

# *Rational Unified Process (RUP) e TMap (Testing Management Approach)*

**Disciplina:** Qualidade de Software



## Conteúdos:

*Rational Unified Process (RUP) e TMap (Testing Management Approach).*

## Habilidade(s):

- Utilizar os métodos RUP e TMap para compreender o teste de desenvolvimento de um *software*.

# Bloco 1

---

Compreendendo os aspectos principais sobre o  
*Rational Unified Process.*

# Quem é você?

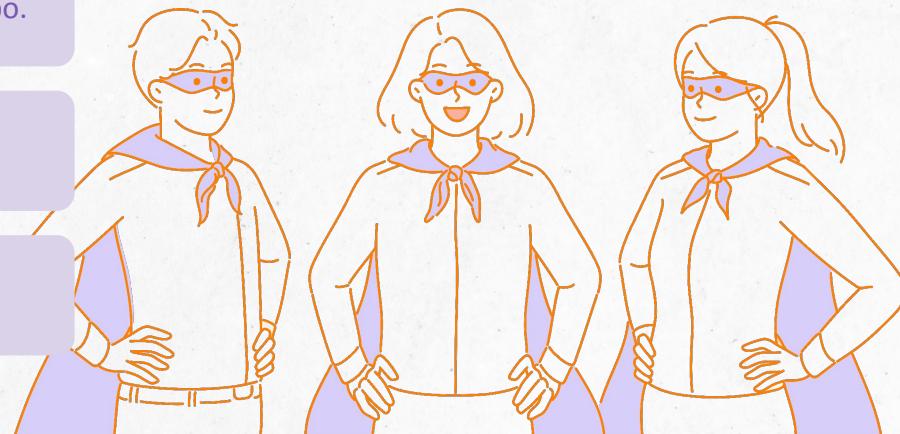
Na hora de fazer um trabalho em grupo, qual super-herói você é?

1 **Capitão Procrastinação.** Deixo tudo para a última hora.

2 **Multitarefa.** Consigo fazer várias coisas ao mesmo tempo.

3 **Solução Fácil.** Sou ótimo para resolver problemas.

4 **Verbo de Ação.** Eu tomo a iniciativa.



# A importância da equipe em um projeto

Durante o desenvolvimento de *software*, a equipe precisa garantir uma alta qualidade de produção para atender às necessidades do usuário.

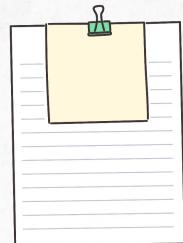
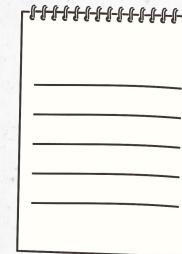
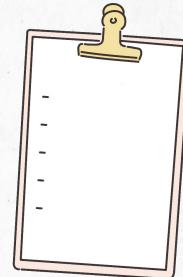
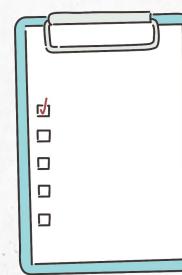
Para isso, cada profissional exerce um papel fundamental. É como se eles fossem os super-heróis da computação, preocupados em entregar o produto final!



# Organizar-se é essencial

Para garantir uma boa organização do projeto, é indispensável que a equipe utilize metodologias eficazes.

Assim como um super-herói precisa de uma tecnologia para trabalhar, os desenvolvedores dispõem de boas estratégias ao longo do seu projeto.



## Na hora de se organizar...

O que faz mais o seu estilo? Bloco de notas, um calendário físico ou um aplicativo de tarefas?

??



Mas...

E se existisse uma plataforma que possibilitasse organizar todos os seus prazos de acordo com cada tarefa?



# Rational Unified Process

Para isso, existe o *Rational Unified Process* (RUP), que, em português, significa Processo Racional Unificado.

**O que é:** um programa de engenharia de software criado para atender às necessidades dos usuários.

**Como foi criado:** através da Rational Software Corporation, adquirido em 2003 pela IBM.

**Por que usar:** gera o modelo de um projeto e define como as coisas devem ser feitas.



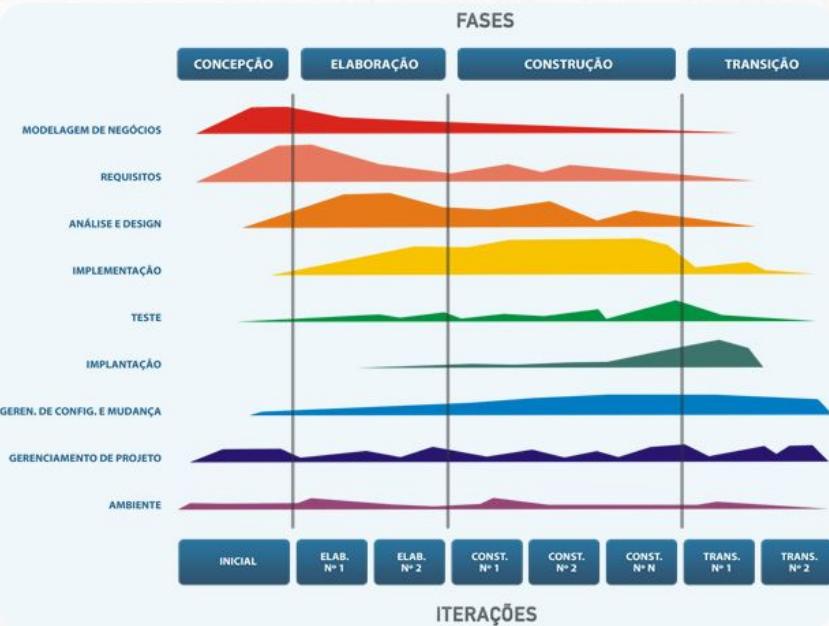
# Interface do RUP

Através desse programa, é possível:

definir quem é o responsável por cada atividade;

como e quando as atividades devem ser realizadas;

acessar a descrição de todas as metas estabelecidas.



Interface do programa *Rational Unified Process*.

# Fases do RUP

O programa constrói quatro fases principais para tratar o planejamento, o levantamento de requisitos, a análise, a implementação, o teste e a implantação.



**Fase de concepção ou iniciação:** são definidos os objetivos do projeto. Aqui, é onde acontece a comunicação com o cliente.



**Fase de elaboração:** análise profunda do problema apresentado no projeto, revisando os riscos que podem estar presentes.



**Fase de construção:** etapa em que os componentes do *software* são construídos.



**Fase de transição:** entrega do *software* e fase de testes.

# Bloco 2

---

Compreendendo os aspectos principais sobre a  
realização dos testes.

## Imagine o seguinte cenário

Você precisa organizar o seu projeto, mas o RUP não está funcionando. Sendo assim, você deve organizar as atividades a seguir em cada uma das quatro fases. Como você faz?

a) Criar um documento inicial de visão do projeto.

b) Elaborar um plano de desenvolvimento.

c) Identificar as necessidades dos usuários.

d) Desenvolver os componentes do aplicativo.

e) Realizar testes de aceitação.

# Gabarito

E aí, acertou?

a) Criar um documento inicial de visão do projeto.

Fase de concepção.

b) Elaborar um plano de desenvolvimento.

Fase de elaboração.

c) Identificar as necessidades dos usuários.

Fase de concepção.

d) Desenvolver os componentes do aplicativo.

Fase de construção.

e) Realizar testes de aceitação.

Fase de transição.

## E se...

E se você pudesse ter o trabalho de testar algo para o resto da sua vida todos os dias, o que você escolheria?



# A importância dos testes de *software*

Como você definiria um teste?

Na área dos *softwares*, existem algumas possíveis definições.

É analisar um programa com a intenção de descobrir erros e defeitos.

É exercitar ou simular a operação de um programa ou sistema.

É medir a qualidade e funcionalidade de um sistema.

É avaliar se o *software* está fazendo o que deveria fazer, de acordo com os seus requisitos, e, se não está fazendo, o que não deveria fazer.

É qualquer atividade que, a partir da avaliação de um atributo ou capacidade de um programa ou sistema, possibilita determinar se ele alcança os resultados desejados.



# Características básicas dos testes de softwares

De forma geral, os testes são essenciais para levantar as principais falhas de um programa, garantindo que ele atenda aos requisitos estabelecidos.

Eles costumam:

determinar se o produto atingiu todas as especificações do projeto.

verificar se o *software* funciona como o esperado na implementação.

estabelecer a qualidade do produto.

mostrar que os resultados estão de acordo com os padrões levantados.

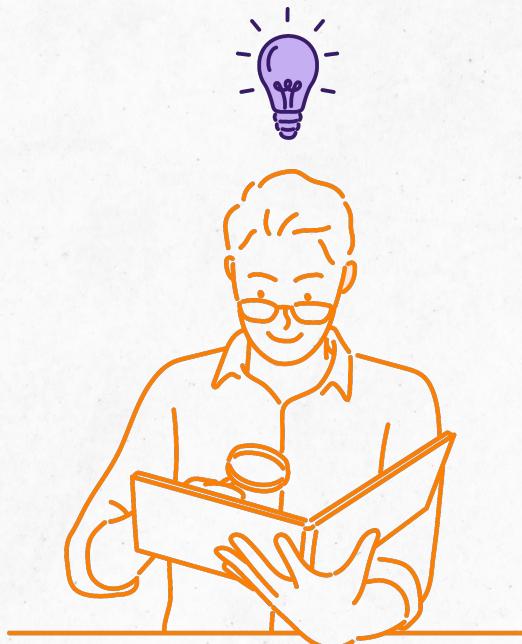
# ***Test Maturity Model Integration (TMMI)***

Entender o que precisa ser avaliado nem sempre é uma tarefa fácil. Por isso, o programa *Test Maturity Model Integration* (TMMI) pode ser um grande aliado.

É um guia e estrutura desenvolvida pela TMMI Foundation para melhorar os processos de teste.

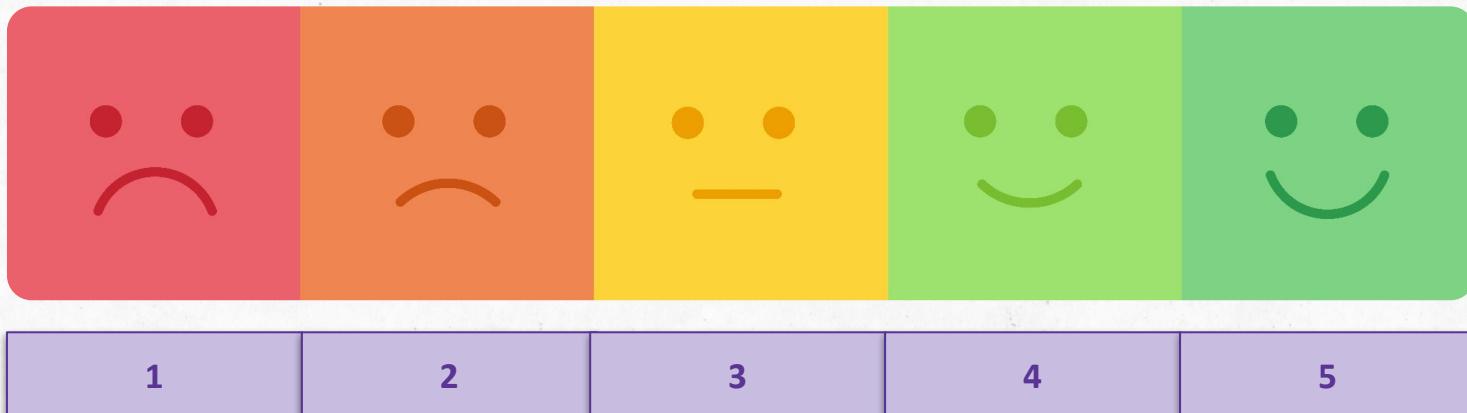
Inclui objetivos genéricos e práticas que podem ser usadas para institucionalizar o processo de teste em uma organização.

Pode ser usado para buscar certificação em um nível de maturidade específico ou como referência.



## Níveis do TMMI

O TMMI define cinco níveis de maturidade, do caótico ao otimizado. Cada nível possui requisitos chamados de **áreas de processo**, que são práticas específicas que devem ser adotadas para melhorar a qualidade dos processos de teste.



# Bloco 3

---

Vamos praticar os conhecimentos que foram  
aprendidos até aqui.



## Dê um *play* no conhecimento!



Assista como elaborar um mapa mental de forma correta.

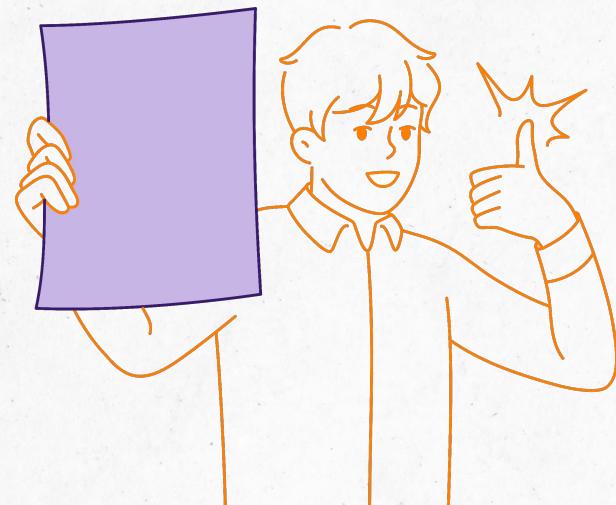
# Está preparado?

Separem-se em grupos e peguem todos os materiais de papelaria disponíveis para realizar a atividade.

Desenvolvam uma ideia de um aplicativo fictício e, em seguida, utilizem a cartolina para elaborar um mapa mental de como os seus requisitos poderiam ser distribuídos no RUP e TMMI.

Se liga no que precisa ter na cartolina:

- a ideia do aplicativo;
- como ele seria desenvolvido no RUP, caso existisse;
- como ele seria desenvolvido no TMMI, caso existisse.



# Bloco 4

---

Por que é tão importante testar?

# Imagine os seguintes cenários

Você precisa sacar dinheiro, mas o aplicativo do banco não está funcionando.

Você está perdido no meio de uma rua desconhecida e o aplicativo do mapa não para de travar.

Você comprou um produto, mas o aplicativo deu erro e efetuou duas compras.

Você precisa mandar uma mensagem para a sua mãe, mas o teclado do seu celular não aparece corretamente.

Nós lidamos com uma série de dificuldades e complicações nos aplicativos que usamos no dia a dia.

Você sabe por que isso acontece?

# Pane no sistema, alguém me desconfigurou

Na verdade, as falhas que ocorrem no *software* são processos naturais de todo programa.

É comum que os seres humanos cometam erros e provoquem *bugs* no sistema durante a codificação de um *software*.

Você conhece a “tela azul”? É chamado de “*blue screen of death*” (em tradução livre, tela azul da morte) e ocorre quando um sistema sofre um erro crítico e precisa ser reiniciado.

A problem has been detected, your beloved operating system has been shut down to prevent damage to your computer. An attempt to reset the world and recover mental health has failed.

If this is the first time you've seen this stop error screen, restart your computer.  
If this screen appears again, follow these steps while praying:

Pray more. Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any updates you might need.

If problems continue, stop believing in computer gods, disable or remove any newly installed hardware or software, disable enhanced memory options, stop using the blockchain and Hodling.

If you need to get your s\*\*t done, borrow another computer that is not a total tool.  
Press F8 to select Killing Spree Mode, and then select Unleash The Beast.

Technical information:

\*\*\* STOP: 0x00000024 (0x001902F8, 0xF1534434, 0xF153414, 0xF8408BAB)  
\*\*\* NttN.sys - Address F7850024 base at F784A000, DateStamp 45d6a04b

Collecting data for crash test dummy dump...  
Initializing disk for crash test dummy dump...  
Beginning wipeout of physical memory...  
Physical memory wipeout complete.  
Now you don't remember anything.  
Your OS is perfectly fine.  
You love it.  
Press any key.

??

Então...

Como construir um *software* confiável?



## A resposta correta é:

À medida que um *software* é executado ao longo do tempo e não apresenta falhas, ele pode ser considerado confiável!



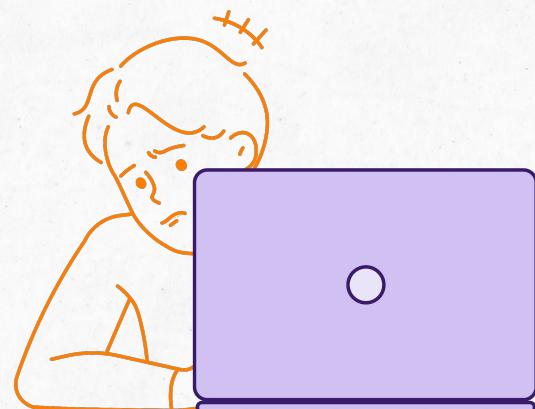
## Alguns exemplos

Embora seja comum, erros na codificação de um *software* podem levar a riscos graves para a um indivíduo.

Confira alguns *bugs* famosos a seguir:

Em 2015, a Bloomberg sofreu uma falha em um *hardware* e um *software* que geram atividades de um mercado financeiro. A consequência foi um impacto em mais de 320,000 pessoas ao redor do mundo.

Em 2016, a Nissan, empresa de automóveis, detectou uma falha de *software* nos seus sensores de *airbag*. Com isso, mais de 3 milhões de carros apresentaram falha no sensor de proteção.



A large orange line-art illustration of a woman with long hair, wearing a white t-shirt and orange pants, shouting into a megaphone. Her mouth is wide open, and there are three small orange wavy lines above her head indicating sound. The megaphone has a speech bubble icon on its handle.

## Mas, atenção!

O teste por si só não garante a qualidade do *software*. Embora detecte falhas e defeitos, ele é apenas uma parte da Garantia de Qualidade de *Software*.

O essencial é investir em prevenção!

# Bloco 5

---

Entendendo a diferença entre Garantia da Qualidade e  
Controle da Qualidade.

# Qual é a diferença?

Você consegue indicar corretamente as principais diferenças entre as duas?



Garantir a qualidade

VS



Controlar a qualidade

# Gabarito

Garantir a qualidade	Controlar a qualidade
Certifica que o processo é definido e apropriado.	Foca na descoberta de defeitos.
Alguns exemplos são a metodologia e padrões de desenvolvimento.	Um exemplo é se os requisitos definidos estão corretos.
Orientada ao processo.	Orientada ao produto.
Orientada à prevenção.	Orientado à detecção.
Foco em monitoração e melhoria.	Certifica-se que o produto atenda aos requisitos.
As atividades são focadas no início do ciclo de vida.	As atividades são focadas nas etapas finais do ciclo de vida.
Garante que as coisas estão sendo feitas da maneira certa.	Garante que o seu trabalho segue como o esperado.

# A importância da equipe

Durante as fases de Garantia e Controle de Qualidade, cada membro da equipe exerce um papel fundamental.

A seguir, confira um guia dos principais.

## Analista de qualidade (QA)

Molda o *software* através de testes  
rigorosos.

## Scrum Master/ Agile Master

Atua como um solucionador de problemas  
e removedor de obstáculos, assegurando  
que o processo flua suavemente e que o  
time alcance o seu potencial máximo.

# A importância da equipe

## **Product Owner (Dono do Produto)**

Determina quais funcionalidades devem ser desenvolvidas e em qual ordem de prioridade.

## **Analista de Segurança**

Promove a cultura DevSecOps, integrando a segurança desde o início do processo, identificando possíveis vulnerabilidades e assegurando que o *software* esteja imune a ameaças.

## **Testador / Automatizador / Arquiteto de Teste**

Enquanto os testadores elaboram casos de teste minuciosos, os automatizadores transformam-nos em *scripts* automatizados, e os arquitetos de teste constroem a infraestrutura necessária.

## **Líder de Teste**

Eles planejam, estimam, coordenam e orquestram toda a operação de teste, assegurando que cada movimento seja calculado e que a missão de garantir a qualidade seja cumprida com sucesso.

# Bloco 6

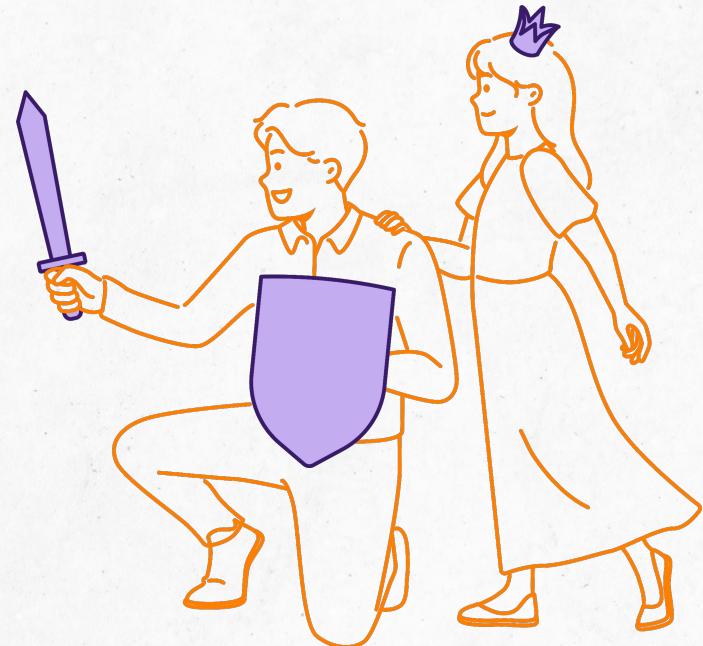
---

Hora de praticar!

## Vamos realizar uma dinâmica?

Vocês devem se lembrar dos papéis das equipes que participam de um projeto de *software*, citados no bloco anterior. Nesta atividade, vocês precisarão incorporar cada um deles, mas de uma forma diferente.

Cada um será um personagem de um jogo de aventura e, juntos, vocês receberão vários desafios, a fim de avaliar o desempenho da equipe como um todo.



# Distribuição de papéis

Separuem-se em grupos e distribuam os papéis entre si.

1 Analista de Qualidade (QA): o Guardião dos Padrões de Qualidade.

2 Scrum Master/*Agile Master*: o Facilitador Ágil.

3 *Product Owner* (Dono do Produto): o Visionário das Prioridades.

4 Analista de Segurança: o Defensor da Fortaleza Digital.

5 Testador/Automatizador/Arquiteto de Teste: o Teia Tecida com Precisão.

6 Líder de Teste/Gerente de Teste: os Estrategistas de Teste.

# Desafios

Cada equipe receberá quatro desafios, que deverão solucionar ao longo da atividade. Sendo assim, o objetivo é que as atividades sejam distribuídas para cada grupo de maneira estruturada, a fim de se aproximar do cenário real de um projeto.

Anotem em folhas de ofício as ideias que tiverem sido debatidas.

1

Desenvolvam requisitos necessários para um aplicativo de viagens em um cenário da **Idade Média**, com castelos, tavernas e estradas.

2

Após descreverem as funcionalidades, digam como implementariam a solução.

3

A equipe deve descrever a maneira como garantiriam a segurança do aplicativo.

4

Conforme o que foi construído, o grupo precisa estabelecer pelo que cada membro ficaria responsável no projeto fictício.

É hora de discutir!



# Fechamento

O que eu já sabia?

O que eu não sabia?

O que eu quero aprender?

# Referências Bibliográficas

PROZ EDUCAÇÃO. *Qualidade de software*. 2023.