multirrótulo de todos esses objetos é executada em apenas 4 linhas, no código apresentado na seção (ii.2). Mais uma vez, implementa-se um laço com *q* iterações para que todos os classificadores binários sejam acionados e realizem a classificação de cada rótulo. Fechando o programa, na seção (ii.3), as classificações multirrótulo são impressas na tela.

3.4. Considerações Finais

Este capítulo apresentou os conceitos fundamentais sobre classificação multirrótulo e uma noção geral dos problemas que esta tecnologia ajudou e está ajudando a solucionar, tanto em ambiente acadêmico como nas empresas. Sem ter a pretensão de cobrir toda a área, o texto teve como objetivo principal apresentar o conteúdo necessário para aqueles que pretendem começar a estudar o tema e que, futuramente, intencionam incorporar processos de classificação aos seus próprios sistemas de informação. A seguir, são sugeridos tópicos para pesquisas futuras, destinados aos que se interessarem em conhecer mais sobre CMR:

- O presente trabalho apresentou as três abordagens elementares para a construção de classificadores multirrótulo: LP, RPC e BR. No entanto, muitas outras técnicas importantes foram propostas ao longo dos últimos anos. O trabalho de Zhang and Zhou (2014) apresenta uma comparação entre as oito técnicas para CMR consideradas mais representativas na atualidade, ou seja, as técnicas mais citadas na literatura (entre elas, BR e ML-kNN). Já o artigo de revisão de Gibaja and Ventura (2014) propõe uma taxonomia que categoriza dezenas de métodos distintos. Conhecer algumas destas técnicas é importante para aqueles que desejam atuar na área de CMR.
- De maneira análoga, o texto apresentou apenas os métodos básicos para avaliar a qualidade de classificadores (*holdout* e validação cruzada) e introduziu algumas das medidas de desempenho mais comumente utilizadas. Informações detalhadas a respeito dos métodos *holdout*, validação cruzada e outros podem ser obtidas em [Flach, 2012; Han et al., 2016; Japkowicz and Shah, 2011; Witten et al., 2016]. Para um estudo detalhado sobre medidas de desempenho preditivo de classificadores multirrótulo, recomenda-se a referência [Pereira et al., 2018], onde uma análise de 16 diferentes medidas é realizada com o intuito de permitir com que usuários de sistemas CMR possam melhor entende-las.
- Ao longo do texto, foram apresentados exemplos simples de *scripts* para classificação multirrótulo elaborados em Java. Aqueles que desejam trabalhar com esta linguagem, poderão consultar as referências [Gonçalves, 2013; Read et al., 2016, Tsoumakas et al., 2011; Weka, 2018a, Weka, 2018b] para se aprofundar no tema.
- No caso da linguagem R, alguns pacotes específicos para CMR foram recentemente disponibilizados. Por exemplo, o pacote ‘mldr’ [Charte and Charte, 2015] fornece uma série de métodos para explorar e transformar bases de dados multirrótulo, permitindo a visualização de suas principais propriedades em tabelas e gráficos. Já o pacote ‘utiml’ [Rivolli, 2018] foi projetado com o objetivo de servir como uma plataforma para o desenvolvimento de protótipos e para a avaliação de classificadores multirrotulo (mesmo papel desempenhado pelas plataformas Mulan e Meka na linguagem Java).

Referências

Almeida, A. M. G., Cerri, R., Paraiso, E. C., Mantovani, R. G. and Junior, S. B. (2018). Applying multi-label techniques in emotion identification of short texts. In: Neurocomputing, 2018, preprint.

Boutell, M. R., Luo, J., Shen, X. and Brown, C. M. (2004). Learning multi-label scene classification. Pattern Recognition, v. 37, p. 1757−1771.

Briggs, F. et al. (2013). “The 9th Annual MSLP Competition: New Methods for Acoustic Classification of Multiple Simultaneous Bird Species in a Noisy Environment”. In: MLSP 2013, pages 1–8.

Charte, F. and Charte, D. (2015). Working with multi-label datasets in R: the mldr package. The R Journal, v. 7(2), p. 149−162.

Chen, L., Fan, C., Yang, H., Hu, S., Zou, L. and Deng, D. (2018). Face age classification based on a deep hybrid model. In Signal, Image and Video Processing, 12(8), pages 1531–1539

Cherman, E. and Monard, M. C. (2009) "Um Estudo sobre Métodos de Classificação Multirrótulo", In: Anais do IV Congresso da Academia Trinacional de Ciências, PTI, p. 1–10.

Diplaris, S.; Tsoumakas, G. and Mitkas, P. A. (2005) "Protein Classification with Multiple Algorithms", In: Advances in Informatics (LNCS 3476), Edited by P. Bozanis and E. Houstis, Springer, USA.

Elisseeff, A. and Weston, J. (2001). A kernel method for multi-labelled classification. In Advances in Neural Information Processing Systems, 14, pages 681–687.

Flach, P., Machine Learning: The Art and Science of Algorithms that Make Sense of Data, Cambridge University Press, 2012.

Frank, E., Hall, M. A. and Witten, I. H. (2016). The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, 4th ed., 2016. Disponível em: < https://www.cs.waikato.ac.nz/ml/weka/Witten_et_al_2016_appendix.pdf >. Acesso em: 25 set. 2018.

Gibaja, E. and Ventura, S. (2014). Multi-label learning: a review of the state of the art and ongoing research. In: WIREs Data Mining and Knowledge Discovery, 4(6), pages 411–444,Wiley.

Gibaja, E. and Ventura, S. (2015). A tutorial on multi-label learning. In: ACM Computing Surveys (CSUR), 47(3), pages 52:1–52:38, ACM.

Godbole, S. and Sarawagi, S. (2004). Discriminative methods for multi-label classification. In: Advances in Knowledge Discovery and Data Mining, 3056, pages 22–30.

Gonçalves, E. C. (2013). Mineração de dados na prática com a weka api. In: SQL Magazine, 107, pages 32–40, Devmedia.

Gonçalves, E. C., Freitas, A. A. and Plastino, A. (2018) "A Survey of Genetic Algorithms for Multi-Label Classification", In: Proc. of the 2018 IEEE Congress on Evolutionary Computation, IEEE, p. 981–988.

Gonçalves, E. C., Plastino, A. and Freitas, A. A. (2013) "A Genetic Algorithm for Optimizing the Label Ordering in Multi-Label Classifier Chains", In: Proc. of the 2013 IEEE 25th Int'l Conf. on Tools with Artificial Intelligence, IEEE, p. 469–476.

Gonçalves, E. C., Rabelo, F. A., Souza, I., Oliveira, M. R., Pereira, M. A. L., Araújo, R. B. S. (2006) "Construção de um Data Mart para a Análise dos Hábitos de Compra das Famílias Brasileiras", In: Anais do III Simpósio Mineiro de Sistemas de Informação, PUC-MG, p. 1–9.

Han, J., Kamber, M. and Pei, J., Data Mining: Concepts and Techniques, 3rd ed., Morgan Kaufmann, 2011.

Huang, Y-H., Hung, C-M. and Jiau, H. C. (2015). A multi-label approach using binary relevance and decision trees applied to functional genomics. In: Journal of Biomedical Informatics, v. 54, p. 85−95.

Hüllermeier, E., Fürnkranz, J., Cheng, W. and Brinker, K. (2008). Label ranking by learning pairwise preferences. In: Artificial Intelligence, 172(16-17), p. 1897−1916.

Japkowicz, N. and Shah, M., Evaluating Learning Algorithms: A Classification Perspective, Cambridge University Press, 2011.

Klimt, B. and Yang, Y. (2004) "The Enron Corpus: A New Dataset for Email Classification", In: Proc. of the ECML 2004, Springer, p. 217–226.

Kuhn, M. (2018). Package 'caret'. Disponível em: <https://cran.r-project.org/web/packages/caret/caret.pdf>. Acesso em: 25 set. 2018.

Lander, J. P. R for Everyone. 2nd ed., Addison-Wesley, 2017.

Madjarov, G., Kocev, D., Gjorgjevikj, D. and Džeroski, S. and Brinker, K. (2012). An extensive comparison of methdos for multi-label learning. In: Pattern Recognition, 45(9), p. 3084−3104.

Marsland, S., Machine Learning: An Algorithmic Perspective, 2nd ed., CRC Press, 2015.

Meyer et al. (2018). Package 'e1071'. Disponível em: <https://cran.r-project.org/package=e1071 >. Acesso em: 25 set. 2018.

Pereira, R. B., Plastino, A., Zadrozny, B. and Merschmann, L. H. C. (2018). Correlation analysis of performance measures for multi-label classification. In: Information Processing and Management, 54(3), pages 359–369.

Pestian, J. P. et al. (2007). "A Shared Task Involving Multi-Label Classification of Clinical Free Text". In: Proc. of the Workshop on BioNLP 2007, pages 87–102.

R Core Team (2017). R: A Language and Environment for Statistical Computing. Disponível em: <https://www.R-project.org/>. Acesso em: 25 set. 2018.

Read, J., Pfahringer, B., Holmes, G. and Eibe, F. (2011). Classifier chains for multilabel classification. In Machine Learning, 85(3), pages 333–359.

Read, J., Reutemann, P., Pfahringer, B., Holmes, G. (2016). Meka: a multi-label/multi-target extension to Weka. In Journal of Machine Learning Research, 17, pages 1–5.

Rivolli, A. (2018). Package 'utiml'. Disponível em: <https://cran.r-project.org/web/packages/utiml>.

da Silva, P. N. et al. (2015). Automatic classification of carbonate rocks permeability from 1H NMR relaxation data. In: Expert Systems with Applications, 42(9), pages 4299–4309.

Trohidis, K., Tsoumakas, G., Kalliris, G. and Vlahavas, I (2011). Multi-label classification of music by emotion. EURASIP J. Audio Speech Music Process., 2011:4.

Tsoumakas, G., Katakis, I., and Vlahavas, I. (2008) "Effective and Efficient Multilabel Classification in Domains with Large Number of Labels", In: Proc. of the ECML/PKDD 2008 Workshop on Mining Multidimensional Data, Springer.

Tsoumakas, G., Katakis, I., and Vlahavas, I. (2010) "Mining Multi-Label Data", In: Data Mining and Knowledge Discovery Handbook, Edited by O. Maimom and L. Rokach, Springer, USA.

Tsoumakas, G. and Vlahavas, I. (2007) "Random k-Labelsets: An Ensemble Method for Multi-Label Classification", In: Proc. of the ECML 2007, Springer, p. 17–21.

Tsoumakas, G., Xioufis, E. S., Vilcek, J. and Vlahavas, I. P. (2011). Mulan: A java library for multilabel learning. In Journal of Machine Learning Research, 12, pages 2411–2414.

Tsoumakas, G. et al. (2014). "WISE 2014 Challenge: Multi-label Classification of Print Media Articles to Topics", In: Proc. of the Int'l Conf. on Web Information Systems Engineering, Springer-Verlag, p. 541–548.

Turnbull, D., Barrington, L., Torres, D. and Lanckriet, G. (2008). Semantic annotation and retrieval of music and sound effects. In IEEE Transactions on Audio, Speech, and Language Processing, 16(2), pages 467–476.

Ururahy, J. M. K., Bastos, J. L. G. V., Federici, A. C. M., Gonçalves, E. C. (2018) "Classificação Multirrótulo de Documentos Texto Utilizando a Relevância Binária e o Algoritmo Naïve Bayes", In: Anais do III Seminário de Estatística com R, Galoá.

Weka Javadoc (2018). Disponível em: <http://weka.sourceforge.net/doc.stable/>. Acesso em: 19 set. 2018.

Weka Wiki (2018). Using weka in your java code. Disponível em: <https://waikato.github.io/weka-wiki/use_weka_in_your_java_code/>. Acesso em: 19 set. 2018.

Witten, I., Frank, E., Hall, M. and Pal, C. Data Mining: Practical Machine Learning Tools and Techniques. 4th ed., Morgan Kaufmann, 2016.

Xia, X., Feng, Y., Lo, Z., Chen, Z. and Wang, X. (2014) "Towards More Accurate Multi-Label Software Behavior Learning", In: Proc. of the 2014 CSMR-WCRE, IEEE, p. 134–143.

Xiao, J., Ehinger, K. A., Hays, J., Torralba, A. and Oliva, A. (2016). SUN database: Exploring a large collection of scene categories. In: International Journal of Computer Vision, 116(1), pages 3–22.

Zhang, W., Liu, F., Luo, L. and Zhang, J. (2015). Predicting drug side effect by multi-label learning and ensemble learning. In: BMC Bioinformatics, 16(365), pages 1–11.

Zhang, M.-L. and Zhou, Z.-H. (2007). ML-kNN: A lazy learning approach to multi-label learning. In: Pattern Recognition, 40(7), pages 2038–2048.

Zhang, M.-L. and Zhou, Z.-H. (2014). A review on multi-label learning algorithms. In: IEEE Transactions on Knowledge and Data Engineering, 26(8), pages 1819–1837.

Autor

Eduardo Corrêa Gonçalves cursou Doutorado em Ciência da Computação pela Universidade Federal Fluminense (UFF) com período de sanduíche na University of Kent, no Reino Unido. Atualmente, trabalha como desenvolvedor de banco de dados no Instituto Brasileiro de Geografia e Estatística (IBGE) e como professor colaborador na Escola Nacional de Ciências Estatísticas (ENCE/IBGE). Atua principalmente nas seguintes linhas de pesquisa: Big Data, Banco de Dados, Algoritmos e Processamento de Linguagem Natural. Endereço do Currículo Lattes:

http://buscatextual.cnpq.br/buscatextual/visualizacv.do?id=K4137241P3

Chapter

4

(Auto) Sintonia-fina em Sistemas de Bancos de Dados nas Organizações

Ana Carolina Brito de Almeida, Sérgio Lifschitz

Abstract

In a vast quantity of data (Big Data), organizations are increasingly demanding agility in processing database queries. In this mini-course we intend to address the issue of the database (self) tuning in the organizations. Database tuning is a complex task that improves database performance. This technique demands deep and specific knowledge on architecture, parameters and design of the database system. The course is intended for presentation on two main topics: an overview of the (self) tuning process in Relational Database Management Systems (DBMSs) and Applications of self (tuning), in practice, by organizations.

Resumo

Na atual era de grandes volumes de dados (big data), as organizações estão demandando cada vez mais a agilidade no processamento de requisições aos bancos de dados. Neste minicurso pretendemos abordar o tema de (auto) sintonia fina (self-tuning) em sistemas de bancos de dados nas organizações. A sintonia fina de banco de dados é uma tarefa complexa que visa melhorar o tempo de resposta de operações realizadas na base de dados. Essa técnica exige grande conhecimento da arquitetura, parâmetros e projeto do sistema de banco de dados. O curso está pensado para apresentação sobre dois tópicos principais: uma visão geral do processo de (auto) sintonia em Sistemas de Gerência de Bancos de Dados (SGBDs) Relacionais e Aplicações de auto (sintonia), na prática, pelas organizações.

4.1. Introdução

Os sistemas de informações estão cada vez mais complexos e gerando grandes volumes de dados (*big data*) para serem armazenados em SGBDs. *Big Data* é definido por três

V's: volume, variedade e velocidade [Berman 2013]. O volume refere-se a grande quantidade de dados; a variedade significa que tais dados podem vir em formatos diferentes, incluindo as bases de dados tradicionais, documentos, imagens e registros considerados complexos; e por fim, a velocidade indica que o conteúdo dos dados é atualizado constantemente, através da absorção de coleções de dados complementares, dados previamente arquivados, dados legados e dados de diversas fontes.

Diante disso, torna-se imprescindível para os sistemas de informações críticos obterem um bom desempenho do SGBD com respostas ágeis e precisas sobre esse grande volume de dados. A tarefa de *tuning* de bancos de dados ou sintonia fina surgiu com o objetivo de atingir o bom desempenho, buscando diminuir o tempo de resposta das consultas submetidas aos bancos de dados bem como aumentar a vazão (*throughput*), ou seja, aumentar o número de consultas executadas no mesmo período de tempo [Shasha and Bonnet 2003].

É necessário ressaltar a diferença entre a tarefa de sintonia fina de banco de dados e a tarefa de otimização de consultas, em particular no contexto de SGBDs relacionais. A otimização de consultas é o processo realizado por um SGBD para converter um comando declarativo, escrito, por exemplo, na linguagem SQL (*Structured Query Language*), em um plano de execução. Nesse plano, são especificados todos os operadores que são aplicados aos dados e de que forma eles são ordenados e compostos. Assim, a otimização de consultas é um processo automático, realizado por componentes do SGBD. Já a atividade de sintonia fina é realizada, na maioria das vezes, pelo administrador do banco de dados (DBA - *DataBase Administrator*) ou por um especialista em ajuste de desempenho.

A sintonia fina descreve uma série de atividades que podem ser realizadas para otimizar o desempenho dos bancos de dados, sendo considerada uma tarefa complexa. Realizam-se ajustes das configurações dos bancos de dados, parâmetros e projeto físico, seleção de estruturas de acesso, duplicação de estruturas físicas, sempre de acordo com a carga de trabalho submetida ao banco. Entende-se carga de trabalho como sendo um conjunto de consultas ou atualizações, ponderadas por suas frequências de ocorrência.

Por ser uma tarefa complexa, torna-se inviável, em muitos casos, monitorar um sistema de banco de dados, tomando decisões em tempo real, sem apoio de ferramentas especialistas. É importante que tais ferramentas sejam capazes de adaptar-se dinamicamente às modificações da carga de trabalho [Bruno 2011] [Lifschitz et al. 2004]. Diante dessa necessidade de automação, surge o conceito da auto-sintonia de banco de dados, sintonia fina automática de banco de dados ou *self-tuning* de banco de dados. Auto-sintonia de banco de dados significa o ajuste automático de configurações que buscam melhorar o desempenho através da diminuição do tempo de resposta sobre as execuções de transações ou comandos submetidos ao banco de dados. Praticamente todos os SGBDs de mercado oferecem apoio de sintonia automática, mas poucos são os profissionais que realmente conhecem as ferramentas ou mesmo têm condição de utilizá-las. Existem diversas propostas de ferramentas que auxiliam a sintonia fina através da automatização de estratégias para a seleção de índices e visões materializadas, particionamento, reescrita de consultas, entre outras.

O estudo da auto-sintonia de sistemas computacionais não é recente, como pode-se constatar em [Ferrari, 1978]. Entretanto, a sua relevância para a área de banco de

dados e sistemas de informações aumentou significativamente nos últimos anos. Espera-se que, no futuro, os sistemas demandem de uma quantidade menor de profissionais especializados para a manutenção de seu desempenho e de sua operação. Em um cenário ideal, os sistemas conseguiriam alcançar um desempenho adequado sem qualquer necessidade de ajuste manual. Mais do que isto, à medida que as cargas de trabalho submetidas ao sistema mudassem, este iria procurar se adaptar dinamicamente para continuar apresentando um desempenho aceitável.

Além disso, o próprio desenvolvedor de sistemas pode contribuir para a melhoria do desempenho, com conhecimentos de boas práticas em sintonia fina, não precisando aguardar por uma intervenção do administrador do banco de dados. Por exemplo, o desenvolvedor pode realizar reescrita das consultas que são elaboradas, de forma automática, por *frameworks* de desenvolvimento.

Este minicurso está organizado da seguinte maneira:

- ✓ **Introdução e visão geral de (auto) sintonia fina de bancos de dados:** apresentação dos principais conceitos envolvidos, em particular processamento e otimização de consultas e atividades de sintonia fina (*tuning*) de bancos de dados. Pretende-se abordar vários aspectos de sintonia fina, porém dando ênfase ao *tuning* de projeto físico.

- ✓ **Aplicações de auto (sintonia) nas organizações:** apresentação de ferramentas e alguns exemplos práticos, ilustrando diversas situações reais e frequentemente encontradas no dia a dia de uma empresa. Estes exemplos envolvem cenários de otimizações de consultas, demonstrando o custo e desempenho de consultas antes e após a sua reescrita de forma simples. Além disso, sugere-se boas práticas para serem adotadas tanto por equipes de desenvolvimento de sistemas de informação quanto por equipes de administradores de bancos de dados. Destaca-se o cuidado que os desenvolvedores precisam ter ao adotar a utilização de *frameworks* que elaboram consultas de forma automática para serem submetidas aos bancos de dados.

O presente minicurso objetiva complementar e aprofundar os conhecimentos de estudantes e profissionais interessados na área de banco de dados e em sistemas de informações que fazem uso de bancos de dados. Embora a ênfase seja dada no modelo relacional, diversos assuntos abordados e discutidos se aplicam também a outros modelos de dados.

4.2. Processamento e Otimização de Consultas

O SGBD é um sistema considerado complexo e que consiste de muitos módulos. Entre esses módulos inclui-se o módulo de implementação do catálogo, do processador de linguagem de consultas, da interface, do controle de concorrência, de segurança e recuperação entre outros [Elmasri and Navathe 2016].

Para contextualizar o presente minicurso, descreve-se o módulo do processador de linguagem de consultas, pois é o mais impactante em termos de sintonia-fina, no escopo de projeto físico do sistema de banco de dados. Nesse módulo que os comandos

DML (*Data Manipulation Language* – Linguagem de Manipulação de Dados) submetidos aos bancos de dados são avaliados e um plano de acesso e execução é gerado.

O processo de avaliação de uma consulta, de uma forma geral, é composto por cinco etapas (Figura 4.1):

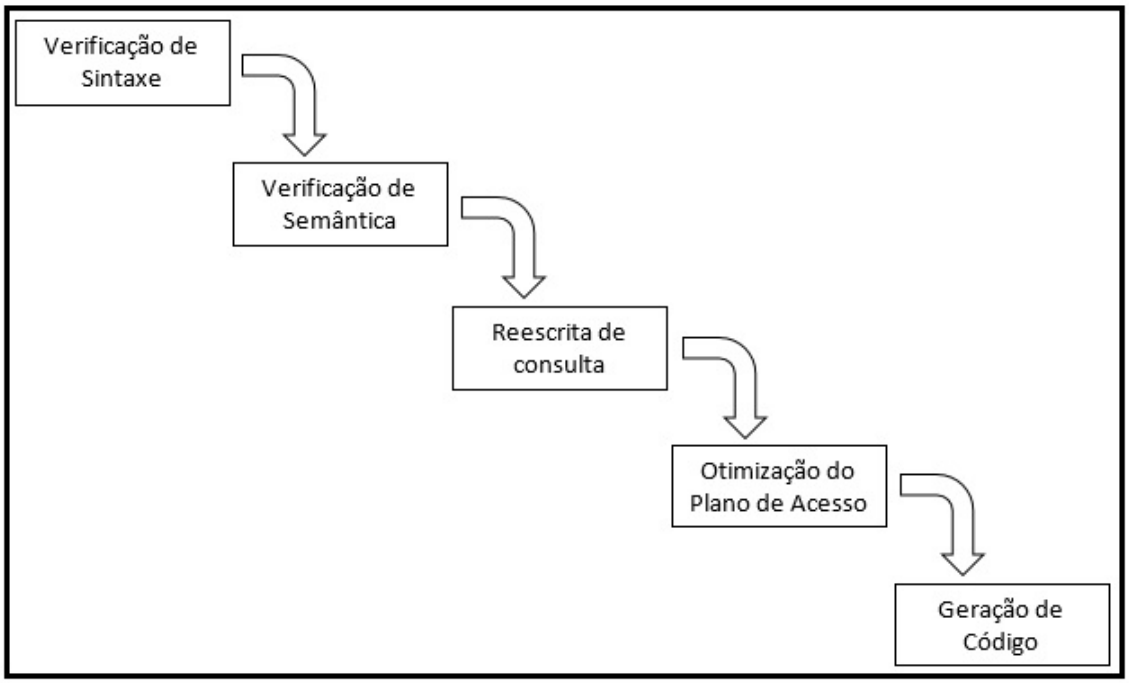


Figura 4.1 Etapas do processo de avaliação de consultas

- ✓ **Verificação de sintaxe** – verifica-se a sintaxe da consulta e detecta-se os erros sintáticos baseados nos *tokens* conhecidos (ex.: SELECT, FROM) pelo SGBD em sua gramática;
- ✓ **Verificação de semântica** – verifica se os objetos referenciados no comando SQL existem e se estão acessíveis para o usuário que submeteu o comando;
- ✓ **Reescrita de consulta** – o comando SQL é reescrito a partir de uma linguagem em alto nível (SQL) para uma representação mais adequada para sua manipulação interna (Álgebra relacional).
- ✓ **Otimização do plano de acesso** – para a execução do comando SQL, o SGBD deve possuir sua representação em unidades mais simples, as quais ele seja capaz de mapear, isoladamente, para suas rotinas básicas. Uma combinação destas unidades, em uma determinada ordem de execução, é chamada de plano de acesso e execução. Usualmente, vários planos de acesso e execução levam ao resultado desejado, mas, quando executados, possuem diferentes tempos de execução. Ao SGBD cabe montar o plano de acesso e execução que seja mais eficiente. De acordo com o número de tabelas utilizadas e a complexidade da consulta (que pode incluir várias subconsultas, por exemplo), o espaço de busca pode ser bastante grande. É importante que o tempo de otimização da consulta

não seja superior ao tempo que pode ser ganho com a escolha de um plano mais adequado. A otimização só vale a pena se o custo associado da execução do plano selecionado for melhor do que a execução de um plano qualquer escolhido aleatoriamente;

- ✓ **Geração de código** – o plano de acesso e execução selecionado na etapa anterior é transformado para chamadas às rotinas básicas do SGBD, as quais são executadas pelo mesmo.

Nesse ponto cabe observar que é possível mostrar que o processo de otimização de consultas em um banco de dados relacional é NP-difícil no número de relações envolvidas. Assim, sabe-se que, na prática, os SGBDs relacionais não determinam a solução ótima, mas sim, a melhor solução possível baseada em heurísticas.

4.3. Desempenho do Sistema de Banco de Dados

[Shasha and Bonnet 2003] enumeram cinco princípios básicos que permeiam ao considerar desempenho (Figura 4.2). São eles: (i) Pense globalmente, corrija localmente; (ii) Particione para quebrar os gargalos; (iii) Custos iniciais são altos, custos de execução são baixos; (iv) Deixe no servidor o que é do servidor; (v) Esteja preparado para conflitos em escolhas (*Trade-Offs*).

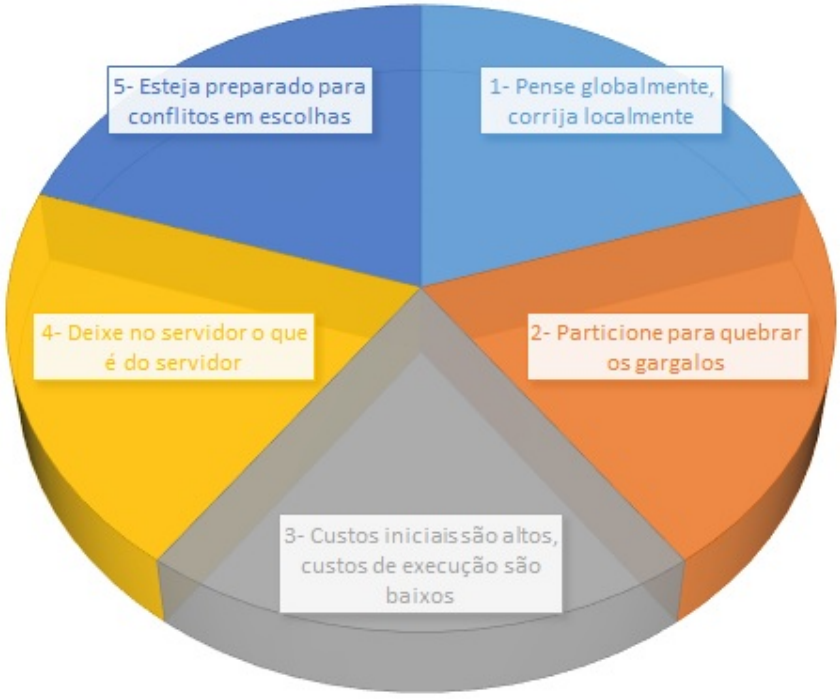


Figura 4.2 Princípios básicos – Desempenho do banco de dados

O primeiro princípio alerta que poucas intervenções simples, se aplicadas corretamente, podem aumentar o desempenho de forma considerável no sistema de banco de dados como um todo. Um exemplo clássico, citado por [Shasha and Bonnet 2003], é que o DBA pode analisar as estatísticas de *hardware* para verificar a utilização do processador, a atividade de entrada/saída (I/O), paginação entre outros itens. Caso o

valor de requisições ao disco seja alto, o DBA pode optar por comprar mais disco rígido para diminuir esse valor. No entanto, essa decisão pode não resolver o problema. A causa do problema pode ser uma consulta, frequentemente executada, que realiza muitas varreduras em uma tabela. Nesse caso, bastaria que o DBA criasse um índice ou movesse os arquivos de dados para discos diferentes. Essa solução seria mais barata e mais efetiva do que comprar mais *hardware*.

O segundo princípio ressalta que quando o DBA identificar um gargalo no banco de dados, deve tentar primeiro acelerá-lo. Caso não consiga, deve particioná-lo. Esse particionamento pode ser por espaço, recurso lógico ou por tempo. Por exemplo, o particionamento de recursos físicos por espaço pode ser útil em um cenário de um banco nacional com várias agências espalhadas pelo Brasil. Considerando que a maioria dos clientes acessam os dados de sua conta bancária próximo ao local de onde reside, uma solução natural seria colocar os dados dos clientes em um local L dentro de um subsistema L. Esse local L poderia ser cada região do Brasil (Sudeste, Sul, Nordeste etc), onde existiria um servidor de banco de dados com dados sobre contas bancárias somente da sua região. Dessa forma, os clientes estariam próximos aos dados das suas contas bancárias, não impactando ou concorrendo com consultas de outros clientes do restante do Brasil.

O terceiro princípio indica o uso de menos recursos de inicialização, considerado mais caro. Por exemplo, é muito comum o cenário onde um programador desenvolve um sistema de informação que acessa o banco de dados. Nesse caso, é melhor que o programador opte por uma única chamada de conexão ao banco de dados, se o comando de consulta ao banco de dados envolver o mesmo conjunto de dados e posteriormente, crie um bloco de comando de repetição para percorrer o conjunto resultado da consulta. Essa solução é melhor do que criar um bloco de repetição, onde em cada interação é aberta uma conexão com o banco de dados e cada chamada tenha o seu próprio comando de consulta. Imaginando que o comando de consulta retorne dez milhões de linhas, é melhor submeter o comando uma única vez ao banco de dados e posteriormente, percorrer o seu conjunto de linhas resultante do que realizar dez milhões de chamadas ao banco de dados.

O quarto princípio visa o equilíbrio das tarefas entre o servidor de banco de dados e os demais servidores (ex.: servidor de aplicação) e clientes. Por exemplo, é importante levantar onde reside a informação relevante. Imagina-se o seguinte cenário: um sistema de informação precisa de alguma resposta do banco de dados toda vez que ocorrer uma determinada mudança no registro do cliente para que seja disparado um e-mail para o mesmo. Dado que a informação necessária e que deve ser acompanhada reside no banco de dados, é melhor que seja criado um gatilho (*trigger*) no servidor do banco, que seja disparado toda vez que ocorrer alguma mudança do que um método no sistema de informação que fique consultando a tabela periodicamente com requisições ao banco. O método, no servidor de aplicação, pode submeter requisições ao banco sem necessidade. Já o gatilho só vai ser disparado quando, de fato, ocorrer alguma alteração nos dados da tabela de cliente.

O quinto e último princípio chama atenção para os possíveis impactos de uma decisão ou escolha. A melhoria de desempenho de um sistema requer a combinação de memória, disco ou recursos computacionais. Por exemplo, a criação de um índice para

agilizar uma determinada consulta requer mais espaço em disco e mais espaço na memória de acesso randômico. Além disso, em momentos de inserções e atualizações no banco de dados requer mais tempo de processamento e mais acessos ao disco, quando não usa o índice. Dessa forma, o DBA deve estar ciente sobre os conflitos de interesse em suas escolhas e decisões.

4.4. Sintonia-fina de Bancos de Dados

A sintonia-fina, no contexto deste minicurso, envolve a realização de ajustes em sistemas de banco de dados de forma a obter um tempo de resposta menor e/ou aumentar a vazão para determinada aplicação. A tarefa de sintonia-fina é considerada difícil, pois requer conhecimento profundo da aplicação que utiliza os dados, do SGBD que gerencia os bancos de dados, do sistema operacional e da parte física do *hardware*.

Para realizar a tarefa de sintonia fina do banco de dados, de forma adequada, é essencial ter um bom entendimento dos planos de acesso e da maneira como o SGBD os elabora. É através desse plano que se pode verificar a validade de determinadas operações de sintonia fina, comparando-se os planos de acesso obtidos antes e após suas realizações.

Os planos de acesso são montados a partir da utilização dos métodos de acesso e das operações de manipulação de dados. Para a escolha das operações a serem utilizadas e a elaboração dos planos de acesso, os SGBDs devem tomar algumas decisões. Tais decisões podem ser baseadas em regras ou em custos.

Os otimizadores baseados em regras utilizam um conjunto de regras para escolher a ordem das operações que devem realizar. Essas regras são hierarquizadas de forma a fazer com que as operações potencialmente mais restritivas sejam realizadas antes das demais. Por exemplo, o otimizador prioriza uma operação que busque por uma única linha baseada no identificador dessa linha do que uma operação que precise realizar uma varredura completa na tabela.

Já os otimizadores baseados em custo atribuem um valor de custo para cada operação a ser realizada. O plano escolhido é aquele que apresenta o menor custo total. Todos esses custos são calculados com o uso de informações extraídas das estatísticas armazenadas no metadado do banco. Tais estatísticas podem compreender informações referentes às tabelas como um todo (ex.: número de tuplas e de blocos e o tamanho médio de uma tupla), aos atributos (ex.: número de valores distintos em cada um) e aos índices (ex.: número de níveis). No entanto, nem sempre as estatísticas estão atualizadas no momento da geração do plano de acesso, podendo acarretar em um plano mal estruturado. Manter todas as estatísticas atualizadas apresenta um custo que, em ambientes com grande quantidade de atualização e elevado nível de concorrência por recursos, pode ser bastante alto. A geração e utilização das estatísticas de forma adequada são fatores que afetam diretamente o desempenho de um SGBD. Dessa forma, o DBA deve avaliar, cuidadosamente, quando atualizar as estatísticas.

Quando são utilizados otimizadores baseados em custos, esses custos atribuídos aos diversos planos podem ser usados como um parâmetro de comparação. Já quando o otimizador baseado em regras é utilizado, faz-se necessário maior conhecimento das várias operações que são realizadas e dos modos como podem ser reordenadas. Em

ambos os casos, deve-se avaliar as alternativas existentes, que vão desde a alteração de parâmetros de inicialização até a utilização de dicas, para induzir o SGBD a adotar um plano de execução que tenha melhor desempenho.

A sintonia fina em bancos de dados busca, na grande maioria das vezes, reduzir a quantidade de operações de E/S (Entrada e Saída), que são comparativamente lentas em relação às operações em memória principal, sendo a sua realização o principal gargalo no processamento de consultas e atualizações em SGBDs. Nesse sentido, pode-se realizar alterações que sejam pontuais, afetando um comando específico ou operações que apresentam maior abrangência, podendo favorecer ou prejudicar diversas operações do SGBD.

Pode-se dividir a tarefa de sintonia-fina de acordo com o componente que se deseja otimizar. São eles: consulta, transação, memória e projeto físico.

4.4.1. Sintonia-fina de consulta

A sintonia-fina de consulta é a menos impactante de todas, pois não onera outro recurso. Existem dois caminhos para detectar que uma consulta está com o desempenho lento no banco de dados: (i) a consulta está realizando muitos acessos ao disco, por exemplo, uma consulta que varre uma tabela inteira; (ii) se o plano de execução da consulta não estiver usando índices que são considerados relevantes para a sua execução.

Caso seja detectado esse tipo de lentidão, é necessário analisar se é possível reescrever a consulta de forma que o otimizador possa interpretá-la de outra forma e optar por um caminho mais barato para executá-la. Na etapa de aplicações de sintonia-fina apresenta-se alguns exemplos de melhorias de desempenho apenas com a reescrita de consultas.

4.4.2. Sintonia-fina de transação

O conceito de transação provê um mecanismo para descrever unidades lógicas de processamento do banco de dados [Elmasri and Navathe 2016]. Uma transação inclui uma ou mais operações de acesso ao banco de dados – operações de inserção, remoção, atualização ou recuperação.

A sintonia-fina de transação inclui atividades que gerenciam o controle concorrente das transações no banco de dados. Entre essas atividades estão: eliminar bloqueios quando são desnecessários através da modificação da ordem dos comandos que seguram os bloqueios; usar o nível de isolamento apropriado para as transações; usar as características de multi-versionamento no sistema para consultas longas e que apenas leem dados; gerenciar a granularidade dos bloqueios; analisar o intervalo de verificação de *deadlocks*.

4.4.3. Sintonia-fina de memória

Os SGBDs utilizam a memória para armazenar os dados mais recentemente acessados. Dessa forma, as operações de atualização que são realizadas por usuários podem fazer com que existam, em memória, várias versões da mesma informação. Além disso, a memória armazena informações sobre as árvores de execução de comandos SQL, sobre

a instância utilizada e a base de dados acessada e os processos do servidor e seus parâmetros de controle.

Os SGBDs utilizam diversas áreas da memória (ex.: buffer pool, redo log buffer). A sintonia-fina de memória busca dimensionar corretamente essas áreas de memória para aumentar desempenho. Usualmente, busca-se aumentar o *hit ratio*, ou seja, a taxa de acerto de se encontrar um dado necessário já na memória. A proporção do número de ocorrências é dividida pela referência total da memória da CPU e é chamada de *hit ratio*.

4.4.4. Sintonia-fina de projeto físico

O projeto físico do banco de dados descreve como os dados são armazenados como arquivos no computador através da representação da informação, tais como: formatos de registro, ordenação dos registros e caminhos de acesso [Elmasri and Navathe 2016]. Caminho de acesso é uma estrutura de busca que realiza a busca para determinados registros de forma mais eficiente (ex.: índices).

O problema do projeto físico é formalizado por [Bruno 2011] como:

> *Dado uma carga de trabalho **W** consistindo de consultas e atualizações e um limite de armazenamento **A**, obtenha a configuração de índices **C** que caiba em **A** e resulte em consultas na **W** executando tão eficientemente quanto possível.*

Assim, a sintonia-fina de projeto físico inclui a seleção das estruturas de índices adequadas, mas também as visões materializadas a serem utilizadas, as tabelas a serem particionadas e os tipos de particionamento mais adequados e até mesmo, a desnormalização de tabelas, como forma de acelerar o desempenho das consultas.

Índices são estruturas auxiliares que visam permitir o acesso mais rápido aos dados. Para isso, os índices são organizados de forma diferente da utilizada para os dados. Estes podem estar armazenados em diversos tipos de arquivos, sendo, os mais comuns, os arquivos de tipo *heap*, onde os dados estão armazenados em sequência aleatória, e ordenado, onde os dados são mantidos fisicamente ordenados por um conjunto de um ou mais atributos (ou colunas) de uma tabela.

As visões materializadas são réplicas dos dados originais em formatos mais adequados para a realização de consulta. Neste caso, tem-se que estar atento para o espaço em disco disponível e a atualização dos dados. Como visões materializadas são réplicas dos dados, a cada atualização dos dados originais as visões devem ser atualizadas também, gerando mais atividade na base de dados. Alguns SGBDs provêem mecanismos automáticos para a realização destas atualizações. Em alguns casos, porém, é desejável realizar a atualização das visões materializadas de forma assíncrona com a atualização das tabelas que a originam.

O particionamento é outra técnica que pode ser utilizada para reduzir a quantidade de operações de E/S. O particionamento horizontal, implementado como um recurso em muitos SGBDs, permite que sejam criados subconjuntos da tabela (ou

índice) original. Em consultas onde o atributo utilizado para realização da partição é utilizado como filtro, os otimizadores podem acessar somente a partição adequada, reduzindo o espaço de busca e a quantidade de operações de E/S. As partições podem ser alocadas, inclusive, em diferentes discos.

Outra forma de realizar a sintonia-fina de projeto físico é a desnormalização, pois a realização de junções pode gerar um problema de desempenho. Em algumas situações é aceitável recorrer a essa técnica no esquema do banco de dados. Isso ocorre, muitas vezes, em ambientes onde são realizadas poucas ou nenhuma operação de atualização de dados. É importante ressaltar que a desnormalização pode trazer vários problemas e sua realização requer uma avaliação detalhada dos impactos sobre as aplicações que acessam a base de dados.

4.5. Auto Sintonia-fina de Bancos de Dados

A auto sintonia fina de bancos de dados refere-se ao ajuste, de forma automática, dos parâmetros, configurações e estruturas de um SGBD com o objetivo de processar mais eficientemente uma determinada carga de trabalho.

Na literatura, existe um conjunto de princípios básicos para os componentes de auto sintonia. Alguns deles são:

- ✓ Definição do ambiente em que o componente de auto sintonia está inserido. Esse ambiente significa determinar o escopo em que a auto sintonia deve ser aplicada, ou seja, de forma local ou global. No escopo local, o objetivo do componente é automatizar uma atividade de sintonia específica e no escopo global, considera-se a sintonia do sistema de banco de dados como um todo.

- ✓ Modificação do sistema através de um ciclo de controle de realimentação. Um ciclo de controle de realimentação é um método para controlar o efeito das decisões de sintonia sobre o desempenho do sistema. O método se divide em três fases: observação, predição e reação. Na fase de observação ocorre a coleta de métricas e estatísticas que permitem avaliar se existe algum problema de desempenho em algum componente específico ou no sistema como um todo. A fase de predição consiste em estimar o desempenho futuro do sistema com base nas métricas coletadas e tomar decisões sobre quais ações podem ser executadas caso seja necessário sintonizar o sistema. Por fim, na fase de reação, o componente de auto sintonia implementa as ações de sintonia do sistema.

- ✓ Coleção de estatísticas e informações de forma que haja pouco impacto no desempenho do sistema. Determinar se há um problema de desempenho no sistema que possa ser resolvido através de ações de sintonia exige que seja realizado um diagnóstico. Este diagnóstico é baseado em um conjunto de métricas e estatísticas coletadas durante a operação do sistema.

- ✓ Utilização de modelos matemáticos e estimativas para prever o desempenho futuro do sistema. Um componente de auto sintonia precisa prever o desempenho futuro do sistema com base em métricas coletadas periodicamente. Os modelos formais sobre o desempenho do sistema podem ajudar com informações valiosas sobre o comportamento do mesmo.

✓ Realização de ajustes simples *on-line* e ajustes complexos *off-line*. A enumeração de ações possíveis para sintonizar o sistema e a sua execução podem, em algumas situações, impactar muito negativamente o desempenho do sistema. Portanto, realizar este tipo de computação *on-line*, enquanto o sistema processa requisições, pode ser impraticável. Para não ocorrer esse tipo de impacto negativo, pode-se registrar a necessidade de tomar a ação para posteriormente, executá-la em *off-line*, ou seja, em um momento em que o processamento de requisições do sistema não seja comprometido.

Os itens citados anteriormente servem para ilustrar os princípios que podem ser aplicados na construção de componentes de auto sintonia em geral. Outras técnicas relevantes podem ser consultadas em [Barrientos et al., 2004].

[Lifschitz et al. 2004] sugerem duas direções principais para a pesquisa em auto sintonia-fina: global e local em concordância com o primeiro item abordado durante a descrição dos princípios.

A auto sintonia-fina global possui como objetivo a construção de sistemas que possam, automaticamente, manter um equilíbrio entre os recursos usados pelos componentes do SGBD para atingir um bom desempenho e vazão como um todo.

A auto sintonia-fina local tenta solucionar problemas específicos de desempenho de banco de dados, tais como: seleção e gerenciamento de índices, localização dos dados físicos, políticas de substituição de páginas no buffer entre outros. Os trabalhos de auto sintonia local concentram os seus esforços em resolver os seguintes problemas:

✓ Projeto físico: o problema consiste em escolher a melhor organização física para um dado esquema e uma carga de trabalho específica.

✓ Alocação de dados: dados N elementos de armazenamento, devemos determinar como podemos alocar e realocar dinamicamente fragmentos de arquivos ou relações nestes elementos de tal forma que o equilíbrio de carga seja o melhor possível, mesmo diante de mudanças nos padrões de acesso da carga de trabalho.

✓ Controle de carga: em um sistema que processa múltiplas transações de forma concorrente e que se utiliza de bloqueios (*locks*) para garantia das propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade), o objetivo é regular o nível de multiprogramação do sistema de forma a evitar que a sua vazão (*throughput*) caia devido a conflitos de bloqueio (*thrashing* de contenção de sados).

✓ Substituição de páginas: o problema é manter em memória o conjunto de páginas mais populares do banco de dados, mesmo quando a carga de trabalho a que o sistema é submetido muda.

✓ Ajuste de *buffers*: procura-se estudar quantos *buffers* distintos devem ser configurados no banco de dados e de que forma devem ser distribuídos os objetos (tabelas e índices) para estes buffers para que o desempenho aumente.

✓ Refino de estatísticas: o objetivo é escolher quais estatísticas devem ser criadas no banco de dados e refinar estas estatísticas sem necessidade de executar comandos específicos para coleta no sistema.

4.6. (Auto) Sintonia-fina na Prática

Existem diversas ferramentas comerciais e acadêmicas para auxiliar o DBA na tarefa de (auto) sintonia-fina.

Pode-se considerar que as ferramentas de auto-sintonia de banco de dados envolvem três componentes principais: espaço de busca, modelo de custo e estratégia de enumeração [Bruno 2011].

✓ **Espaço de busca:** no contexto de sintonia fina de projeto físico, usando a estrutura de acesso de índice, considera-se como espaço de busca, o conjunto de índices candidatos para um determinado banco de dados e uma dada carga de trabalho. Em uma visão mais ampla, podemos considerar como sendo o conjunto de estruturas de acesso candidatas para uma base de dados de acordo com uma determinada carga de trabalho.

✓ **Modelo de custo:** é elaborado fora do otimizador. Ele avalia qual seria o custo de execução dos comandos que fazem parte de uma determinada carga de trabalho sobre uma configuração simulada durante a tarefa de sintonia fina. Quando os custos de execução são estimados sobre configurações hipotéticas, ou seja, que consideram estruturas sem estarem realmente materializadas, chamamos de camada de otimização *what-if*.

✓ **Estratégia de enumeração:** considera as limitações do ambiente em que o SGBD se encontra para enumerar configurações candidatas. Por exemplo, a estratégia pode considerar um espaço em disco limitado disponível para a criação de índices. Com a limitação de espaço físico, a estratégia pode simplificar o espaço de busca, passando a considerar apenas índices simples.

Considerando os resultados complexos da área e o aumento da sofisticação das máquinas de consultas e cargas de trabalho, as técnicas recentes de sintonia fina usam heurísticas para simularem configurações que sejam relevantes no espaço de busca analisado. Na próxima seção, cita-se alguns trabalhos que fazem uso de heurísticas.

4.6.1. Ferramentas

Existem diversas ferramentas acadêmicas (ex.: [Morelli et al. 2012], *DBX* [Almeida et al. 2015], *Outer-Tuning* [Almeida et al. 2018] e comerciais (ex.: *Automatic Database Diagnostic Monitor – ADDM* [Oracle 2018], *SQL Server Database Engine Tuning Advisor* [Microsoft 2017a]), que auxiliam o DBA a selecionar as melhores estruturas e parâmetros de configurações para o SGBD. Essas ferramentas também auxiliam com outras técnicas, como a reescrita de consulta, a desnormalização e a *clusterização* entre outras.

A *DBX* é uma ferramenta para a manutenção automática do projeto físico em bancos de dados relacionais baseada em planos de execução hipotéticos (Figura 4.3 e Figura 4.4) [Almeida et al. 2015]. Essa ferramenta é implementada na linguagem Java e totalmente desacoplada de qualquer SGBD específico. Ela realiza a manutenção de índices e visões materializadas, reagindo dinamicamente às alterações na carga de trabalho. O diferencial da ferramenta é que ela considera tanto índices completos quanto índices parciais, além de detectar a fragmentação de um índice e sugerir a sua recriação ou remoção. As visões materializadas também são acompanhadas durante todo o seu ciclo de vida, ou seja, quando deixam de proporcionar benefícios, a ferramenta sugere a sua remoção.

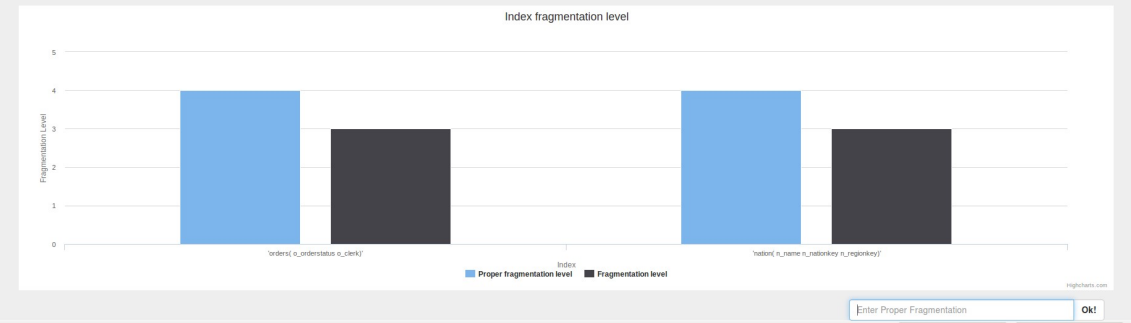


Figura 4.3 DBX – Gráficos sobre o nível de fragmentação de índices

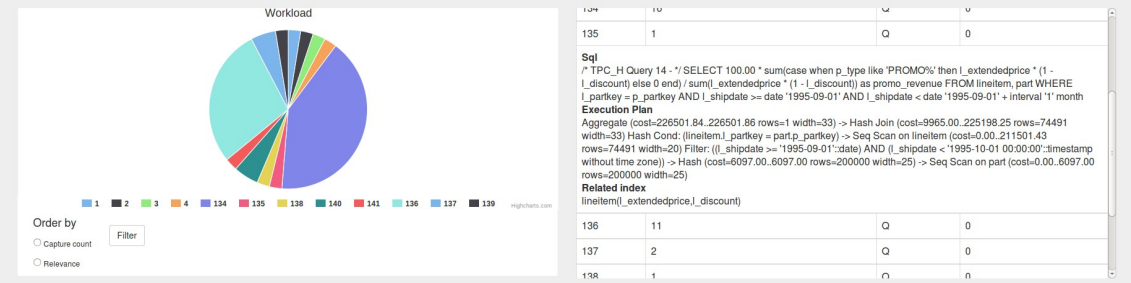


Figura 4.4 DBX – Detalhes sobre a carga de trabalho e índice sugerido para consulta

A *Outer-Tuning* é uma ferramenta que faz uso de uma ontologia de aplicação para apoiar a tarefa de (auto) sintonia-fina de banco de dados [Almeida et al. 2018]. Essa ferramenta pode ser instanciada ou estendida com qualquer heurística, desde que os conceitos utilizados por ela sejam definidos na ontologia de domínio. Atualmente, a ferramenta possui heurísticas que sugerem índices e visões materializadas e que podem ser aplicadas aos SGBDs PostgreSQL e Oracle.

Inicialmente, na Figura 4.5(A), o usuário do pode visualizar as heurísticas definidas na ontologia de tarefa e selecionar aquelas que deseja considerar para as futuras sugestões de sintonia fina. Posteriormente, o usuário informa para a ferramenta o modo que deseja trabalhar: semiautomático ou automático (sem intervenção humana). A ferramenta inicia a captura da carga de trabalho, em tempo real e apresenta, de forma gráfica, o momento em que o comando de consulta ou atualização de dados é executado no banco de dados e a sua duração, em segundos (Figura 4.5(B)). Caso o usuário queira detalhes maiores sobre a execução do comando, ele pode verificar mais abaixo na

mesma tela (Figura 4.5(C)). Caso o usuário queira acompanhar as ações de sintonia fina que estão sendo analisadas e sugeridas pela ferramenta, pode fazer isso através do menu *Tuning actions*. Nessa tela (Figura 4.5(D)), o usuário pode visualizar um gráfico com o cruzamento das informações de ganho esperado com a ação de sintonia fina (eixo x) e custo estimado de criação da estrutura de acesso (eixo y). O tamanho do círculo indicado no gráfico representa a quantidade de consultas SQL que a ação pode beneficiar. Quanto maior o tamanho do círculo, maior será o número de comandos beneficiados pela ação de sintonia fina na carga de trabalho. O *pop-up* apresentado é um resumo sobre a ação de sintonia fina proposta com as seguintes informações: ganho esperado, custo de criação, tipo de ação (ex.: índice ou visão materializada) e número de comandos beneficiados pela ação.

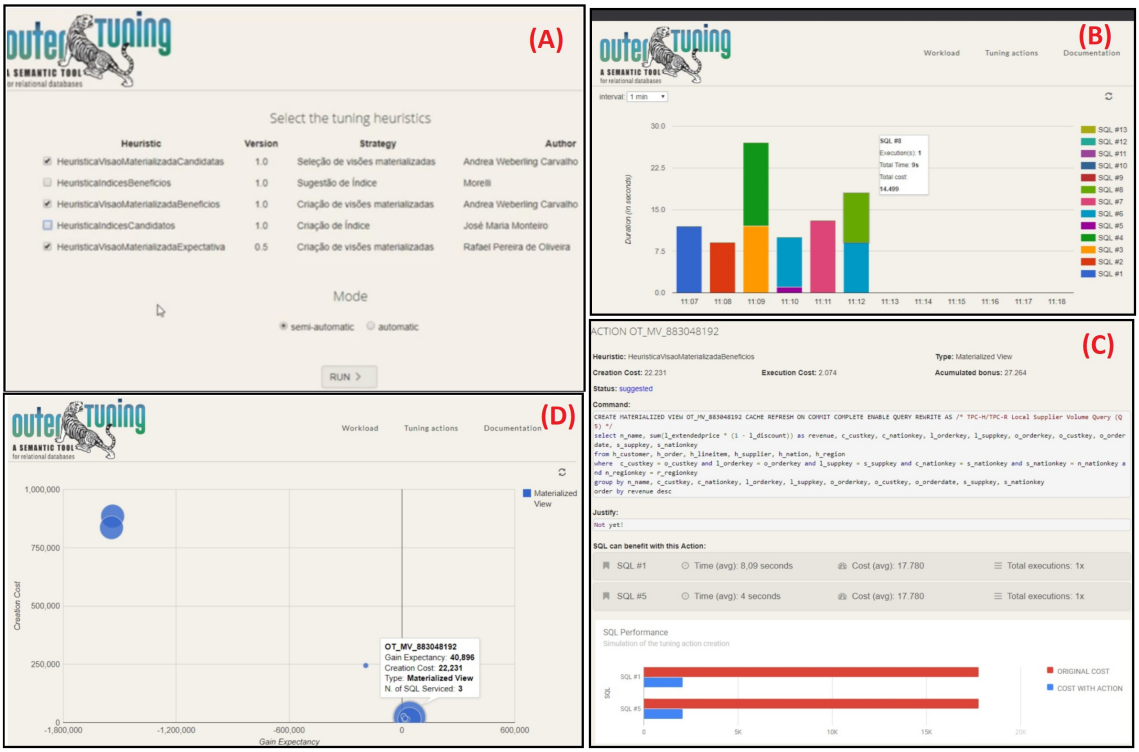


Figura 4.5 Outer-Tuning – Telas

As ferramentas automáticas de sintonia-fina oferecidas pela Oracle para o SGBD Oracle 18c são: *Automatic Database Diagnostic Monitor – ADDM*, *SQL Tuning Advisor*, *SQL Access Advisor*, *SQL Plan Management* e *SQL Performance Analyzer* [Oracle 2018].

A *ADDM* realiza uma análise *top down* de um instante particular do SGBD. Ela identifica os sintomas de desempenho para, então, refiná-los de forma que descubra as reais causas dos problemas de desempenhos levantados. Seu objetivo é sugerir automaticamente uma forma de sintonia-fina do SGBD, buscando diminuir o tempo perdido pelo servidor ao processar uma requisição do usuário, incluindo tempo de espera e CPU. Essa ferramenta provê recomendações de correção e quantifica os benefícios esperados. Na Figura 4.6 tem-se um exemplo de uma recomendação da ferramenta ADDM, indicando que, se a ação indicada for realizada, o ganho de tempo

estimado será de 31%. Além disso, a figura apresenta o raciocínio da ferramenta para sugerir a recomendação.

```
FINDING 1: 31% impact (7798 seconds)
-----------------------------------
SQL statements were not shared due to the usage of literals.
This resulted in additional hard parses which were consuming significant database time.
RECOMMENDATION 1: Application Analysis, 31% benefit (7798 seconds)
  ACTION: Investigate application logic for possible use of bind variables
    instead of literals. Alternatively, you may set the parameter
    "cursor_sharing" to "force".
  RATIONALE: SQL statements with PLAN_HASH_VALUE 3106087033 were found to be
    using literals. Look in V$SQL for examples of such SQL statements.
```

Figura 4.6 Exemplo de relatório gerado pela ferramenta ADDM

A razão da sugestão é a análise das expressões em linguagem SQL que foram submetidas e consideradas pela ferramenta, para que suas condições sejam trocadas para os tipos de variáveis *bind* (Figura 4.7) ao invés de literais (Figura 4.8).

Uso de variável *bind*

```
variavelNota := 5;
SELECT matriculaAluno, nomeAluno
FROM aluno WHERE notaAluno = variavelNota;
```

Figura 4.7 Exemplo de uso de variável *bind*

Uso de literal

```
SELECT matriculaAluno, nomeAluno
FROM aluno WHERE notaAluno = 5;
```

Figura 4.8 Exemplo de uso de literal

É recomendável utilizar variáveis de ligação (*bind*) ao invés de valores fixos (literais) porque podem existir, por exemplo, diversas consultas iguais com alteração somente de seus valores. Com isso, substituindo tais valores fixos por referência a variáveis, e informando o valor desejado, a cada vez, como atributo da variável, o texto do SQL não muda. Portanto, não haverá verificação sintática (processo seguido para verificação e busca do valor solicitado pela consulta) repetitiva e, consequentemente, maior custo de CPU e tempo.

O *SQL Tuning Advisor* (Figura 4.9) detecta comandos SQL considerados problemáticos e recomenda melhorias para o comando. Essa ferramenta verifica se as estatísticas estão obsoletas ou se não existem; constrói perfis do SQL, ou seja, um conjunto de informações específicas para o comando SQL; analisa se algum caminho de acesso alternativo pode melhorar o desempenho de forma significativa; e identifica comandos SQL que estão sendo executados com planos de execução abaixo do ideal. As recomendações relatam as estatísticas dos objetos, sugerem a criação de índices novos, a reestruturação do comando SQL ou a criação do perfil do SQL.

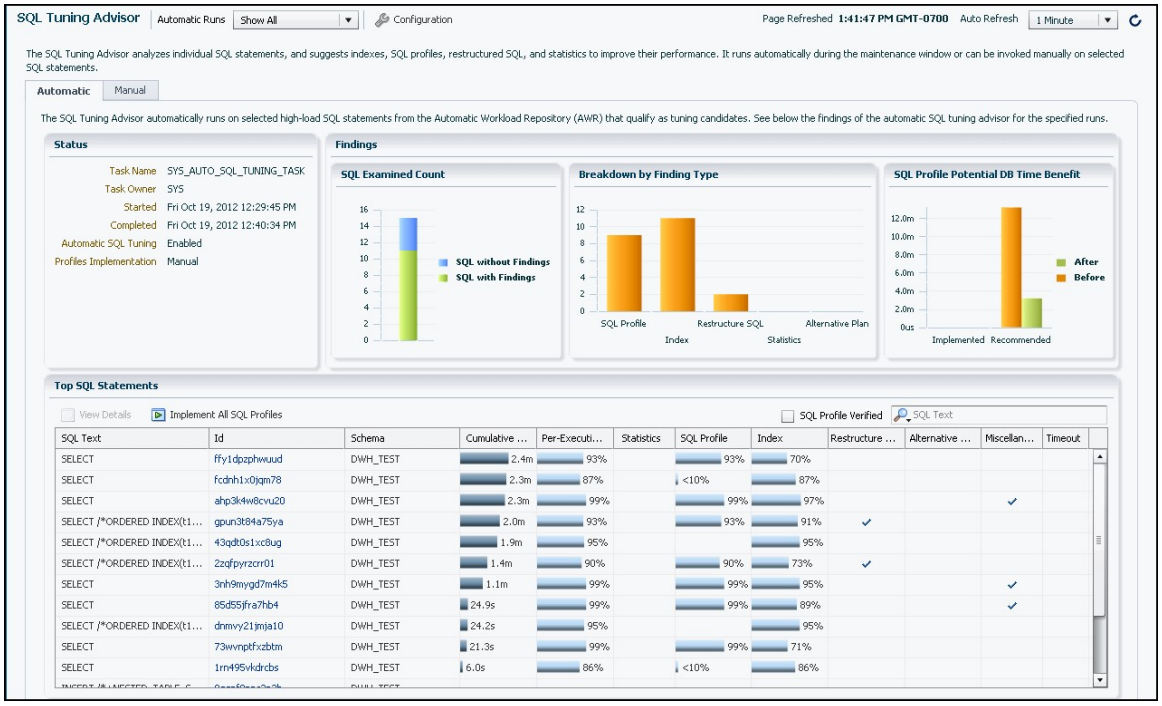


Figura 4.9 SQL Tuning Advisor – Tela [Oracle 2017]

O *SQL Access Advisor* analisa a carga de trabalho real ou pode atuar sobre uma carga de trabalho hipotética do esquema. Essa ferramenta analisa o conflito entre as possíveis escolhas (*trade-offs*). Ela avalia o custo-benefício entre o uso de espaço e o desempenho das consultas, recomendando a configuração de visões materializadas e índices (Figura 4.10).

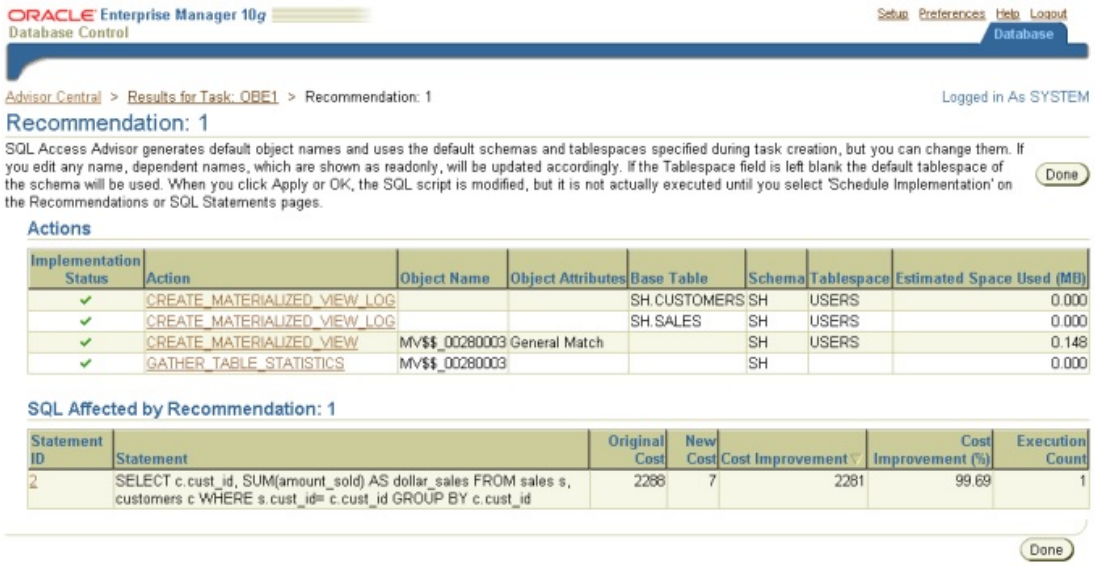


Figura 4.10 *SQL Access Advisor* - Exemplo da tela de recomendação [Oracle 2006]

O *SQL Plan Management* é um mecanismo de prevenção que permite que o otimizador gerencie os planos de execução, automaticamente, garantindo que o banco de dados usa somente planos conhecidos e já verificados (Figura 4.11).

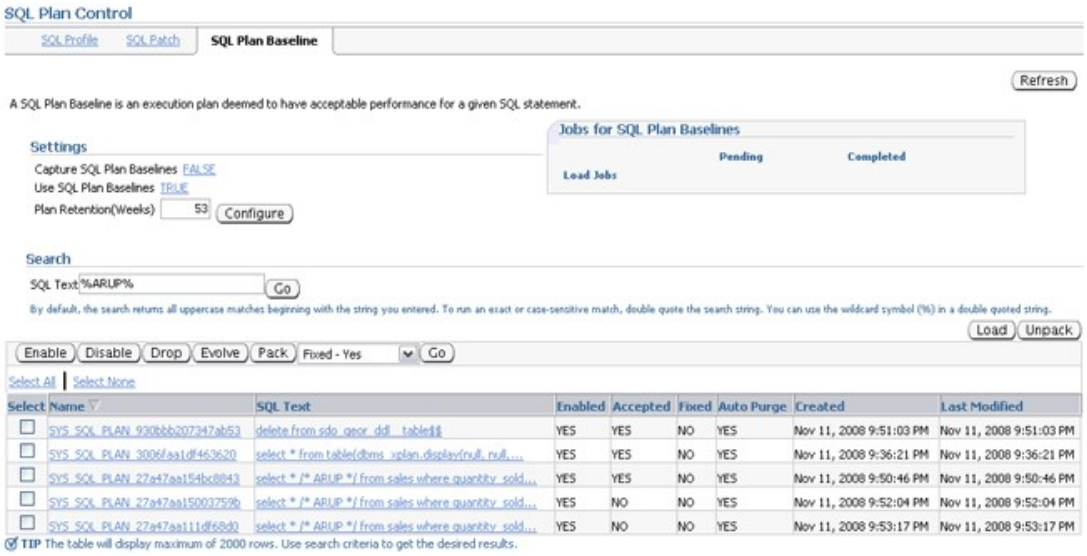


Figura 4.11 Baselines de planos de execução para serem usados pelo SQL Plan Management [Nanda 2009]

O *SQL Performance Analyzer* identifica o efeito de uma mudança na carga de trabalho SQL através da identificação de divergência de desempenho para cada comando SQL (Figura 4.12). Essa ferramenta apresenta informações sobre a tarefa, o conjunto de sintonia-fina que foi usado e se os comandos SQL encontraram algum erro. Além disso, exibe estatísticas globais baseada na comparação de métricas usadas para análises, a melhoria global na carga de trabalho e o impacto de regressão. De acordo com a Figura 4.12, tem-se que a melhoria sugerida não impacta negativamente em nenhum outro comando SQL, pois os itens textuais (item 1) e gráficos (item 2) referentes ao item de regressão (*Regression Impact*) constam zerados.

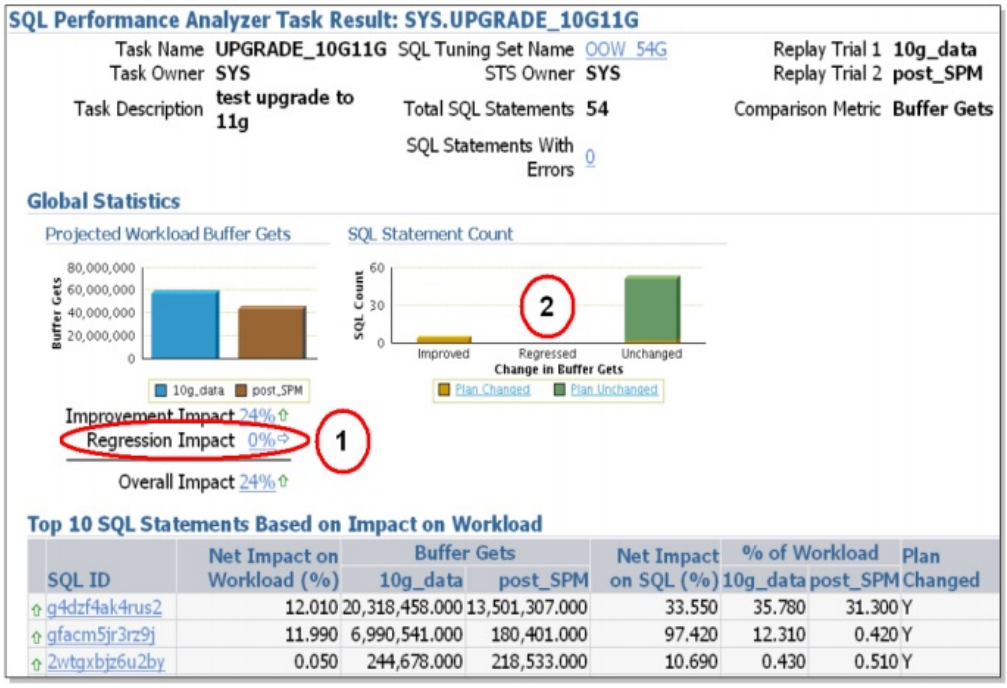


Figura 4.12 SQL Performance Analyzer - Exemplo de resultado de análise [Yagoub et al., 2007]

A ferramenta comercial da Microsoft para o SGBD SQL Server 2017 chama-se *Database Engine Tuning Advisor - DTA* [Microsoft 2017a]. Essa ferramenta analisa a carga de trabalho e sugere a criação de índices, visões indexadas ou partições de tabelas sem a necessidade de o usuário possuir entendimento sobre as estruturas do banco de dados ou detalhes internos do SQL Server (Figura 4.13).

Existem diversas outras ferramentas e abordagens de sintonia fina não citadas neste minicurso. As ferramentas citadas são apenas exemplos da existência de auxílios para o DBA. A maioria das ferramentas busca sugerir uma ou mais ações de sintonia fina e considera que o usuário seja um DBA, ou seja, saiba interpretar os resultados e as ações dessas ferramentas. As ferramentas comerciais citadas aqui não permitem qualquer extensão.

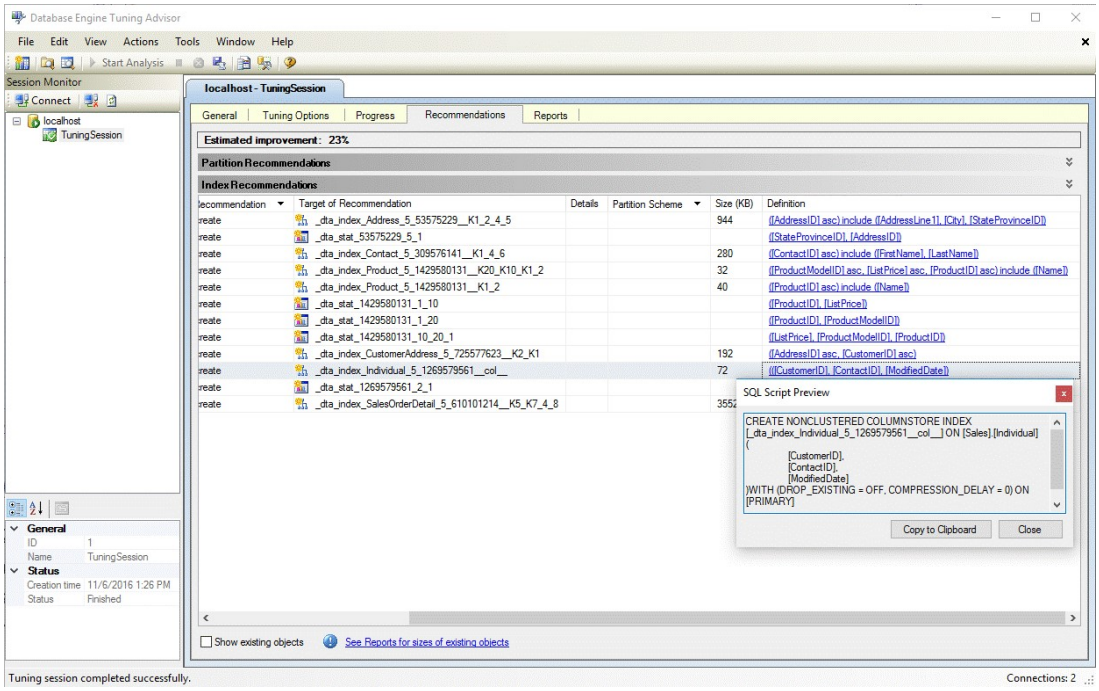


Figura 4.13 *Database Engine Tuning Advisor* – Exemplo de recomendação [Microsoft 2017b]

4.6.2. Estudos de Caso e Boas Práticas

Nesta subseção são descritos alguns exemplos de melhorias de desempenho no banco de dados e por consequência, relatadas algumas boas práticas.

Para todos os exemplos, usa-se o esquema de bancos de dados de uma locadora [Morelli, 2009] implementado no SGBD PostgreSQL 9.6.9[1] (Figura 4.14):

[1] https://www.postgresql.org/

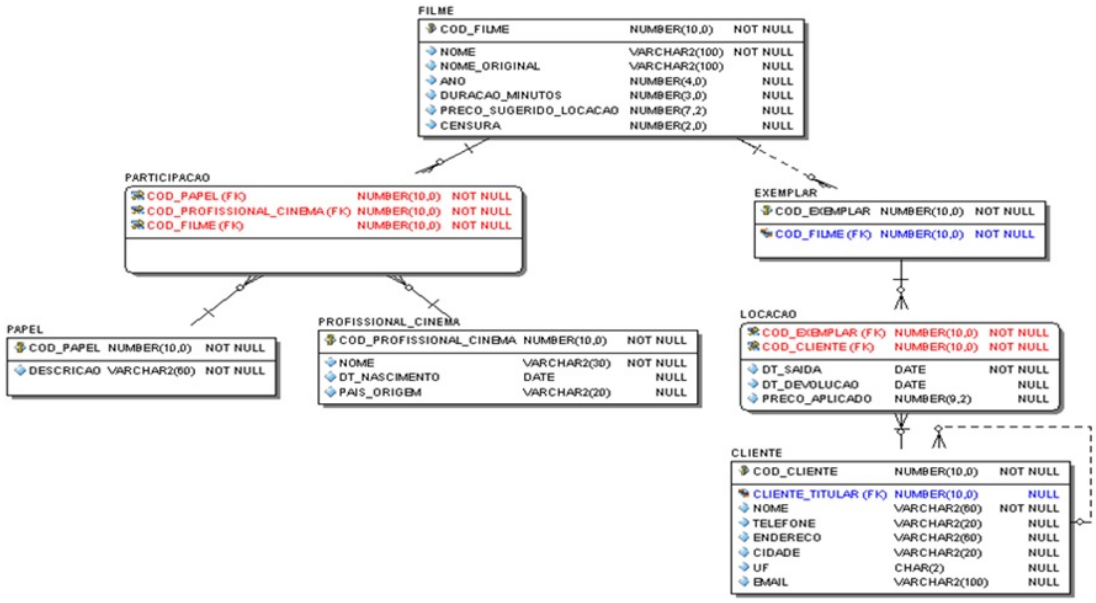


Figura 4.14 Esquema de Locadora [Morelli, 2009]

Para simular um grande volume de dados, cria-se uma tabela chamada de **HISTORICO_LOCACAO** com as mesmas características da tabela **LOCACAO**, mas contendo em torno de 15.000.000 de linhas a mais do que a existente (**LOCACAO**).

4.6.2.1 Uso do DISTINCT

A cláusula DISTINCT deve ser evitada. O DBA deve sempre avaliar a lógica a aplicação para verificar se o uso do DISTINCT é realmente necessário. No caso de consultas sobre uma única tabela, o DBA deve verificar se o resultado da consulta contém alguma coluna definida como chave única. Nesse caso, se existir, não há necessidade do uso do DISTINCT, visto que o comando retornará todas as linhas, pois sempre terá uma coluna com valor distinto em todas as linhas (chave única). Para consultas com junções, o DBA deve verificar o tipo de junção, as colunas da junção e as colunas do resultado.

Considerando a seguinte consulta (Figura 4.15):

```
SELECT distinct cod_exemplar, cod_cliente, dt_saida
FROM historico_locacao;
```

Figura 4.15 Consulta com uso do DISTINCT

O Plano de execução original dessa consulta, gerado pelo otimizador do SGBD é apresentado na Figura 4.16.

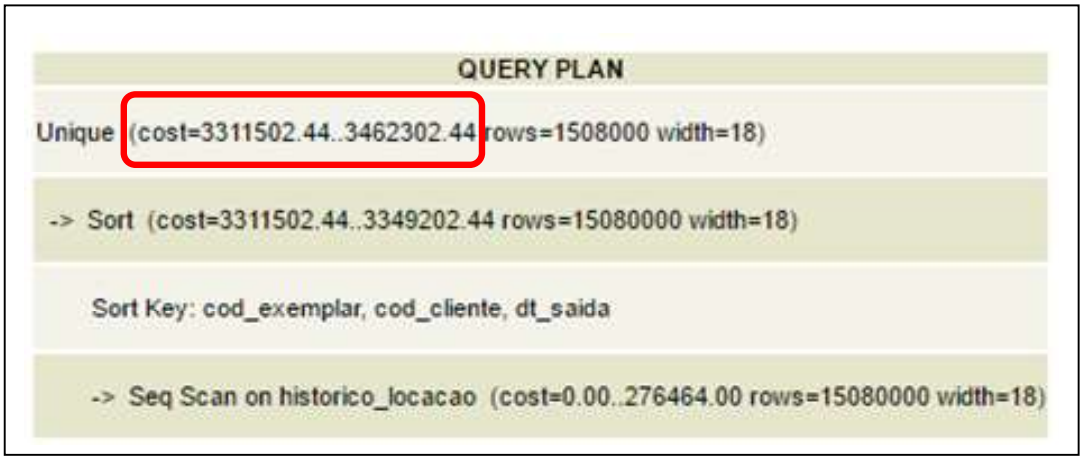


Figura 4.16 Plano de execução da consulta com uso do DISTINCT

Ao analisar consulta, nota-se que as colunas da cláusula de projeção (SELECT) resultante são as `COD_EXEMPLAR` e `COD_CLIENTE`. Essas colunas, segundo o esquema apresentado, corresponde as colunas que compõem a chave primária da tabela. Por definição, a chave primária de uma tabela no SGBD relacional deve ser única, logo, não é necessário usar a cláusula DISTINCT. Ele é totalmente desnecessário. Ao retirar a cláusula DISTINCT e solicitar novamente um plano de execução ao otimizador, obtém-se um custo total bem menor (Figura 4.17), sem as operações de ordenação.

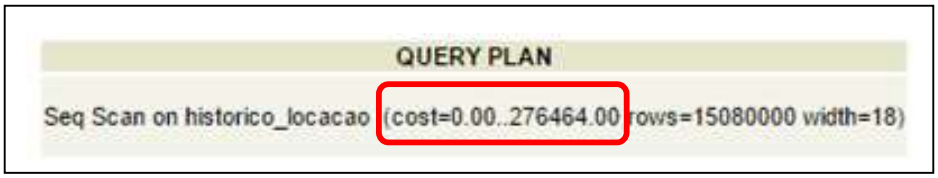


Figura 4.17 Plano de execução otimizado da consulta que usava DISTINCT

4.6.2.2 Uso do HAVING

Se a cláusula HAVING puder ser substituída pela cláusula WHERE, tal substituição deve ser realizada. Essa boa prática é indicada, pois a cláusula WHERE elimina linhas antes da operação de agrupamento e facilita a utilização de índices na comparação dos valores de busca.

Considerando a seguinte consulta (Figura 4.18):

```
SELECT max(preco_aplicado)
FROM historico_locacao
GROUP BY cod_exemplar
HAVING cod_exemplar = 15;
```

Figura 4.18 Consulta com uso do HAVING

O Plano de execução original dessa consulta, gerado pelo otimizador do SGBD é apresentado na Figura 4.19.

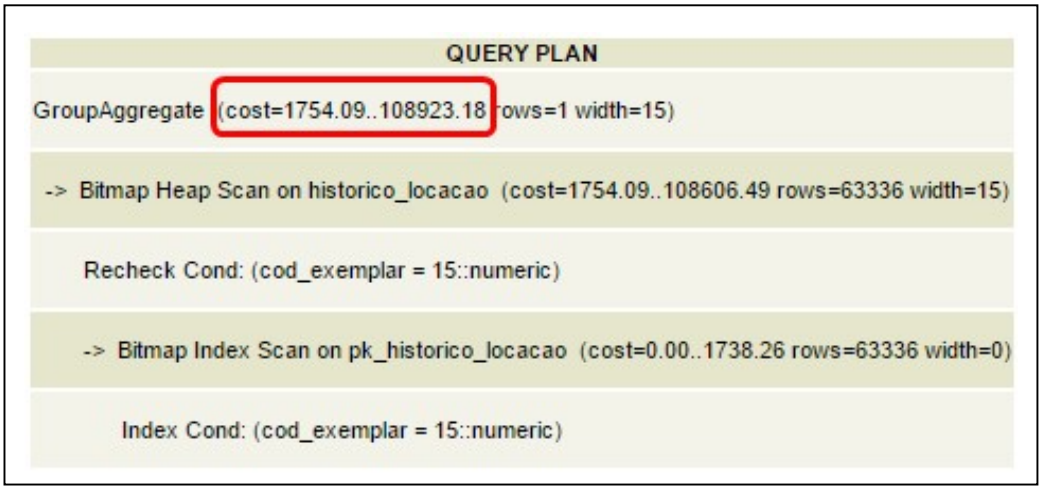


Figura 4.19 Plano de execução da consulta com uso do HAVING

Ao analisar o plano de execução, verifica-se que existe um custo muito alto de agregação, mesmo usando um índice *bitmap* da chave primária. Analisando a consulta, percebe-se que a restrição solicitada na cláusula HAVING não depende do agrupamento, ou seja, ela pode ser executada na cláusula WHERE, para cada linha da tabela. Ao realizar essa modificação, o otimizador gera um plano de execução com custo menor, apresentado na Figura 4.20.

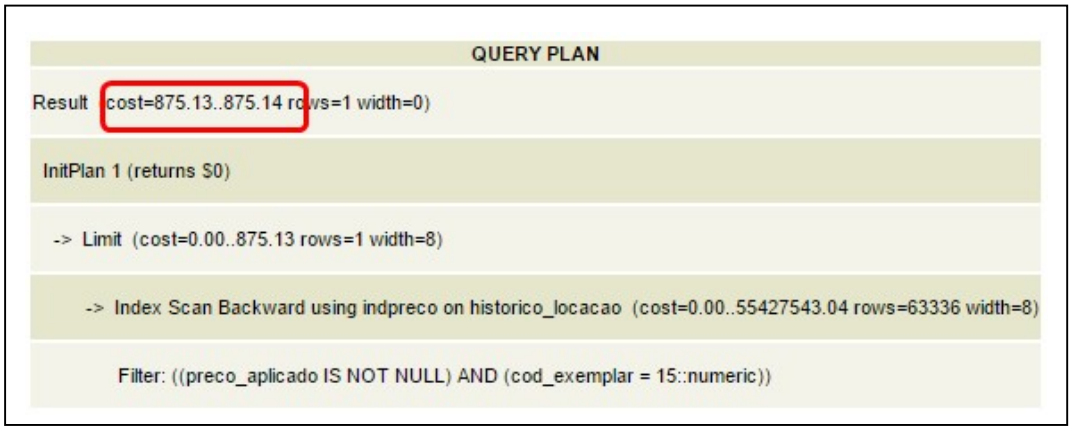


Figura 4.20 Plano de execução otimizado para a consulta que usava a cláusula HAVING

Conforme ilustrado nos exemplos anteriores, a técnica de reescrita de consultas pode ser usada para melhorar o desempenho do banco de dados, diminuindo o custo total de execução de consultas. É importante alertar que o uso de *frameworks*, por desenvolvedores, deve ser cuidadosamente avaliado. O *framework* pode facilitar o desenvolvimento de sistemas de informações, mas pode prejudicar o desempenho do banco de dados. Alguns *frameworks* trazem a facilidade de elaborar as suas próprias consultas, criando diversas subconsultas de forma desnecessária. Com isso, ao submeter essas requisições ao banco de dados, pode-se degradar o desempenho de forma

considerável. Ao analisar as consultas geradas pelo *framework*, o próprio desenvolvedor pode otimizá-la de forma a não prejudicar o banco de dados.

4.6.2.3 Uso de Estruturas Auxiliares

Além da técnica de reescrita de consultas, o DBA também pode optar por criar novas estruturas de acesso. Por exemplo, considera-se a consulta descrita na Figura 4.21.

```
SELECT c.nome, SUM(hl.preco_aplicado) Pago
FROM cliente c
      INNER JOIN historico_locacao hl
      ON c.cod_cliente = hl.cod_cliente
WHERE c.uf = 'ES'
      AND hl.preco_aplicado = 2
GROUP BY c.nome;
```

Figura 4.21 Consulta com junção

O plano de execução, gerado pelo otimizador do SGBD, é apresentado na Figura 4.22.

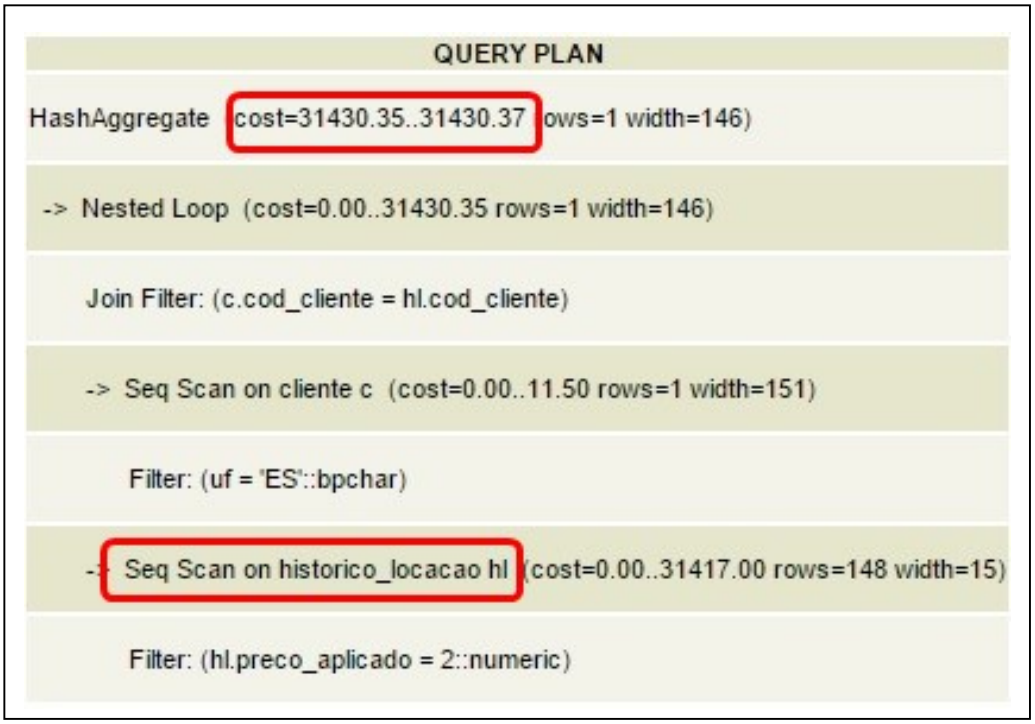


Figura 4.22 Plano de execução da consulta com junção

Analisando o plano de execução da consulta, observa-se que o otimizador realiza uma varredura (*Seq Scan*) na tabela `HISTORICO_LOCACAO`, para executar o filtro

sobre a coluna **PRECO_APLICADO** (`hl.preco_aplicado = 2`). Para diminuir o custo da varredura, o DBA pode optar pela criação de um índice sobre a coluna **PRECO_APLICADO**, desde que haja espaço e esse índice não prejudique as atualizações da tabela. O comando usado para a criação do índice se encontra na Figura 4.23.

```
CREATE INDEX indhist ON historico_locacao(preco_aplicado);
```

Figura 4.23 Comando SQL para criação de índice

Ao criar o índice no banco de dados e analisar o novo plano de execução da consulta (Figura 4.24), certifica-se de que o índice realmente beneficia a consulta e proporciona um decréscimo considerável no custo total do plano. Além disso, a criação do índice proporciona uma agilidade bem maior no tempo de resposta da consulta.

Figura 4.24 Plano de execução otimizado da consulta com junção

4.6.2.4 Uso de Subconsultas

O DBA deve estar atento ao uso de alternativa de subconsultas não correlacionadas, subconsultas correlacionadas e junções. Nas subconsultas não correlacionadas, não existe menção à consulta externa na subconsulta. As subconsultas correlacionadas

referenciam algum atributo de uma das tabelas da consulta externa. As junções permitem realizar a eventual recomposição de dados que foram separados por projeto.

Entre essas três alternativas, tem-se que as subconsultas não correlacionadas apresentam uma tendência pela não utilização de índices sobre a coluna selecionada na subconsulta (e que é a comparada entre a consulta e a subconsulta), enquanto as junções apresentam, usualmente, melhor desempenho.

Por exemplo, imagine uma consulta para verificar os exemplares de filmes que estão locados. Pode-se avaliar três formas de consultas possíveis. Uma forma poderia usar consulta não correlacionada (Figura 4.25), outra correlacionada (Figura 4.26) e outra por junção (Figura 4.27). Todas elas com os seus respectivos planos de execução e custos.

Nota-se que o otimizador gerou o mesmo plano de execução, com utilização de um algoritmo de junção *hash*, para os comandos com a presença das subconsultas correlacionada e não correlacionada. O plano de execução para o caso da utilização da junção também utiliza uma junção *hash*, mas a ordem de acesso as tabelas é invertida com relação à quando são utilizadas subconsultas. Além disso, tem-se que o custo da realização das consultas que utilizam as cláusulas IN e EXISTS é de, aproximadamente, 29.61 unidades, enquanto o custo do comando com a utilização de junção é de, aproximadamente, 3.34 unidades. Dessa forma, verifica-se que apenas com a reescrita da consulta, a mesma pode ter um ganho de mais de 85% em seu custo.

```
SELECT *
FROM exemplar e
WHERE e.cod_exemplar in
  (SELECT cod_exemplar
   FROM locacao l
   WHERE dt_devolucao
         is null);
```

QUERY PLAN
Hash Join (cost=29.61..32.81 rows=78 width=10)
Hash Cond: (e.cod_exemplar = l.cod_exemplar)
-> Seq Scan on exemplar e (cost=0.00..2.04 rows=104 width=10)
-> Hash (cost=28.63..28.63 rows=78 width=5)
-> HashAggregate (cost=27.85..28.63 rows=78 width=5)
Group Key: l.cod_exemplar
-> Seq Scan on locacao l (cost=0.00..24.08 rows=1508 width=5)
Filter: (dt_devolucao IS NULL)

Figura 4.25 Primeira forma de consulta – Exemplares locados

```
SELECT *
FROM exemplar e
WHERE exists
 (SELECT *
  FROM locacao l
  WHERE l.cod_exemplar
       = e.cod_exemplar
       and dt_devolucao
           is null);
```

```
QUERY PLAN
Hash Join  (cost=29.61..32.81 rows=78 width=10)
Hash Cond: (e.cod_exemplar = l.cod_exemplar)
-> Seq Scan on exemplar e  (cost=0.00..2.04 rows=104 width=10)
-> Hash  (cost=28.63..28.63 rows=78 width=5)
   -> HashAggregate  (cost=27.85..28.63 rows=78 width=5)
      Group Key: l.cod_exemplar
      -> Seq Scan on locacao l  (cost=0.00..24.08 rows=1508 width=5)
         Filter: (dt_devolucao IS NULL)
```

Figura 4.26 Segunda forma de consulta – Exemplares locados

```
SELECT *
FROM exemplar e,
  locacao l
WHERE l.cod_exemplar
     = e.cod_exemplar
     and dt_devolucao
         is null;
```

```
QUERY PLAN
Hash Join  (cost=3.34..48.16 rows=1508 width=42)
Hash Cond: (l.cod_exemplar = e.cod_exemplar)
-> Seq Scan on locacao l  (cost=0.00..24.08 rows=1508 width=32)
   Filter: (dt_devolucao IS NULL)
-> Hash  (cost=2.04..2.04 rows=104 width=10)
   -> Seq Scan on exemplar e  (cost=0.00..2.04 rows=104 width=10)
```

Figura 4.27 Terceira forma de consulta – Exemplares locados

4.7. Conclusão

Este minicurso apresenta uma discussão sobre os principais conceitos envolvidos na (auto) sintonia-fina de bancos de dados relacionais. Além disso, apresenta-se algumas ferramentas acadêmicas e comerciais que podem auxiliar o DBA nessa tarefa considerada complexa. Porém parece claro que as ferramentas, isoladamente, não conseguem resolver todas as situações práticas e o papel do DBA nas organizações continua fundamental no dia a dia. Cabe ao DBA saber se adaptar para utilizar mais e melhor as ferramentas de apoio e poder dedicar mais tempo às atividades mais complexas que são mais dependentes do conhecimento do negócio [Morelli and Lifschitz, 2004].

Como uma forma de demonstrar casos práticos que possam ser aplicados para a melhoria de desempenho de bancos de dados e por consequência, de sistemas de informações, apresenta-se exemplos reais de melhorias com uso de técnicas de sintonia-fina.

Os profissionais especializados na atividade de sintonia-fina de bancos de dados precisam conhecer bem desde a etapa de projeto lógico do banco de dados até estruturas de armazenamento e índices disponíveis. Em particular, sabe-se que há, por vezes, dificuldades existentes exclusivamente por desconhecimento de fundamentos básicos dos sistemas de bancos de dados. Por exemplo, pode ser interessante modificar estruturas de tabelas na base de dados de estruturas *heap* para estruturas de tipo *hash* caso o padrão de acesso aos dados seja mais adequado para acessos diretos (ou aleatórios). Em alguns SGBDs, isso é realizado com um simples comando (como o *modify* do Ingres) e os ganhos de desempenho são notáveis. Entretanto, o desconhecimento do comando do SGBD e, principalmente, do princípio de independência de dados (mudanças no nível físico não devem impactar o nível conceitual ou lógico), faz com que tarefas relativamente simples se tornem complicadas e que sistemas de banco de dados fiquem pouco eficientes desnecessariamente.

Espera-se que o participante tenha uma visão geral sobre (auto) sintonia-fina de bancos de dados, tema importante para a área de bancos de dados e para todos os sistemas de informações que fazem uso de bancos de dados, reforçando a formação básica dos profissionais e estudantes da área.

Referências

Almeida, A. C., Brayner, A., Filho, J. M. S. M., Lifschitz, S., Oliveira, R. P. (2015) "DBX: Um Framework para Auto-sintonia fina baseado em planos hipotéticos", 30º Simpósio Brasileiro de Bancos de Dados - SBBD - Demos and Applications Session, p. 149 – 154, Petrópolis, RJ, Brasil.

Almeida, A. C., Haeusler, E. H., Lifschitz, S., Oliveira, R. P., Schwabe, D. (2018) "Outer-Tuning: sintonia fina automática baseada em ontologia", 33º Simpósio Brasileiro de Bancos de Dados - SBBD - Demos and Applications Session, p. 29 – 34, Rio de Janeiro, RJ, Brasil.

Barrientos, A. (2004). "Uma arquitetura para auto-sintonia global de sgbds baseada em agentes". Dissertação de Mestrado do Departamento de Informatica, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio).

Berman, J. J. (2013) "Principles of big data: preparing, sharing, and analyzing complex information". Morgan Kaufmann, Elsevier.

Bruno, N. (2011) "Automated Physical Database Design and Tuning". CRC Press.

Elmasri, R., Navathe, S. (2016) "Fundamentals of database systems". 7th Edition, London: Pearson.

Ferrari, D. (1978). "Computer Systems Performance Evaluation". Prentice Hall.

Lifschitz, S., Milanés, A., Salles, M. (2004) "Estado da arte em auto-sintonia de SGBD relacionais. ", Relatório Técnico, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), MCC35.

Microsoft (2017a) “Database Engine Tuning Advisor”, https://docs.microsoft.com/pt-br/sql/relational-databases/performance/database-engine-tuning-advisor?view=sql-server-2017, Setembro.

Microsoft (2017b) “Columnstore index recommendations in Database Engine Tuning Advisor (DTA)”, https://docs.microsoft.com/en-us/sql/relational-databases/performance/columnstore-index-recommendations-in-database-engine-tuning-advisor-dta?view=sql-server-2017, Setembro.

Morelli, E. (2009) “Oracle DBA Essencial Vol. 1 - SQL”. Editora Brasport.

Morelli, E., Lifschitz, S. (2004) “Auto-sintonia em SGBDs: o fim do DBA?”. SQL Magazine 8.

Nanda, A. (2009) “Use SQL plan management in Oracle Database 11*g*to optimize execution plans”, https://blogs.oracle.com/oraclemagazine/baselines-and-better-plans, acessado em Outubro, 2018.

Oracle (2006) “Performance Tuning using the SQL Access Advisor”, https://www.oracle.com/technetwork/database/manageability/twp-manage-tuning-using.pdf, acessado em Outubro, 2018.

Oracle (2017) “Configuring the Automatic SQL Tuning Advisor”, https://docs.oracle.com/database/121/ADMQS/GUID-AB077F2A-033B-4881-A916-ADBA1B20A8AC.htm#ADMQS1038, Setembro.

Oracle (2018) “SQL Tuning Guide”, https://docs.oracle.com/en/database/oracle/oracle-database/18/tgsql/introduction-to-sql-tuning.html#GUID-B770B7C7-2D0D-4E8A-9A29-811A5E2A9D4F, acessado em Setembro, 2018.

Shasha, D., Bonnet, P. (2003) “Database tuning – Principles, experiments, and troubleshooting techniques”. Morgan Kaufmann publishers.

Yagoub, K., Gongloor, P. (2007) “SQL Performance Analyzer”, https://www.oracle.com/technetwork/database/database-technologies/performance/spa-white-paper-ow07-132047.pdf, acessado em Outubro, 2018.

Autores

Ana Carolina Brito de Almeida é professora adjunta da Universidade do Estado do Rio de Janeiro (UERJ), alocada no Departamento de Informática e analista judiciário com especialidade em Informática no Tribunal Regional Federal da 2ª Região (TRF2), alocada na Seção de Administração de Bancos de Dados. Realizou pós-doutorado com ênfase em *Big Data* na Universidade Federal do Rio de Janeiro (UFRJ) (Out/2014 a Abr/2015). Doutora em Informática com especialização em *Tuning* de Bancos de Dados pela Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio) (2008 a 2013). Mestre em Sistemas e Computação com especialização em Bioinformática pelo Instituto Militar de Engenharia (IME/RJ) (2004 a 2006). Pesquisadora na área de bancos de dados com ênfase em: (i) sistemas contemplando (auto) sintonia de bancos de dados e (ii) ontologias. Já foi consultora em banco de dados e ministrou cursos na Universidade Petrobras. Detalhes em `http://lattes.cnpq.br/8306729029606464`.

Sérgio Lifschitz é professor associado da Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), alocado no Departamento de Informática. Doutor em Informática com especialização em Bancos de Dados pela École Nationale Supérieure des Télécommunications, ENST Paris, França (Out/1990 a Jan/1994). Mestre em Engenharia Elétrica (Especialização: Sistemas e Geometria Computacional - Mar/1987 a Ago/1990) e Engenheiro Eletricista (Especialização em Sistemas - Mar/1981 a Julho/1986), ambos pela PUC-Rio. Pesquisador na área de bancos de dados com ênfase em (i) computação autônoma e sistemas contemplando auto-sintonia e auto-gerenciamento e (ii) ferramentas e sistemas de gerência de dados para aplicações em bioinformática. Detalhes em `http://lattes.cnpq.br/8164403687403639`.

Capítulo 5

Testes de Desempenho de Software: Teoria e Prática

Thiago Silva de Souza

Abstract

Performance is one of the most important quality features in a software product. Evaluating software performance is not a trivial task because it requires not only practical knowledge of a test automation tool but also understanding of related theoretical aspects. However, the terminology regarding performance testing is quite inconsistent. In addition, the technical literature provides little information as to how performance requirements should be elicited and documented and how performance tests should be planned, implemented, and executed. This mini-course aims to present a performance testing classification as well as provide hands-on experience ranging from eliciting the performance requirement to testing using the Apache JMeter tool.

Resumo

O desempenho é uma das características de qualidade mais importantes em um produto de software. Avaliar o desempenho de um software não é uma tarefa trivial, pois requer não apenas conhecimento prático de uma ferramenta de automação de testes mas, também, compreensão dos aspectos teóricos relacionados. No entanto, a terminologia a respeito de testes de desempenho é bastante inconsistente. Além disso, a literatura técnica traz pouca informação a respeito de como requisitos de desempenho devem ser elicitados e documentados e como testes de desempenho devem ser planejados, implementados e executados. Este minicurso apresenta uma classificação sobre testes de desempenho e proporciona uma experiência prática cobrindo desde a elicitação do requisito de desempenho até o seu teste, utilizando a ferramenta Apache JMeter.

5.1. Introdução

O teste de software é um processo relacionado ao desenvolvimento de software que tem como principal objetivo revelar falhas por meio da análise dinâmica de programas [ISO/IEC/IEEE 2013]. Trata-se de uma análise dinâmica porque há a necessidade de

colocar o software em execução para avaliar se determinados dados de entrada geram determinados dados de saída (resultados esperados), sob determinadas condições de contexto.

Testes de software podem ser classificados em função de diversas dimensões (nível, tipo e técnica de teste, entre outras). Em relação ao seu tipo, os testes podem ser classificados em **testes funcionais** ou **testes não-funcionais**. A definição do tipo de teste depende do tipo de requisito que se deseja avaliar. **Requisitos funcionais** demandam testes funcionais, enquanto que **requisitos não-funcionais** (RNF) exigem a realização de testes não-funcionais.

A classificação em requisitos funcionais e não-funcionais, aparentemente, não é a ideal. Trata-se de uma classificação onde um dos conceitos é dado pela negação do outro. Em outras palavras, todos os requisitos que não são funcionais são classificados como não-funcionais. Assim, temos requisitos totalmente diferentes sob a mesma classificação. Esse tipo de classificação contribui para uma grande confusão conceitual, provocando distorções que se espalham por todo o processo de desenvolvimento do software, inclusive o teste.

Uma dessas distorções diz respeito à estimativa de esforço e custo. Muitas empresas de software têm cobrado pela implementação de RNF de forma genérica, precificando requisitos de desempenho da mesma forma que precificam requisitos de usabilidade ou de segurança, considerando que todos são "requisitos não-funcionais". No entanto, cada tipo de RNF exige esforços e custos específicos, o que justificaria preços diferenciados.

Portanto, há diversos tipos de RNF, classificados das mais variadas formas. Atualmente, a classificação de requisitos de software mais aceita pela indústria é aquela representada no modelo de qualidade de produtos de software da norma ISO 25010 [ISO/IEC 2011], representado na Figura 5.1. Esse modelo é organizado em oito características e 31 subcaracterísticas de qualidade. A maior parte dessas características (sete) e subcaracterísticas (28) se refere a requisitos não-funcionais, dentre as quais se encontra a característica de "Eficiência no desempenho" (ou apenas "Desempenho"), que é o foco deste trabalho.

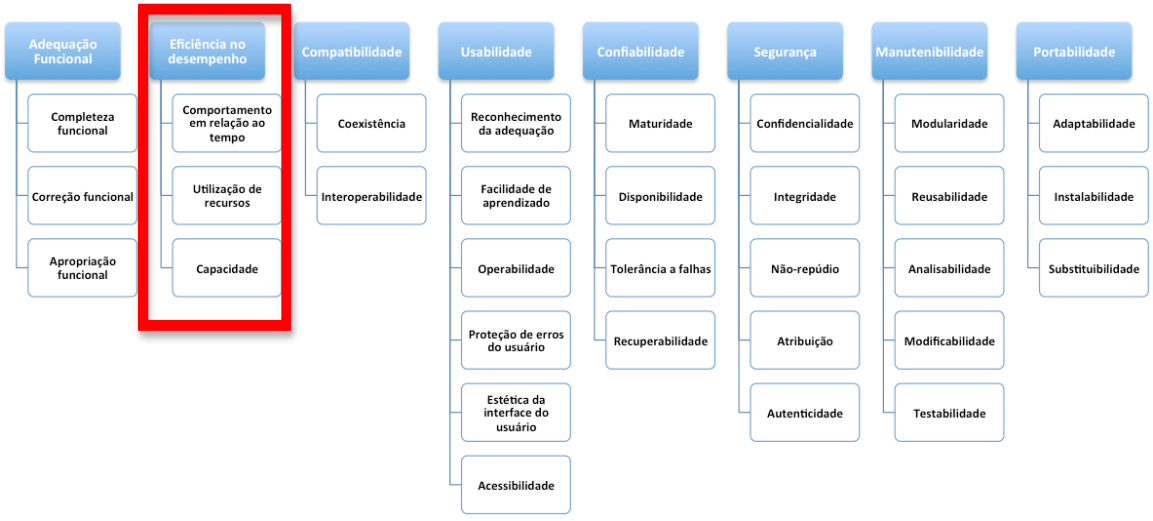


Figura 5.1: Características de qualidade de produtos de software presentes na norma ISO 25010 [ISO/IEC 2011].

O restante deste trabalho está organizado da seguinte maneira: a seção 5.2 aborda aspectos relacionados à elicitação e documentação de requisitos de desempenho; a seção 5.3 descreve os principais conceitos relacionados, com ênfase na classificação de diferentes tipos de testes de desempenho; a seção 5.4 introduz a ferramenta de testes de desempenho Apache JMeter; a seção 5.5 demonstra aspectos do JMeter por meio de um exemplo prático; a seção 5.6 relaciona uma série de recomendações práticas para realização de testes de desempenho; por fim, a seção 5.7 apresenta as conclusões deste trabalho.

5.2. Requisitos de Desempenho de Software

O processo de Engenharia de Requisitos cumpre papel fundamental para o processo de Teste de Software. É nesse processo que os requisitos do sistema são elicitados (identificados), analisados, documentados e validados. A qualidade dos testes é fortemente influenciada pela qualidade dos requisitos. Quando se trata de requisitos de desempenho, é necessário realizar as atividades desse processo com o viés que esse tipo de requisito demanda. Portanto, as subseções 5.2.1 e 5.2.2 apresentam, respectivamente, orientações a respeito das atividades de elicitação e documentação de requisitos de desempenho baseadas na experiência prática do autor.

5.2.1. Elicitação de Requisitos de Desempenho

A literatura de Engenharia de Software descreve diversas técnicas para elicitação de requisitos, tais como entrevistas, questionários, etnografia, etc. A princípio, essas técnicas servem para elicitar qualquer tipo de requisito. Apesar disso, analistas de requisitos costumam utilizá-las com foco em requisitos funcionais. A utilização das técnicas de elicitação de requisitos para identificar RNF exige adaptação de acordo com o contexto apresentado. Cada tipo de RNF pode ter fontes de informação diferentes. Por exemplo, a principal fonte de informação sobre requisitos funcionais é o próprio cliente e os potenciais usuários do sistema. Já os RNF podem ter fontes variadas, podendo ser, inclusive, necessário elicitar requisitos em conjunto com cliente e pessoal técnico. A Figura 5.2 representa essa variedade de fontes em função do tipo de requisito.

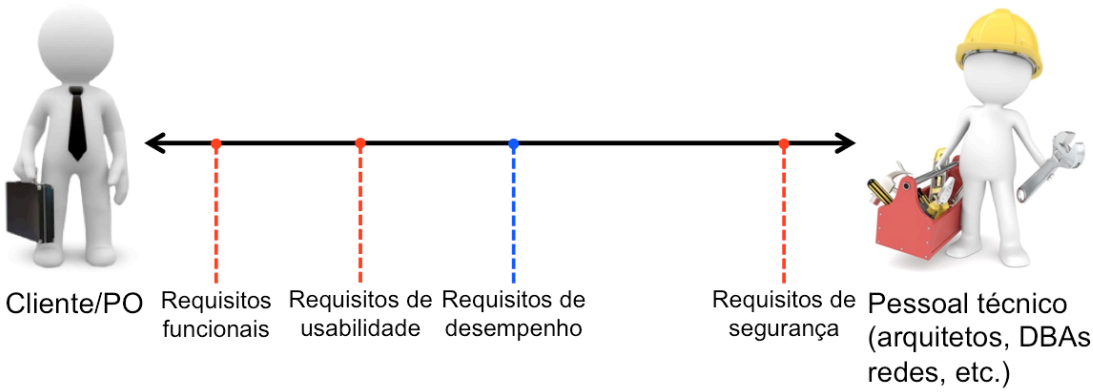


Figura 5.2: Fontes de diferentes tipos de requisitos de software.

Vale destacar que a Figura 5.2 representa um espectro e o fato de um tipo de requisito estar mais próximo de uma das extremidades não significa que a outra não possa contribuir com informações sobre determinado tipo de requisito. Outro ponto importante é que a responsabilidade por determinar todos os requisitos de um projeto é

exclusiva do cliente. Ser uma fonte de informações sobre um determinado tipo de requisito não significa ser o responsável por determinar quais serão os requisitos daquele tipo. Cabe, no entanto, ao pessoal técnico (tanto do lado da equipe de desenvolvimento quanto do lado do cliente, quando pertinente) fornecer informações que apoiem a decisão do cliente sobre cada tipo de requisito. Por exemplo, um arquiteto pode sinalizar que um determinado requisito funcional é inviável de se alcançar, devido às restrições tecnológicas da solução. Assim, o requisito funcional em questão deve ser renegociado com o cliente.

Requisitos de desempenho, em especial, são difíceis de elicitar porque nem sempre o cliente sabe responder a questionamentos como "Qual a expectativa de acessos concorrentes a esta funcionalidade?" ou "Qual o tempo de resposta tolerado para esta consulta?". Em contrapartida, para um membro da equipe técnica do projeto, pode ser um desafio propenso a erros tentar inferir requisitos desse tipo sem a contribuição do cliente. Recomenda-se, portanto, aplicar uma combinação de técnicas de elicitação de requisitos, tais como:

- **Entrevistas** com o cliente, *product owner* (PO) e demais *stakeholders*;
- **Analogia** com sistemas existentes (realizando a análise de *logs* de produção);
- ***Brainstorming*** envolvendo especialistas no negócio e pessoal técnico;
- **Testes de desempenho exploratórios**, submetendo o sistema sob teste a cargas definidas aleatoriamente, definindo os requisitos de desempenho em função dos resultados apresentados. É importante destacar que neste caso os requisitos de desempenho são definidos tardiamente, somente após a disponibilização de uma versão executável do sistema.

5.2.2. Documentação de Requisitos de Desempenho

Há diversas técnicas disponíveis para especificar (documentar) requisitos funcionais, tais como casos de uso e histórias de usuário. Além disso, há vasta publicação a respeito de critérios de qualidade de especificação de requisitos funcionais. No entanto, não se encontram tantos padrões industriais para se especificar RNF.

Cada tipo de RNF demanda um formato de especificação diferente. Um requisito de acessibilidade poderá exigir a identificação do tipo de deficiência do usuário e suas respectivas necessidades - por exemplo, um recurso de alto contraste em uma aplicação Web pode ser muito útil para uma pessoa com baixa visão, porém não terá utilidade para uma pessoa com cegueira total. Já um requisito de desempenho poderá exigir a especificação de um modelo de carga (*workload model*), incluindo, para cada funcionalidade do sistema uma expectativa de usuários concorrentes em momentos de uso típico e de pico e o tempo de resposta desejável ou aceitável.

Geralmente, RNF são documentados em linguagem natural, usando texto livre. Essa característica pode acarretar em descrições insuficientes, ambíguas e inconsistentes, diminuindo sua utilidade no contexto do projeto. Os três exemplos de RNF a seguir foram coletados em projetos reais na indústria. Todos se referem a requisitos de desempenho e apresentam problemas que confundem e, até mesmo, inviabilizam o trabalho de arquitetos, testadores e demais interessados nesses requisitos.

- "A quantidade de acessos **simultâneos** estimada **mínima** é de 200 usuários."
    - Provavelmente, o analista de requisitos quis dizer "acessos concorrentes", o que é diferente de "simultâneos" (paralelos).
    - O que de fato importa é saber a quantidade máxima de acessos concorrentes, para que tanto o hardware quanto o software sejam dimensionados para suportar as cargas de pico.
- "O sistema pode acomodar **vários** usuários ou transações."
    - O requisito está incompleto e impreciso. O que significa "vários"? É necessário indicar claramente quantos acessos concorrentes o sistema deve suportar e em qual espaço de tempo.
- "O tempo **médio** de resposta às operações de atualização e consultas **simples** deverá ser no máximo de **trinta segundos**."
    - Recomenda-se não usar o tempo médio como parâmetro para um requisito de desempenho, tendo em vista que a média é muito sensível a valores extremos (*outliers*). Neste caso, é preferível utilizar medidas relacionadas aos percentis 90, 95 ou 99 de tempo de resposta, dependendo do grau de criticidade do sistema.
    - O requisito é genérico e impreciso, quando agrupa e não indica quais são as "operações de atualização e consultas simples". Além disso, se há operações "simples", provavelmente também há operações "complexas", as quais não são mencionadas no requisito.
    - Provavelmente, um tempo médio de trinta segundos para responder a operações classificadas como "simples" não será tolerado pelos usuários.

O exemplo a seguir, por sua vez, reúne um conjunto de boas práticas de especificação de requisitos de desempenho que viabilizam a sua testabilidade:

- "A funcionalidade 'Consultar pedidos de compra' deve fornecer tempos de resposta de no máximo 2s, considerando 100 usuários concorrentes, com uma rampa de subida de 10s e 200 registros a serem retornados do banco de dados."
    - O requisito é específico: refere-se a apenas uma funcionalidade.
    - Indica objetivamente o tempo de resposta tolerado e em que condições ele deve ser observado, considerando o número de usuários concorrentes, a rampa de subida e a quantidade de registros retornados.

Esse exemplo, no entanto, não representa um padrão ou modelo a ser seguido em qualquer contexto. Características específicas de um projeto poderão exigir um formato de especificação diferenciado visando garantir a testabilidade do requisito de desempenho.

5.3. Testes de Desempenho de Software

As subseções 5.3.1, 5.3.2 e 5.3.3 descrevem, respectivamente, o processo típico de teste de software, as principais medidas de desempenho de software e os tipos de teste de desempenho mais comuns. A definição precisa desses conceitos é fundamental para a compreensão das práticas que envolvem um projeto de testes dessa natureza.

5.3.1. Processo de Teste de Software

As atividades que compõem um processo de teste de software podem variar de acordo com a organização. No entanto, de acordo com a norma ISO/IEC/IEEE 29119-2 [ISO/IEC/IEEE 2013], um processo de testes típico costuma incluir as seguintes atividades:

- **Planejamento de testes**: atividade caracterizada pela elaboração do Plano de Teste, documento frequentemente utilizado para descrever o escopo do projeto de testes e para definir a estratégia de testes, os recursos alocados e o cronograma;
- **Design e implementação de testes**: atividade na qual são derivados os casos e procedimentos de teste a partir dos requisitos do software. Esta atividade requer que os testadores apliquem uma ou mais técnicas de design de testes com o objetivo de alcançar os critérios de conclusão estabelecidos, tipicamente descritos em termos de medidas de cobertura. Esta atividade também contempla a implementação dos procedimentos de teste na forma de *scripts* de teste automatizados;
- **Configuração do ambiente de testes**: atividade em que se estabelece o ambiente operacional no qual os testes serão executados, incluindo o software sob teste e ferramental de apoio a sua operação;
- **Execução de testes**: atividade em que os procedimentos de teste definidos na atividade de design e implementação de testes são executados sobre o ambiente de testes estabelecido, de forma manual ou automatizada;
- **Comunicação de incidentes**: atividade em que os resultados observados com a execução de testes são analisados e eventuais incidentes de teste são registrados e comunicados às partes interessadas;
- **Conclusão do projeto de testes**: esta atividade marca o encerramento do projeto de testes e consiste em uma série de tarefas, incluindo o registro e comunicação dos resultados do projeto de testes às partes interessadas, a "limpeza" do ambiente de testes e a identificação de lições aprendidas.

Testes de desempenho podem ser realizados de acordo com esse processo-base, com as devidas adaptações considerando as características desse tipo de teste. Atividades de diagnóstico de gargalos e *tuning* do sistema, apesar de importantes nesse contexto, não fazem parte do processo de testes - elas compõem o processo de desenvolvimento.

5.3.2. Medidas de Desempenho

Dentre os diversos tipos de RNF representados na Figura 5.1, na forma de características e subcaracterísticas de qualidade, há o requisito de "eficiência no desempenho" ou apenas "**desempenho**". De acordo com esse modelo, o desempenho de um produto de software pode ser observado em função de três dimensões: o comportamento em relação ao tempo, a utilização de recursos e a capacidade do sistema.

O **comportamento em relação ao tempo** é definido como o "grau em que os tempos de resposta e de processamento, bem como as taxas de *throughput* do sistema, atingem seus requisitos, quando as funções do sistema são exercitadas". Já a **utilização de recursos** é definida como o "grau em que a quantidade e variedade de recursos usados pelo sistema atingem seus requisitos, quando as funções do sistema são exercitadas". Por fim, a **capacidade** é definida como o "grau em que os limites máximos de um parâmetro do sistema, incluindo o número de itens a serem armazenados, número de usuários concorrentes, largura de banda, *throughput* das transações e tamanho do banco de dados, atingem seus requisitos" [ISO/IEC 2011].

Cada uma dessas dimensões pode ser medida de diferentes maneiras, conforme sugere a norma ISO/IEC/IEEE 25023 [ISO/IEC/IEEE 2015], como pode ser observado na Tabela 5.1.

Tabela 5.1. Dimensões e medidas de desempenho de software [ISO/IEC/IEEE 2015].

Comportamento no tempo	Utilização de recursos	Capacidade
Tempo médio de resposta	Utilização média do processador	Capacidade de processamento de transações
Adequação do tempo de resposta	Utilização média da memória	Capacidade de acessos de usuários
Tempo médio de entrega	Utilização média dos dispositivos de E/S	Adequação do aumento do número de acessos
Adequação do tempo de entrega	Utilização da largura de banda	
Throughput médio		

Para avaliar se um produto de software atende a essa variedade de requisitos é necessário realizar **testes de desempenho**. De acordo com a norma ISO/IEC/IEEE 29119-1, o teste de desempenho (ou *performance*) mede e avalia o grau em que um item de teste desempenha suas funções designadas dentro de determinados limites de tempo, de uso de recursos (CPU, memória, disco, rede) e de capacidade [ISO/IEC/IEEE 2013].

Testes de desempenho também servem para identificar eventuais "gargalos" (*bottlenecks*) no sistema. Um **gargalo** é uma parte do sistema que limita o desempenho ou capacidade de todo o sistema, afetando principalmente os tempos de resposta e o *throughput* [Bondi 2014]. Os gargalos em um sistema podem estar localizados no próprio sistema (como uma *query* mal elaborada ou não-otimizada), na camada de software que sustenta esse sistema (sistemas operacionais, servidores de aplicação, *firewalls* mal configurados) ou no hardware, incluindo CPU, memória, disco e rede.

A avaliação de cada subcaracterística de desempenho, incluindo a identificação de gargalos, demanda a realização de diferentes tipos de testes de desempenho. A seção a seguir classifica e descreve os principais tipos de testes de desempenho.

5.3.3. Tipos de Testes de Desempenho

Um dos principais problemas da área de Teste de Software é a ausência de uma terminologia e de uma taxonomia consistentes e que sejam aceitas pela indústria. Quando se trata de Testes de Desempenho, em especial, surgem inúmeros termos (*performance*, carga, *stress*, etc.) que, em geral, não são compreendidos da mesma forma, até mesmo por profissionais da mesma organização e com muitos anos de experiência na área. A Figura 5.3 representa o "guarda-chuva" de termos relacionados a testes de desempenho.

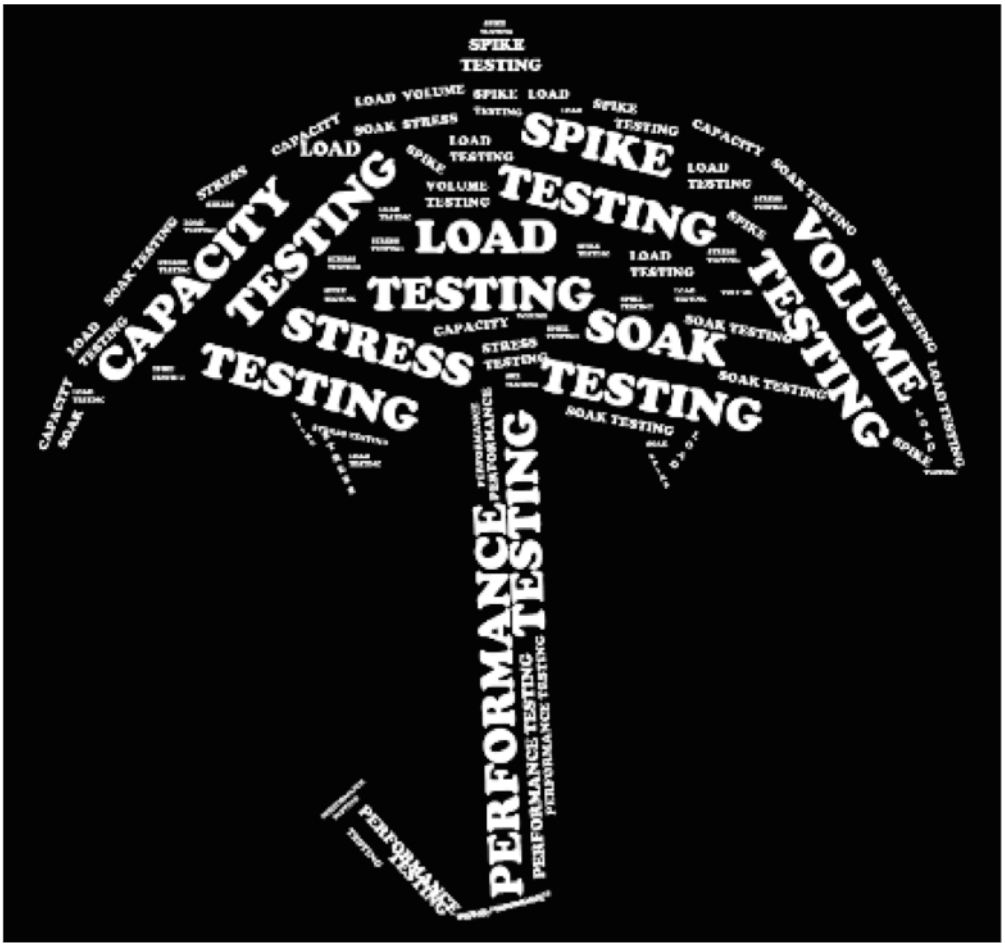


Figura 5.3: "Guarda-chuva" conceitual sobre testes de desempenho.

Essa falta de consenso pode levar tanto a ruídos simples de comunicação quanto a erros graves na cobrança pela prestação de serviços desta natureza. Além disso, o desconhecimento sobre Testes de Desempenho contribui para que os requisitos de desempenho não sejam elicitados e documentados de forma apropriada, podendo ocasionar diferentes impactos sobre o projeto.

Diante deste cenário, a norma ISO/IEC/IEEE 29119-1 sugere uma classificação e um conjunto de definições. Esta classificação considera, genericamente, "Teste de Desempenho" como um tipo de teste que possui cinco subtipos: 1) teste de carga; 2) teste de estresse; 3) teste de capacidade; 4) teste de resistência; e 5) teste de volume, definidos conforme a Tabela 5.2 [ISO/IEC/IEEE 2013]:

Tabela 5.2. Tipos de testes de desempenho conforme a norma ISO/IEC/IEEE 29119-1 [ISO/IEC/IEEE 2013].

Tipo de teste	Definição
Teste de carga	Avaliar o comportamento de um item de teste em condições **esperadas de carga** variável (uso baixo, típico e de pico).
Teste de estresse	Avaliar o comportamento de um item de teste em condições de **carga acima dos requisitos** de capacidade antecipados ou especificados ou da disponibilidade de recursos abaixo dos requisitos mínimos exigidos.
Teste de capacidade	Avaliar o nível no qual o **aumento de carga** (de usuários, transações, armazenamento de dados, etc.) **compromete a capacidade** de um item de teste para sustentar o desempenho requerido.
Teste de resistência	Avaliar se um item de teste pode sustentar uma **carga necessária continuamente** por um período de tempo determinado.
Teste de volume	Avaliar a capacidade do item de teste processar **volumes de dados** especificados (usualmente no ou **próximos ao limite máximo de capacidade** especificada) em termos de *throughput*, capacidade de armazenamento ou ambos.

Apesar de útil, essa classificação ainda se mostra superficial à medida em que não detalha as diferenças entre os cinco tipos de teste apresentados e desconsidera outros tipos de testes relacionados ao desempenho, tais como o *spike testing* e o *soak testing*.

Desta forma, foi realizado um trabalho de harmonização de conceitos, considerando o que já está definido na norma ISO/IEC/IEEE 29119-1 [ISO/IEC/IEEE 2013] e outros tipos de testes de desempenho que estão ausentes da norma. Com base nas definições encontradas, foram identificadas algumas relações de sinonímia e de hiponímia entre os termos, resultando no modelo conceitual representado na Figura 5.4.

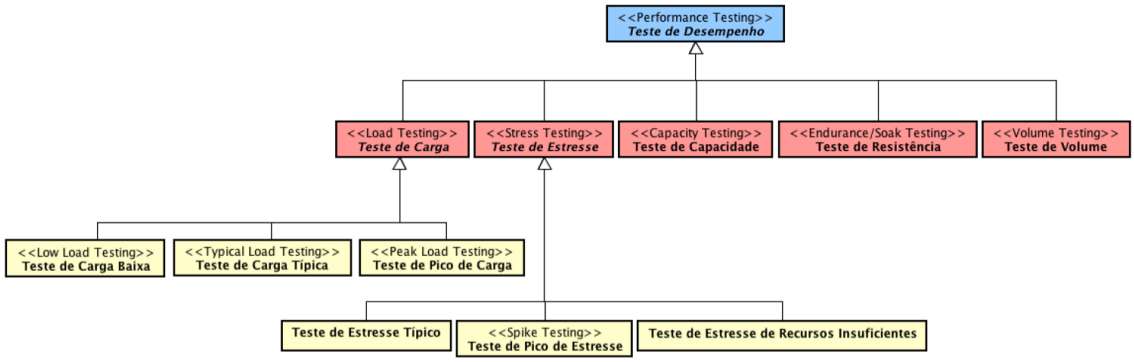


Figura 5.4: Tipos de testes de desempenho.

Para diferenciar os diversos tipos de testes de desempenho representados nesse modelo conceitual, deve-se levar em conta os seguintes parâmetros:

- **expectativa de utilização típica**: a quantidade de usuários/transações concorrentes e/ou de dados a serem processados, esperada para horários de utilização média (regular) do sistema (desconsiderando períodos de pico);
- **número de usuários/transações concorrentes**: a quantidade de usuários/transações concorrentes submetida ao sistema sob teste;

- **quantidade de dados a serem processados**: a quantidade (volume) de dados submetida ao sistema sob teste;
- **capacidade máxima projetada**: a medida da carga máxima suportada pelo sistema, incluindo seu ambiente de execução. Idealmente, a capacidade máxima projetada deve estar acima da expectativa de utilização típica, considerando picos de acesso e previsão de crescimento do sistema.

O **teste de carga baixa** é um subtipo do teste de carga que, semelhantemente a um teste fumaça, tem como objetivo principal avaliar se as funcionalidades críticas de um sistema estão funcionando como esperado. Desta forma, o sistema é submetido a um conjunto de casos de teste que exercitam diversas funcionalidades e possíveis integrações, sob uma carga reduzida, bem abaixo da expectativa de utilização típica, como representa a Figura 5.5.

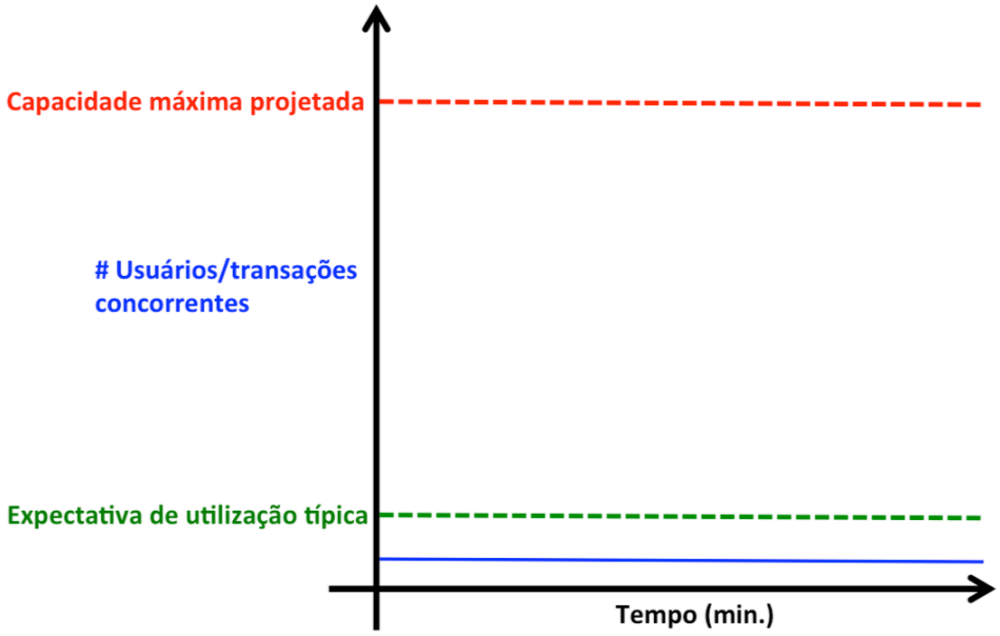


Figura 5.5. Teste de carga baixa.

O **teste de carga típica** é um subtipo do teste de carga que avalia o desempenho do sistema diante da submissão de uma carga correspondente à expectativa de utilização típica, conforme a Figura 5.6.

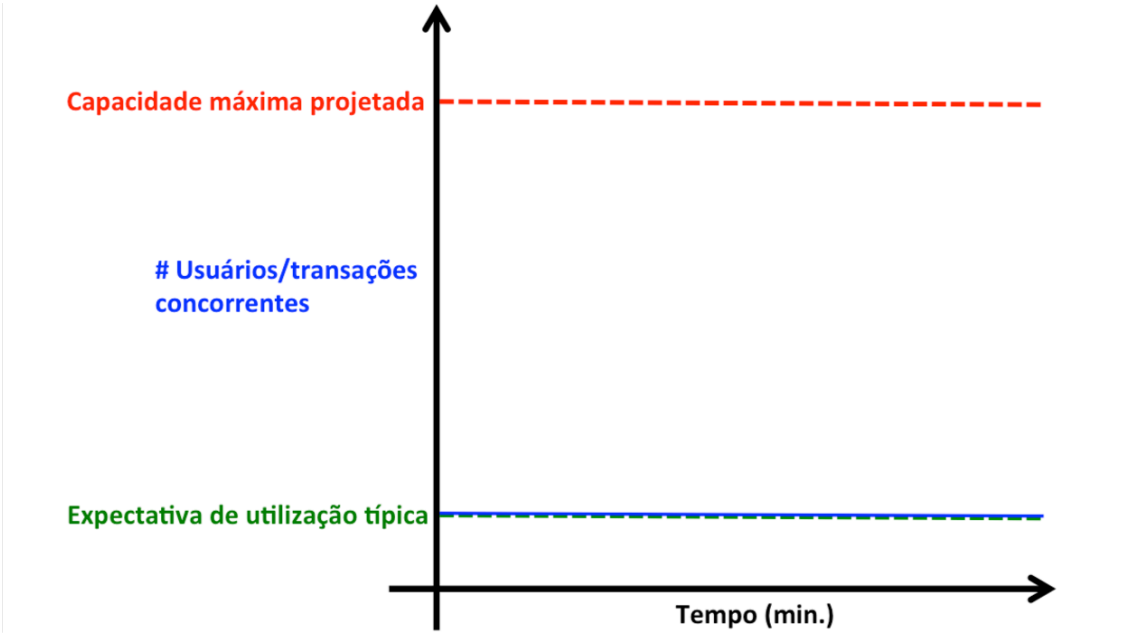


Figura 5.6. Teste de carga típica.

O **teste de pico de carga**, ou *peak load testing*, é o subtipo de teste de carga mais comum. Seu objetivo é avaliar o desempenho do sistema diante de cargas que correspondam aos picos de utilização previstos. Para simular um pico de utilização é necessário estabelecer o período de aceleração ou subida (*ramp-up period*) e, eventualmente, o período de desaceleração ou descida (*ramp-down period*), como mostra a Figura 5.7. O *ramp-up period* é o tempo dentro do qual todos os usuários virtuais (*threads*) devem ser iniciados, submetendo requisições ao sistema sob teste. Já o *ramp-down period* corresponde à faixa de tempo em que os usuários virtuais devem ser finalizados, encerrando o processamento de suas requisições.

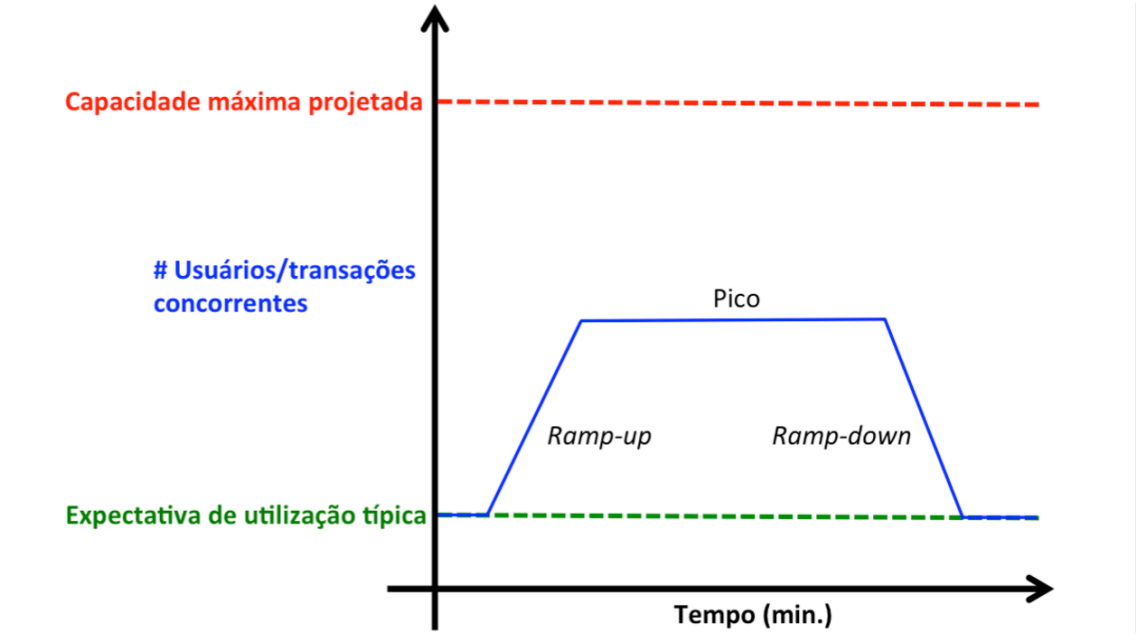


Figura 5.7. Teste de pico de carga.

De forma geral, testes de carga avaliam o desempenho do sistema dentro dos limites de capacidade máxima projetada. Já o testes de estresse extrapolam esses limites, com o objetivo de avaliar o comportamento do sistema diante de condições extremas de utilização. O resultado típico de um teste de estresse realizado sobre um sistema Web é a interrupção ("queda") do servidor HTTP. Há, ainda, casos de exposição de vulnerabilidades de segurança. No entanto, outros resultados podem ser observados, especialmente em plataformas não-Web. Portanto, é preciso se atentar a qualquer anomalia no comportamento do sistema quando submetido a um teste de estresse.

O **teste de estresse típico** é um subtipo do teste de estresse usado para avaliar como o sistema sob teste se comporta diante de cargas superiores à sua capacidade máxima projetada. Também pode ser usado para identificar o "ponto de quebra" (*breaking point*) do sistema, ou seja, o tamanho de carga que faz o sistema parar de funcionar. Esse tipo de teste envolve a submissão de uma carga que aumenta continuamente ao longo do tempo, conforme a Figura 5.8, sendo interrompido somente quando o sistema apresentar alguma anomalia.

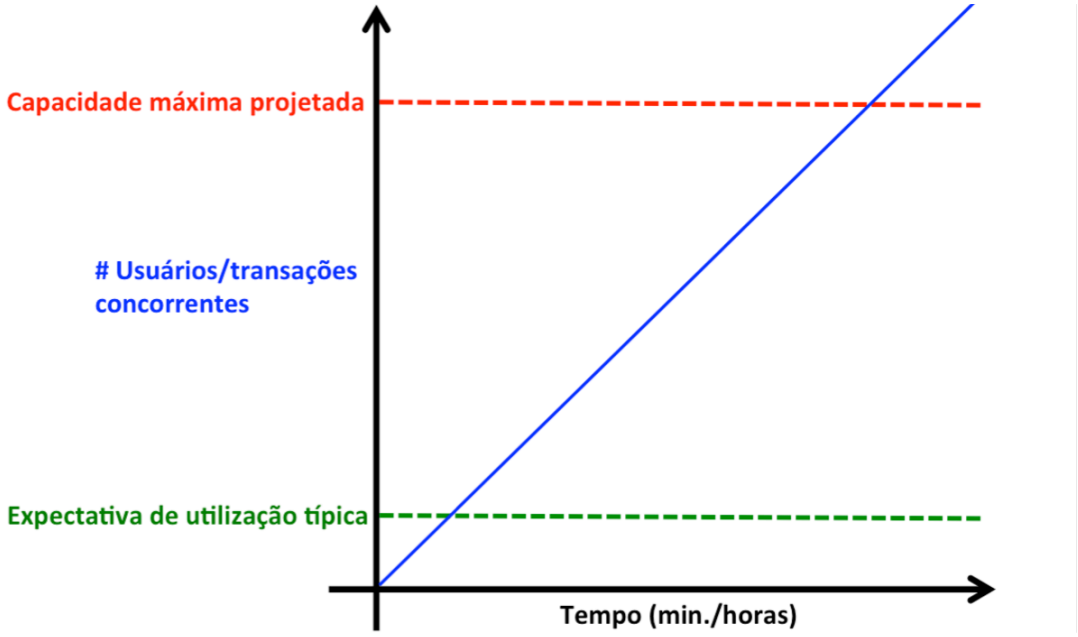


Figura 5.8. Teste de estresse típico.

O **teste de pico de estresse**, ou *spike testing*, é um subtipo do teste de estresse que submete o sistema a acréscimos e decréscimos extremos e repentinos de carga, como representa a Figura 5.9. Esse teste ajuda a determinar se o desempenho do sistema se deteriora quando há um aumento súbito da carga. Outro objetivo desse teste é determinar o tempo de recuperação do sistema. Entre dois picos consecutivos de carga do usuário, o sistema precisa de algum tempo para se estabilizar. Esse tempo de recuperação deve ser o mais baixo possível.

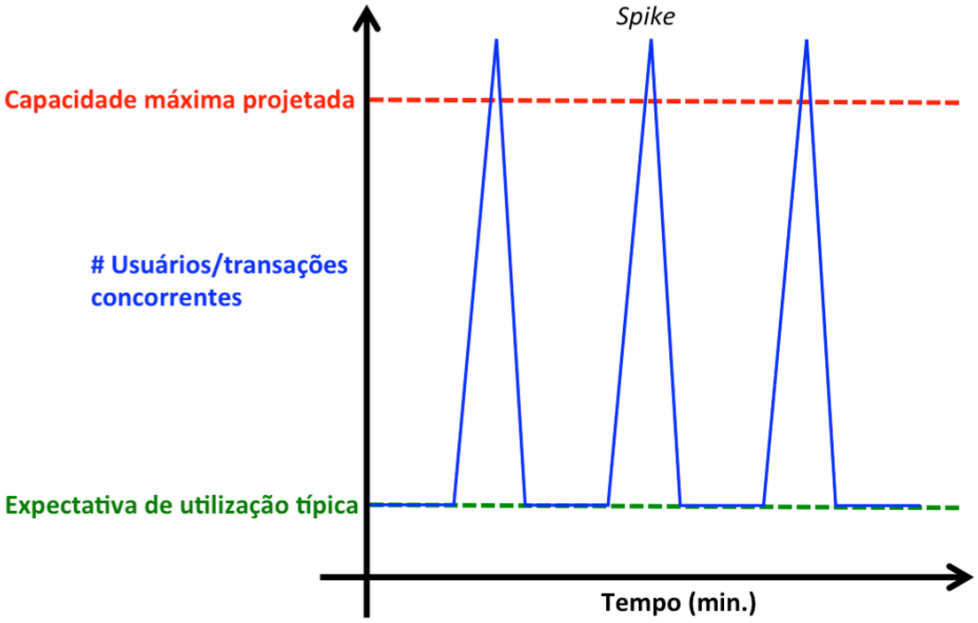


Figura 5.9. Teste de pico de estresse.

O **teste de estresse de recursos insuficientes** é um subtipo do teste de estresse no qual o comportamento do sistema é avaliado sob uma configuração de hardware e software muito aquém das configurações mínimas necessárias (os parâmetros de "expectativa de utilização típica" e "capacidade máxima projetada" são invertidos, como representado na Figura 5.10). Por exemplo, para executar um determinado programa *desktop* é necessário um computador com o sistema operacional Windows 10 e no mínimo uma CPU de dois núcleos, 8GB de RAM e 15GB de espaço em disco. Porém, o que poderia acontecer se esse programa fosse instalado em uma máquina que não atenda a esses requisitos mínimos? A resposta a essa pergunta pode ser obtida por meio de um teste de estresse de recursos insuficientes, em que, por exemplo, o programa fosse instalado em uma máquina com Windows 8 e demais requisitos abaixo do necessário.

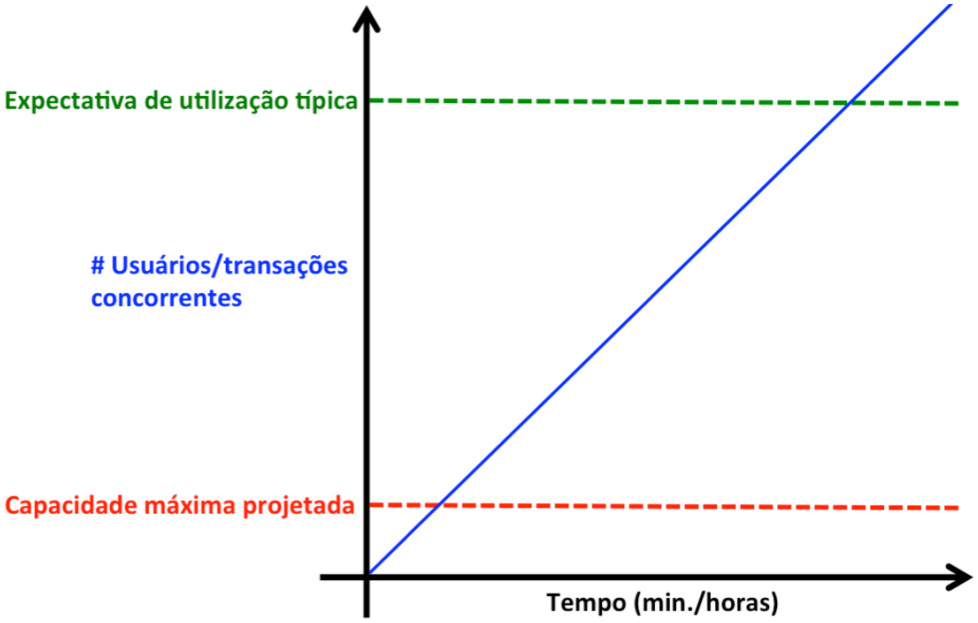


Figura 5.10. Teste de estresse de recursos insuficientes.

O **teste de capacidade**, representado na Figura 5.11, é um tipo de teste de desempenho cujo objetivo é avaliar como o sistema se comporta em termos de tempo de resposta e utilização de recursos, à medida em que se aumenta a carga. Trata-se de um tipo de teste com características parecidas com o teste de estresse típico. Porém, no teste de capacidade, a carga é aumentada de forma iterativa - para cada patamar, analisa-se o desempenho do sistema. Desta forma, testa-se a capacidade do sistema sob diferentes cargas até encontrar aquela cujo desempenho do sistema seja satisfatório para o cliente.

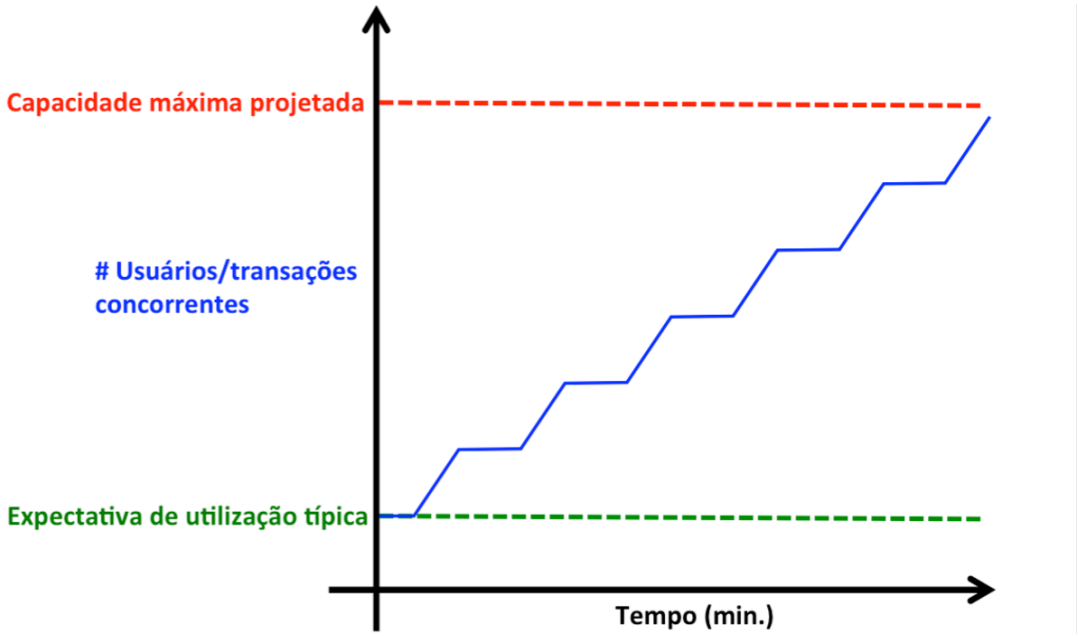


Figura 5.11. Teste de capacidade.

O **teste de resistência**, ou *soak testing*, é um tipo do teste de desempenho de longa duração usado para determinar o desempenho e/ou a estabilidade do sistema ao longo do tempo. Um sistema pode funcionar bem por uma ou duas horas e, em seguida, começar a ter problemas. Geralmente, um teste de resistência consiste em executar testes de carga continuamente (em *loop*) por cerca de 24h ininterruptas, com uma carga acima da expectativa de utilização típica, conforme a Figura 5.12. Esses testes são especialmente úteis para identificar vazamentos de memória (*memory leaks*).

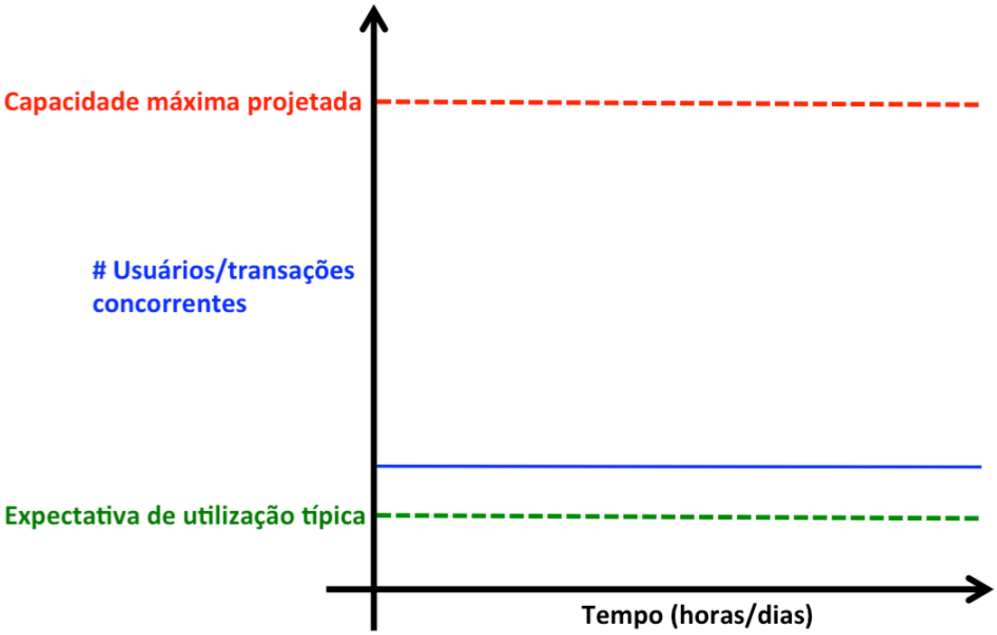


Figura 5.12. Teste de resistência.

De forma geral, testes de desempenho se referem à prática de simular vários usuários acessando o programa de modo concorrente (não necessariamente simultâneo). No entanto, há situações em que se deseja avaliar qual o volume de dados que o sistema é capaz de processar, independentemente da quantidade de usuários concorrentes. Por exemplo, um teste de carga pode envolver a execução de 100 usuários concorrentes realizando pedidos de compra típicos. Já um **teste de volume**, poderia envolver o processamento de uma única requisição com um pedido de compra que explorasse todos os limites de cardinalidade dos atributos, submetendo ao sistema uma grande quantidade de dados. O teste de volume, portanto, implica em submeter o sistema ao processamento da quantidade máxima de dados para a qual foi projetado, como ilustra a Figura 5.13.

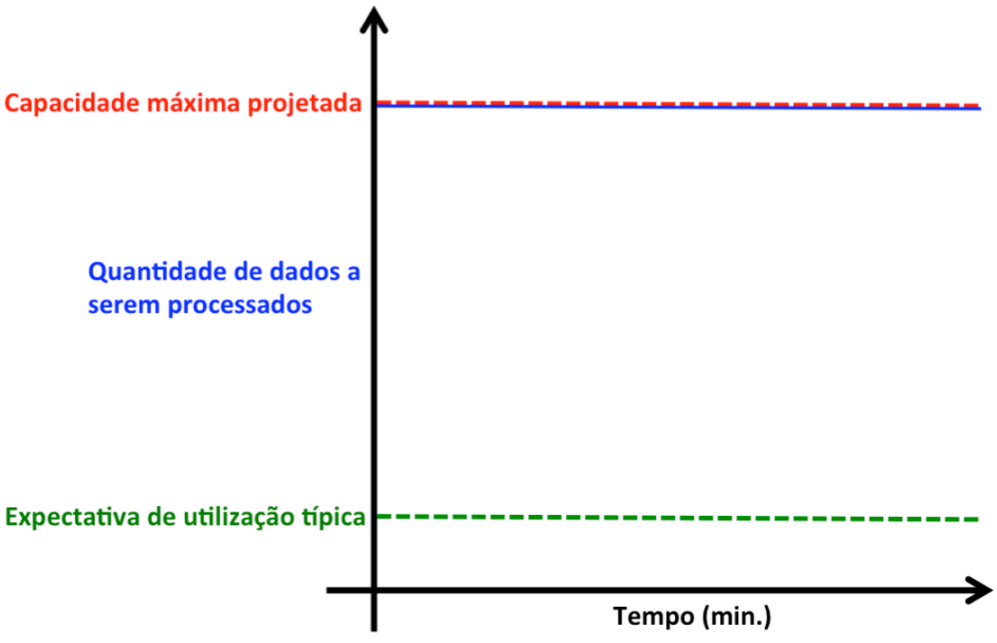


Figura 5.13. Teste de volume.

5.4. Apache JMeter

O Apache JMeter é uma ferramenta de código aberto desenvolvida em Java utilizada para realização de testes de desempenho. A primeira versão do JMeter foi lançada em 1998. Ele foi originalmente projetado para testar aplicações Web, mas desde então passou a cobrir diversas outras tecnologias de software cliente/servidor, como conexões de bancos de dados JDBC, serviços de mensageria JMS, serviços de diretórios LDAP, entre outros [Apache Foundation 2018].

O JMeter também pode ser utilizado para realização de testes funcionais, uma vez que possui recursos que possibilitam automatizar testes desse tipo, tais como asserções. No JMeter, uma asserção corresponde a uma condição que deve ser satisfeita pela resposta a uma requisição [Erinle 2017]. Há diversos tipos de asserções disponíveis, tais como as asserções de resposta, que possibilitam avaliar, entre outros itens, o conteúdo das respostas. Essa flexibilidade faz do JMeter uma das ferramentas de testes mais utilizadas na indústria [Erinle 2014].

Um *script* de teste no JMeter é organizado de forma hierárquica. O elemento mais ao topo dessa hierarquia é o Plano de Teste (*Test Plan*). Um Plano de Teste completo consistirá em um ou mais grupos de usuários (*threads*), controladores lógicos, requisições, ouvintes, temporizadores, asserções e elementos de configuração [Apache Foundation 2018]. A Figura 5.14 representa a interface gráfica do JMeter. À esquerda, é possível observar um exemplo de Plano de Teste com seus respectivos componentes, o que corresponde a um *script* de teste.

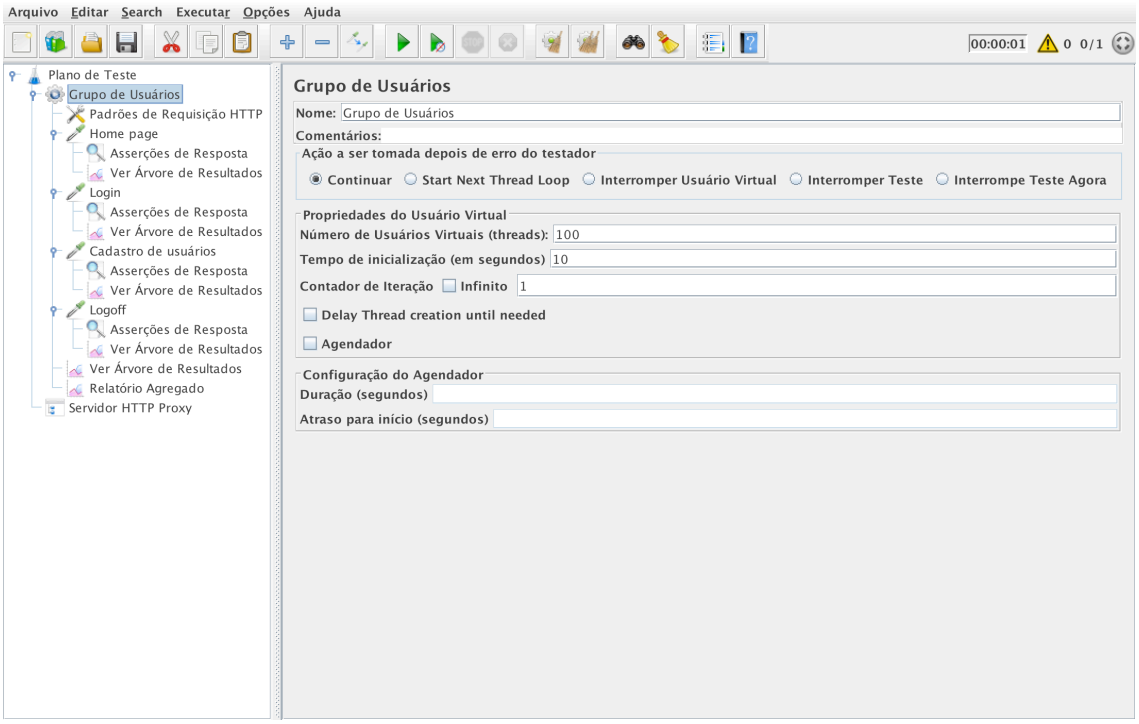


Figura 5.14. Interface gráfica do Apache JMeter.

A seção a seguir descreve os principais recursos do JMeter por meio de um exemplo prático. Para tal, utilizou-se a versão mais recente da ferramenta no momento da escrita desse texto, a versão 5.0.

5.5. Exemplo Prático

Para demonstrar os conceitos apresentados e o uso da ferramenta JMeter, esta seção apresenta um exemplo prático de testes de desempenho sobre uma aplicação Web. Levando-se em conta que o JMeter é uma ferramenta com enorme variedade de recursos, as possíveis soluções (*scripts*) para um mesmo problema tendem ao infinito. Portanto, não se pretende cobrir todos os recursos do JMeter, muito menos apresentar uma solução como a única possível.

Este exemplo prático é uma adaptação livre de exemplos encontrados na obra de Matam & Jain (2017). Trata-se do sistema fictício de comércio eletrônico "Digital Toys", disponível em https://github.com/jmeterbyexample, juntamente com diversos *scripts* de teste do JMeter. A Figura 5.15 apresenta a *home page* do sistema Digital Toys.

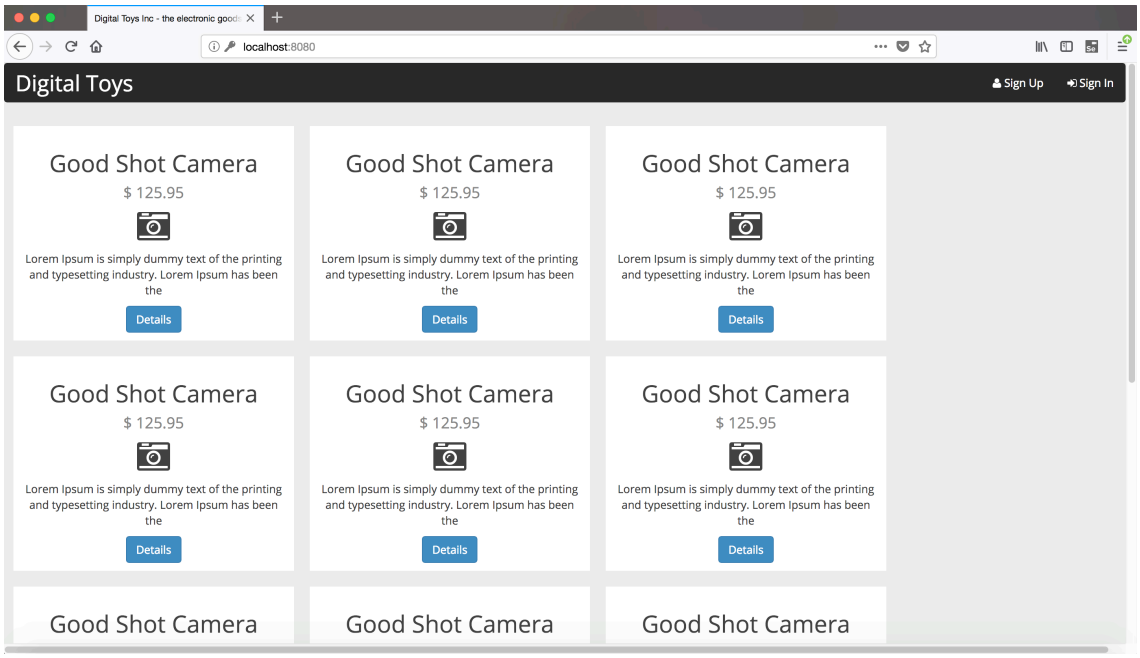


Figura 5.15. *Home page* do Digital Toys.

Como um sistema típico de comércio eletrônico, o Digital Toys possui funcionalidades que possibilitam a compra de produtos, tais como eletroeletrônicos e livros. Dentre as funcionalidades mais importantes do sistema estão as seguintes:

- Realizar compra: funcionalidade mais importante do sistema, tanto do ponto-de-vista arquitetural quanto de negócio. Por meio desta funcionalidade, os clientes do Digital Toys realizam a compra de produtos comercializados pelo sistema.
- Manter produtos: API REST que possibilita ao administrador do sistema Digital Toys realizar o cadastro, a consulta, a alteração e a remoção de produtos.
- Cadastrar cliente: funcionalidade que permite ao usuário interessado em realizar compras no Digital Toy se cadastrar no sistema.
- Autenticar usuário ("*Sign Up*"): recurso que possibilita aos usuários realizarem o *login* no sistema.

- Converter moeda: trata-se de um *web service* SOAP que é consumido sempre que um cliente opta por realizar o pagamento em uma moeda diferente do dólar americano.

Considerando as funcionalidades apresentadas, o testador definiu o modelo de carga representado na Tabela 5.3. Como se pode observar, estima-se que a funcionalidade "Listar produtos" (representada pelo cenário de teste "Listar 1.000 livros"), que compõe o caso de uso "Manter produtos", é a funcionalidade que representa a maior parte (40%) da carga do sistema em momentos de pico. É importante notar que esse cenário de teste indica a quantidade de registros que devem ser retornados pelo sistema, tendo em vista que não seria justo comparar tempos de resposta de duas ou mais execuções com quantidades diferentes de registros retornados.

Tabela 5.3. Modelo de carga do Digital Toys.

#CT	Cenário de teste	% da carga total	Usuários	Rampa	Agentes	Tempo de resposta tolerável
CT1	Cadastrar cliente	10%	100	10s	1	0,5s
CT2	Realizar compra	30%	300	10s	2	3s
CT3	Converter moeda	5%	50	5s	1	2s
CT4	Cadastrar livro	15%	150	10s	1	1s
CT5	Listar 1.000 livros	40%	400	20s	2	8s

Outro aspecto importante de se observar na Tabela 5.3 é que, dada a quantidade de usuários concorrentes, optou-se por executar os cenários de teste CT2 e CT5 a partir de dois agentes. Um **agente** é cada máquina que compõe o ambiente de ativação. O **ambiente de ativação** é o conjunto de máquinas responsáveis por disparar requisições concorrentes sobre o ambiente-alvo. O **ambiente-alvo** é o servidor para o qual são direcionadas as requisições provenientes do ambiente de ativação (trata-se do servidor que hospeda o sistema sob teste).

Sendo assim, a título de exemplo, são descritas a seguir as principais etapas para implementação do *script* de teste referente ao cenário de teste CT1 ("Cadastrar cliente"). Esse cenário é composto pelos seguintes passos, que devem ser reproduzidos pelo *script* do JMeter:

1. Acessar *home page*;
2. Escolher opção para cadastrar cliente ("*Sign Up*");
3. Preencher formulário de cadastro e confirmar ("*Submit*");
4. Sair do sistema ("*Sign Out*").

Definido o conjunto de passos do cenário a ser testado, deve-se proceder a implementação do *script* correspondente no JMeter. Quando o JMeter é aberto, a estrutura do *script* já se apresenta com um Plano de Teste vazio. Todos os elementos do *script* devem ser incluídos hierarquicamente abaixo desse Plano de Teste. A Figura 5.16 representa a interface gráfica do JMeter com um Plano de Teste.

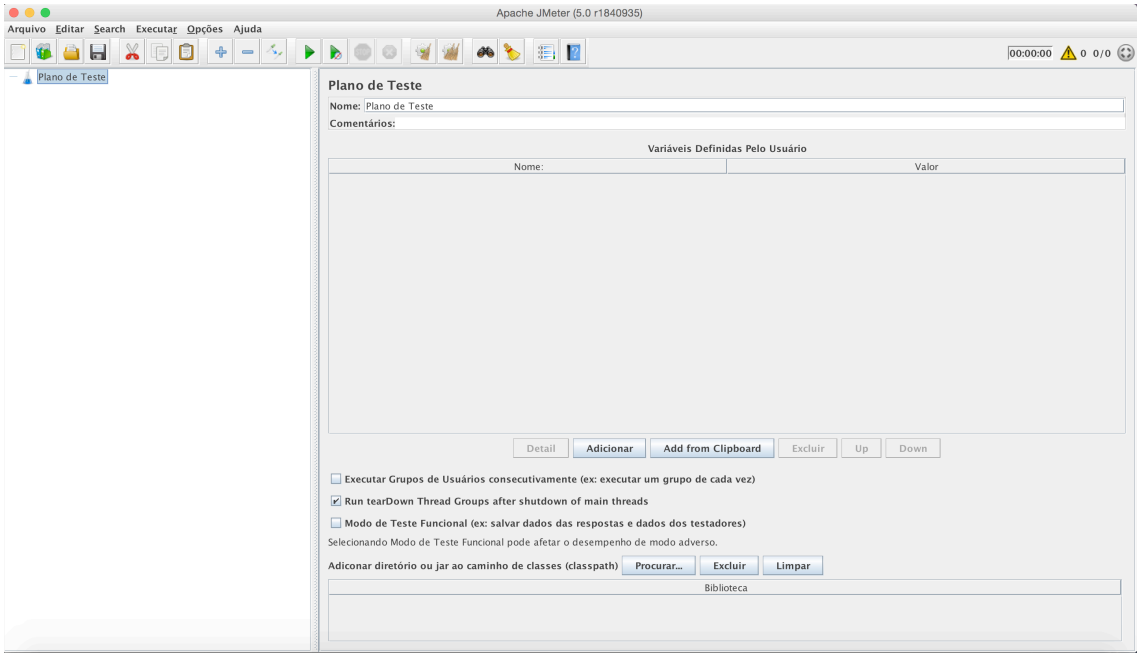


Figura 5.16. Plano de teste do JMeter.

Em seguida, deve-se incluir um Grupo de Usuários (*Thread Group*) em um nível abaixo do Plano de Teste, conforme a Figura 5.17. Nesse componente são definidos, entre outros, o número de usuários concorrentes e a rampa de subida, respectivamente nos campos "Número de Usuários Virtuais (*threads*)" e "Tempo de inicialização (em segundos)". Inicialmente, deve-se mantê-lo com os valores-padrão (1). Após a implementação do *script*, esses campos devem ser atualizados com os respectivos valores, de acordo com o modelo de carga.

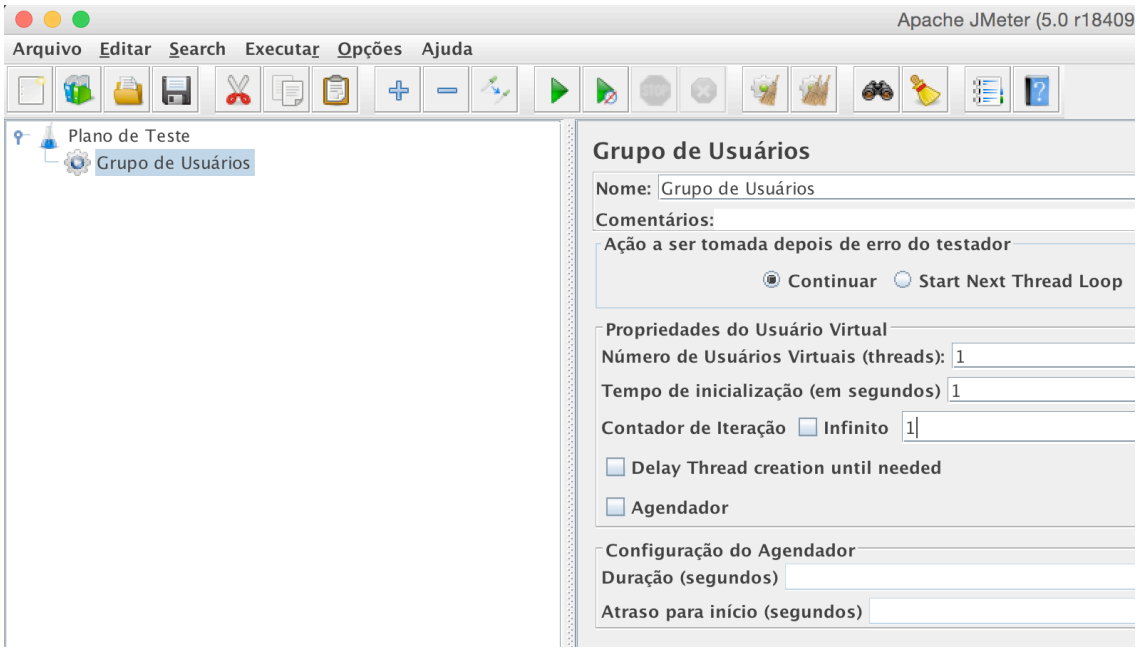


Figura 5.17. Inclusão do Grupo de Usuários.

A próxima etapa consiste na definição das requisições HTTP correspondentes a cada passo do cenário de teste. Esta etapa pode ser realizada de duas formas: manual ou automática. A forma manual consiste de se criar cada requisição HTTP no JMeter, imediatamente abaixo do Grupo de Usuários. No entanto, dependendo da quantidade e da complexidade das requisições necessárias, pode ser mais interessante mapeá-las de forma automática, utilizando um *proxy* definido no próprio JMeter. Para tal, deve-se incluir um "Servidor HTTP Proxy" no nível abaixo do Plano de Teste. O "Servidor HTTP Proxy" deve ser configurado com a indicação da porta que irá receber as requisições durante a execução dos testes, qual o controlador-alvo do *script* que irá receber as requisições mapeadas, entre outras informações. Neste exemplo, conforme a Figura 5.18, foi definida a porta 8888 (que deve ser configurada também no navegador) e as requisições mapeadas serão direcionadas para o controlador "Cadastrar cliente" (controlador simples).

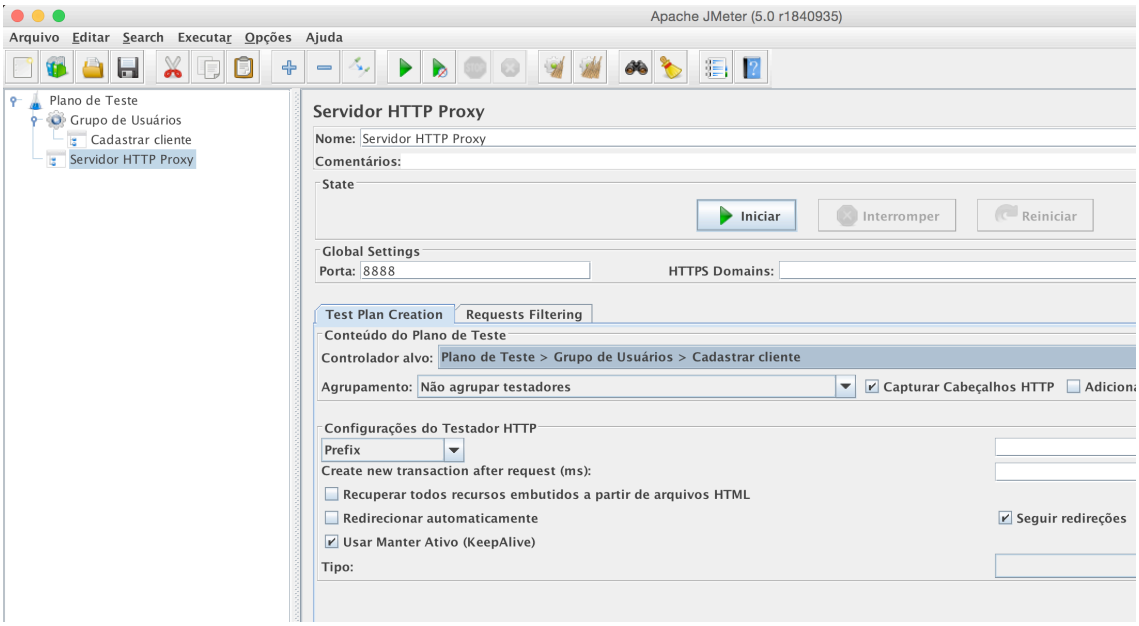


Figura 5.18. Inclusão de um Servidor HTTP Proxy.

Além disso, como o *proxy* é capaz de capturar todas as requisições feitas para quaisquer recursos HTTP, recomenda-se aplicar um filtro de modo a excluir requisições sobre recursos como arquivos CSS ou JS, tornando o *script* mais compreensível. Esse filtro pode ser aplicado na aba "*Requests Filtering*", seção "Padrões de URL a serem excluídos", clicando-se no botão "Adicionar" e informando as extensões de arquivos a serem excluídos. Pode-se, também, clicar no botão "*Add suggested Excludes*", que insere uma expressão regular informando várias extensões de arquivos que são comumente excluídos. A Figura 5.19 representa essa opção.

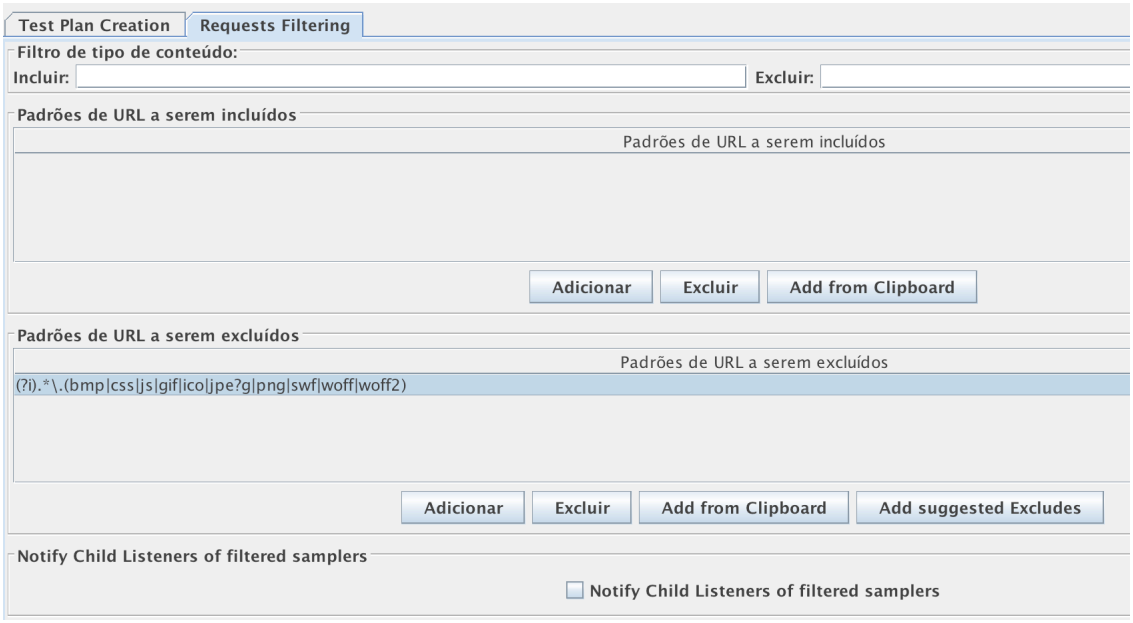


Figura 5.19. Definição de padrões a serem excluídos da gravação.

Após a configuração do "Servidor HTTP Proxy", deve-se clicar no botão "Iniciar" deste próprio componente. Em seguida, deve-se acessar a aplicação sob teste. A partir desse ponto já é possível observar que o JMeter está capturando as requisições realizadas pelo testador. Portanto, o primeiro passo consiste de acessar a *home page* do sistema Digital Toys, representado anteriormente na Figura 5.15. Em seguida, o testador deve clicar na opção "*Sign Up*", que representa o segundo passo do cenário CT1. Com isso, um formulário de cadastro de cliente será exibido, conforme a Figura 5.20. Esse formulário deve ser preenchido e submetido.

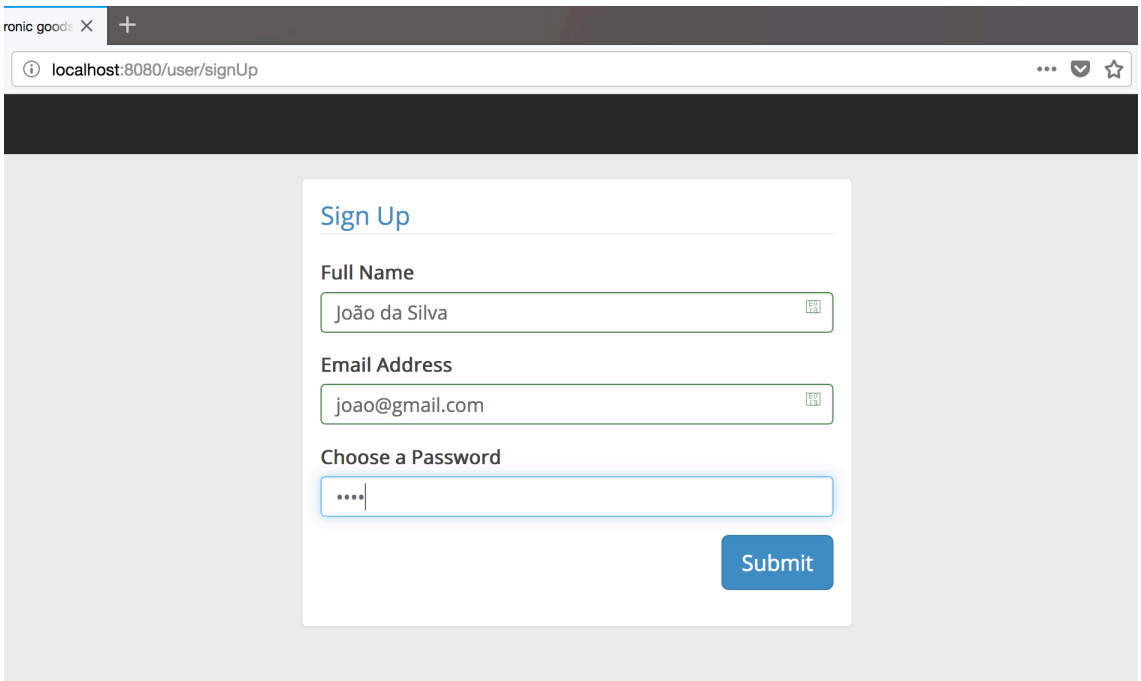


Figura 5.20. Formulário de cadastro de novo cliente.

Após a submissão do formulário de cadastro, o usuário é redirecionado para a *home page* do sistema, porém, com a indicação de que está autenticado. A Figura 5.21 destaca que o usuário recém-criado "João da Silva" está autenticado. Por fim, o testador deve clicar na opção "*Sign Out*" (sair), também representada na Figura 5.21, que corresponde ao último passo do cenário de teste.

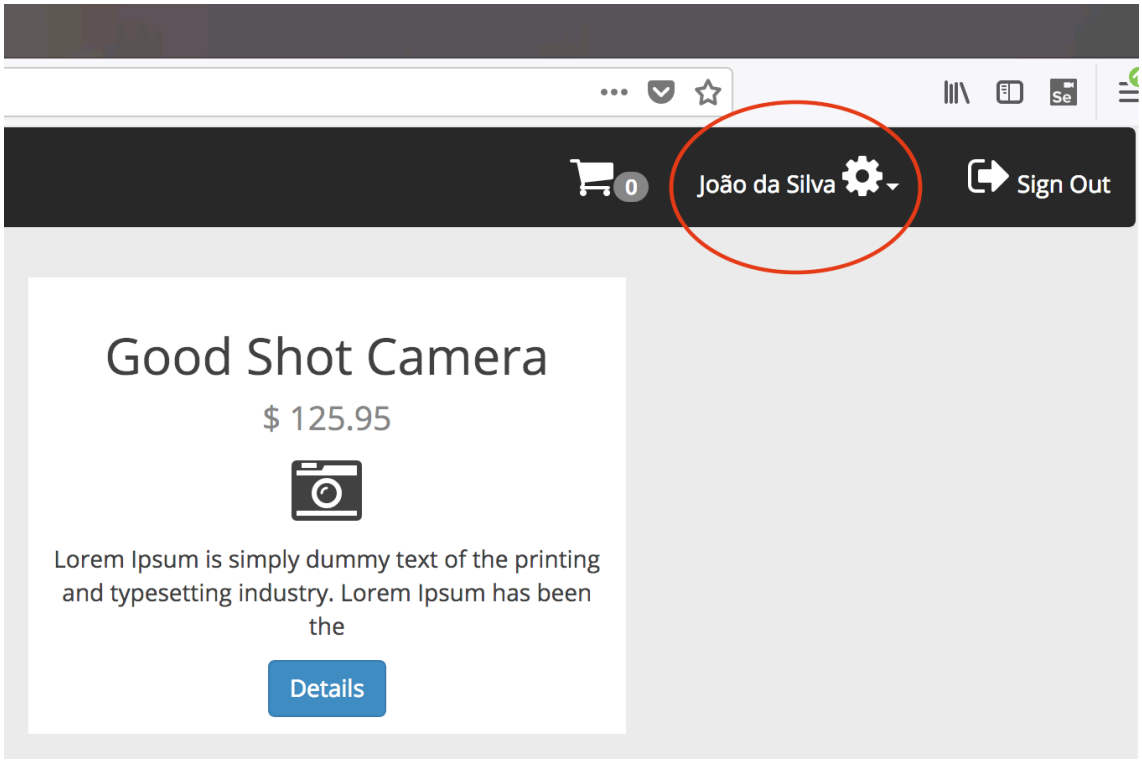


Figura 5.21. Indicação de usuário autenticado no sistema.

Uma vez executados todos os passos do cenário de teste, deve-se finalizar o proxy do JMeter clicando no botão correspondente. Feito isso, deve-se observar as requisições que foram mapeadas pelo *proxy*. A Figura 5.22 representa as quatro requisições HTTP mapeadas dentro do controlador "Cadastrar cliente" (todas foram renomeadas posteriormente). Ainda na Figura 5.22 pode-se observar em detalhes a requisição "Submeter formulário de cadastro". Trata-se de um requisição do tipo POST que passa três parâmetros: "*fullName*", "*email*" e "*password*".

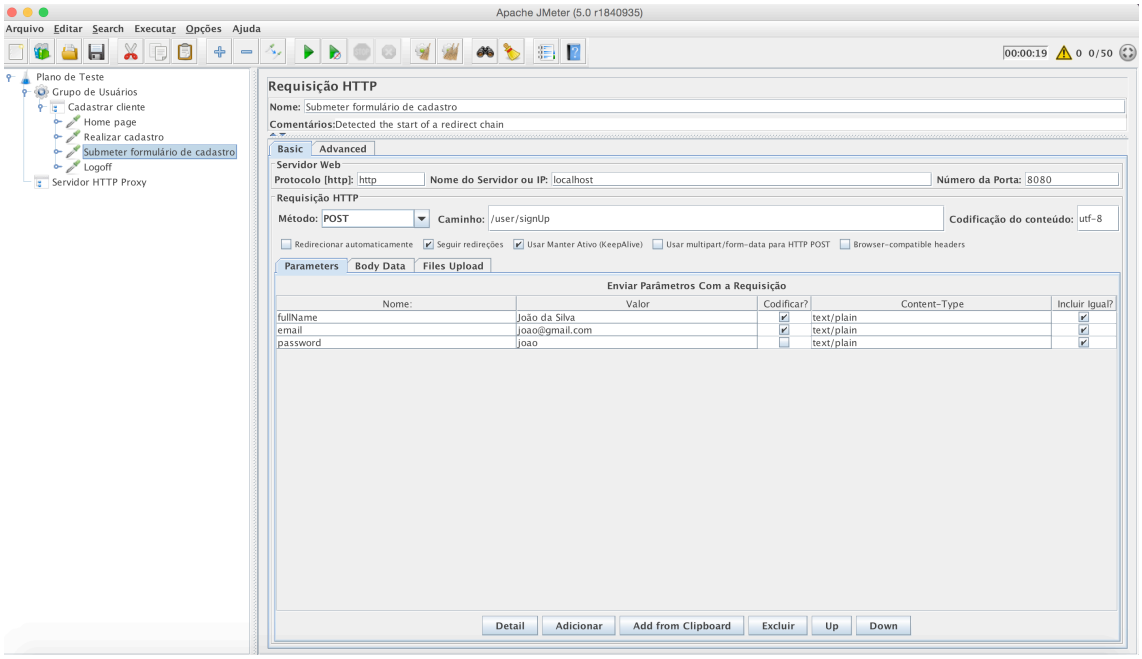


Figura 5.22. Requisições HTTP mapeadas pelo proxy.

Outra etapa importante é a inclusão de ouvintes no *script*. Um ouvinte (ou "*listener*") é uma espécie de relatório que representa os resultados das execuções das *threads* (usuários virtuais). Há diversos tipos de ouvintes para os mais variados gostos. Neste caso, foram incluídos os ouvintes "Ver Árvore de Resultados" e "Relatório Agregado", como mostra a Figura 5.23. Como eles foram adicionados no nível abaixo de Grupo de Usuários, as respostas de todas as requisições serão exibidas neles. Também é possível incluir ouvintes específicos por requisição, bastando que sejam incluídos no nível hierárquico abaixo da requisição desejada.

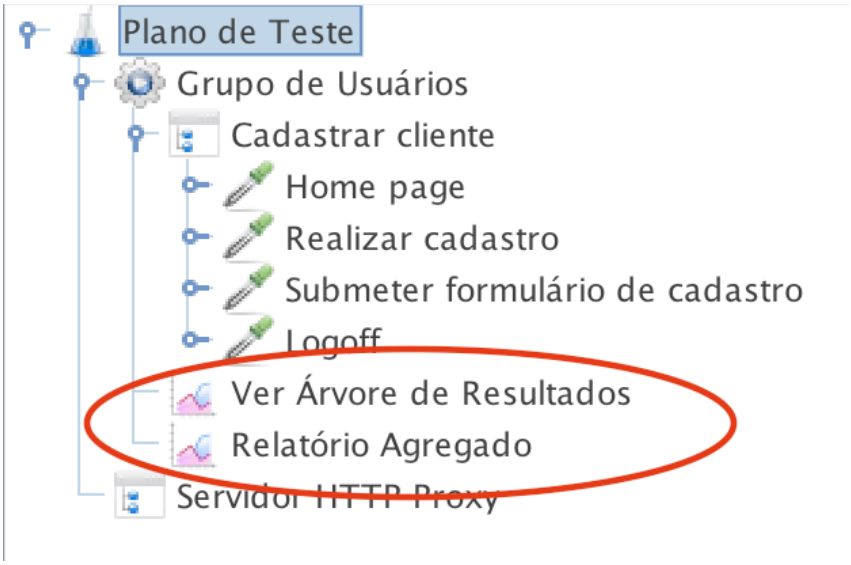


Figura 5.23. Inclusão de ouvintes (*listeners*).

Portanto, após definidos os componentes principais do *script*, deve-se retornar ao Grupo de Usuários para definir a quantidade de usuários concorrentes e a rampa de subida, como representado na Figura 5.24. Desta forma, a cada segundo, serão disparadas dez requisições a partir do ambiente de ativação.

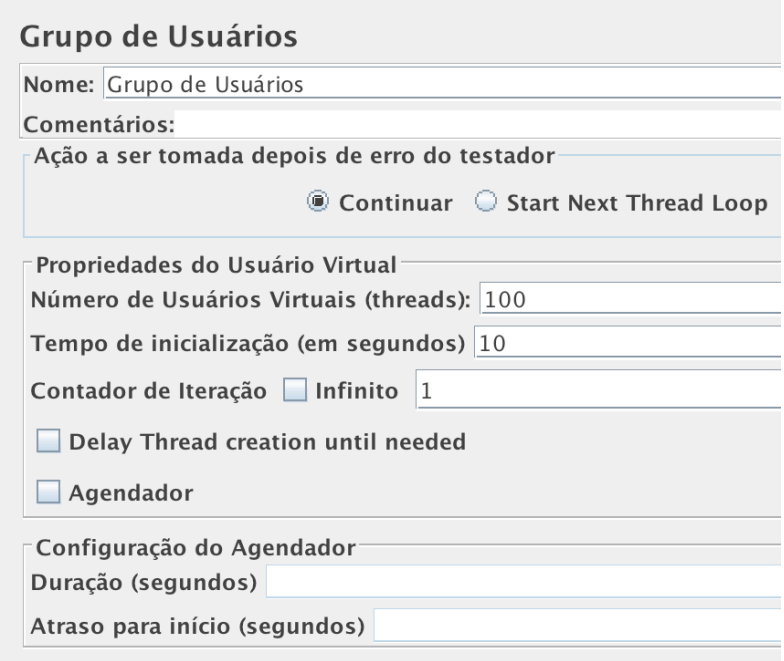


Figura 5.24. Definição da quantidade de usuários concorrentes e da rampa de subida.

Após a configuração do Grupo de Usuários, o testador deve clicar no botão "Iniciar" na barra de ferramentas. A partir de então, as requisições serão disparadas ao ambiente-alvo. Na mesma barra, no canto direito, um contador indica ao longo da execução do teste a quantidade de requisições que ainda faltam ser processadas.

Após o término da execução, o testador deve observar os resultados por meio dos ouvintes. No ouvinte "Ver Árvore de Resultados", representado na Figura 5.25, pode-se observar quais requisições foram processadas com sucesso e quais falharam, quais cabeçalhos e qual o conteúdo de cada resposta, entre outros recursos. A grande vantagem desse ouvinte é a possibilidade de se realizar uma análise individualizada sobre requisições e suas respectivas respostas. No entanto, ele não fornece informações suficientes para determinar se o requisito de desempenho definido para o cenário de teste foi satisfeito ou não.

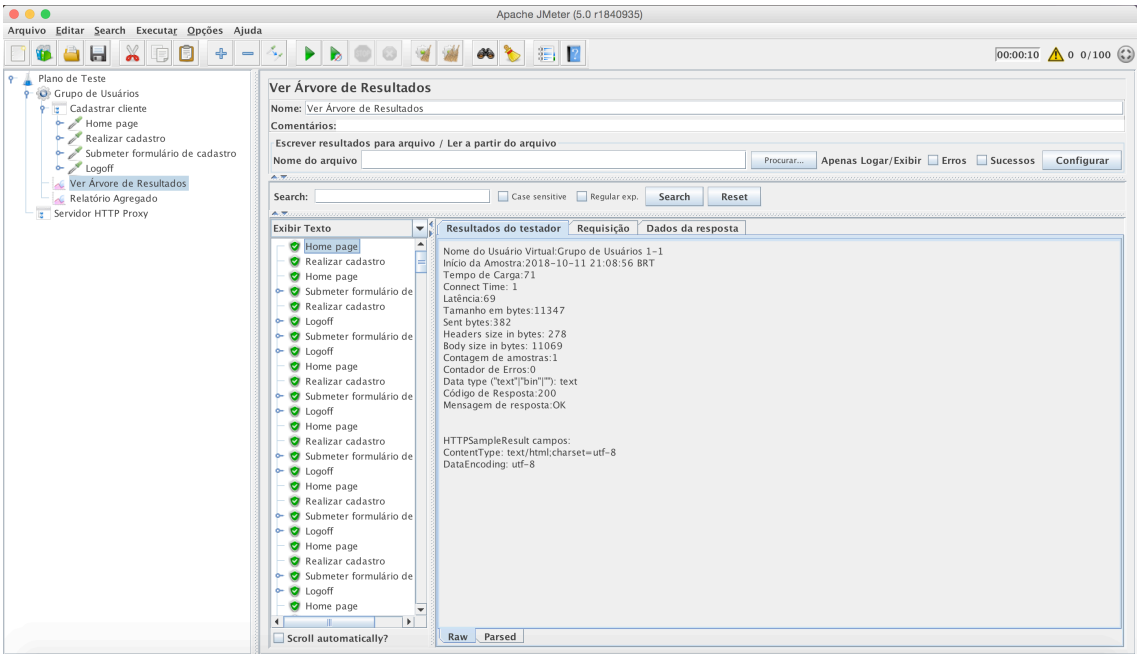
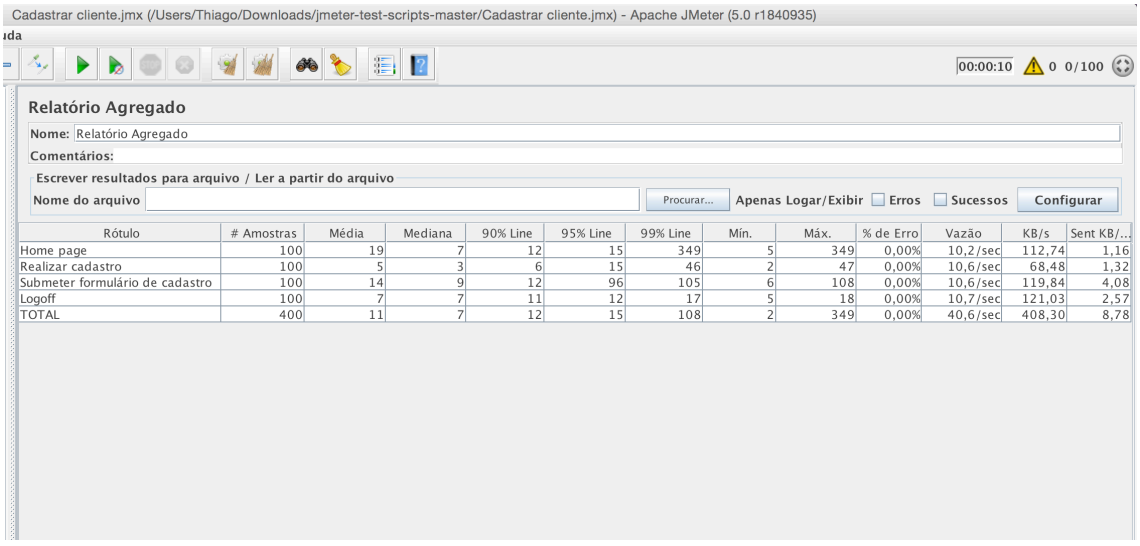


Figura 5.25. Resultados da execução no ouvinte "Ver Árvore de Resultados".

Já o ouvinte "Relatório Agregado", representado na Figura 5.26, traz informações suficientes para determinar se o cenário de teste CT1 passou ou falhou. Neste cenário, o tempo de resposta tolerado para uma carga de 100 usuários em uma rampa de 10s é de 0,5s. Recomenda-se, não utilizar a média como critério para aceitação de requisito de desempenho, tendo em vista que se trata de uma medida estatística muito sensível a valores extremos (*outliers*). Dependendo do nível de criticidade do sistema, recomenda-se utilizar o tempo de resposta nas linhas de 90%, 95% ou 99%. Para um sistema típico de comércio eletrônico a linha de 90% parece ser a mais adequada. Essa linha indica o tempo de resposta no 90º. percentil, isto é, organizadas as 100 respostas em função do tempo de resposta, a linha de 90% corresponde ao tempo de resposta da 90ª. resposta mais lenta.



Rótulo	# Amostras	Média	Mediana	90% Line	95% Line	99% Line	Mín.	Máx.	% de Erro	Vazão	KB/s	Sent KB/...
Home page	100	19	7	12	15	349	5	349	0,00%	10,2/sec	112,74	1,16
Realizar cadastro	100	5	3	6	15	46	2	47	0,00%	10,6/sec	68,48	1,32
Submeter formulário de cadastro	100	14	9	12	96	105	6	108	0,00%	10,6/sec	119,84	4,08
Logoff	100	7	7	11	12	17	5	18	0,00%	10,7/sec	121,03	2,57
TOTAL	400	11	7	12	15	108	2	349	0,00%	40,6/sec	408,30	8,78

Figura 5.26. Resultados da execução no ouvinte "Relatório Agregado".

Portanto, considerando que o tempo total na linha de 90% indicado no Relatório Agregado foi de 12ms, o requisito de desempenho foi satisfeito. No entanto, vale lembrar que neste exemplo o sistema Digital Toys está hospedado na mesma máquina que dispara os testes, o que certamente afeta os resultados, já que o ambiente real seria composto por redes de computadores e inúmeros dispositivos que contribuiriam para aumentar os tempos de resposta.

5.6. Recomendações Práticas

Esta seção apresenta um conjunto de recomendações que servem para orientar a realização de testes de desempenho. Esse conjunto de recomendações reflete a experiência do autor com esse tipo de teste, não representando, portanto, uma lista exaustiva de recomendações.

5.6.1. Conheça os conceitos relacionados a testes de desempenho

Conhecer os conceitos relacionados a testes de desempenho é fundamental para a realização de um bom trabalho. Além disso, a ausência de uma terminologia comum pode afetar negativamente a comunicação da equipe do projeto.

5.6.2. Conheça os requisitos de desempenho do sistema a ser testado

Leia o documento de requisitos não-funcionais (RNF) do projeto e converse com a equipe responsável. Procure identificar e compreender os requisitos de desempenho de modo que se tenham todas as informações necessárias para a realização do teste. Se for necessário, aplique o questionário a seguir, que pode ser complementado com outras perguntas que forem pertinentes.

1) Quais são os requisitos de desempenho descritos no documento de RNF?
2) Resumidamente, como é a arquitetura da aplicação a ser testada?
3) Quais são os casos de uso arquiteturalmente significativos?
4) Para cada caso de uso arquiteturalmente significativo, quais são suas expectativas de:
    a) quantidade de usuários concorrentes em horários de pico?
    b) tempo de resposta aceitável, considerando o pico de usuários concorrentes?
5) Considerando as respostas da pergunta 4, quais cenários de teste se mostram relevantes?
6) Qual a massa de teste necessária para cada cenário de teste escolhido? Como ela pode ser gerada?
7) Há integrações com outros sistemas nos cenários de teste escolhidos? Quais sistemas?
8) Que ambiente será utilizado nos testes de desempenho? Caso não seja o ambiente de produção, o ambiente escolhido é semelhante ao de produção em termos de hardware e software?

9) Há restrições de horários de execução dos testes, considerando o ambiente de testes escolhido?

10) Há testes funcionais automatizados?

5.6.3. Planeje e documente o projeto de testes de desempenho

Testes de desempenho costumam demandar muitos recursos e podem impactar diversos sistemas. Sendo assim, é fundamental que esses testes sejam planejados, de modo a otimizar o uso dos recursos e a evitar o retrabalho ("queimar cartuchos"). Além disso, tratam-se de testes que serão executados diversas vezes, eventualmente, por profissionais diferentes. É preciso garantir o mesmo estado do sistema e de variáveis de contexto a cada execução. Desta forma, é necessário que todos os procedimentos de execução sejam documentados para garantir a repetibilidade dos testes e, consequentemente, a consistência dos resultados. Portanto, documente as informações referentes ao planejamento dos testes de desempenho em um Plano de Testes.

5.6.4. Defina um modelo de carga

No contexto de testes de desempenho, o modelo de carga (*workload model*) é um dos itens mais importantes do Plano de Testes. Um modelo de carga descreve como um sistema será usado no ambiente de produção em termos de distribuição da carga de trabalho [Molyneaux 2014]. Para obter resultados confiáveis, o teste de desempenho deve se basear em um modelo de carga que retrate como o sistema será utilizado, levando-se em conta a quantidade de acessos concorrentes, o volume de dados transmitidos, os tempos de resposta toleráveis, a expectativa de consumo de recursos, etc. A Tabela 5.3, apresentada anteriormente, representa um exemplo de modelo de carga de um sistema que deve suportar 1.000 usuários concorrentes realizando diferentes transações.

5.6.5. Represente a arquitetura física do sistema

Observe o sistema sob teste como um encanamento. A bitola dos canos é equivalente à largura de banda da rede, enquanto que a vazão da água é equivalente ao *throughput* do sistema. A representação do sistema mais próxima de um encanamento é o modelo da arquitetura física, como ilustra a Figura 5.27, onde estão representados os servidores, bases de dados, *links* e equipamentos de rede, entre outros elementos que formam a infraestrutura do ambiente do sistema. Esse tipo de representação pode orientar o planejamento de casos de teste mais significativos. Além disso, esse modelo pode apoiar a descoberta de gargalos durante a execução de testes de desempenho.

Modelos como esse costumam estar presentes em documentos de arquitetura do software. Caso não haja, recomenda-se que sejam representados, com o apoio dos arquitetos do projeto, e incluídos no Plano de Testes.

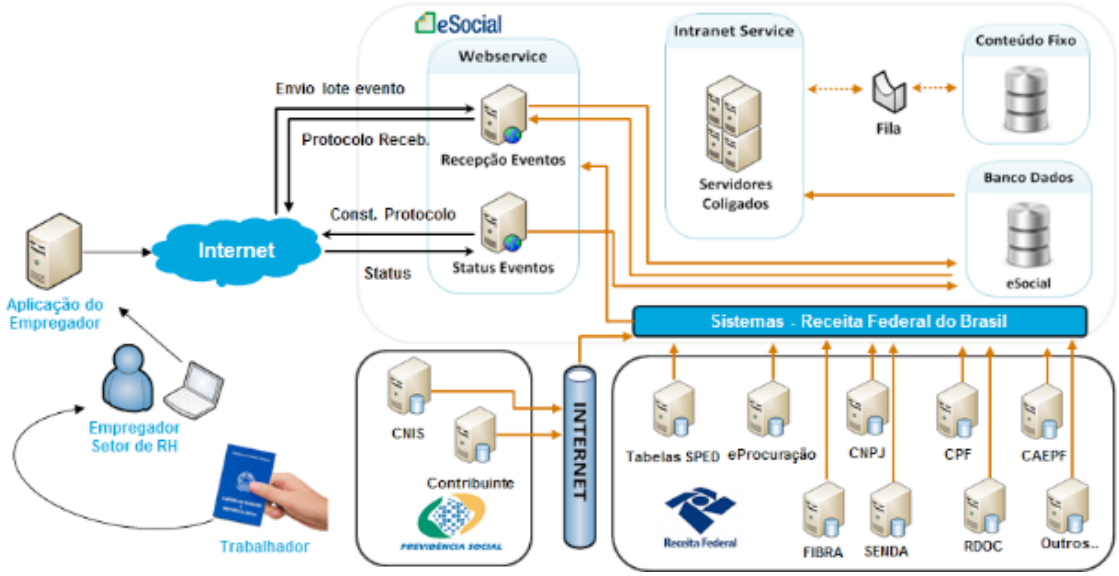


Figura 5.27. Exemplo de arquitetura física de um sistema [Brasil 2018].

5.6.6. Projete testes realistas

Os resultados do teste de desempenho serão mais precisos se for possível simular os cenários de uso do sistema de forma realista, preferencialmente em ambiente de produção. Testes não-realistas irão produzir resultados inúteis e que poderão levar a conclusões equivocadas.

Há pelo menos três elementos determinantes para a definição de casos de teste de desempenho realistas: 1) a quantidade de usuários/transações concorrentes, 2) a relação rampa de subida/número de agentes e 3) o *think time*. Veja os exemplos a seguir:

- Em um sistema com 500 usuários cadastrados e pico de acesso em torno de 150 usuários concorrentes, não seria realista um teste com mil usuários concorrentes. Portanto, a quantidade de usuários cadastrados, ou que se espera ter cadastrados, pode ser usada como parâmetro para delimitar a quantidade de usuários realizando acessos concorrentes em um cenário de teste.
- Um teste com 1.000 *threads* e uma rampa de subida de 100 segundos, quando disparado por apenas um agente, resulta em 10 requisições submetidas sequencialmente ao ambiente-alvo a cada segundo. Mas se o mesmo teste for executado a partir de vinte agentes (50 *threads* em cada), teoricamente, o ambiente-alvo poderá ficar 2s inteiros sem receber requisições, bem como, receber 20 requisições na mesma fração de segundo, provocando um cenário irreal de uso de recursos. Sendo assim, é necessário equilibrar adequadamente a relação entre a rampa de subida e o número de agentes, mesmo que para isso seja necessário executar diversos testes a título de calibração, até encontrar a medida ideal.
- Em cenários de teste onde o usuário navega por diversas páginas e, consequentemente, submete várias requisições ao servidor, é necessário simular o tempo em que ele passa interagindo com a aplicação. Esse tempo é conhecido como *think time*, o qual representa o tempo em que o usuário pensa e toma uma

decisão de interação com o sistema (p. ex., confirmar ou cancelar uma operação). Quando não se considera o *think time*, a execução dos testes de desempenho se torna não-realista, uma vez que as requisições de cada usuário virtual serão executadas imediatamente uma após a outra, o que não aconteceria em um cenário real.

5.6.7. Mantenha os *scripts* e demais artefatos sob controle de versão

Assim como qualquer outro artefato do projeto, os *scripts* e demais artefatos referentes a testes de desempenho devem ser mantidos sob controle de versão, tendo em vista que eles podem ser implementados de forma iterativa e colaborativa. Além disso, pode ser necessário reexecutar um cenário de teste de desempenho que foi executado há anos, para avaliar o impacto de uma manutenção recente em um sistema.

5.6.8. Execute os testes sobre um ambiente-alvo específico para testes de desempenho

Em um cenário ideal, os testes de desempenho seriam realizados sobre o ambiente de produção, mesmo que durante a noite ou fins de semana, pelo fato de proporcionar os resultados mais realistas possíveis. No entanto, são raros os casos em que esse cenário é viável. Também não se recomenda utilizar outros ambientes, como o de desenvolvimento, de testes funcionais ou de homologação, para realizar testes de desempenho por conta dos seguintes motivos:

- Geralmente, os ambientes de desenvolvimento, testes e homologação não refletem completamente as características de hardware e de software do ambiente de produção. É provável que processadores, memória, discos, *links* de rede, *firewalls*, a configuração de programas e o tamanho do bancos de dados sejam diferentes do ambiente de produção. Logo, não se pode concluir que um sistema apresentará tempos de resposta iguais em ambientes tão distintos.
- Quando se realizam testes de desempenho em ambientes compartilhados, o resultado do teste pode sofrer interferências das transações concorrentes que estavam sendo realizadas pelos desenvolvedores, testadores e usuários comuns. Da mesma forma, os testes de desempenho podem afetar o trabalho desses outros usuários, fornecendo respostas mais lentas ou, até mesmo, provocando a interrupção do sistema.

Portanto, recomenda-se o estabelecimento de um ambiente específico para testes de desempenho, criado à semelhança do ambiente de produção.

5.6.9. Execute os testes a partir de um ambiente de ativação específico para testes de desempenho

Em qualquer projeto de teste de software recomenda-se ter um ambiente específico de hardware e software para realização dos testes. Em testes de desempenho, particularmente, recomenda-se que a execução dos testes seja realizada com apoio de um ambiente de ativação dedicado a testes de desempenho, tendo em vista que um ambiente de ativação com hardware e *link* de rede compartilhados poderia afetar os resultados dos testes (por exemplo, provocando maior latência).

A Figura 5.28, que ilustra um cenário composto por três ambientes distintos: 1) um ambiente de implementação, no qual o profissional de teste desenvolve os *scripts* de teste de desempenho; 2) um ambiente de ativação, composto por diversos agentes, configurado para disparar *jobs* da ferramenta Jenkins que executam *scripts* do JMeter; 3) por fim, um ambiente-alvo, composto por duas camadas físicas (*tiers*), representando o sistema que recebe as requisições provenientes do ambiente de ativação.

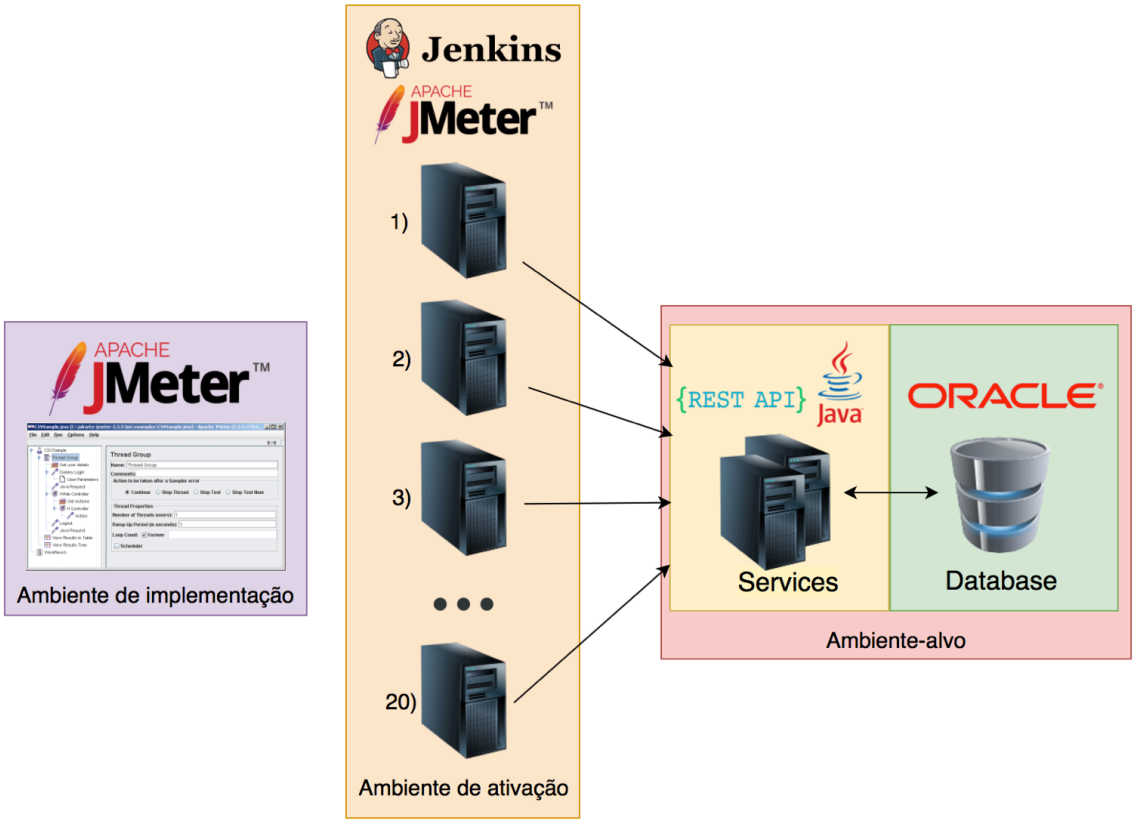


Figura 5.28. Diferentes ambientes utilizados em testes de desempenho.

5.6.10. Não utilize *listeners* no ambiente de ativação

O JMeter possui um conjunto de *listeners* (“ouvintes”) que exibem dados das requisições e das respostas. Em geral, esses *listeners* consomem uma quantidade significativa de memória da máquina que dispara os testes (especialmente os *listeners* “Ver Árvore de Resultados” e “Gráfico de Resultados”. Os *listeners* são muito úteis em tempo de implementação dos *scripts*, porém devem ser evitados nos *scripts* que serão executados pelo ambiente de ativação.

5.6.11. Execute os testes várias vezes e em diferentes horários

Uma única execução de um *script* de teste de desempenho pode não ser conclusiva. A cada execução, os testes podem apresentar tempos de resposta diferentes. Eventualmente, algumas execuções do mesmo cenário de teste apresentam tempos de resposta satisfatórios e outras não. Isso se deve a inúmeros fatores, tais como a latência da rede, esgotamento temporário de recursos, entre outros. Sendo assim, recomenda-se que um mesmo cenário de teste seja executado ao menos três vezes e em diferentes

horários para evitar que fatores momentâneos interfiram nos resultados de todas as execuções do teste.

5.6.12. Acompanhe a execução dos testes com uma ferramenta de monitoramento

O valor agregado ao projeto pelos testes de desempenho é potencializado quando o time realiza a monitoração dos servidores do ambiente-alvo, com o objetivo de identificar os gargalos. Portanto, recomenda-se que toda execução de testes de desempenho seja monitorada via ferramentas como o *Application Performance Management* (APM) [CA Technologies 2018], representado na Figura 5.29.

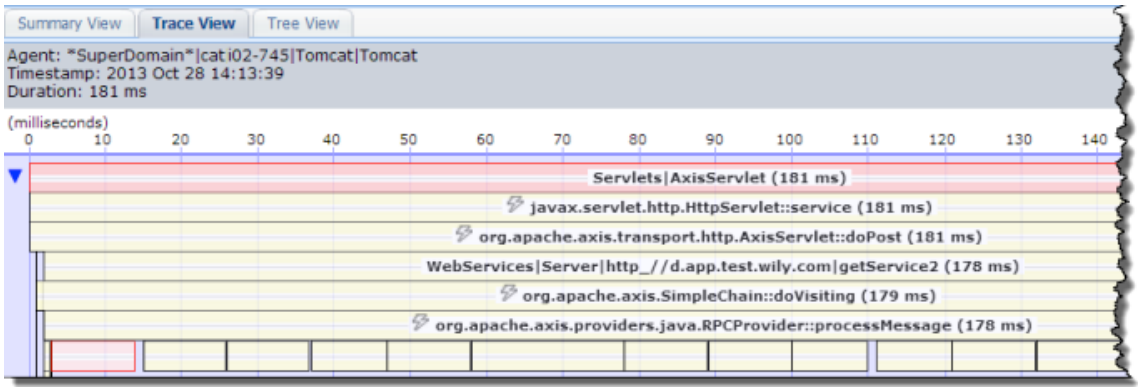


Figura 5.29. Visão da ferramenta APM.

5.6.13. Execute os testes cedo e com frequência

Tradicionalmente, os testes de desempenho são realizados quando o sistema está próximo de ser homologado ou colocado em produção (abordagem reativa). Isso se deve ao fato de que em um cenário ideal esse tipo de teste deva ser realizado em um produto completo e livre de defeitos (funcionais). No entanto, considerando a frequência de entregas nos novos projetos e a decomposição dos sistemas em partes cada vez menores (microsserviços), adotar uma abordagem proativa, antecipando esse tipo de teste, pode trazer uma série de benefícios, mesmo que não se tenha uma percepção definitiva do desempenho do sistema. Entre os benefícios observados, temos os seguintes:

- definição antecipada dos critérios de aceitação de desempenho;
- identificação dos riscos associados ao desempenho;
- descoberta antecipada de gargalos.

5.6.14. Comunique os resultados visualmente

Os resultados dos testes de desempenho são de interesse de toda a equipe. Portanto, comunique seus resultados a todos os interessados, de preferência, de forma visual. Uma sugestão é utilizar a ferramenta Grafana [Grafana Labs 2018], que possibilita a criação de *dashboards*, como representa a Figura 5.30.

Figura 5.30. *Dashboard* do Grafana.

1.7. Conclusão

Este trabalho apresentou aspectos teóricos e práticos a respeito de testes de desempenho de software. Quanto aos aspectos teóricos, mereceu destaque uma classificação que apresenta nove tipos de teste de desempenho tipicamente praticados na indústria. Essa classificação tem especial importância pelo fato de que não há consenso estabelecido na área a esse respeito, podendo levar a falhas de comunicação em projetos e a consequências sérias.

Já os aspectos práticos foram apresentados por meio de um exemplo prático de uso da ferramenta Apache JMeter e de um conjunto de recomendações práticas sobre testes de desempenho. O exemplo prático, apesar de simples, pode ser suficiente para um testador sair da "estaca zero" de conhecimento sobre a ferramenta JMeter. As recomendações prática apresentadas representam a experiência do autor com testes dessa natureza ao longo dos últimos três anos em projetos Web de larga escala. Provavelmente, outras recomendações podem ser adicionadas ao conjunto apresentado, especialmente em outros contextos de projeto de software.

Em relação a trabalhos futuros, especialmente para alunos de pós-graduação em Computação, sugere-se a criação de um método para apoiar a definição de modelos de carga, em especial, a definição da relação entre usuários concorrentes, rampa de subida e número de agentes. Geralmente, essa tarefa é realizada por tentativa e erro, como uma espécie de calibração. Um método que indicasse a quantidade ideal de cada um desses três parâmetros para cada cenário de teste seria útil para a maioria das organizações que realizam esse tipo de teste.

Referências Bibliográficas

Apache Foundation (2018) *Apache JMeter 5*. Disponível em: <https://jmeter.apache.org>. Acesso em: agosto, 2018.

Brasil (2018) *Sistema eSocial: Manual de Orientação do Desenvolvedor*. Disponível em: <http://portal.esocial.gov.br/manuais/manualorientacaodesenvolvedoresocialv1-7.pdf>. Acesso em: setembro, 2018.

Bondi, A. B. (2014). *Foundations of software and system performance engineering: process, performance modeling, requirements, testing, scalability, and practice*. Pearson Education.

CA Technologies (2018) *Application Performance Management (APM)*. Disponível em: <https://www.ca.com/br/products/ca-application-performance-management.html>. Acesso em: setembro, 2018.

Erinle, B. (2017). *Performance Testing with JMeter 3*. Packt Publishing Ltd.

Erinle, B. (2014). *JMeter Cookbook*. Packt Publishing Ltd.

Grafana Labs (2018) *Grafana 5.1.0*. Disponível em: <https://grafana.com>. Acesso em: agosto, 2018.

ISO/IEC. (2011) *ISO/IEC 25010 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models*. International Organization for Standardization. Geneva.

ISO/IEC/IEEE. (2013) *ISO/IEC/IEEE 29119-1 - Software and systems engineering - Software testing - Part 1: Concepts and definitions*. International Organization for Standardization. Geneva.

ISO/IEC/IEEE. (2013) *ISO/IEC/IEEE 29119-2 - Software and systems engineering - Software testing - Part 2: Test process*. International Organization for Standardization. Geneva.

Matam, S. and Jain, J. (2017). *Pro Apache JMeter: Web Application Performance Testing*. Apress.

Molyneaux, I. (2014). *The Art of Application Performance Testing: From Strategy to Tools*. O'Reilly Media, Inc.

Autor

Thiago Silva de Souza é Doutor em Engenharia de Sistemas e Computação pela COPPE/UFRJ, Mestre em Informática pelo PPGI/UFRJ, Especialista em Gerência e Desenvolvimento de Sistemas Distribuídos pelo NCE/UFRJ e Bacharel em Sistemas de Informação pela Universidade do Grande Rio (UNIGRANRIO). Possui diversas certificações internacionais, especialmente nas áreas de Teste de Software e Métodos Ágeis. Atualmente é Analista de Sistemas no Serviço Federal de Processamento de Dados (SERPRO) e Professor Adjunto na UNIGRANRIO. Seus interesses de pesquisa incluem Engenharia de Software Experimental, Teste de Software, Tecnologia de Web Services e Internet das Coisas (IoT). Seu e-mail para contato é profthiagodesouza@gmail.com.