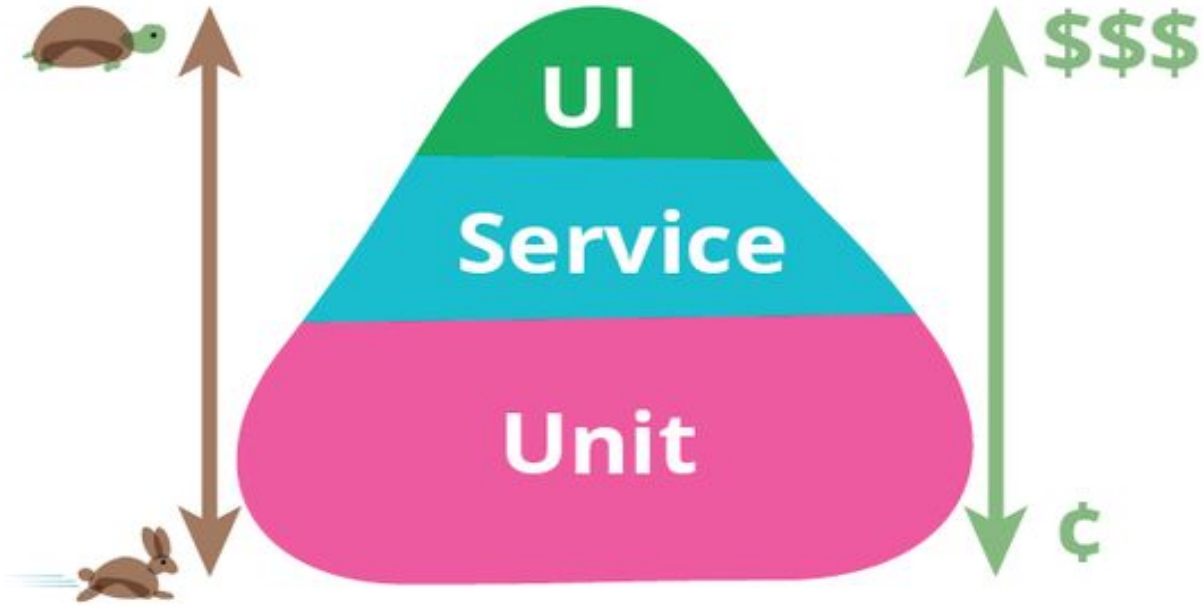
Faculdade IMPACTA

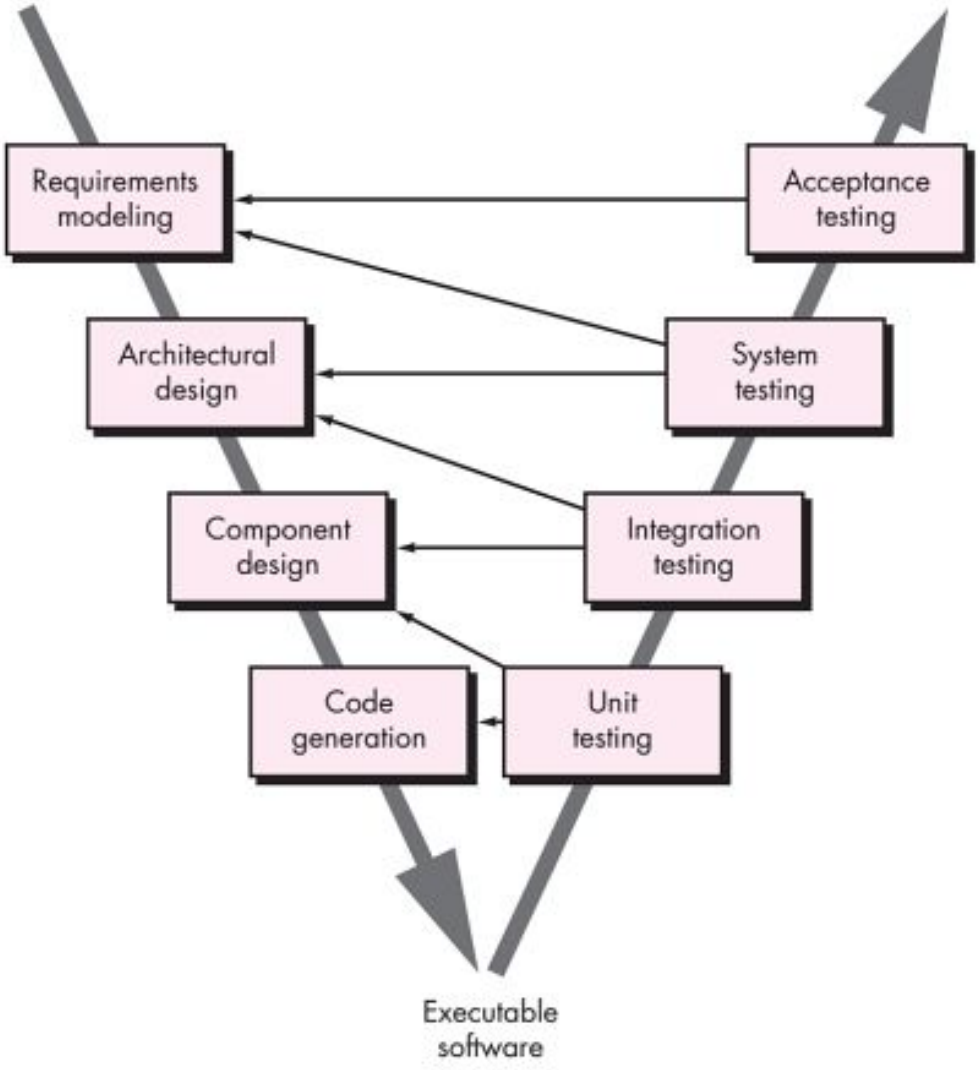
Automated Software Testing

João Henrique Victorino da Silva

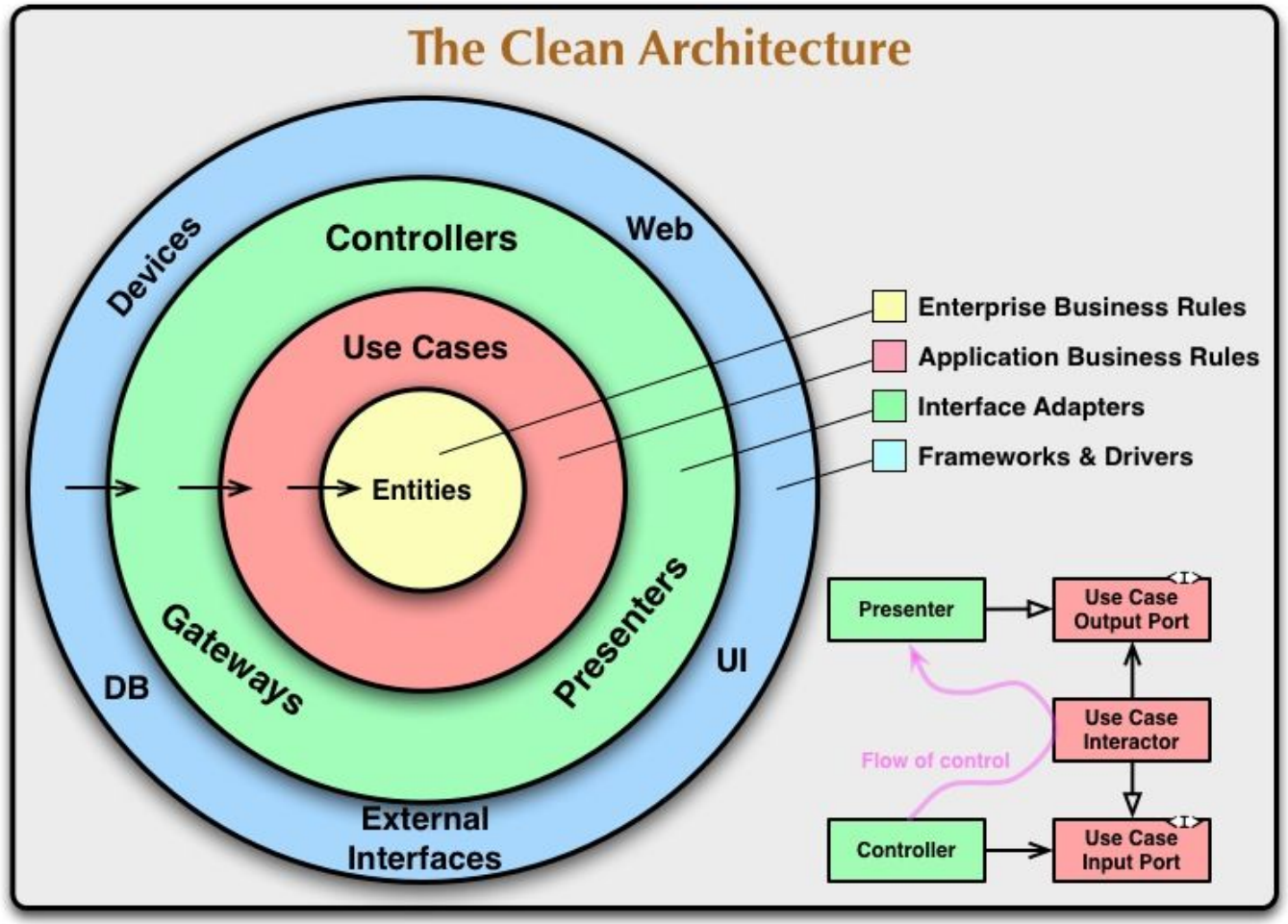
Pirâmide de testes



V-Model


Requirements modeling
Acceptance testing
Architectural design
System testing
Component design
Integration testing
Code generation
Unit testing
Executable software

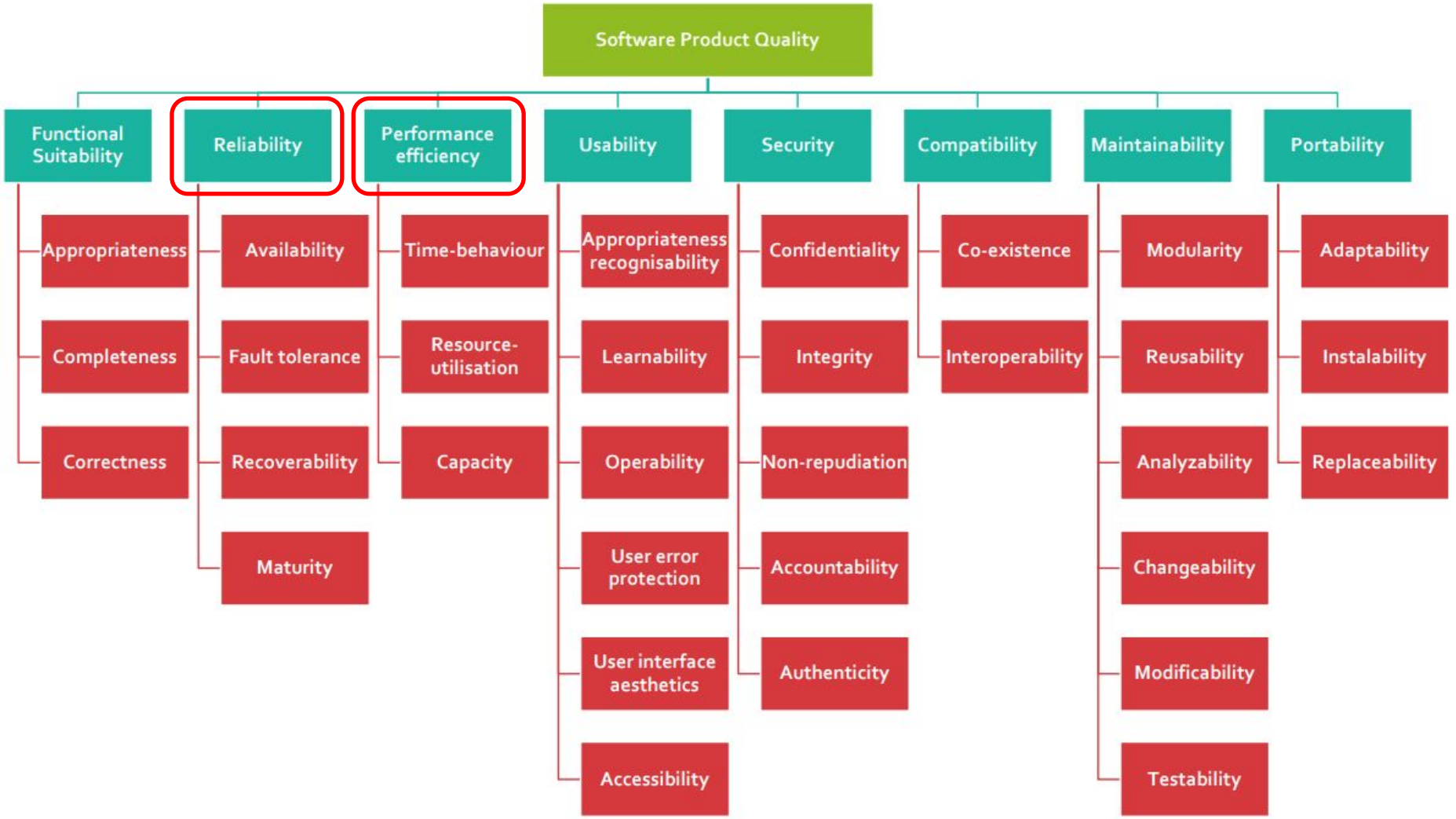
Clean Architecture



Granularidade dos testes

- **Testes de unidade**, classes e métodos isolados
- **Testes de integração**, um componente ou mais
- **Teste de sistema**, todo o sistema, relação entre componentes e infraestrutura (RNF)
- **Teste de aceitação**, todo o sistema, relação entre componentes e infraestrutura (RF)

ISO/IEC 25010 (Quality Model)



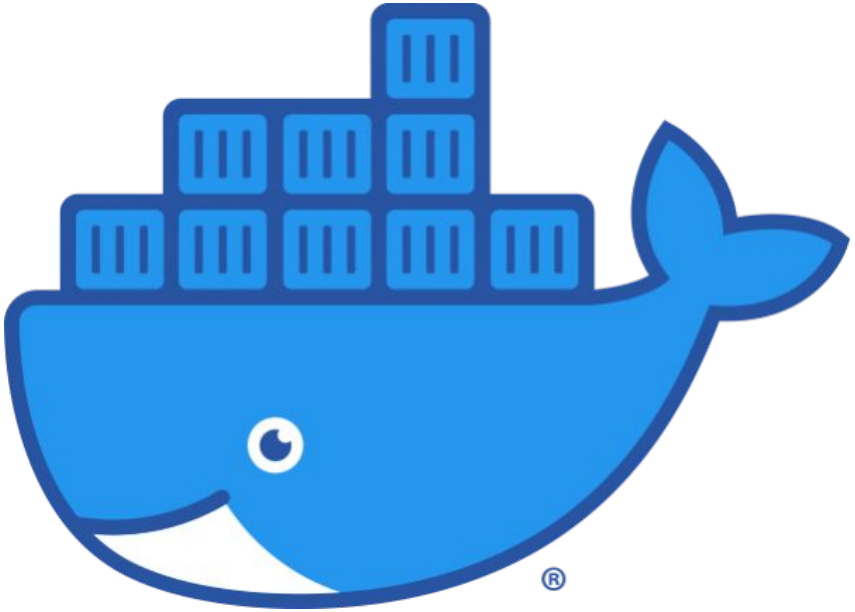
Testes de sistema

Testes não funcionais, é o momento de incluir a infraestrutura, preferencialmente na proporção e configuração de produção:

- **Disponibilidade**, exemplo, Netflix Chaos Monkey
- Segurança, tentativas de ataques e vulnerabilidades
- **Desempenho**, testes de tempo de resposta dada uma sobrecarga (JMeter, k6, etc)
- Compatibilidade, rodar em diversas plataformas, sistemas operacionais e browsers

Docker

https://docs.docker.com/



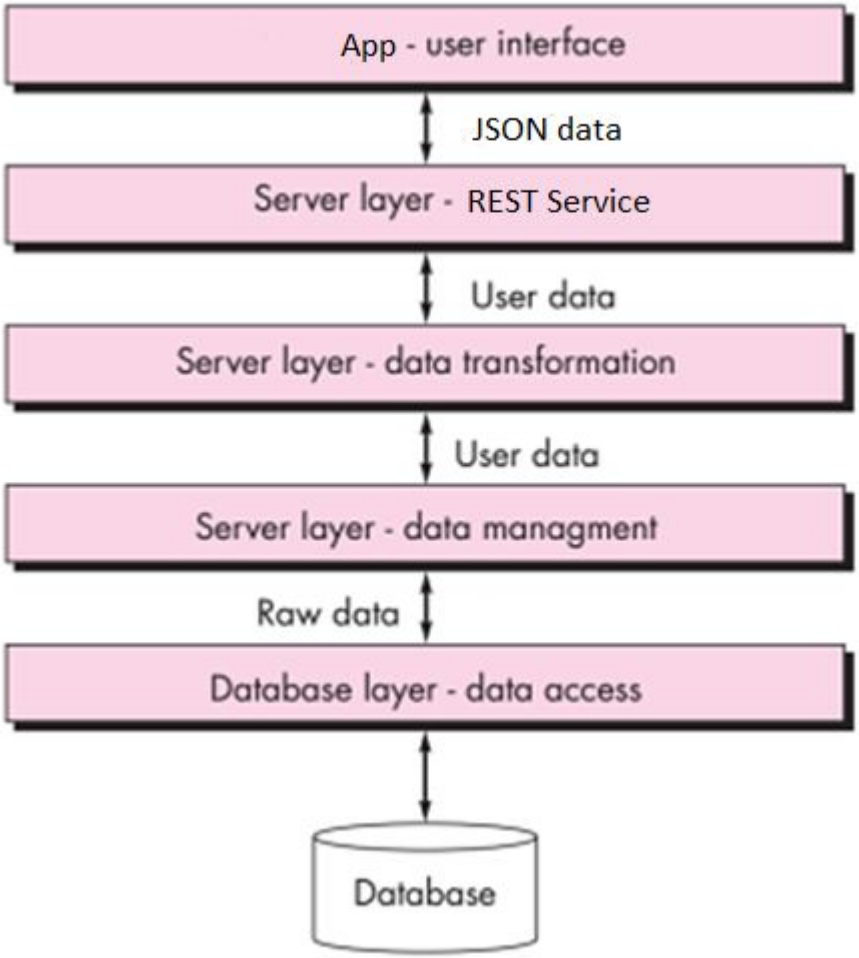
Conteinerizando a aplicação

https://gist.github.com/joaovictorino/49ba7aecce159ed15c078c7e4dde581b

(15 minutos)

“ What cannot be measured,
cannot be improved.”

Testes de desempenho



Desempenho

"É o nível ao qual o sistema consegue atender aos requisitos de vazão e latência em números de transações por tempo ou tempo para executar apenas uma transação"

latência

Tempo entre a chegada de um evento e a geração de uma resposta

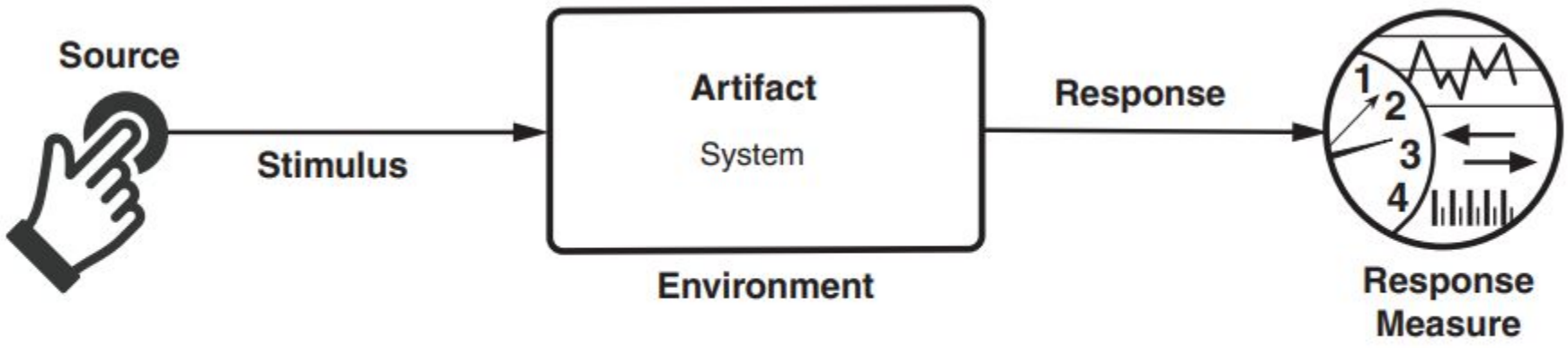
vazão (throughput)

Quantidade de eventos processados dentro de um período de tempo

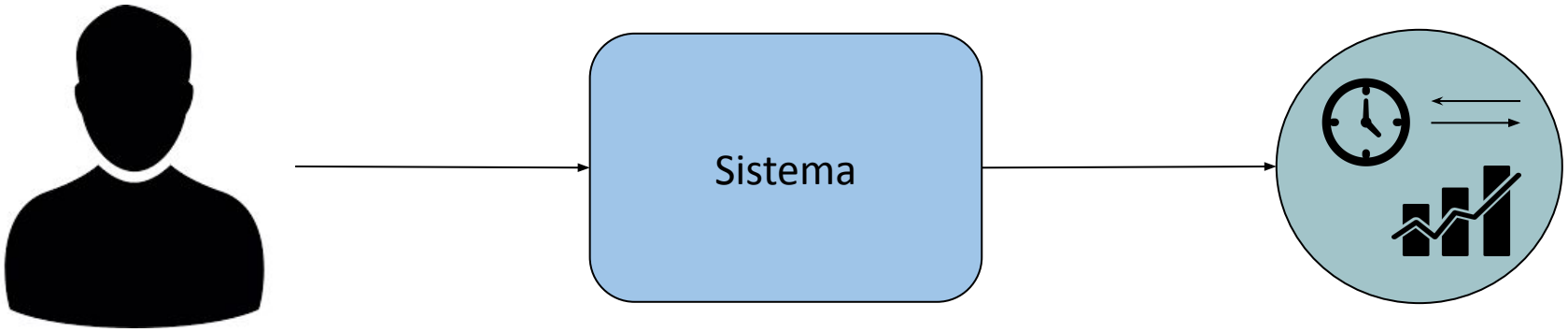
Cenário geral (Desempenho)

Fonte de estímulos	Pode vir de dentro e de fora do sistema, como: usuários, sistemas externos, componentes internos, etc
Estímulo	Eventos que chegam ao sistema de forma periódica, esporádica ou aleatória
Artefato	Sistema, módulo, componente, etc
Ambiente	Em modo normal, sobrecarga ou pico de demanda
Resposta	Processamento do estímulo, pode alterar seu estado de normal para sobrecarga
Medida da resposta	Tempo gasto para processar os eventos que chegam, variação de tempo, número de eventos processados dentro de um tempo (throughput) ou eventos que não puderam ser processados

Desempenho (Cenário Concreto)



Desempenho (Cenário Concreto)



Usuários

Iniciar transação

Sistema sob condições normais de uso

Transação processada

Com latência de 2 segundos (p99)

Tipos de testes de desempenho

- **Teste de carga** (load) avalia o desempenho do seu sistema sob condições normais esperadas.
- **Teste de estresse** avalia o desempenho de um sistema em seus limites quando excede a carga média esperada.
- **Teste de imersão** (soak) avalia a confiabilidade e o desempenho do seu sistema durante longos períodos.
- **Teste de pico** (spike) valida o comportamento do sistema em casos de aumentos repentinos e curtos.
- **Teste de quebra** (breakpoint) aumenta gradualmente a carga para identificar os limites de capacidade do sistema.

https://k6.io/docs/test-types/load-test-types/

Tipos de testes de desempenho

Tipo	Acessos	Duração	Quando?
Carga	Média de produção	5 - 60 minutos	Frequentemente para verificar se o sistema mantém o desempenho médio
Estresse	Acima da média	5 - 60 minutos	Quando o sistema pode receber acessos acima da média para verificar como ele gerencia
Imersão	Média de produção	Horas	Depois de mudanças para verificar o comportamento do sistema sob uso contínuo
Pico	Muito alta	Poucos minutos	Quando o sistema se prepara para eventos sazonais ou recebe picos de tráfego frequentes
Quebra	Até quebrar	Até quebrar	Algumas vezes para encontrar os limites do sistema

K6

https://k6.io/

Iniciando os testes de desempenho

https://gist.github.com/joaovictorino/cac03fbb7efc8c09a5dc0b401495965b

(10 minutos)

Testando a transferência por API

https://gist.github.com/joaovictorino/7ab933a49b2d27c5a2eeed80a631407c

(10 minutos)

Trabalhando com massa de dados

https://gist.github.com/joaovictorino/5557d233d2dac9bb2f90c9ab1757f468

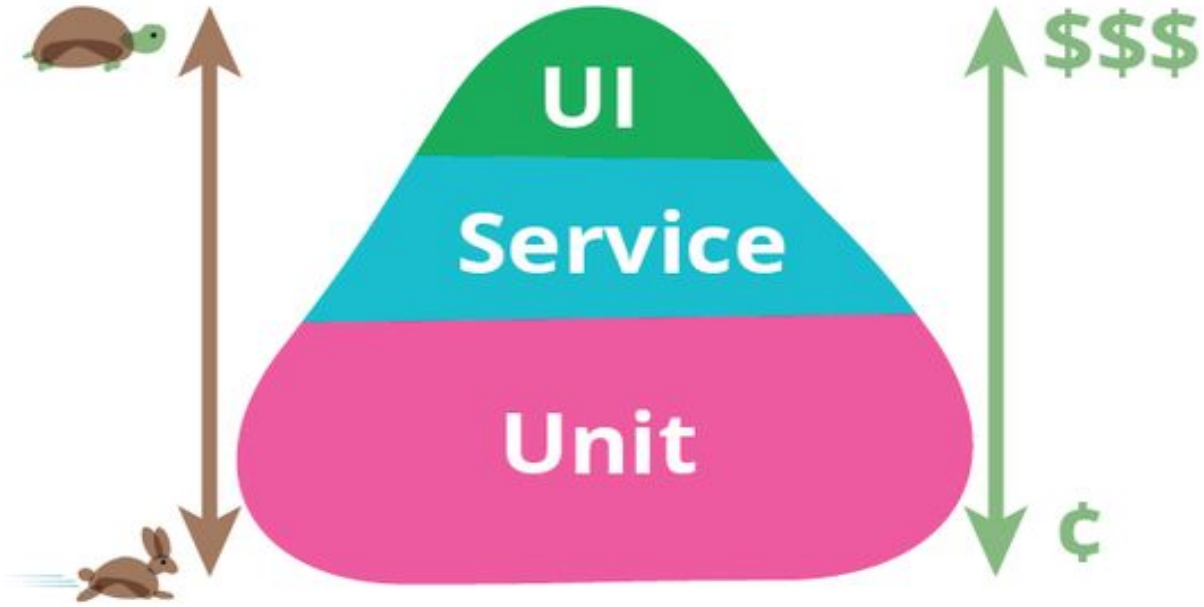
(10 minutos)

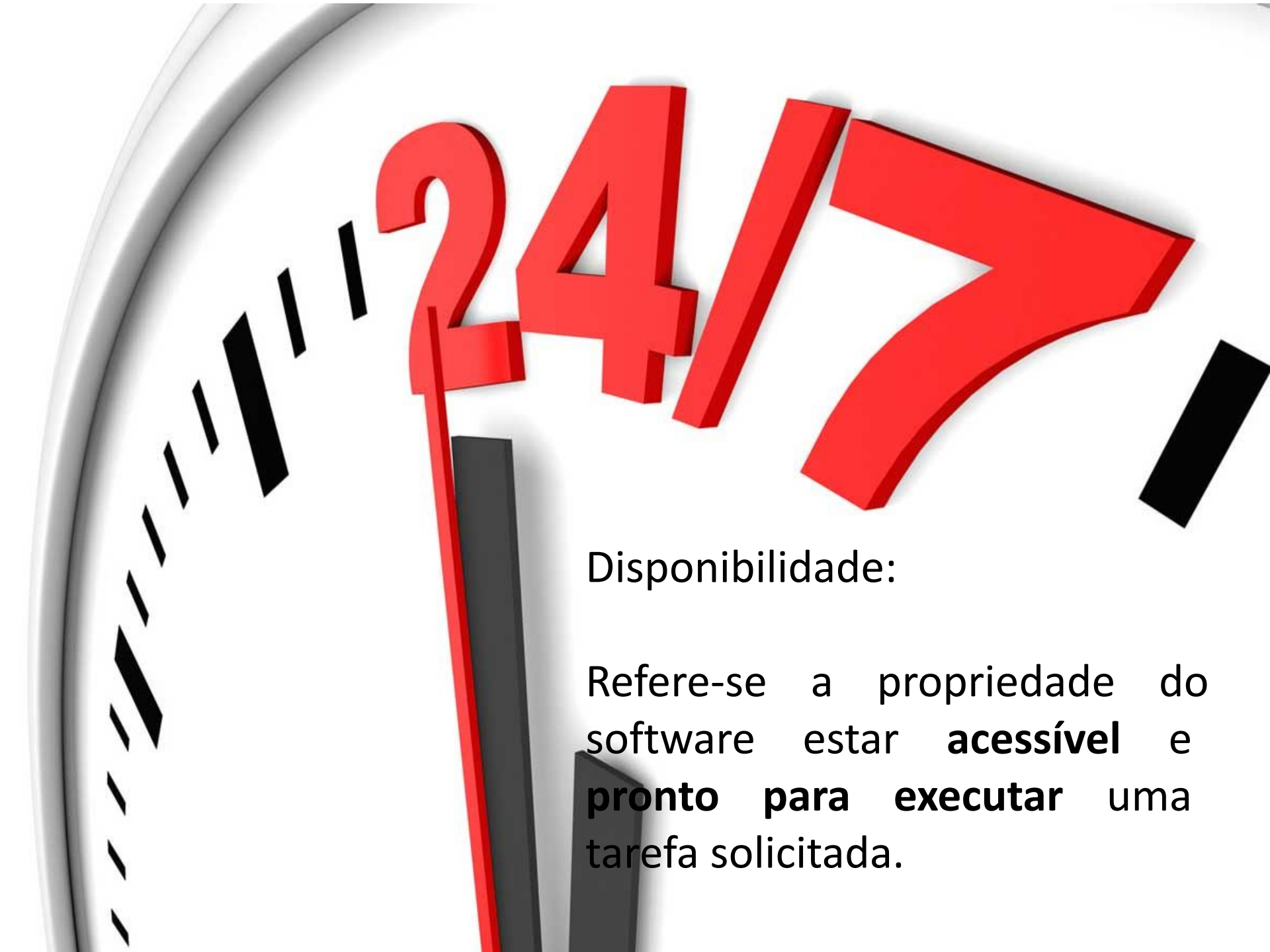
Testes de sistema

Testes não funcionais, é o momento de incluir a infraestrutura, preferencialmente na proporção e configuração de produção:

- **Disponibilidade**, exemplo, Netflix Chaos Monkey
- Segurança, tentativas de ataques e vulnerabilidades
- **Desempenho**, testes de tempo de resposta dada uma sobrecarga (JMeter, k6, etc)
- Compatibilidade, rodar em diversas plataformas, sistemas operacionais e browsers

Pirâmide de testes





Disponibilidade:

Refere-se a propriedade do software estar **acessível** e **pronto para executar** uma tarefa solicitada.

Disponibilidade

Mascarar ou **reparar erros** de modo que o tempo de interrupção não seja superior a um intervalo de tempo especificado.

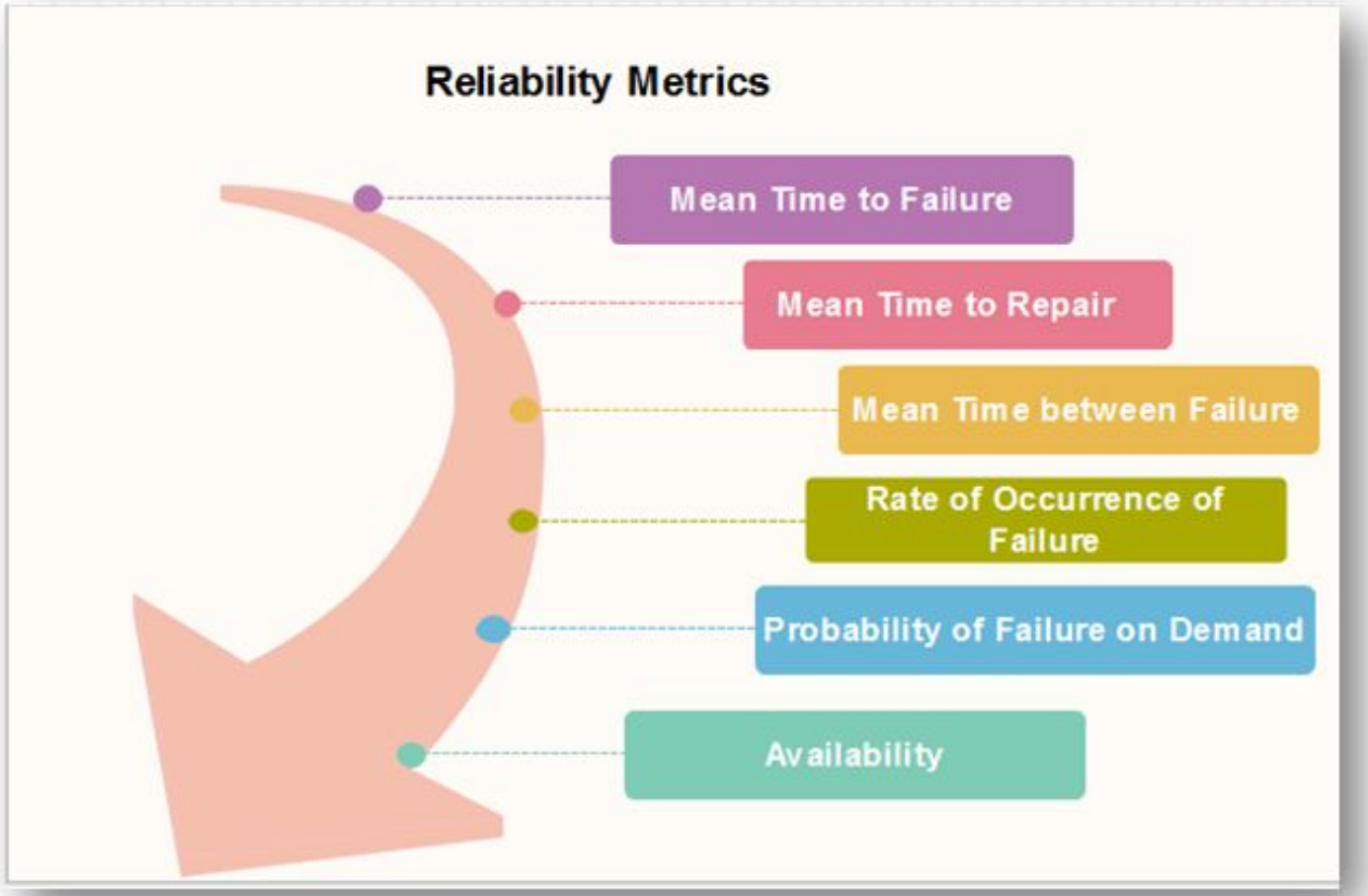
Tempo disponível ou indisponível (SLA)

$$Availability = \frac{Successful\ Requests}{Total\ Requests} = \frac{71{,}950}{72{,}000} = 99.930\%$$

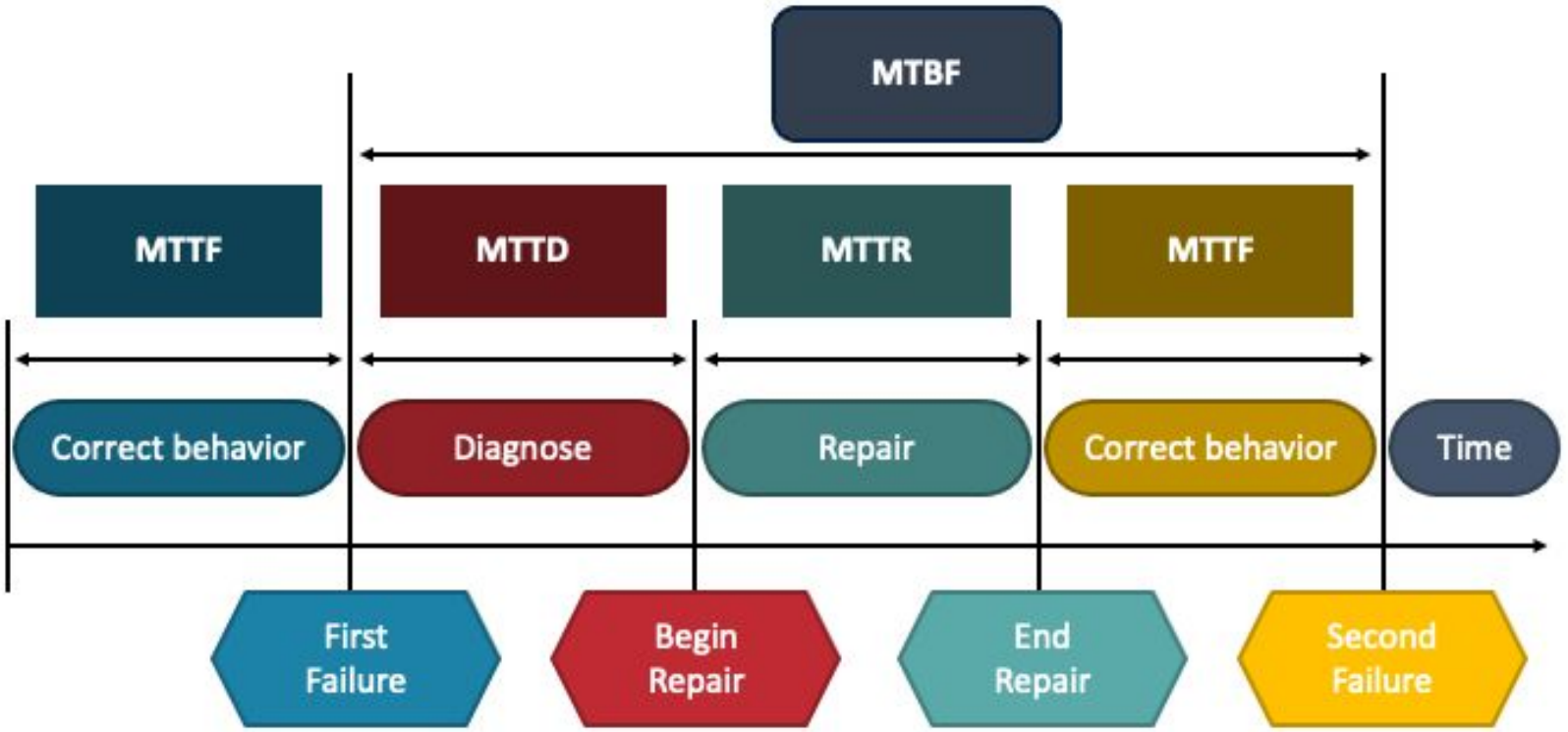
Table 1 - Levels of Availability			
Level of Availability	Percent of Uptime	Downtime per Year	Downtime per Day
1 Nine	90%	36.5 days	2.4 hrs.
2 Nines	99%	3.65 days	14 min.
3 Nines	99.9%	8.76 hrs.	86 sec.
4 Nines	99.99%	52.6 min.	8.6 sec.
5 Nines	99.999%	5.25 min.	.86 sec.
6 Nines	99.9999%	31.5 sec.	8.6 msec

https://uptime.is/

Métricas


Reliability Metrics
Mean Time to Failure
Mean Time to Repair
Mean Time between Failure
Rate of Occurrence of Failure
Probability of Failure on Demand
Availability

Métricas de disponibilidade



Métricas de disponibilidade

Fault recovery mechanism	Estimated MTTR
Launch and configure new virtual server	15 minutes
Redeploy the software	10 minutes
Reboot server	5 minutes
Restart or launch container	2 seconds
Invoke new serverless function	100 ms
Restart process	10 ms
Restart thread	10 µs

https://docs.aws.amazon.com/whitepapers/latest/availability-and-beyond-improving-resilience/availability-and-beyond-improving-resilience.html

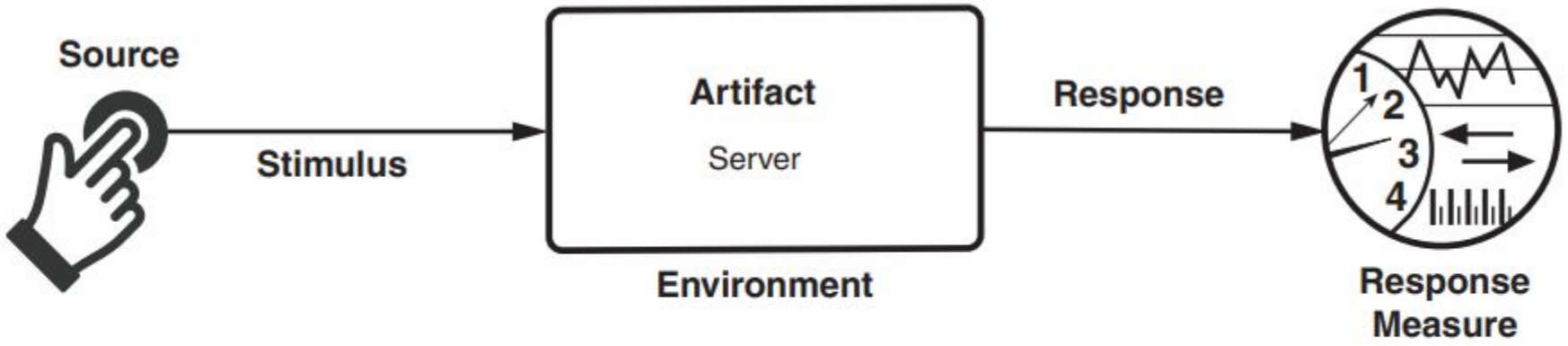
erro (*fault*) x falha (*failure*)

Um erro pode se transformar em uma falha se não for corrigido ou mascarado.

Cenário geral (Disponibilidade)

Fonte de estímulos	Interno ou externo ao sistema
Estímulo	Erro: omissão, quebra, tempo de resposta ou resposta
Artefato	Canais de comunicação, processos, processadores e persistência
Ambiente	Em modo de funcionamento normal ou degradado (poucas funcionalidades ou solução de contorno)
Resposta	Sistema detecta o erro e pode: gravar, notificar interessados, desabilitar recursos com erros, ficar indisponível por um período de tempo ou continuar a funcionar em modo normal ou degradado
Medida da resposta	Tempo que o sistema deve estar disponível, tempo que o sistema pode funcionar degradado e tempo de reparo

Disponibilidade (Cenário Concreto)



Disponibilidade (Cenário Concreto)

Componente de integração com agência externa

Agência externa de crédito

Erro de omissão

Operação normal

Informar operador, logar e tentar novamente mais tarde

Sem inatividade (SLA 98%)

Kubernetes

https://kubernetes.io/docs/home/

kubernetes

Habilitando o Kubernetes

https://gist.github.com/joaovictorino/466d0b885ed33d9c2f2d26b793bdf3a6

(20 minutos)

Subindo a aplicação

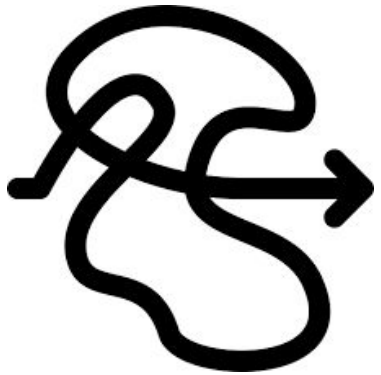
https://gist.github.com/joaovictorino/47c3775911a1d83dcf1b6360701e6c05

(20 minutos)

Chaos Engineering

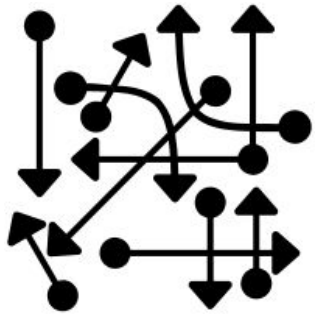
Chaos Engineering is the discipline of experimenting on a system in order to build confidence in the system's capability to withstand turbulent conditions in production.

Fonte:
https://principlesofchaos.org/



Principles of Chaos Engineering

- Construa uma hipótese, algo que possa ser medido externamente no sistema (taxas de erros, exceções, latência, vazão, etc)
- Crie eventos distintos (aplicação e infraestrutura)
- Roda experimentos em produção
- Automatize os experimentos para rodar constantemente
- Minimize os efeitos dos problemas no usuário final



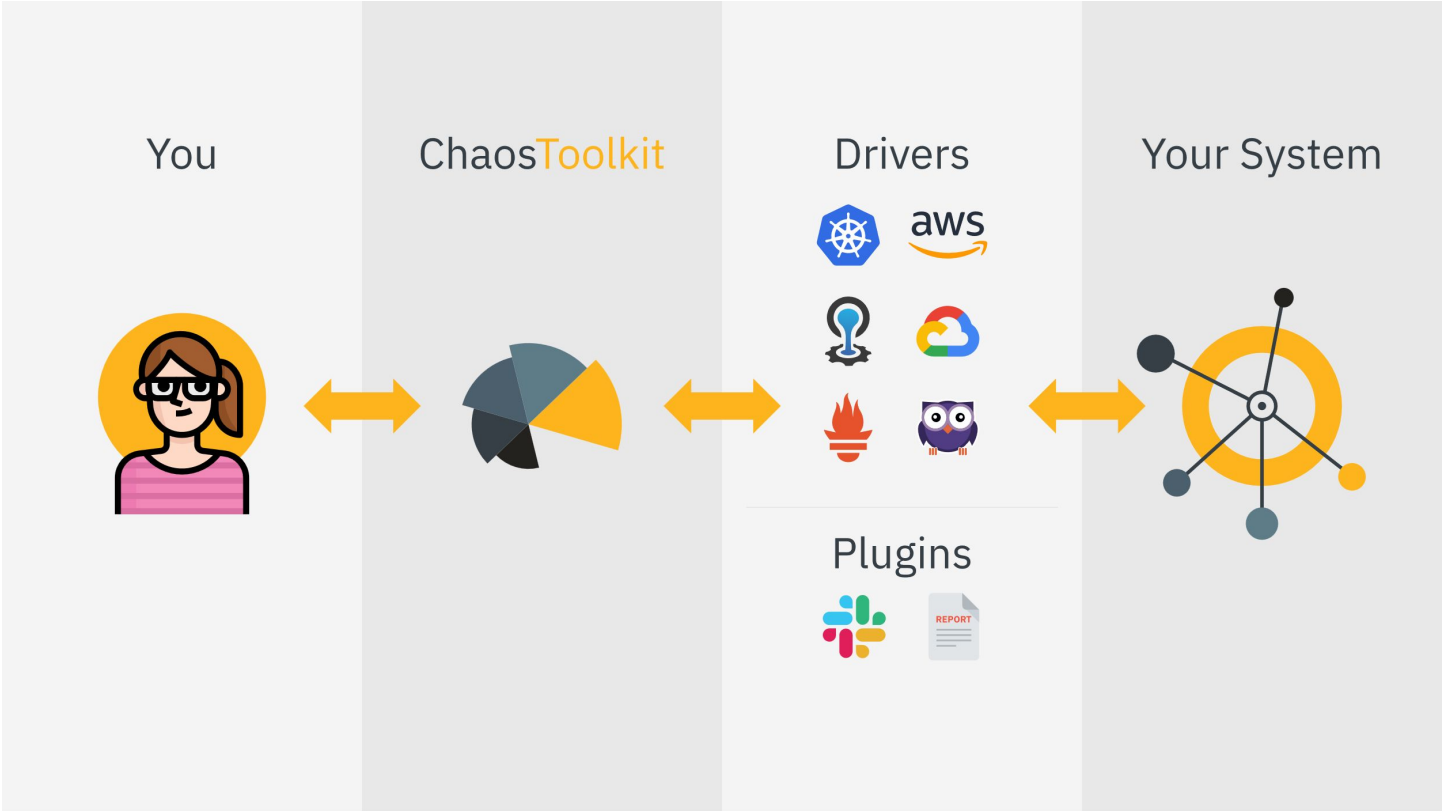
Chaos Engineering

- Execute durante o dia
- Avise ao time que os testes estão acontecendo e permaneça monitorando o ambiente a procura de comportamentos estranhos
- Existem diversos tipos de falhas, desde desativar um servidor até desabilitar sua comunicação com outros servidores
- Dê tempo para corrigir os problemas e executar testes novamente

Chaos Monkey

https://github.com/netflix/chaosmonkey

Chaos Toolkit



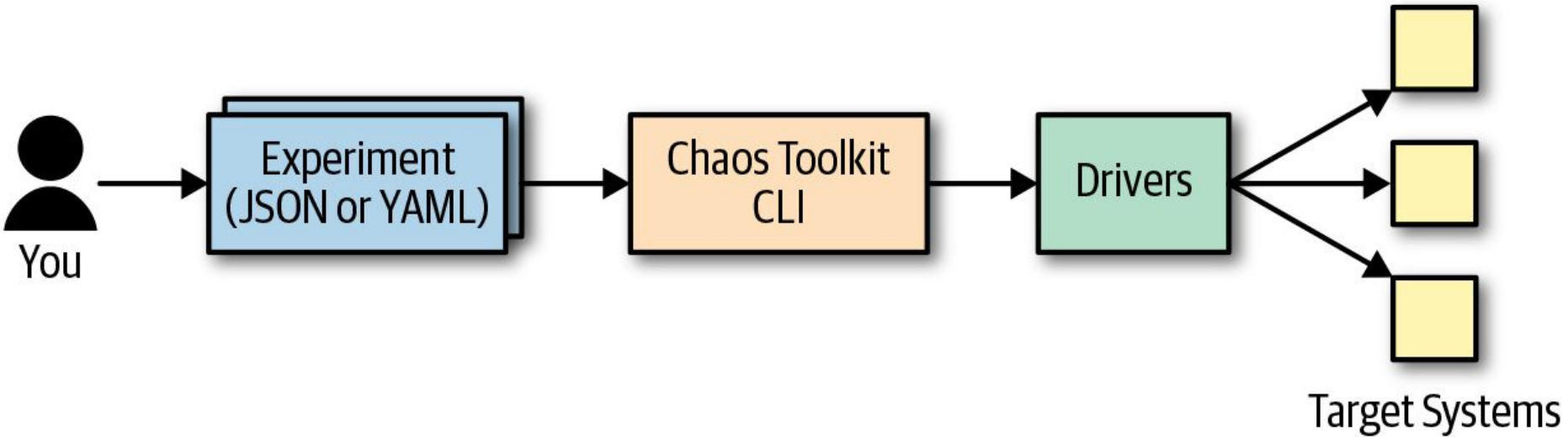
https://chaostoolkit.org

Instalando o Chaos Toolkit

https://gist.github.com/joaovictorino/e4278a346e27e07aa91b2fc00ed3cbb9

(10 minutos)

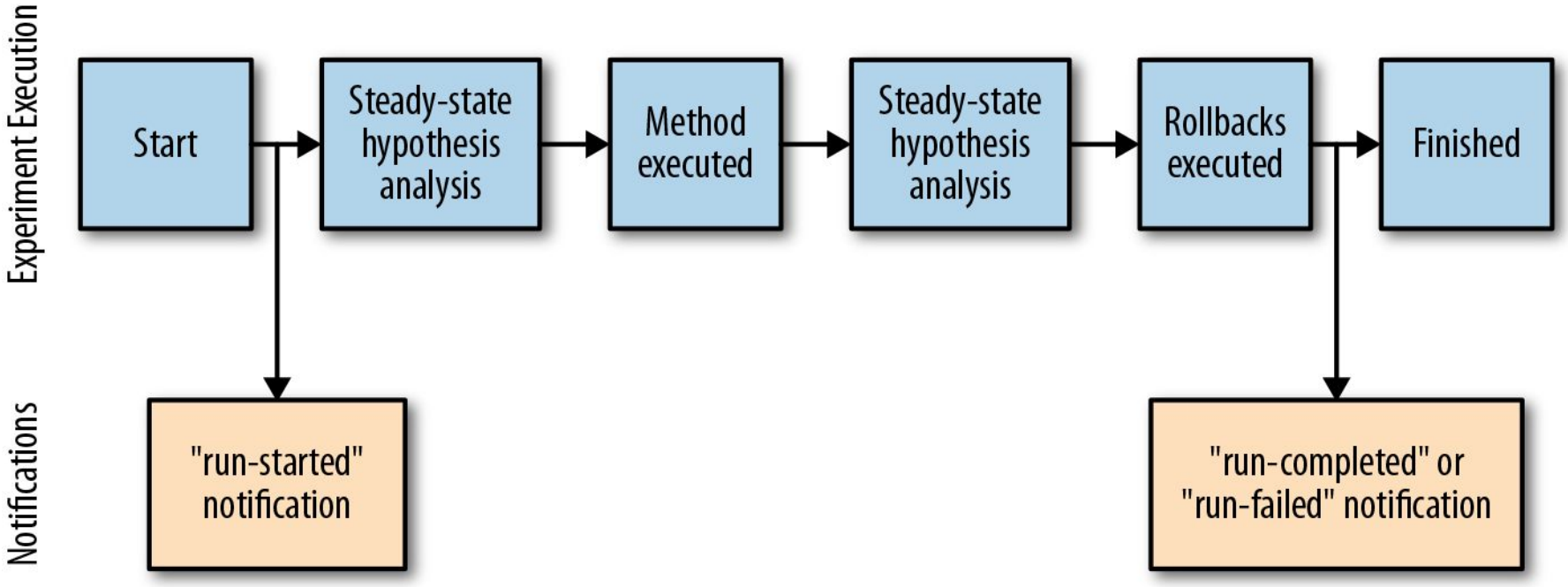
Chaos Toolkit



Conceitos Chaos Toolkit

- experiment
  - steady-state-hypothesis
    - probe
  - method
    - action
    - probe
  - Rollbacks (opcional)
    - action

Experimento



Criando experimento de aplicação

https://gist.github.com/joaovictorino/d8ef42fd0545346a182590647710e695

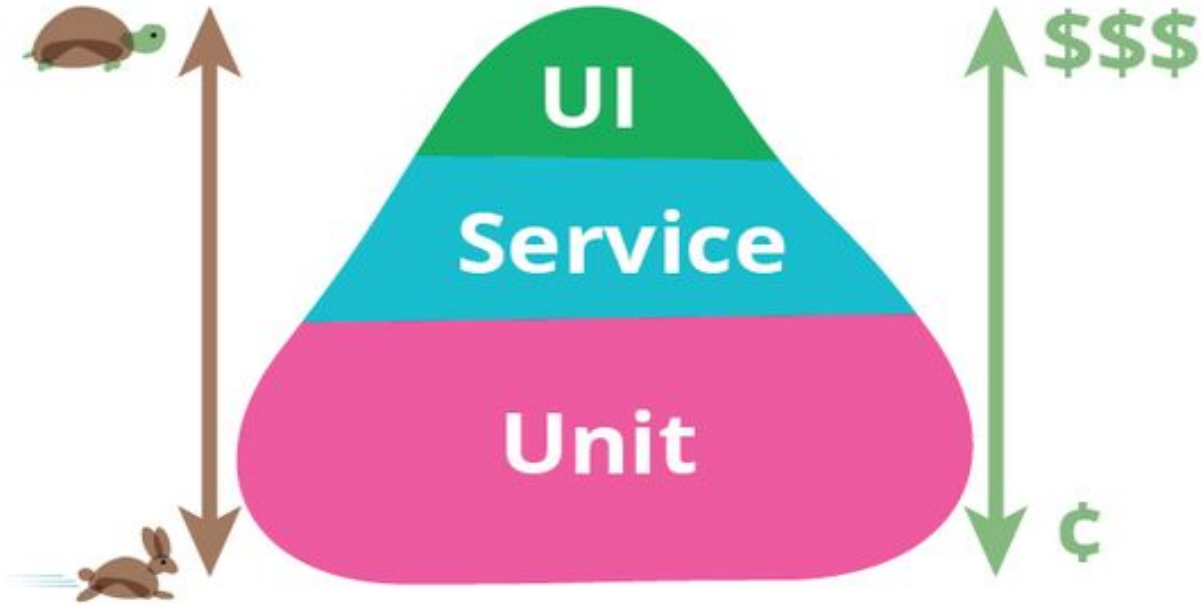
(10 minutos)

Criando experimento de banco

https://gist.github.com/joaovictorino/41c7b3380b854d86de0d470c14ca4308

(10 minutos)

Pirâmide de testes



Faculdade IMPACTA

BOA NOITE!

Referência no ensino de Tecnologia em **Graduação**, **Pós**, **EAD**, **MBA**, **cursos livres** e **treinamentos empresariais**. Lidera iniciativas em Inteligência Artificial, Ciência de Dados, Robótica, Engenharia da Computação, Big Data, UX e Transformação Digital. Criou a incubadora Impacta Open Startup, exclusiva para seus estudantes. É campeã na formação de times para Hackathons, eventos de inovação patrocinados por gigantes: NASA, IBM, Deloitte, Shell, Santander, Itaú, Globo e Fiesp. É uma das marcas mais admiradas pela Comunidade Tech da América Latina. Atua no mercado desde 1988 e formou mais de 1 milhão de pessoas e certificou mais de 25 mil empresas de diversas áreas da Economia. Ver mais em: impacta.edu.br e impacta.com.br