|  |  |
| --- | --- |
|  | Logo fatec - 2016 - Denilce |

Coordenadoria de Tecnologia da Informação

Tecnologia em Análise e Desenvolvimento de Sistemas

Utilização de Inteligência Artificial na automação de testes de aplicações Web

Fernanda Pereira dos Santos Sousa

Sorocaba

Dezembro – 2019

|  |  |
| --- | --- |
|  |  |
|  |  |

**Coordenadoria de Tecnologia da Informação**

Tecnologia em Análise e Desenvolvimento de Sistemas

Utilização de Inteligência Artificial na automação de testes de aplicações Web

Fernanda Pereira dos Santos Sousa

Profa. Dra. Maria das Graças Junqueira Machado Tomazela – Orientadora

Sorocaba

Dezembro – 2019

**Dedicatória**

A dedicatória é opcional. Aproveite esse espaço para dedicar seu trabalho a pessoas e instituições que fizeram parte da sua vida e que merecem reconhecimento na construção da sua vida acadêmica e profissional até hoje.

Exemplo de dedicatória

Dedico este trabalho aos meus pais que com muita perseverança e paciência sempre incentivaram meus estudos e proporcionaram a educação que hoje tenho.

**Agradecimento**

Os agradecimentos são opcionais.

È importante reconhecer nos agradecimentos

todas as pessoas e instituições que de alguma

forma auxiliaram na construção

do seu trabalho de graduação

Exemplo de agradecimento

Agradeço a todos que de uma forma ou de outra colaboraram para com este trabalho, em especial aos meus grandes amigos Luke Skywalker e seu pai Darth Vader que me ajudaram com a força nas horas difíceis.

Agradeço em especial aos professores do curso de Análise e Desenvolvimento de Sistemas da Fatec Sorocaba que com muita dedicação contribuíram para minha formação profissional. Em especial ao meu orientador, prof. Feitosa, pela paciência e contribuições fundamentais durante as sessões de orientação.

**Índice**

[Índice: faça um índice como neste exemplo 7](#_Toc19351054)

[Lista de Figuras 8](#_Toc19351055)

[Lista de Tabelas 8](#_Toc19351056)

[Resumo 1](#_Toc19351057)

[1. Objetivo (incluir na introdução) 2](#_Toc19351058)

[2. Introdução 3](#_Toc19351059)

[3. Teste de software 5](#_Toc19351060)

[3.1. Tipos de teste (incluir teste unitário) 5](#_Toc19351061)

[3.1.1 Teste Funcional 6](#_Toc19351062)

[3.1.2 Teste não funcional 6](#_Toc19351063)

[3.1.3 Teste caixa-branca 6](#_Toc19351064)

[3.1.4 Teste relacionado à mudança 6](#_Toc19351065)

[3.2. Níveis de teste 7](#_Toc19351066)

[3.2.1 Teste de Componentes 7](#_Toc19351067)

[3.2.2 Teste de Integração 8](#_Toc19351068)

[3.2.3 Teste de Sistema 9](#_Toc19351069)

[3.2. 4 Teste de Aceite 9](#_Toc19351070)

[4. Inteligência Artificial 11](#_Toc19351071)

[4.1. Conceitos de Inteligência Artificial 11](#_Toc19351072)

[4.2. Técnicas de IA 12](#_Toc19351073)

[4.2.1 Raciocínio baseado em casos 12](#_Toc19351074)

[4.2.2 Redes neurais 16](#_Toc19351075)

[4.2.3 Algoritmos genéticos 17](#_Toc19351076)

[4.2.4 Arvores de decisão (explicar dentro de regra de produção) 18](#_Toc19351077)

[4.2.5 Lógica Fuzzy 19](#_Toc19351078)

[4.3 Regras de produção 20](#_Toc19351079)

[6. Trabalhos relacionados 22](#_Toc19351080)

[7. Proposta de modelo de teste automatizado 23](#_Toc19351081)

[7.1. Definição do escopo do teste 23](#_Toc19351082)

[7.2. Apresentação de caso de teste 23](#_Toc19351083)

[7.3. Utilização de regras de produção 23](#_Toc19351084)

[7.4. Telas da Aplicação 33](#_Toc19351085)

[7.5. Analise da solução proposta 47](#_Toc19351086)

[8. Considerações finais 48](#_Toc19351087)

[Bibliografia 49](#_Toc19351088)

[3. Utilização de figuras 53](#_Toc19351089)

[4. Utilização de tabelas 55](#_Toc19351090)

[5. Utilização de Itens 56](#_Toc19351091)

[Bibliografia - Exemplo 58](#_Toc19351092)

[Glossário (OBRIGATÓRIO) 61](#_Toc19351093)

# Índice: faça um índice como neste exemplo

Resumo i

1. Objetivo.......,..... 1

2. Assunto: Visão Geral i

3. Desenvolvimento do Tema 1

3.1 Definição de Informação 1

3.1.1 Histórico e Desenvolvimento 1

3.2 Sistemas de Informação Gerenciais 1

3.3 A Gestão de Recursos Humanos 3

3.3.1 Introdução 3

3.3.2 Histórico da Administração de Recursos Humanos 6

3.3.2.1 A Administração Científica 6

3.3.2.2 A Escola das Relações Humanas 7

3.3.2.3 Relações Industriais 7

3.3.2.4 Administração de Recursos Humanos 8

3.3.3 Desenvolvimento 10

3.3.4 Situação Atual 12

3.3.5 Tendências 14

3.4 Organizações 17

3.4.1 Introdução: As organizações 17

3.4.2 Conceito de Organização 17

3.4.3 A complexidade das organizações 18

3.4.4 As diferentes eras da organização 19

3.4.5 Organizações como sistemas abertos 23

3.5 As influências da Tecnologia da Informação na Gestão de Recursos Humanos 24

3.5.1 Situação Atual 24

3.5.2 Tendências 25

3.5.2.1 Influência da Internet nas comunicações e formas de aprendizado 25

3.6 Estudo de casos 26

3.6.1 Estudo de caso I: A Área de Recursos Humanos da HP 26

3.6.1.5 Alinhamento RH / TI – o caso da HP 30

3.6.2 Estudo de caso II: Secretaria da Fazenda do Estado de São Paulo 31

3.6.2.6 Alinhamento RH / TI – o caso da Secretaria da Fazenda 36

4 Conclusões 38

5 Bibliografia 40

6. Glossário 41

# Lista de Figuras

Figura 1 – Visão geral da gestão de pessoas 10

Figura 2 – Enfoque sistêmico nas organizações 12

Figura 3 – Esquema da estratégia empresarial e gestão de pessoas 13

Figura 4 – Fronteiras virtuais nas organizações 24

Figura 5 – Gestão das tecnologias da informação e gestão de pessoas 25

# Lista de Tabelas

Tabela 1 – Principais mudanças na Área de Recursos Humanos (PricewaterhouseCoopers) 17

Tabela 2 – Principais mudanças na Área de Recursos Humanos (Chiavenato) 22

Tabela 3 – Resumo do Estudo de Caso 1 29

Tabela 4 – Ferramentas de T.I. e seu relacionamento com os processos da ARH (estudo de caso 1) 30

Tabela 5 – Ferramentas de T.I. e seu relacionamento com os processos da ARH (estudo de caso 2) 36

**Utilização de Inteligência Artificial na automação de testes de aplicações Web**

# Resumo

As metodologias ágeis de desenvolvimento de software vêm ganhando cada dia mais força no meio empresarial, já que fornecem valor ao cliente à cada nova entrega, porém isso exige um maior esforço de teste, pois para garantir integração entre os módulos e pleno funcionamento da aplicação, testes de regressão são necessários à cada entrega. Considerando que este é um tipo de teste repetitivo e cada vez mais longo, esse trabalho tem como objetivo propor uma metodologia que seja capaz de montar scripts automatizados de teste a partir de passos de Casos de Teste (CT) e use técnicas de inteligência artificial no reconhecimento de padrões e elaboração de facilitadores para a automação, com a pretensão apoiar os testadores na programação desses scripts e agilizar o processo de teste. Espera-se, por meio de um trabalho puramente teórico, a elaboração de uma metodologia capaz de juntar passos automatizados de teste e empregar algoritmos inteligentes que possam traçar passos implícitos no CT ao analisar passos já existentes em uma base de conhecimento. Ao término do projeto, objetiva-se a ampla utilização desse mecanismo em projetos de desenvolvimento de software que exijam testes longos e/ou repetitivos, potencializando assim a capacidade e efetividade da equipe de testes.

Palavras-chave: Testes automatizados, Inteligência artificial, Casos de teste

# 1. Introdução

O cenário empresarial e tecnológico atual mostra uma grande quantidade de aplicativos e aplicações web sendo requisitadas e liberadas para consumo diariamente, isso gera uma grande carga de trabalho, não só para desenvolvedores como para testadores, sendo os testes muitas vezes mais morosos que o próprio desenvolvimento.

Ao observar esse fato, pode-se analisar as metodologias usadas no desenvolvimento, por exemplo, utilizando-se a metodologia Cascata, a maior parte do teste será executada ao final de todo o período de desenvolvimento, gerando uma altíssima carga de testes. Já na metodologia Ágil o teste é feito em pequenos blocos, assim como o desenvolvimento, diminuindo sua carga e aumentando sua frequência, exigindo um maior número de testes de integração, repetindo várias e várias vezes o mesmo tipo de teste.

Com essa repetição, surgiram os mecanismos de automação de testes, que, ao utilizar frameworks próprios e linguagens de programação já usadas por desenvolvedores, são capazes de executar diversas vezes e autonomamente um mesmo caso de teste. Porém, essa automação deve ser inteiramente construída pelo testador para cada caso de teste criado, o que pode levar muito tempo e esforço.

Com esse pensamento pretende-se analisar a possibilidade do uso de uma nova abordagem, que poderá ser aplicada facilmente em projetos de desenvolvimento web.

A metodologia proposta é focada na possibilidade de criação, de uma maneira mais rápida, códigos de automação de testes unitários e de regressão a partir de casos de teste padronizados, atribuindo linhas de código já cadastradas a passos semelhantes de diversos casos de teste. Dessa forma o testador pode cadastrar cada tipo de ação apenas uma vez na base de conhecimento e reutilizá-la em todos os testes subsequentes.

Para o desenvolvimento de tal metodologia se torna necessário o uso de tecnologias de padronização de escrita de casos de teste, para deixar as informações mais completas e uniformes para análise, ferramentas de inteligência artificial para o reconhecimento de passos necessários para complementar a automação do teste além de uma base de conhecimento para se manter e consultar os passos já criados em cada CT.

É esperado o uso de regras de produção, *frameworks* e linguagens de programação voltadas ou adaptáveis para automação de testes.

Como maneira de ilustrar a metodologia proposta, este trabalho apresenta um programa fictício, HappyPath, visando facilitar o entendimento geral do funcionamento interno da metodologia.

Dito isto, a abordagem apresentada neste trabalho tem como objetivo propor uma metodologia de teste utilizando um mecanismo de Inteligência Artificial, visando a aumentar a acurácia do teste, e melhorar o desempenho dos testadores.

A importância do trabalho é dada pela a agilidade atribuída ao processo de teste, facilitando assim o trabalho dos testadores em projetos com grande carga de testes.

Visando atingir o objetivo declarado uma pesquisa teórica foi realizada, analisando monografias, dissertações e artigos da área estudada, dessa forma aumentando o conhecimento específico e verificando o estado da arte.

Este trabalho está organizado em capítulos, sendo o Capítulo 1 destinado a situar o leitor a condição atual do mercado, objetivos do trabalho, sua estrutura. O Capítulo 2 visa conceituar testes de software. O Capítulo 3 tem como objetivo explicar conceitos e definições relacionadas à Inteligência Artificial. O Capítulo 4 trás trabalhos relacionados mostrando o estado da arte do conteúdo em estudo. O Capítulo 5 é destinado a especificação do funcionamento da metodologia, trazendo o programa fictício HappyPath como forma de ilustrá-la. O Capítulo 6 é destinado a conclusões finais do trabalho.

# 2. Teste de software

O teste de software é o processo de avaliação de um software em que se procura falhas causadas por defeitos na aplicação ou pontos de melhoria, podendo assim indicá-los para que sejam consertados e inseridos na aplicação que será destinada ao usuário final, desta forma aumentando a confiabilidade do sistema.

É esperado que cada equipe de desenvolvimento de software tenha uma ou mais pessoas responsáveis pelos testes, sugere-se ser o próprio desenvolvedor a testar.

De forma que ele torne-se mais confiavel para o usuário final

Pesquisar as definições do presmamm e somerville, falar um pouquinho de cada e seguir um deles

## 2.1. Tipos de teste (incluir teste unitário)

Os testes de software podem ser classificados em diversos tipos conforme suas funções, como avaliar as funcionalidades do software, a usabilidade, a segurança, a estrutura do componente, etc.

Um tipo de teste é um grupo de atividades de teste destinado a testar características específicas de um sistema de software, ou parte de um sistema, com base em objetivos de teste específicos. Tais objetivos podem incluir:

• Avaliar as características de qualidade funcional, tais como integridade, correção e adequação.

• Avaliar as características de qualidade não funcionais, como confiabilidade, eficiência de desempenho, segurança, compatibilidade e usabilidade.

• Avaliar se a estrutura ou arquitetura do componente ou sistema está correta, completa e especificada

• Avaliar os efeitos das alterações, como a confirmação da correção dos defeitos (teste de confirmação) e procurar alterações não intencionais no comportamento como resultado de alterações no software ou no ambiente (teste de regressão).

### 2.1.1 Teste Funcional

O teste funcional é responsável por verificar características funcionais do sistema, ou seja, “o que” o sistema deve fazer segundo os requisitos funcionais do software. Essas informações podem estar documentadas nas especificações de requisitos, regras de negócio, casos de uso ou estórias de usuários (*user story*).

### 3.1.2 Teste não funcional

O teste não-funcional é responsável por avaliar características não funcionais do sistema, ou seja, o “quão bem” o sistema funciona, verificando questões como segurança, usabilidade, eficiência, portabilidade, compatibilidade e manutenibilidade.

### 2.1.3 Teste caixa-branca

O teste caixa-branca refere-se ao teste que se baseia na estrutura interna do software, seja ela o código, o fluxo de dados ou a arquitetura do sistema, ou seja, trata-se de um teste mais profundo, que vai além do que o usuário final pode usar.

### 2.1.4 Teste relacionado à mudança

Quando há mudanças na aplicação, seja correção de defeitos ou inserção de novos recursos, um teste deve ser feito para garantir que tudo está funcionando corretamente, para isso existem o teste de confirmação e o teste de regressão. O teste de confirmação é responsável por, depois da correção de um defeito, executar novamente os casos de teste que falharam ou novos casos se a correção do defeito trouxer uma nova função. Já o teste de regressão é responsável por analisar outras partes da aplicação que possam ter sido afetadas pela mudança ocorrida no código a procura de falhas colaterais. O conjunto de testes de regressão aumenta a cada nova alteração, ficando os testes cada vez mais demorados e repetitivos, o que torna essa categoria de teste propensa à automação.

## 2.2. Níveis de teste

Níveis de teste são testes de um mesmo tipo, que possuem os mesmos objetivos, portanto podem ser agrupados e gerenciados em conjunto.

### 2.2.1 Teste de Componentes

O teste de componente (também conhecido como teste de unidade ou módulo) se concentra em componentes que são testáveis separadamente. Os objetivos incluem:

•Reduzir o risco.

•Verificar os comportamentos funcional e não funcional do componente ao projetado e especificado.

•Construir a confiança na qualidade do componente

•Encontrar defeitos no componente

•Evitar que os defeitos espalhem para níveis mais altos de teste.

Em alguns casos, especialmente em modelos de desenvolvimento incremental e iterativo (p.e.,Ágil), em que as alterações de código estão em andamento, os testes de regressão de componentes automatizados desempenham um papel fundamental na criação daconfiança de que as alterações não impactaramos componentes existentes.

O teste de componente geralmente é realizadoisoladamente do resto do sistema, dependendo do modelo deciclo de vida de desenvolvimento de software e desistema, que pode exigir objetos simulados, virtualização de serviços, equipamentos, simuladorese controladores. O teste de componente pode cobrir funcionalidade (p.e.,correção de cálculos), características não funcionais (p.e.,busca de vazamentos de memória) e propriedades estruturais (p.e.,teste de decisão).

### 2.2.2 Teste de Integração

O teste de integração se concentra nas interações entre componentes ou sistemas. Os objetivos do teste de integração incluem:

•Reduzir risco.

•Verificar se os comportamentos funcionais e não funcionais das interfaces estão projetados e especificados.

•Construir confiançana qualidade das interfaces.

•Encontrar defeitos (que podem estar nas próprias interfaces ou nos componentes ou sistemas).

•Evitar que os defeitosespalhem para níveis mais altos de teste.

Assim como no teste de componentes, em alguns casos, o teste de regressão de integração automatizadagaranteque as alterações não interromperam as interfaces, os componentes ou sistemas existentes.

Existem dois níveis diferentes de teste de integração descritos neste syllabus, que podem ser realizados em objetos de teste de tamanho variável, como segue:

•O teste de integração de componentes foca nas interações e interfaces entre componentes integrados. O teste de integração de componentes é executado após o teste do componente e geralmente é automatizado. No desenvolvimento iterativo e incremental, os testes de integração de componentes geralmente fazem parte do processo de integração contínua.

•O teste de integraçãodo sistema concentra-se nas interações e interfaces entre sistemas, pacotes e microserviços. O teste de integração do sistema também pode abranger interações e interfaces fornecidas por entidadesexternas (p.e., serviços da Web). Nesse caso, a entidadeemdesenvolvimento não controla as interfaces externas, o que pode criar vários desafios para testes (p.e., garantir que os defeitos de bloqueio de teste no código da organização externa sejam resolvidos, organizar ambientes de teste etc.). O teste de integração do sistema pode ser feito após o teste do sistema ou em paralelo com as atividades de teste do sistema em andamento (tanto no desenvolvimento sequencial quanto no desenvolvimento iterativo e incremental).

### 2.2.3 Teste de Sistema

O teste de sistema se concentra no comportamento e nas capacidades de todo um sistema ou produto, geralmente considerando as execuções das tarefas de ponta a ponta do sistema e os comportamentos não funcionais exibidos ao executar tais tarefas. Os objetivos do teste do sistema incluem:

•Reduzir o risco.

•Verificar se os comportamentos funcionais e não funcionais do sistema são como projetados e especificados.

•Validar se o sistema está completo e funcionará como esperado.

•Criar confiança na qualidade do sistema como um todo.

•Encontrar defeitos.

•Evitar que os defeitos espalhem para níveis mais altos de teste ou produção.

Para determinados sistemas, a verificação da qualidade dos dados pode ser um objetivo. Assim como no teste de componente e no teste de integração, e em alguns casos os testes automatizados de regressão do sistema fornecem a confiabilidade de que as alterações não quebraram os recursos existentes ou a execução dos recursos de ponta a ponta. O teste do sistema geralmente produz informações que são usadas pelas partes interessadas para tomar decisões de liberação. O teste do sistema também pode satisfazer requisitos oupadrões legais e regulatórios.

O ambiente de teste deve preferencialmente corresponder ao seu destino, ou ao ambiente de produção.

### 2.2. 4 Teste de Aceite

O teste de aceite, como o teste do sistema, geralmente se concentra no comportamento e na capacidade de todo um sistema ou produto. Os objetivos do teste de aceite incluem:•Estabelecer confiança na qualidade do sistema como um todo.

•Validar que o sistema está completo e funcionará como esperado.

•Verificar se os comportamentos funcionais e não funcionais do sistema são os especificados.

O teste de aceite pode produzir informações para avaliar a situação do sistema para implantação e uso pelo cliente (usuário final). Os defeitos podem ser encontrados durante o teste de aceite, mas encontrar defeitos muitas vezes não é um objetivo, e encontrar um número significativo de defeitos durante o teste de aceite pode, em alguns casos, ser considerado um grande risco de projeto. O teste de aceite também pode satisfazer requisitos ou padrões legais ou regulatórios.

Formas comuns de testes de aceite incluem o seguinte:

•Teste de aceite do usuário

•Teste de aceite operacional

•Teste de aceite contratual e regulatório

•Teste Alfa e Beta.

# 3. Inteligência Artificial

A inteligência artificial pode ser definida, segundo Castrounis (2019), como um conjunto de conceitos e ferramentas que representam o verdadeiro significado de inovação e com um imenso potencial disruptivo e transformativo.[1]

Pode ainda ser definida, segundo Luger (2009), como o ramo da ciência da computação que diz respeito à automação do comportamento inteligente.

AI is a set of concepts and tools that represents the true meaning of innovation, and has huge disruptive and transformative potential.

Citação com autor incluído no texto: Castrounis (2019)

Citação com autor não incluído no texto: (CASTROUNIS, 2019)

Artificial intelligence (AI) may be defined as the branch of computer science that is concerned with the automation of intelligent behavior

Artificial intelligence is still a young discipline, and its structure, concerns, and methods are less clearly

defined than those of a more mature science such as physics.

Artificial intelligence has always been more concerned with expanding the capabilities of computer science than with defining its limits.

Citação com autor incluído no texto: Luger (2009)

Citação com autor não incluído no texto: (LUGER, 2009)

## 3.1. Conceitos de Inteligência Artificial

Abandoning the philosophical question of what it means for an artificial entity to think or have intelligence, Alan Turing [1950] developed an empirical test of artificial intelligence, which is more appropriate to the computer scientist endeavoring to implement artificial intelligence on a computer. The Turing test is an operational test; that is, it provides a concrete way to determine whether the entity is intelligent. The test involves a human interrogator who is in one room, another human being in a second room, and an artificial entity in a third room. The interrogator is allowed to communicate with both the other human and the artificial entity only with a textual device such as a terminal. The interrogator is asked to distinguish the other human from the artificial entity based on answers to questions posed by the interrogator. If the interrogator cannot do this, the Turing test is passed and we say that the artificial entity is intelligent.

Citação com autor incluído no texto: Neapolitan e Jiang (2013)

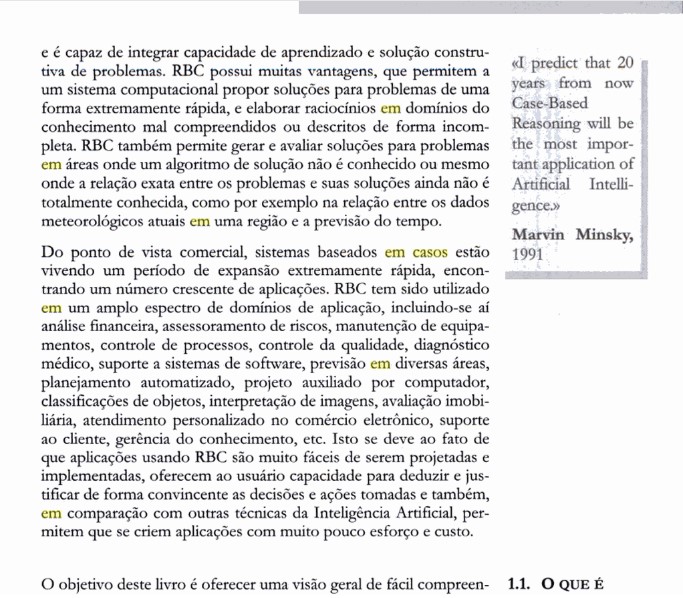
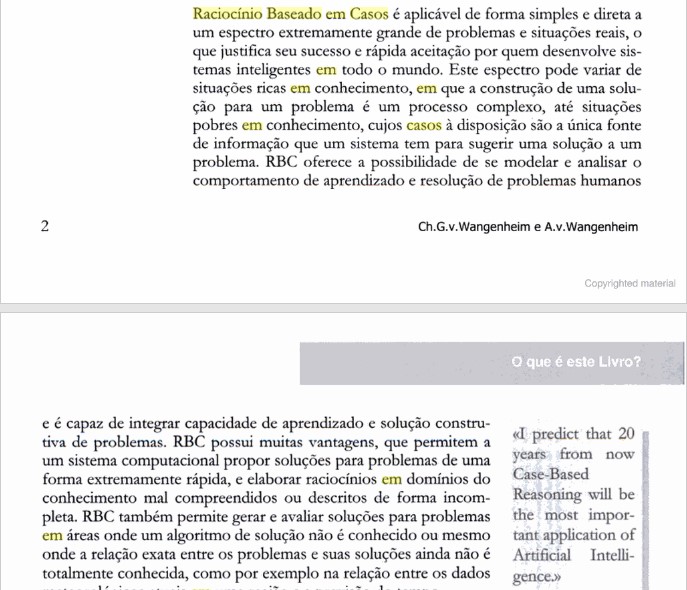
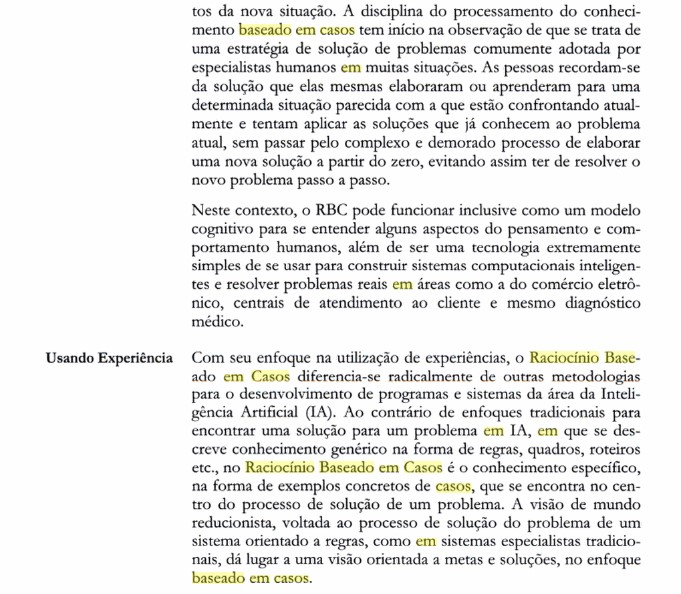
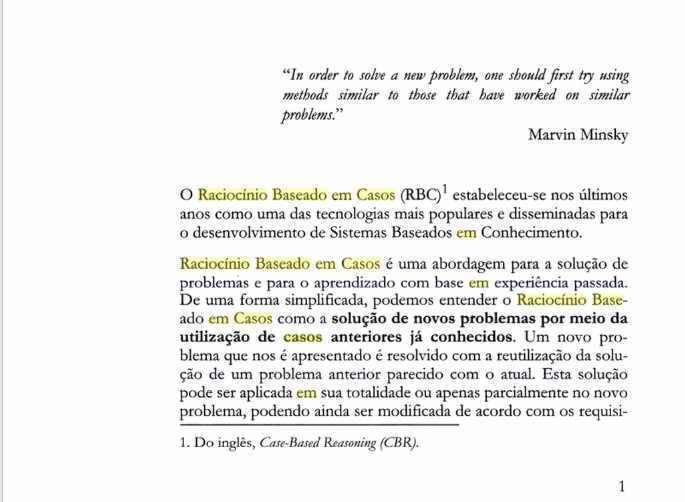
Citação com autor não incluído no texto: (NEAPOLITAN; JIANG, 2013)

## 3.2. Técnicas de IA

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus luctus sapien.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus luctus sapien ac libero.

### 3.2.1 Raciocínio baseado em casos

Imagens do texto na pasta ver depois

Citação com autor incluído no texto: von Wangenheim e von Wangenheim (2003)

Citação com autor não incluído no texto: (VON WANGENHEIM; VON WANGENHEIM, 2003)

### 3.2.2 Redes neurais

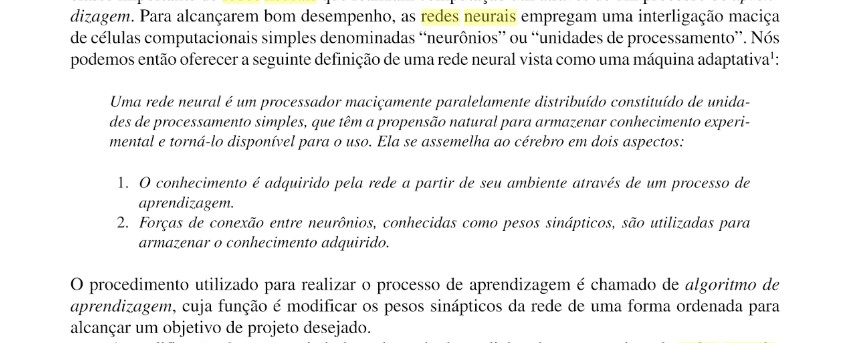
A redes neurais são compostas por “neurônios” ou “unidades de processamento”, são células computacionais simples, que são interligadas. Elas se assemelham ao cérebro em dois aspectos (HAYKIN, 2001):

1 – O conhecimento é adquirido pela rede a partir do seu ambiente por meio de um processo de aprendizagem.

2 – Forças de conexão entre neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido.” (HAYKIN, 2001)

Sendo que esse processo de aprendizagem é conhecido como algoritmo de aprendizagem, que modifica os pesos sinápticos de uma maneira ordenada.

Imagens do texto na pasta ver depois



Citação com autor incluído no texto: Haykin (2001)

Citação com autor não incluído no texto: (HAYKIN, 2001)

### 3.2.3 Algoritmos genéticos

Algoritmos genéticos é uma técnica da inteligência artificial criada ao observar a teoria da evolução proposta por Darwin, em um conjunto de potenciais soluções para um problema, as melhores seriam selecionadas até que a melhor seja escolhida como solução.

Pode-se observar a origem dos estudos de redes neurais ao observar sua comparação com o evolucionismo, assim como apresentado no texto *Introducción a los Algoritmos Genéticos y la Programación Genética* (Introdução aos Algoritmos Genéticos e a Programação Genética) (GESTAL et al., 2010)

“Durante milhões de anos as diferentes espécies se adaptam para poder sobreviver em um meio em constante mudança. Da mesma maneira se poderia ter uma população de potenciais soluções a um problema, das quais se iriam selecionando as melhores até que se adaptassem perfeitamente ao meio, e neste caso o problema resolver (Michalewicz & Fogel, 2000) (Bäck, 1996) (Whitley, 1994).”(Tradução nossa).

Si algo funciona bien, ¿por qué no imitarlo?. La respuesta a esta pregunta

nos lleva directamente a los orígenes de la computación evolutiva. Durante millones de años las diferentes especies se han adaptado para poder sobrevivir

en un medio cambiante. De la misma manera se podría tener una población

de potenciales soluciones a un problema, de las que se irían seleccionando las mejores hasta que se adaptasen perfectamente al medio, en este caso el problema a resolver (Michalewicz & Fogel, 2000) (Bäck, 1996) (Whitley, 1994). Em términos muy generales se podría definir la computación evolutiva como una

familia de modelos computacionales inspirados en la evolución.

Más formalmente, el término de computación evolutiva se refiere al estudio de los fundamentos y aplicaciones de ciertas técnicas heurísticas basadas en los principios de la evolución natural (Tomassini, 1995). Estas técnicas

heurísticas podrían clasificarse en 3 grandes categorías o grupos, dando lugar a la ecuación evolutiva recogida en la Figura 1.1.

Citação com autor incluído no texto: Gestal et al. (2010)

Citação com autor não incluído no texto: (GESTAL et al., 2010)

### ~~3.2.4 Arvores de decisão (explicar dentro de regra de produção)~~

Segundo Monard e Baranaukas (2003) a arvore de decisão é uma estrutura de dados definida recursivamente, sendo um nó folha correspondente a uma classe ou um nó de decisão

Algoritmos que induzem ´arvores de decis˜ao pertencem `a fam´ılia de algoritmos Top Down Induction of Decision Trees — TDIDT. Uma ´arvore de decis˜ao (ou AD) ´e uma estrutura de dados

definida recursivamente como:

• um n´o folha que corresponde a uma classe ou

• um n´o de decis˜ao que cont´em um teste sobre algum atributo. Para cada resultado do

teste existe uma aresta para uma sub´arvore. Cada sub´arvore tem a mesma estrutura que

a ´arvore.

E f´acil perceber que a ´arvore pode ser representada como um conjunto de regras. Cada ´

regra tem seu in´ıcio na ra´ız da ´arvore e caminha at´e uma de suas folhas.

Citação com autor incluído no texto: Monard e Baranaukas (2003)

Citação com autor não incluído no texto: (MONARD; BARANAUKAS, 2003)

### 3.2.4 Lógica Fuzzy

Gomide e Gudwin (1994) definiram a logica fuzzy como algo diferente dos sistemas lógicos tradicionais sendo um caso limite do raciocínio aproximado o raciocínio predominante

A lógica fuzzy é a lógica baseada na teoria dos conjuntos fuzzy. Ela difere dos sistemas lógicos tradicionais em suas características e seus detalhes. Nesta lógica, o raciocínio exato corresponde a um caso limite do raciocínio aproximado, sendo interpretado como um processo de composição de relações nebulosas. Na lógica fuzzy, o valor verdade de uma proposição pode ser um subconjunto fuzzy de qualquer conjunto parcialmente ordenado, ao contrário dos sistemas lógicos binários, onde o valor verdade só pode assumir dois valores: verdadeiro (1) ou falso (0). Nos sistemas lógicos multi-valores, o valor verdade de uma proposição pode ser ou um elemento de um conjunto finito, num intervalo, ou uma álgebra booleana. Na lógica nebulosa, os valores verdade são expressos linguisticamente, (e.g. : verdade, muito verdade, não verdade, falso, muito falso, ...), onde cada termo linguístico é interpretado como um subconjunto fuzzy do intervalo unitário.

Citação com autor incluído no texto: Gomide e Gudwin (1994)

Citação com autor não incluído no texto: (GOMIDE; GUDWIN, 1994)

## 3.3 Regras de produção

A representação do conhecimento por regras de produção é baseada nas propostas concebidas pelo matemático Emil Post (1943) que via nos sistemas de produção um modelo computacional geral de solução de problemas. Na década de oitenta passou-se a utilizar esta técnica como suporte para o modelo mental. O termo "sistema de produção" é atualmente usado para descrever os sistemas que taiêm em comum o fato de serem constituídos de um conjunto de regras para descrever condições e ações. As regras são armazenadas como uma coleção de declarações SE-ENTÃO. SE < premissas > ENTÃO A parte SE da regra é chamada de corpo, parte antecedente ou lado esquerdo e deve ser avaliada em relação à base de conhecimentos como um todo. Quando existe o ajuste buscado pelo mecanismo de avaliação a ação correspondente especificada no lado direito, ou parte conseqüente, é executada. As condições na parte antecedente da regra devem ser satisfeitas para que a ação, na parte conseqüente, seja considerada. Se qualquer premissa falhar o lado direito também falha. A representação do conhecimento por regras de produção é a forma mais utilizada em sistemas especialistas. A justificativa é a naturalidade que representa para o homem pois o par "condição-ação", para raciocinar e decidir, também é usado pela mente humana Estima-se que cerca de oitenta por cento dos sistemas existentes utilizam esta forma de representação do conhecimento. Como ilustração ao uso de regras de produção são mostradas a seguir algumas regras que poderiam fazer parte, por exemplo, de um sistema especialista para orientação de candidatos ao curso de mestrado.

Regra: SE um candidato se inscrever para o curso de mestrado e o candidato preencher todos os requisitos exigidos e existirem vagas disponíveis ENTÃO o candidato passa a ser aluno do curso de mestrado Regra: SE o aluno de mestrado for aprovado em todas as disciplinas do currículo ENTÃO ele deve escrever uma dissertação Regra: SE a dissertação está concluída será submetida a uma banca examinadora e a banca examinadora aprovar o trabalho ENTÃO o aluno receberá o grau de mestre Além da naturalidade para a interpretação humana a utilização de regras de produção apresenta outros aspectos positivos, como a modularidade e a uniformidade. As regras podem ser manipuladas como peças independentes e novas regras podem ser incluídas a qualquer tempo o que é uma característica importante pois o conhecimento de qualquer sistema especialista tende a aumentar com o passar do tempo. A uniformidade fica caracterizada no padrão único utilizado para todas as regras do sistema. Esta uniformidade pode gerar alguns inconvenientes para o usuário mas também traz muitos benefícios como facilidade dê manutenção e de uso do sistema por pessoas não familiarizadas com o mesmo.

**Citação com autor incluído no texto:**Heinzle (1995)

**Citação com autor não incluído no texto:**(HEINZLE, 1995)

# 4. Trabalhos relacionados

Citar Sistema especialista com regras de produção e Inteligência Artificial na Verificação e Teste de Software para Desenvolvimento Ágil

# 5. Proposta de modelo de teste automatizado

Uma aplicação será desenvolvida para a criação de Casos de Teste padronizados. Cada Caso de Teste deve possuir um “Ponto de partida” – delimitando a tela em que o Caso de Teste deve iniciar – e um ou mais passos – funções necessárias. O “Ponto de partida” deve ser selecionado em um *dropbox* e os passos serão adicionados uma a um ao selecionar uma opção em cada *dropbox* de “Condição inicial”, “Ação” e “Resultado esperado / Conclusão”. Ao selecionar a função desejada um número de *textboxes* igual ao de parâmetros necessários serão exibidos e não devem ser deixados em branco.

Em seu primeiro uso o testador deverá alimentar a aplicação com as funções desejadas incluindo seu Nome, Número de parâmetros e Código da função (escrito em Java/Selenium).

## 5.1. Definição do escopo do teste

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus luctus sapien.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus luctus sapien ac libero.

## 5.2. Apresentação de caso de teste

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus luctus sapien.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus luctus sapien ac libero.

## 5.3. Utilização de regras de produção

A metodologia Regras de Produção será utilizada na criação do teste automatizado (Java/Selenium) para traçar um caminho partindo da Página Inicial até o Ponto de Partida definido em um Caso de Teste, para isso serão verificados todos os Casos de Teste observando suas Condições Iniciais e Resultados Esperados. Considerando um cenário em que haja Casos de Teste em todas as páginas da aplicação, uma espécie de mapeamento será produzido partindo do Ponto de Partida desejado e “fazendo o caminho inverso” até chegar na página inicial.

Para exemplificar o funcionamento é possível considerar os seguintes Casos de Teste registrados na aplicação:

|  |  |  |
| --- | --- | --- |
| Caso de Teste: Abas Principais | | |
| Condição Inicial | **Ação** | **Resultado Esperado** |
| Página [Home] visível | Clicar em aba [Administração] | Página [Administração] visível |
| Página [Administração] visível | Clicar em aba [Coordenação] | Página [Coordenação] visível |
| Página [Coordenação] visível | Clicar em aba [Usuário] | Página [Usuário] visível |
| ... | ... | ... |

Fonte (autoria própria)

|  |  |  |
| --- | --- | --- |
| Caso de Teste: Opções da Página Administração | | |
| Condição Inicial | **Ação** | **Resultado Esperado** |
| Página [Administração] visível | Clicar em [Requisição de acesso] | Página [Requisição de acesso] visível |
| Página [Requisição de acesso] visível | Clicar em aba [Administração] | Página [Administração] visível |
| Página [Administração] visível | Clicar em [Usuários] | Página [Usuários] visível |
| ... | ... | ... |

|  |  |  |
| --- | --- | --- |
| Caso de Teste: Criação de Requisição de Acesso | | |
| Condição Inicial | **Ação** | **Resultado Esperado** |
| Página [Requisição de acesso] visível | Clicar em [Nova requisição de acesso] | Página [Criar requisição de acesso] visível |
| Página [Criar requisição de acesso] visível | Preencha o campo [País] com [Brasil] | Campo [País] preenchido com [Brasil] |
| ... | ... | ... |

A aplicação usará as Regras de Produção da seguinte forma: Cada passo dos Casos de Testes será considerado como uma regra (Condição Inicial = SE, Ação = ENTÃO, Resultado Esperado = Verdade), ao iniciar no Ponto de Partida a aplicação considera-o como verdade e busca qual Condição Inicial levou até àquele resultado.

Tendo em vista que o Caso de Teste escolhido para conversão em código automatizado tem como Ponto de Partida a página “Requisição de acesso” (Caso de Teste Criação de Requisição de Acesso), então “Página [Requisição de acesso] visível” é considerada verdade e procura-se a Condição Inicial que leva a esse resultado, encontrando assim a Condição Inicial “Página [Administração] visível” (Caso de Teste Opções da Página Administração). O código relacionado a esse passo é reservado, a Condição Inicial é comparada com a página inicial buscada e não sendo igual o processo continua. Se inicia a busca pela Condição Inicial que tem como resultado esperado “Página [Administração] visível” encontrando assim a Condição Inicial “Página [Home] visível” (Caso de Teste: Abas Principais), o código relacionado a esse passo é reservado acima do anterior e ao comparar a página encontrada com a página inicial buscada se tem um resultado positivo e o processo é encerrado.

Dessa forma dois trechos de código foram incluídos automaticamente antes do começo do Caso de Teste escolhido, assim o Caso de Teste convertido terá a seguinte estrutura:

|  |  |  |
| --- | --- | --- |
| Condição Inicial | Ação | Resultado Esperado |
| Página [Home] visível | Clicar em aba [Administração] | Página [Administração] visível |
| Página [Administração] visível | Clicar em [Requisição de acesso] | Página [Requisição de acesso] visível |
| Página [Requisição de acesso] visível | Clicar em [Nova requisição de acesso] | Página [Criar requisição de acesso] visível |
| Página [Criar requisição de acesso] visível | Preencha o campo [País] com [Brasil] | Campo [País] preenchido com [Brasil] |
| ... | ... | ... |

Num sistema baseado em regras de produção o conhecimento a ser processado é representado através do uso de regras com uma arquitetura previamente definida. Estas regras utilizam um par condição-ação onde as condições são premissas e a ação é a conclusão. No processo de inferência o sistema busca uma primeira regra arbitrariamente, ou em alguns casos aquela predefinida como regra inicial, e tenta atender as premissas da regra. As premissas descritas na regra são apresentadas ao usuário em forma de questionamentos. As respostas fornecidas pelo usuário são então armazenadas na lista de verdades fazendo com que estas informações fiquem disponíveis ao longo do processo de raciocínio e possam ser utilizadas para a validação de outras regras. Se as respostas fornecidas pelo usuário atenderem as premissas da regra e a regra contiver na sua parte conclusiva uma solução para o problema o processo de inferência estará concluído com sucesso.

Se, por outro lado, a regra não permitiu alcançar uma sohição para o problema,

o sistema seguirá avaliando outras regras, sempre acumulando o conhecimento

adquirido ao longo do processo na sua lista de verdades. O processo continua até que

seja alcançada uma regra que leve à solução do problema, ou quando não for mais

possível continuá-lo.

Um exemplo de um pequeno conjunto de regras para diagnóstico de problemas

em veículos é mostrado abaixo.

Regra 1

SE tem combustível no tanque

E tem combustível no carburador

ENTÃO o motor recebe combustível

Regra 2

SE o motor recebe combustível

E o motor vira

ENTÃO o problema é nas velas

Regra 3

SE o motor não vira

E as lâmpadas não ascendem

ENTÃO o problema é na bateria ou nos cabos

Regra 4 SE o motor não vira

E as lâmpadas ascendem

ENTÃO o problema é o motor de partida

Observa-se no exemplo apresentado que existe um encadeamento lógico entre as regras. Esta rede de encadeamentos é chamada de árvore de busca. O raciocínio com regras de produção envolve, portanto, a aplicação de um algoritmo para fazer a busca dos possíveis caminhos da árvore. Este algoritmo, por sua vez, deve oferecer recursos para que o usuário possa optar por estratégias diferenciadas de raciocínio ou encadeamento.

3.2 - RACIOCÍNIO E ENCADEAMENTO

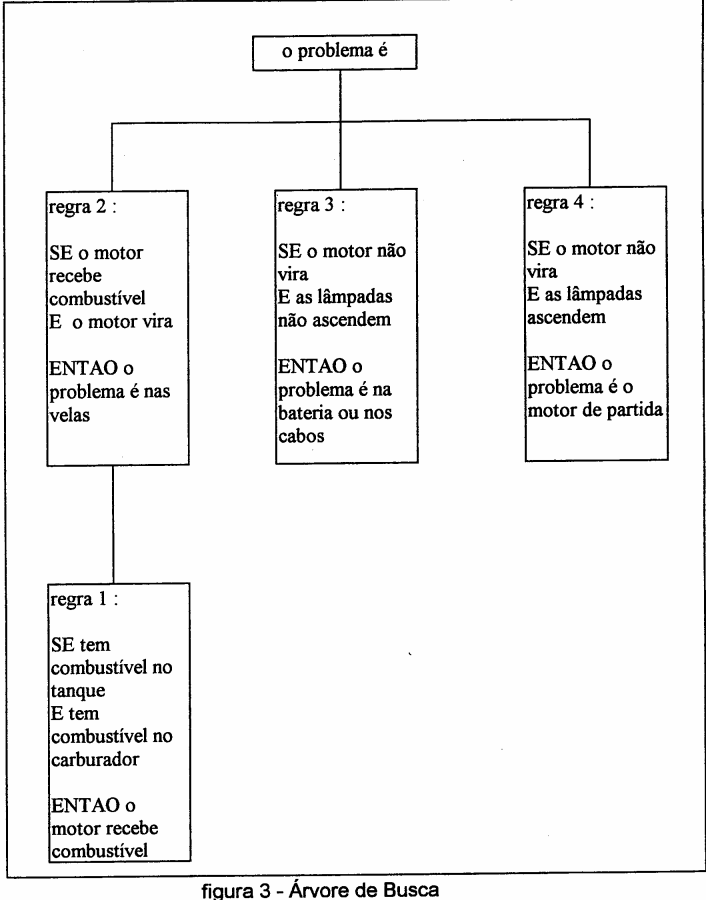
3.2.1 - ENCADEAMENTO PROGRESSIVO

O método de encadeamento progressivo, também chamado de encadeamentopara-frente ou “forward”, começa com a seleção de uma regra qualquer da base de conhecimentos. Simultaneamente é criada a lista de verdades onde serão armazenadas todas as informações tidas como verdadeiras. No início do processo de inferência alguns questionamentos ao usuário serão obrigatórios já que a lista de verdades está ainda vazia. As respostas do usuário são então colocadas na lista de verdades. Após a utilização de uma regra, as conclusões desta regra também devem ser adicionadas à lista de verdades. Na seqüência, o sistema procura por uma regra que tenha entre as suas premissas a conclusão da regra avaliada. Este encadeamento que vai da conclusão de uma regra para as premissas de outra regra é chamado de encadeamento progressivo. Utilizando-se o exemplo do diagnóstico de problemas em veículos apresentado anteriormente, o sistema avaliaria inicialmente a regra 1. As premissas da regra 1 são "a existência de combustível no tanque" e "a existência de combustível no carburador". Como neste estágio a lista de verdades ainda se encontra vazia e não existe outrã regra na base de conhecimentos que permita verificar a veracidade destas premissas o sistema questionará diretamente o usuário a respeito delas. As respostas do usuário às premissas da regra 1 seriam então colocadas na lista de verdades. Caso estas respostas confirmem as premissas, também a conclusão da regra 1, ou seja "o motor recebe combustível", seria introduzida na lista de verdades e o sistema buscaria uma outra regra que tivesse entre as suas premissas a conclusão da regra 1. A regra 2 seria selecionada e o sistema tentaria provar as suas premissas. A primeira delas, "o motor recebe combustível” seria atendida imediatamente pois esta informação já está na lista de verdades. A segunda premissa, "o motor vira", não pode ser comprovada pois não se tem nenhuma informação acerca desta premissa na lista de verdades. A questão deve ser então apresentada ao usuário. Se a resposta atender a premissa, a busca terminaria com sucesso com o sistema concluindo que o problema está nas velas.. Se, por outro lado, a resposta fosse contrária, o sistema passaria a verificar a condição "as lâmpadas não ascendem" que é a segunda premissa da regra 3. Novamente não é possível buscar a informação na lista de verdades nem através de outra regra, devendo ser apresentada a questão ao usuário. Se a resposta atender a condição, o processo de inferência estaria encerrado com o sistema concluindo que o problema é nas velas. Se a resposta não atender a premissa, a regra 4 passaria a ser avaliada.

A regra 4 pode ser automaticamente avaliada pelo sistema, pois as informações relacionadas a todas as suas premissas já estariam na lista de verdades. O processo teria continuidade até que alguma conclusão fosse alcançada, ou não existissem mais regras a serem avaliadas. Alguns aspectos computacionais, que foram ignorados no exemplo, teriam que ser observados em caso de implementação. Um sistema real deve considerar a existência de caminhos alternativos ao longo do processamento, inclusive com um mecanismo que possibilite retomo ou retrocesso, chamado de "backtracking", quando necessário. Neste caso as regras 2 e 3 também podem levar a uma solução; o sistema deve, por isto, oferecer ao usuário a alternativa de buscar outras soluções além daquela alcançada e deve também sustentar as conclusões através de recursos de justificação. 3.2.2 - ENCADEAMENTO REGRESSIVO O encadeamento regressivo, também chamado de “backward” ou para-trás, difere do encadeamento progressivo pelo fato de que inicia numa conclusão e passa então a usar as regras para provar esta conclusão. Para provar que a conclusão é verdadeira, o sistema verifica se as premissas da regra são verdadeiras. Para provar a veracidade de uma premissa, o método procura por outra regra que tenha esta premissa como sua conclusão. Quando a regra é encontrada, procura-se prová-la demonstrando a veracidade de cada uma de suas premissas. Na seqüência , novamente o sistema procura por uma regra cuja conclusão atenda a premissa da regra em avaliação. O processo termina quando não mais existirem regras a serem provadas. Também é criada, neste método, a lista de verdades que conterá todas as informações tidas como verdadeiras. Esta lista, além das respostas fornecidas diretamente pelo usuário, engloba também as conclusões do próprio sistema.

No caso do exemplo, o sistema teria que, inicialmente, òptar entre as regras 2, 3 ou 4, já que todas elas levam a uma conclusão ou solução do problema. Se o sistema resolver o conflito, por exemplo, em favor daquela de menor número, a regra 2 passaria a ser avaliada. Neste caso, o sistema assume que o problema é nas velas e tentará provar as premissas da regra 2, que são "o motor recebe combustível” e "o motor vira". O problema é então decomposto em dois subproblemas, cada um deles relativo a uma das premissas. Como a primeira premissa desta regra é a conclusão da regra 1, o sistema passa a considerar esta regra. Neste momento existem três condições a serem confirmadas: "tem combustível no tanque", "tem combustível no carburador" e "o motor vira". Nenhuma das três condições está contida como parte conclusiva de outra regra, fazendo com que o sistema questione diretamente o usuário a respeito delas. As respostas são então colocadas na lista de verdades. Se todas as respostas confirmarem as premissas, o processamento termina com sucesso e o sistema responderia que o problema está nas velas. Se, no entanto, as respostas não atenderem as premissas, as regras 1 e 2 seriam desprezadas e o sistema passaria a considerar a regra 3, assumindo que o problema é na bateria ou nos cabos, e tentará provar as premissas correspondentes. O processo continua até que se consiga chegar a uma conclusão ou não restem mais regras para serem avaliadas. A figura 3 mostra a árvore de busca para o encadeamento regressivo. O encadeamento regressivo mostra-se mais apropriado nas situações em que se tem uma conclusão preliminar que precisa ser confirmada. A seqüência de avaliação das regras neste caso parecerá mais lógica ao usuário pois os questionamentos apresentados estarão concentrados no objetivo que se quer provar.

As regras que foram utilizadas como exemplo na demonstração dos mecanismos de raciocínio progressivo e regressivo têm um encadeamento bastante simples e direto. As conexões existentes entre as regras são sempre únicas, fazendo com que as opções de pesquisa na árvore sejam reduzidas, facilmente visualizadas e não exigindo mecanismos sofisticados para determinar a seqüência de avaliação das regras. Entretanto, numa situação real, a árvore de inferência cresce em complexidade, especialmente no que se refere às conexões entre regras, exigindo do algoritmo de busca, além da especificação da direção, a determinação da ordem em que estas conexões, ou caminhos, serão examinados.



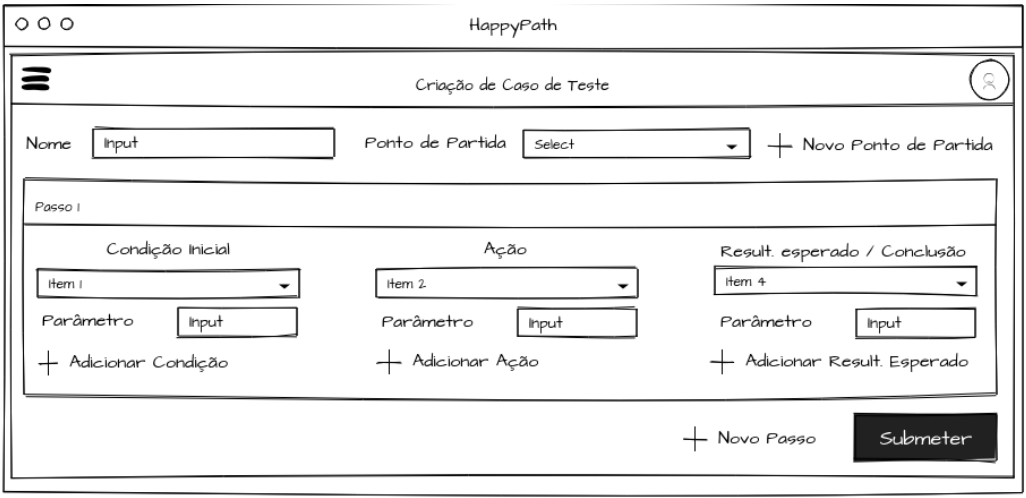
**Citação com autor incluído no texto:**Heinzle (1995)

**Citação com autor não incluído no texto:**(HEINZLE, 1995)

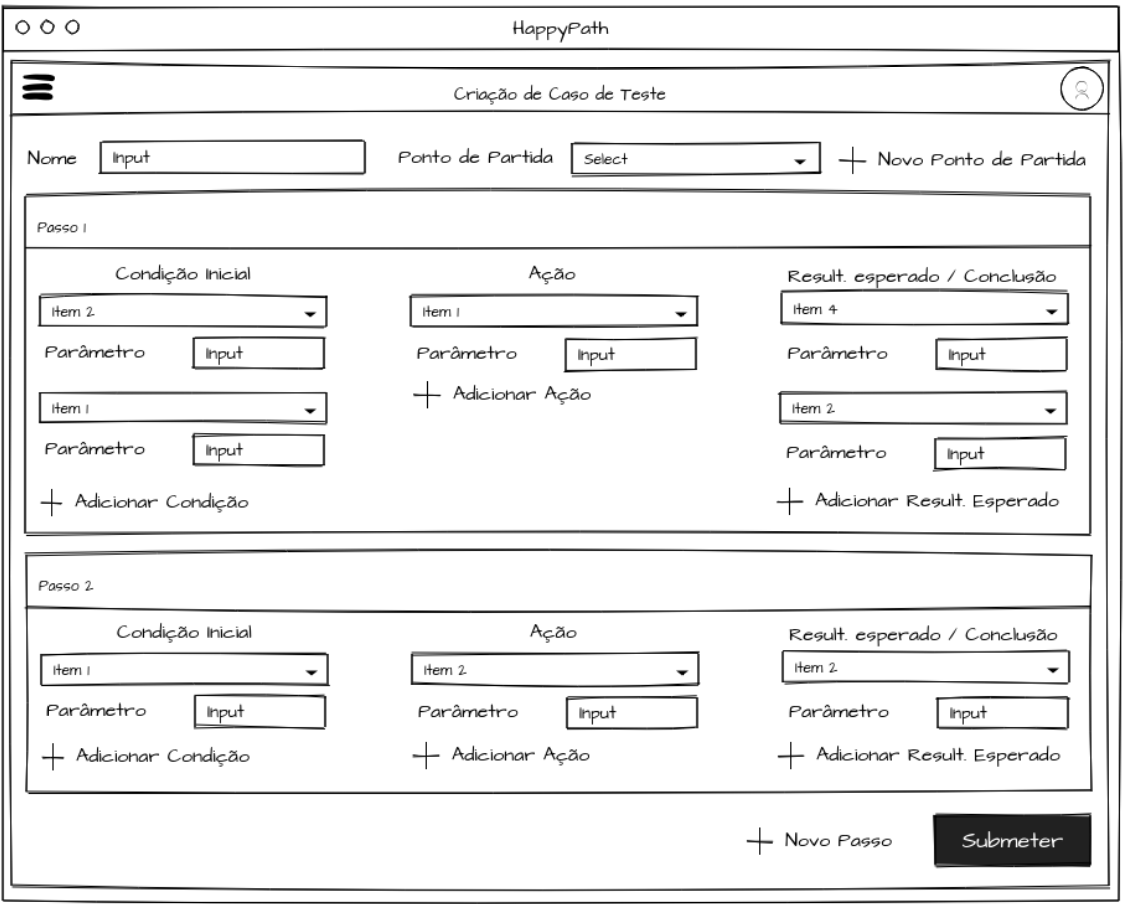
## 5.4. Telas da Aplicação

Como forma de exemplificar a metodologia, o protótipo da aplicação HappyPath foi desenvolvido, segue as telas desenvolvidas:

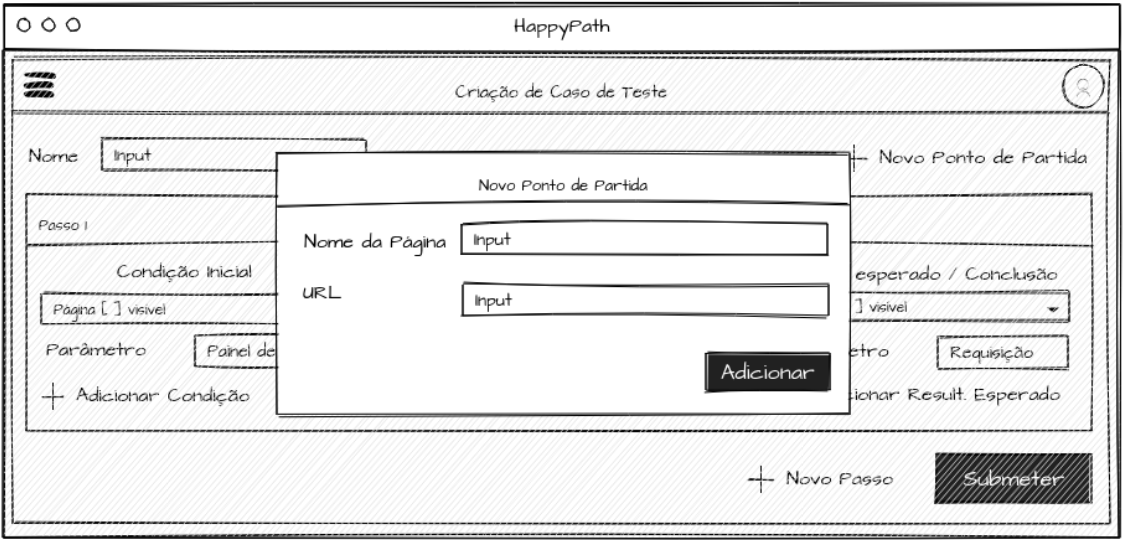
A tela de Criação de Casos de Testes apresenta um campo de texto para a inclusão do Nome do Caso de Teste, um campo de seleção para escolher o Ponto de Partida e por padrão o primeiro passo do CT, contendo os campos de seleção da Condição Inicial, Ação e Resultado Esperado/Conclusão, cada um exibindo um número de caixas de texto para a inclusão de Parâmetros igual à registrada na opção escolhida.

  
legenda

Será possível incluir mais Condições Iniciais, Ações e Resultados Esperados / Conclusões ao clicar nos sinais de adição abaixo de cada campo e adicionar novos passos ao clicar no sinal de adição ao lado do botão Submeter.

  
legenda

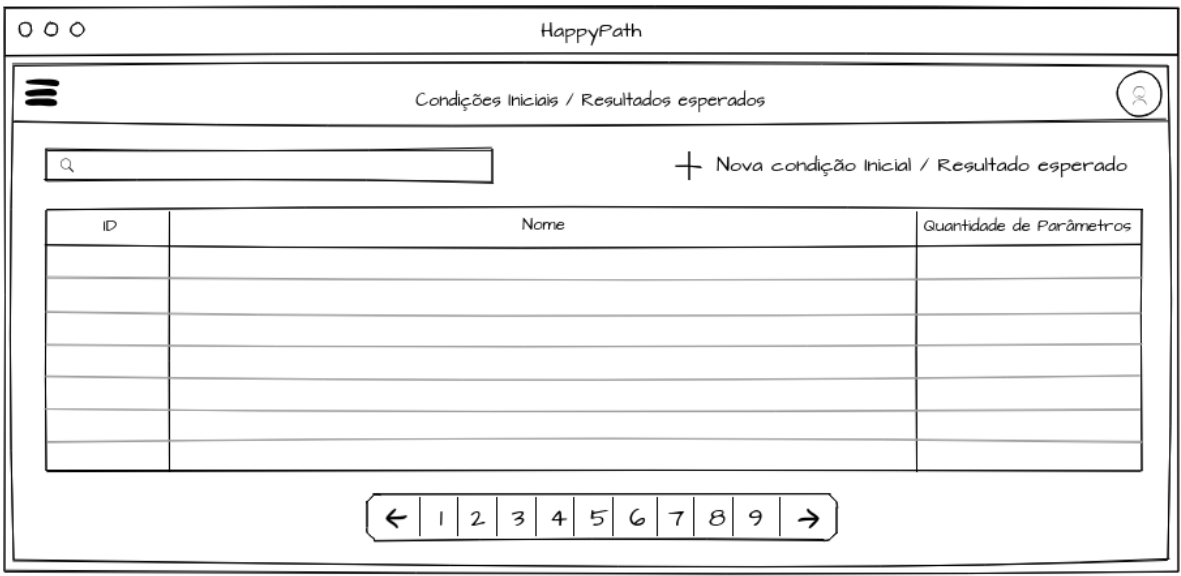
Caso o Ponto de Partida desejado não esteja cadastrado será possível criá-lo através do sinal de adição ao lado do campo de seleção.

  
legenda

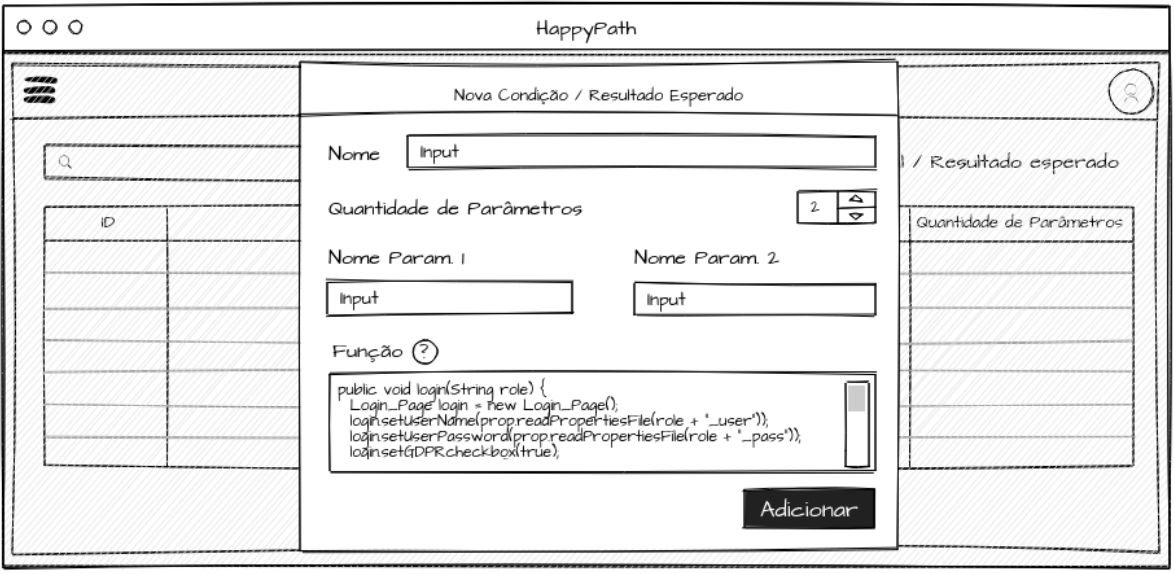
As opções de Condições Iniciais, Ações e Resultados Esperados/Conclusões devem ser criadas em telas diferentes.

As Condições Iniciais e Resultados Esperados/Conclusões serão adicionadas na mesma tela (Condições Iniciais / Resultados Esperados) e armazenadas na mesma tabela do Banco de Dados por se tratar de situações semelhantes. Nesta tela serão exibidos todos os itens criados identificados por ID, Nome e Quantidade de Parâmetros, será possível pesquisar por esses identificadores e navegar entre as páginas.

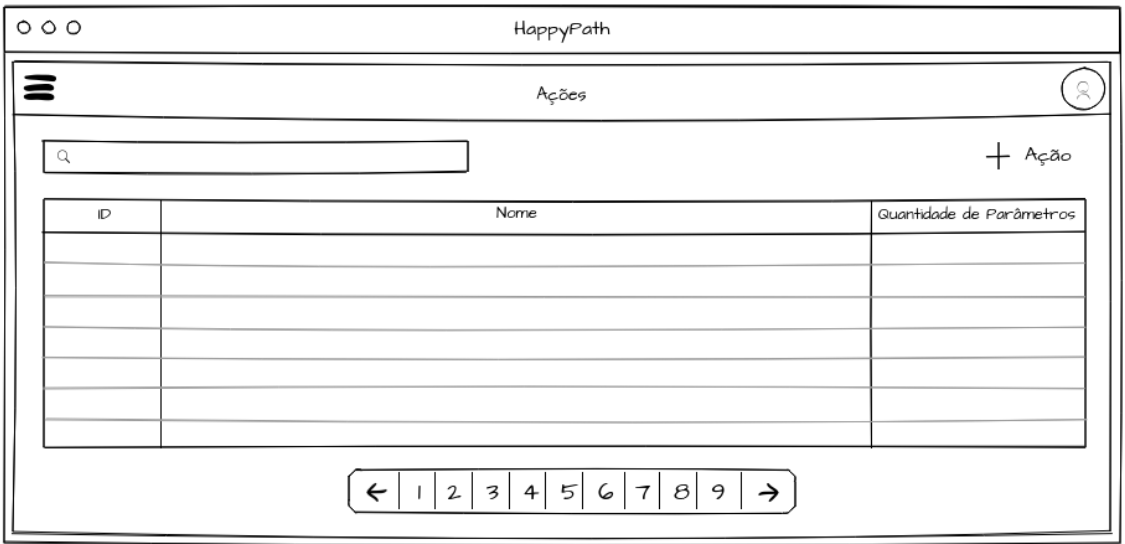
Será possível adicionar um novo item clicando no sinal de adição Nova condição inicial / resultado esperado.

  
legenda

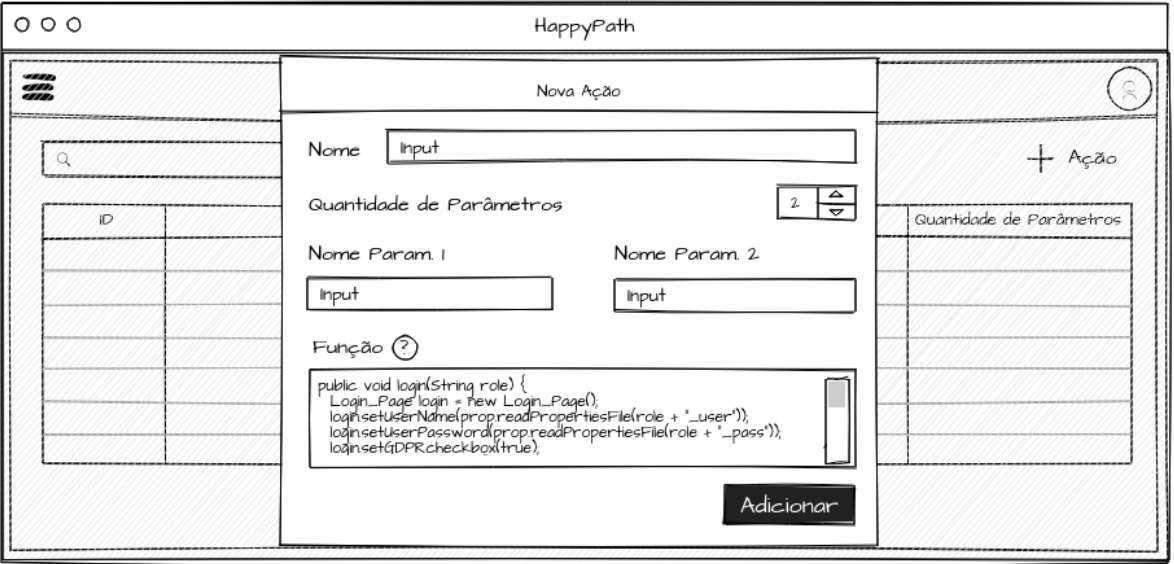
Na tela Nova Condição / Resultado Esperado será apresentado um campo de texto para a inclusão do nome da condição inicial ou resultado esperado / conclusão, um campo para seleção da quantidade de parâmetros que devem ser passados, uma quantidade igual ao número inserido de parâmetros de caixas de texto para o nome dos parâmetros e uma área de texto para a inclusão do código relacionado a essa regra, escrito em Java/Selenium.

  
legenda

Da mesma forma funcionará com as Ações, uma tela conterá todas as ações cadastradas identificadas por ID, Nome e Quantidade de Parâmetros, nessa tela será possível pesquisá-las por esses identificadores ou navegar entre as páginas. Para adicionar uma nova Ação deve-se clicar sobre o sinal de adição no topo da página.

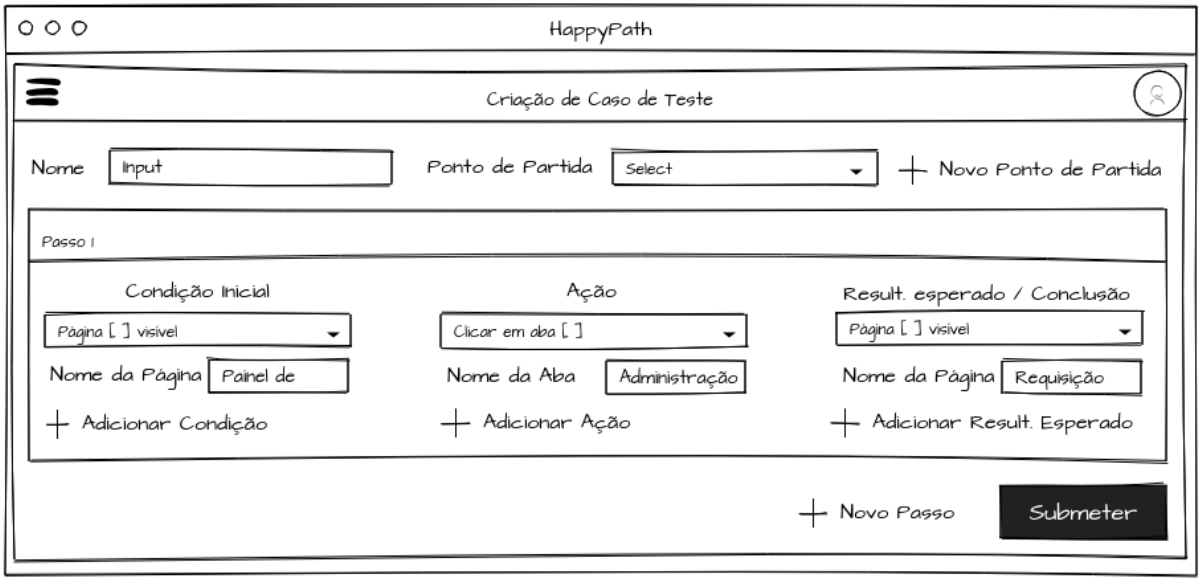
  
legenda

A tela Nova Ação possuirá um campo de texto para a inclusão do nome da ação, um campo para seleção da quantidade de parâmetros que devem ser passados , uma quantidade igual ao número inserido de parâmetros de caixas de texto para o nome dos parâmetros e uma área de texto para a inclusão do código relacionado a essa regra, escrito em Java/Selenium.

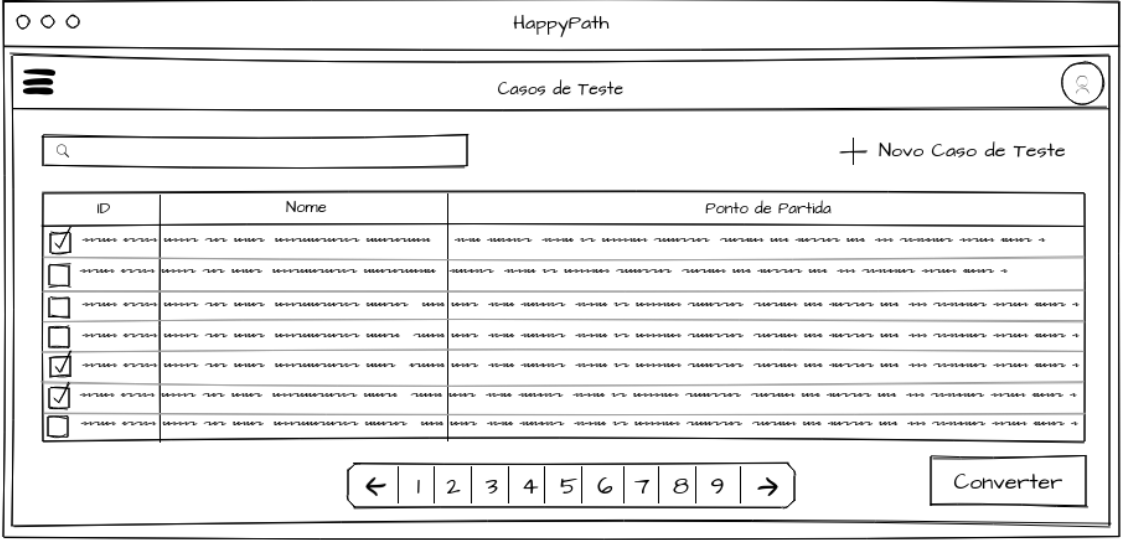
  
legenda

A criação das condições iniciais, ações e resultados esperados devem seguir um padrão para se manterem genéricas, e assim poderem ser reutilizadas.

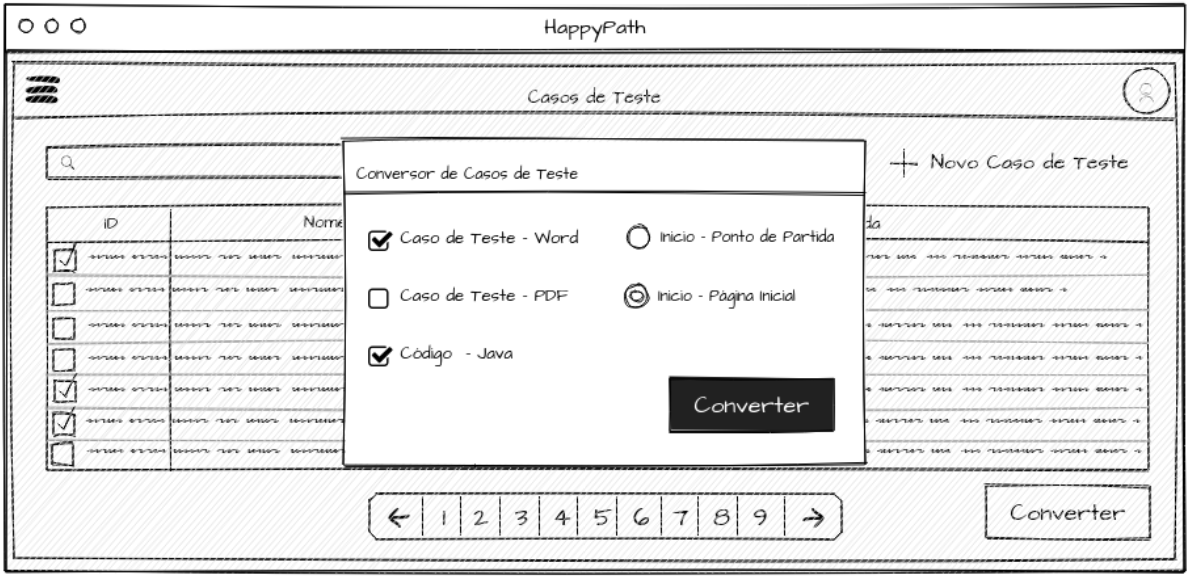
Os nomes devem ser intuitivos para que o usuário entenda a função executada pelo código, como no exemplo abaixo, Figura X, em que o nome da condição inicial é “Página [ ] visível” e os colchetes (“[ ]”) indicam onde o parâmetro passado será incluído. Nesse caso, a condição inicial será “Página [Painel de Controle] visível”, caso haja mais de um parâmetro a ser passado, a ordem considerada será a presente no nome da condição / resultado esperado / ação, por exemplo “Preencher campo [ ] com [ ]”, os parâmetros Nome e Marina seriam passados, e respeitando a ordem de inserção a regra seria entendida como “Preencher campo [Nome] com [Marina].

  
legenda

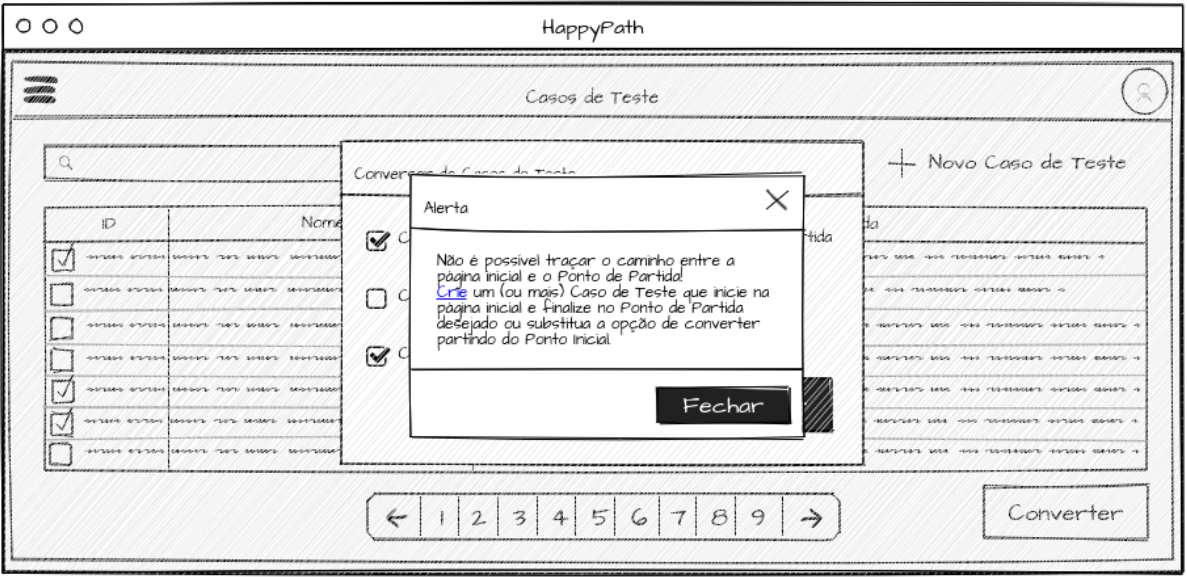
Na tela de Casos de Teste, conforme Figura X, são exibidos em uma grid todos os Casos de Teste já criados, identificados pelo ID, Nome e Ponto de Partida. Há um botão para a criação de novos Casos de teste e um para a conversão dos Casos de Teste selecionados.

  
legenda

Ao selecionar os Casos de Teste desejados e clicar em Converter um modal, será apresentado para a seleção do tipo de conversão desejada, em forma de Caso de Teste em Word ou PDF ou em forma de Código em Java/Selenium, também é dada a opção do início do Caso de Teste no ponto de partida indicado ou na página inicial da aplicação. A Figura X ilustra esse cenário.

  
legenda

Caso a opção escolhida for iniciar a partir da página inicial, ao clicar em converter a aplicação verificará, utilizando a Inteligência Artificial (Regras de Produção), se existem Casos de Teste que levam da página inicial até o Ponto de Partida declarado no CT, caso não haja, uma mensagem de erro será apresentada indicando que não é possível traçar um caminho da página inicial até o Ponto de Partida e que, nesse caso, será necessário ou criar um ou mais Casos de Teste que tenham como Ponto de Partida a página inicial e Levem até o Ponto de Partida do CT (funcionamento explicado no capítulo 7.3), ou que selecionar a opção de conversão partindo do Ponto de Partida do CT.

  
legenda

## 5.5. Analise da solução proposta

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus luctus sapien.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus luctus sapien ac libero.

# 6. Considerações finais

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus luctus sapien.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus luctus sapien ac libero.

# Bibliografia

MORE: Mecanismo online para referências, versão 2.0. Florianópolis: UFSC Rexlab, 2013. Disponível em: ‹ http://www.more.ufsc.br/ › . Acesso em: 16 mar. 2019.

CASTROUNIS, Alex. **AI for People and Business.**Sebastopol: O’reilly Media, Inc., 2019. Disponível em: <https://learning.oreilly.com/library/view/ai-for-people/9781492036562/copyright-page01.html>. Acesso em: 16 mar. 2019.

LUGER, George F. **Artificial intelligence : structures and strategies for complex problem solving.**6. ed. Boston: Pearson Education, Inc, 2009. Disponível em: <http://dspace.fue.edu.eg/xmlui/bitstream/handle/123456789/3379/2387.pdf?sequence=1>. Acesso em: 16 mar. 2019.

NEAPOLITAN, Richard E.; JIANG, Xia. **CONTEMPORARY ARTIFICIAL INTELLIGENCE.**Boca Raton: Crc Press - Taylor & Francis Group, 2013. Disponível em: <https://learning.oreilly.com/library/view/contemporary-artificial-intelligence/9781466573192/OEBPS/9781466559400\_cvi.htm>. Acesso em: 16 mar. 2019.

VON WANGENHEIM, Christiane Gresse; VON WANGENHEIM, Aldo. **Raciocínio baseado em casos.**Barueri: Manole Ltda., 2003.

HAYKIN, Simon. **Redes Neurais: Princípios e Prática.**2. ed. São Paulo: Artmed Editora S.a., 2001.

GESTAL, Marcos et al. **Introducción a los Algoritmos Genéticos y la Programación Genética.**A Coruña: Universidade da Coruña, Servizo de Publicacións, 2010. Disponível em: <http://www.galeon.com/dantethedestroyer/algoritmos.pdf>. Acesso em: 16 mar. 2019.

MONARD, Maria Carolina; BARANAUKAS, José Augusto. “Indução de regras e árvores de decisão”. **Sistemas Inteligentes Para Engenharia.** Rezende: So Editora Manole Ltda, 2003. Disponível em: <http://dcm.ffclrp.usp.br/~augusto/publications/2003-sistemas-inteligentes-cap5.pdf>. Acesso em: 16 mar. 2019.

GOMIDE, Fernando Antonio Campos; GUDWIN, Ricardo Ribeiro. **MODELAGEM, CONTROLE, SISTEMAS E LÓGICA FUZZY.**1994. Universidade Estadual de Campinas (UNICAMP). Disponível em: <https://pdfs.semanticscholar.org/c4ea/4484bdd2449053d7b7384b05c0e2def86449.pdf>. Acesso em: 16 mar. 2019.

HEINZLE, Roberto. **PROTÓTIPO DE UMA FERRAMENTA PARA CRIAÇÃO DE SISTEMAS ESPECIALISTAS BASEADOS EM REGRAS DE PRODUÇÃO.**1995. 161 f. Dissertação (Mestrado) - Curso de Engenharia, Departamento de Engenharia de Produção e Sistemas, Universidade Federal de Santa Catarina, Florianópolis, 1995. Cap. 2. Disponível em: <https://repositorio.ufsc.br/bitstream/handle/123456789/157957

**2.1. Subtítulo: use letra Arial, tamanho 12, negrito**

Utilize subtítulos para organizar o trabalho. Sempre coloque uma linha em branco entre o sub-título e o texto. Numere os subtítulos de acordo com o capítulo atual, se necessário pode-se utilizar vários níveis, veja o exemplo abaixo

**2.1.1. Níveis de subtítulos**

Mantenha o padrão de cabeçalho e texto para todos os subtítulos, observe que a numeração identifica que o assunto do subtítulo está um nível abaixo do subtítulo anterior.

**2.1.2. Referências Bibliográficas**

Use referências bibliográficas quando for utilizar textos, observações, ideias de outros autores, coloque o nome do autor e o ano da publicação entre parênteses, veja os parágrafos abaixo.

Segundo Flávio de Toledo (Toledo, 1993), são vários os conceitos caracterizam a estrutura e o comportamento das organizações no período que vai da Antiguidade às vésperas da Revolução Industrial.

Os vários conceitos que caracterizam a estrutura e o comportamento das organizações no período que vai da Antiquidade às vésperas da Revolução Industrial foram descritos por Toledo em seu estudo clássico (Toledo, 1993).

“*Vários estudos caracterizam a estrutura e o comportamento das organizações no período que vai da Antiquidade às vésperas da Revolução Industrial*” (Toledo, 2003a; Jardim, 2005).

**2.1.3. Apud**

A Palavra Apud é utilizada em referências bibliográficas para fazer uma citação indireta, ou seja, citar um trecho que não foi lido diretamente na obra original, mas citado por outro autor. É utilizada no texto com o significado de “citado por” indicando ao leitor que a citação é feita conforme o que foi lido e referenciado por um outro autor que teve acesso à obra original.

É aconselhável não fazer muitas citações usando a palavra apud, devendo-se utiliza-la apenas nos casos de obras originais de difícil acesso por exemplo: publicações antigas, obras raras ou textos em línguas dificilmente acessíveis.

Na lista de referência deve-se colocar somente a obra consultada

Veja os exemplos abaixo:

Para Apple (1994 apud Silva, 2002) quer reconheçamos ou não, o currículo e as questões educacionais mais genéricas sempre estiveram atrelados à história dos conflitos de classe, raça, sexo e religião.

...quer reconheçamos ou não, o currículo e as questões educacionais mais genéricas sempre estiveram atrelados à história dos conflitos de classe, raça, sexo e religião, tanto nos Estados Unidos quanto em outros países. (Apple, 1994 apud Silva, 2002)

Porter (apud Carvalho e Souza,1999) considera que “a vantagem competitiva surge fundamentalmente do valor que uma empresa consegue criar para seus compradores e que ultrapassa o custo de fabricação pelas empresas”.

Segundo Weller (1999 apud Collins, 2003) “para superar este problema serão necessárias adotar ações corretivas e preventivas”.

# 3. Utilização de figuras

Para utilizar figuras, deve-se identificar a figura e referenciar a sua origem caso esteja sendo retirada de uma outra obra. Se a figura foi elaborada pelo autor, deverá receber uma nomenclatura, mas não será necessário referenciar. Use as figuras sempre centralizadas, sem texto em volta, use letra times new roman tamanho 10 – itálico, sem negrito na referência das figuras. Veja os exemplos abaixo:

*Figura 1 – Visão geral da gestão de pessoas.*



*Fonte: adaptado de CHIAVENATO, 2002*

*Figura 2 – Visão de um notebook tradicional*

j0345667

Fonte: elaborado pelo autor

# 4. Utilização de tabelas

A diferença entre tabelas e figuras é que as tabelas possuem dados, que podem ser textos ou números que serão comparados. Use a letra e o tamanho que for mais conveniente, tente manter um padrão no texto, porém se não for possível use outros formatos para as tabelas. Veja o exemplo abaixo

*Tabela 1 – Principais mudanças na Área de Recursos Humanos.*

|  |  |
| --- | --- |
| Antes | Agora |
| Tinha plenos poderes para selecionar, efetivar e demitir; | Tem profissionais de formação diversa no comando da equipe |
| Punia disciplinarmente por meio de cartas de advertências e até suspensões | Funciona como consultor interno, dando orientação aos chefes na hora da contratação, demissão e resolução de conflitos com os funcionários; |
| Assumia toda a responsabilidade pela concessão e revisão de empréstimos, benefícios e aumentos salariais; | Investe na terceirização de funções como seleção, treinamento e pesquisa salarial; |
| Direcionava e executava os programas de treinamento, inclusive os operacionais; | Tem desempenho voltado para os negócios, alinhando suas políticas à estratégia da empresa; |
| Participava de reuniões estratégicas apenas como ouvinte e não dava palpite nas decisões | Ganhou credibilidade com a presidência e é chamado a participar das decisões |

*Fonte: adaptado de**PricewaterhouseCoopers, 2000*

# 5. Utilização de Itens

Se necessário utilize marcadores para os itens que serão citados no texto, é importante usar sempre os mesmos marcadores para todo o texto, procure usar marcadores discretos. Veja o exemplo abaixo.

Segundo Chiavenato (Chiavenato,2002), as grandes organizações, também chamadas de organizações complexas, possuem certas características como:

* + - Complexidade. As organizações são distintas dos grupos e sociedades em termos de complexidade estrutural. A complexidade estrutural refere-se à diferenciação horizontal e vertical.
    - Anonimato. A ênfase é colocada sobre as tarefas ou operações e não sobre as pessoas. O importante é que a operação seja executada, não importa por quem.
    - Rotinas padronizadas. Para operar procedimentos e canais de comunicação. Apesar da atmosfera impessoalizada, as organizações apresentam a tendência a desenvolver grupos informais face a face dentro delas.
    - Estruturas personalizadas e não oficiais. Constituem a organização informal que funciona em paralelo com a estrutura formal.
    - Tendência à especialização e à proliferação de funções, que tende a separar as linhas de autoridade formal daquelas de competência profissional ou técnica.
    - Tamanho. O porte é um elemento final e intrínseco às grandes organizações, pois decorre do número de participantes e de órgãos que formam sua estrutura organizacional.

**6. Dicas Gerais**

Releia o texto depois de pronto para verificar a coerência do texto, peça para outra pessoa também ler. Evite usar figuras coloridas ou texto com cores, normalmente temos problemas com a impressão de textos coloridos.

Peça para uma pessoa com experiência para procurar erros de gramática e redação.

# Bibliografia - Exemplo

Como apresentar a bibliografia: exemplos

**IMPORTANTE**: UTILIZAR O PADRÃO MORE (Mecanismo Online para Referências) da UFSC – [www.more.ufsc.br](http://www.more.ufsc.br)

**Inclua o MORE em suas referências**

MORE: Mecanismo online para referências, versão 2.0. Florianópolis: UFSC Rexlab, 2013. Disponível em: ‹ http://www.more.ufsc.br/ › . Acesso em: XX XXX XXXX.

**de AQUINO, Cleber Pinheiro** .- *Administração de Recursos Humanos . São Paulo : Atlas, 1992.*

**BERNARDO, André. A História do Gerenciamento de Projetos.** Responsabilidade do autor do vídeo. YouTube, 2013. Duração: 5min52seg. Disponível em:<<https://www.youtube.com/watch?v=le0GTYjlvl4>>. Acesso em: abril de 2017.

**CASTRO, Alfredo Pires de.; dos REIS, Almiro (neto) ; et alli** - *Manual de Gestão de Pessoas e Equipes. São Paulo : Editora Gente, 2003.*

**CHIAVENATO, Idalberto** *- Recursos Humanos Edição Compacta . São Paulo : Atlas, 2002. 7ª edição.*

**CRUZ, Tadeu** *- Sistemas de Informação: Tecnologia da Informação e a empresa do século XXI. Ed. Atlas, 1998.*

**DRUCKER, Peter F***. - Fator Humano e Desempenho. Ed. Livraria Pioneira, 1997. 3ª Edição.*

**FISCHER, André Luiz; NOGUEIRA, Arnaldo José França Mazzei** *– As Pessoas na Organização. São Paulo : Editora Gente, 2002, 2ª edição.*

**GATES, Bill** *. - A* estradadofuturo*. São Paulo : Companhia das Letras, 1995*

**GUFFEY, Mary E.** - *APA style electronic formats*, originalmente publicado em Business Communication Quarterly, Mar., pp. 59-76, [<http://www.westwords.com/GUFFEY/apa.html>](http://www.westwords.com/GUFFEY/apa.html) Acesso em: abril de 2017

**KEEN, P. G. W***. – Guia Gerencial para a Tecnologia da Informação. Ed. Campus 1996. 2ª Edição.*

**OLIVEIRA, Jayr Figueiredo de** – Sistemas de Informação: Um Enfoque Gerencial Inserido no Contexto Empresarial e Tecnológico. Ed. Érica, 2000.

**SANTOS, Fernando César Almada***. - Estratégia de Recursos Humanos: Dimensões Competitivas. São Paulo: Atlas, 1999a.*

**SANTOS, Fernando César Almada***. - Estratégia de Recursos Humanos: Dimensões Corporativaqs. São Paulo: Atlas, 1999b.*

**TACHIZAWA, Takeshy; et alli***. - Gestão com pessoas: uma abordagem aplicada às estratégias de negócios. Ed. FGV, 2001.*

**TOLEDO, Flávio de.** *- O que são recursos humanos. São Paulo : Ed. Brasiliense, 1993a*

**TOLEDO, Flávio de.** *- O que são recursos humanos II. São Paulo : Ed. Brasiliense, 1993b*

**TORREÃO, Paula. História do Gerenciamento de Projetos**, 2007. Disponível em:< <https://pontogp.wordpress.com/2007/04/23/historia-do-gerenciamento-de-projetos/>>. Acesso em: abril de 2017.

**VASCONCELLOS, Eduardo ; MARCOVITCH, Jacques**. *Gerenciamento da Tecnologia: um instrumento para a competitividade empresarial. Ed. Edgard Blücher Ltda, 1997.*

**WALTON, Richard E***. – Tecnologia de Informação: O uso de TI pelas empresas que obtêm vantagem competitiva. São Paulo: Atlas, 1998*

# Glossário (OBRIGATÓRIO)

Faça um glossário de termos técnicos e siglas utilizados no texto.

**Exemplo:**

**Caso de teste** – Documento de engenharia de software que contém condições para a realização de teste de software (*Test Case*).

**Framework** – Abstração que captura funcionalidades em comum entre vários projetos, facilitando seu uso de maneira genérica.

**Inteligência Artificial** – Dispositivos que possuem capacidades humanas, como perceber situações raciocinar e tomar decisões (IA, *AI*, *Artificial Intelligence*).

**Scripts automatizados de teste** – Codificação usada para testar um software autonomamente.

**Modelo Cascata** – Modelo de desenvolvimento de software em que cada etapa do desenvolvimento é fechada e sequencial, fluindo de um estágio para outro como em uma cascata, sem voltar para o estágio anterior (*Waterfall Model*).

**Modelo Ágil** – Modelo de desenvolvimento de software em que os indivíduos e as iterações possuem prioridade aos processos e ferramentas e a resposta a mudanças é dinâmica e rápida (*Agile Model*).

**Workflow** – Software que procura gerenciar e descrever o fluxo de dados entre as tarefas e processos da organização.