EXPLICAR O PROCESSO UNIFICADO PARA O DESENVOLVIMENTO DE SOFTWARE

UNIFIED MODELING LANGUAGE (UML)

A compreensão do Processo Unificado exige que tenhamos o entendimento da Unified Modeling Language, comumente denominada de UML, que é uma linguagem de modelagem padrão para a elaboração de artefatos no contexto do paradigma de desenvolvimento orientado a objetos.

Saiba mais

Até meados da década de 1990, o paradigma de desenvolvimento estruturado era o padrão mais utilizado na indústria de software, ou seja, os métodos vigentes aplicavam os conceitos estruturados nos artefatos gerados, tal como o diagrama de fluxo de dados na etapa de análise. Importante destacar que a unidade básica do software era a função.

Considerando a evolução da complexidade do software, alguns metodologistas perceberam que os conceitos da orientação a objetos permitiam um melhor trato da complexidade, particularmente nos requisitos reuso e manutenibilidade. Esses metodologistas começaram a propor métodos com artefatos direcionados ao referido paradigma. A unidade básica do software passou a ser o objeto, que inclui dados ou atributos, e funções ou métodos, encapsulados.

Em algum momento, percebeu-se a necessidade de um padrão para a modelagem de sistemas que fosse aceito e utilizado amplamente, surgindo a UML em 1996 como a linguagem “unificadora” das diversas propostas de métodos orientados a objetos. Em 1997, a UML foi aprovada pelo Object Management Group (OMG), tornando-se um padrão de modelagem orientada a objeto na indústria do software.

A UML é uma linguagem visual, independente de linguagem de programação e de processo de desenvolvimento. Um processo de desenvolvimento que utilize a UML como linguagem de modelagem envolve a criação de artefatos de software gráficos, denominados de diagramas, que podem ser complementados por descrições textuais.

Por que é definida como linguagem?

Uma linguagem serve para a comunicação, possuindo uma sintaxe e uma semântica associada. A sintaxe determina o padrão de escrita, tal como, a estrutura de uma frase formada por “sujeito + verbo + predicado” e a semântica está associada ao significado da frase, tal como, “Carlos foi ao clube”. Fazendo

uma analogia com a UML, a Figura 6 ilustra a estrutura de uma classe, cujo padrão inclui o nome da classe, atributos e métodos, ou seja, esse padrão de desenho, ou sintaxe do modelo, é definido pela UML.

Nome da Classe
+ atributoPublico: tipo # atributoProtegido: tipo - atributoPrivado: tipo
+ operacao(argumento): tipoRetorno

E a semântica?

A partir do momento que construímos um diagrama com um conjunto de classes, assim como ilustrado na Figura 7, temos então a formação de uma rede semântica, na qual os objetos e associações representam uma abstração da estrutura estática de certo domínio do problema.

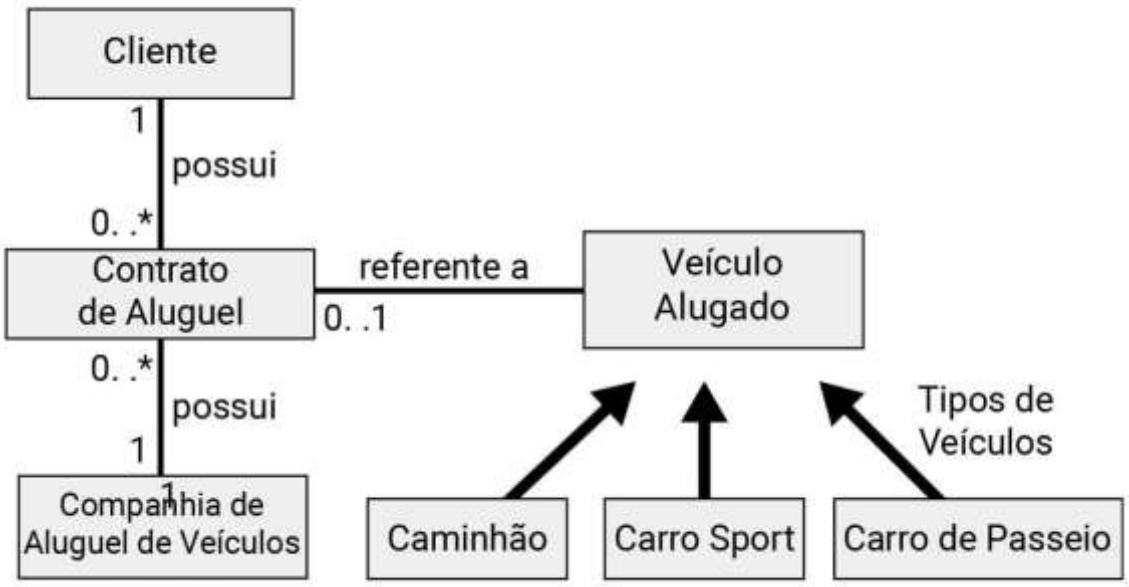

Figura 7 – Exemplo de Diagrama de Classes

E a comunicação?

Como já afirmado, uma linguagem serve para comunicação. No caso do desenvolvimento de software, um modelo permite a comunicação entre engenheiros de software, entre estes e os usuários para validação dos referidos modelos, entre estes e os gerentes de projeto ou programadores; enfim, caso você, como engenheiro de software, desenhe um diagrama de classes, a semântica será compreendida por qualquer outro engenheiro de software que faça uso do paradigma orientado a objetos.

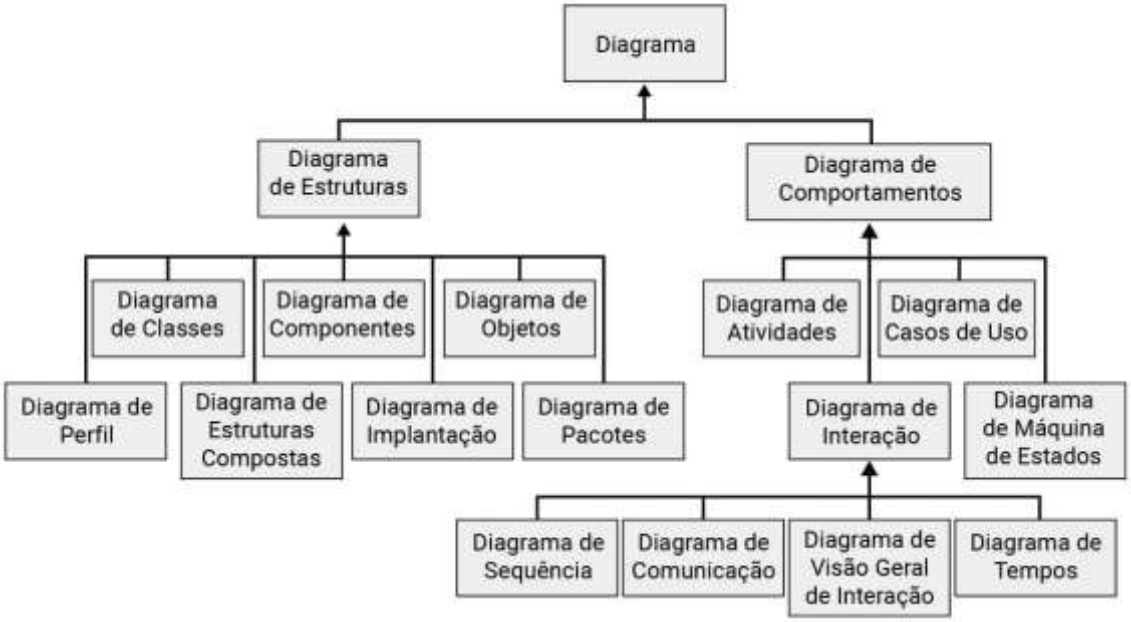

Figura 8 – Diagramas da UML

A Figura 8 ilustra os diagramas especificados na UML, com destaque para as duas categorias: diagramas comportamentais e diagramas estruturais. Os diagramas comportamentais têm ênfase na abstração funcional do projeto de software e os diagramas estruturais, nas abstrações das estruturas de dados e de componentes.

PROCESSO UNIFICADO

O Processo Unificado, também denominado de Rational Unified Process (RUP), é um processo de desenvolvimento de software criado pela Rational Software Corporation, empresa fundada pelos criadores da UML, posteriormente adquirida pela IBM.

O RUP é considerado um processo formal que permite ser utilizado na solução de problemas com alta complexidade, entretanto, pode ser customizado para projetos de qualquer escala.

O RUP possui três princípios:

GUIADO POR CASOS DE USO
O planejamento do desenvolvimento é feito em função dos casos de uso, de modo que o processo é iniciado pela abstração funcional, ou seja, pelos serviços desejados pelos usuários.

CENTRADO EM ARQUITETURA

O RUP enfatiza o papel da arquitetura permitindo ao arquiteto de software manter o foco nas metas corretas, tais como, compreensibilidade, confiança em mudanças futuras e reutilização.

A arquitetura deve permitir a realização dos requisitos, abrangendo um conjunto de decisões estruturais e comportamentais, alinhado às duas categorias de diagramas ilustradas na Figura 8, com destaque para a modularização da solução.

ITERATIVO E INCREMENTAL

O RUP aplica o modelo de processo de software iterativo e incremental. De forma evolucionária, novos requisitos são incorporados ocorrendo o versionamento das entregas de software ao usuário final, fundamental ao desenvolvimento de software moderno. Não necessariamente deve ocorrer uma entrega ao final de uma iteração.

A Figura 9 ilustra o aspecto bidimensional do RUP. Na horizontal, temos as fases que permitem capturar o aspecto temporal do projeto, que podem ser utilizadas como marcos durante esse projeto, e na vertical, as atividades iterativas. Vejamos as descrições dessas fases.

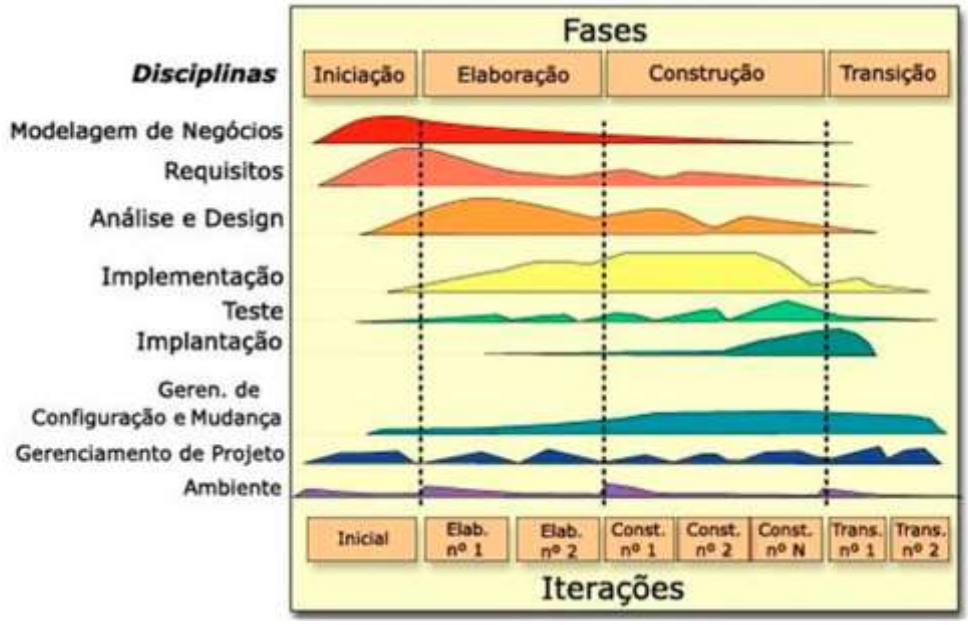

Figura 9 – O Processo Unificado

O Processo Unificado (Rational Unified Process)

Veja mais detalhes sobre as etapas do modelo unificado a seguir:

INICIAÇÃO

A fase de iniciação, também denominada de concepção, exige do engenheiro de software uma visão inicial e geral do sistema a ser desenvolvido, cabendo destacar a possibilidade da utilização dos modelos de estado ou de atividades no desenho dos processos de negócio, isto é, a aplicação da primeira atividade do fluxo de trabalho que é a modelagem de negócio. Tal modelagem permite aos engenheiros de software compreender o negócio e garantir que todas as partes interessadas tenham um entendimento comum do domínio do problema.
A partir dessa compreensão do negócio, pode-se obter os requisitos fundamentais do projeto, isto é, requisitos funcionais e não funcionais. Em seguida, realizar a modelagem dos casos de uso mais críticos, incluídos os principais serviços disponibilizados aos usuários, e a modelagem de classes, para entendimento da estrutura de dados a ser gerenciada pelo sistema.
A identificação dos principais requisitos, junto aos modelos de casos de uso e classes iniciais, permite o estabelecimento do escopo do projeto. Nessa fase, a arquitetura do sistema limita-se a um esquema provisório com os principais subsistemas e as funções e recursos que o compõem.
Temos agora dois grandes desafios aplicáveis a muitos projetos de software: verificar a viabilidade técnica e financeira do projeto, bem como garantir o seu financiamento, visto que ambos os desafios incluem também a análise de riscos.
Podemos avançar com o projeto? Caso positivo, vamos realizar o planejamento detalhado da fase de Elaboração e geral das demais fases. Caso negativo, encerramos nossas atividades.

ELABORAÇÃO

Com o projeto de software aprovado, iniciamos a fase de elaboração, que é realizada de forma iterativa e incremental, de acordo com a Figura 9. Vamos reduzir o nível de abstração dos modelos de casos de uso e de classes iniciais, finalizando a atividade de análise do processo de software.
Na sequência, temos a atividade de projeto na qual ocorre o refinamento de alguns modelos da análise e a criação de novos, assim como o modelo de interação, que representa os aspectos dinâmicos do sistema, além de construir um protótipo da interface com o usuário. A representação arquitetural é expandida em cinco visões do software: modelo de casos de uso, modelo de análise, modelo de projeto, modelo de implementação e modelo de implantação. Cada visão inclui um conjunto de modelos com seus respectivos diagramas. Nessa fase, devemos, também, minimizar os riscos e realizar o planejamento da fase de construção.

CONSTRUÇÃO

A fase de construção segue o modelo de arquitetura gerando os componentes, ou aplica-se o reuso, que tornam os casos de uso operacionais para os usuários.

Os modelos gerados na fase de elaboração são concluídos para refletir a versão final do incremento de software. Com o avanço da codificação, são realizados os testes unitários, os testes de integração entre os componentes e os testes de validação, até o teste beta, em que os usuários realizam a validação em ambiente de desenvolvimento controlado.

TRANSIÇÃO
A fase de transição tem como objetivo realizar a implantação do software no ambiente de produção. Inicialmente, o software é disponibilizado aos usuários para realização do teste beta em ambiente não controlado pelo desenvolvimento, tendo como foco a correção de defeitos e mudanças necessárias em função dos feedbacks dos usuários.
Nessa fase, podem ocorrer a elaboração dos manuais do usuário, treinamento dos usuários, procedimentos de instalação, montagem de estrutura de suporte ou definição do processo de manufatura. Na conclusão dessa fase, o incremento torna-se uma versão de produção utilizável pela comunidade de usuários.

FLUXO DE TRABALHO

Após o detalhamento das fases do RUP ilustradas na Figura 9, vamos entender as atividades transversais. Essas atividades, que correspondem às atividades genéricas de um processo de desenvolvimento de software, apresentadas anteriormente, compõem o denominado fluxo de trabalho, sendo as atividades distribuídas em todas as fases.

Como ocorre o relacionamento entre as dimensões fases do projeto X fluxos de projeto da Figura 9?

Cada atividade do fluxo de trabalho se desenvolve de forma mais intensa em determinada fase. Por exemplo, os fluxos de trabalho da modelagem de negócio e requisitos são intensos na fase de concepção, justamente quando são definidos os requisitos de alto nível do negócio; a análise e *design*, i.e., projeto, tem uma intensidade na concepção em função dos modelos de análise e projeto, entretanto podemos observar alguma atividade na fase de construção, pois alguns modelos são refinados nesta fase do software.

Por que ocorre a atividade de implementação na fase de elaboração?

Essa fase não é intensa em análise e projeto?

Lembra-se da prototipação? Podemos nessa fase validar algum requisito do software aplicando a prototipação que inclui atividade de codificação, i.e., implementação.

O RUP define três grupos de atividades de apoio ao desenvolvimento do software:

Ambiente

Inclui as atividades necessárias para configurar os processos e as ferramentas que darão suporte à equipe de desenvolvimento.

Gerenciamento de configuração e mudança

Permite a estruturação sistemática dos artefatos, tais como documentos e modelos, que precisam estar sob controle de versão, devendo essas alterações serem visíveis.

Gerência de Projeto

Enfatiza a importância do gerenciamento de projetos; nessa atividade, sugerimos a utilização das melhores práticas contidas no **PMBOK**.

RESUMINDO

Neste módulo, pudemos avaliar a importância do Processo Unificado no contexto do desenvolvimento de software. Inicialmente, apresentamos a UML, linguagem de modelagem unificada, a qual define o padrão dos artefatos de software de acordo com o paradigma de desenvolvimento orientado a objetos a serem utilizados em dado modelo de processo de desenvolvimento de software.

Prioritariamente, o RUP é um modelo de processo de software para o desenvolvimento de acordo com o paradigma orientado a objetos. Esse processo tem um alto grau de formalismo, possuindo os seguintes princípios: baseado em casos de uso, centrado em arquitetura e iterativo e incremental. Enfim, se o problema for complexo, o RUP é um forte candidato a ser escolhido por algum engenheiro de software como requisito não funcional “modelo de processo de software”.

Atenção

Vale destacar que o RUP pode ser ajustado a qualquer escala de complexidade e a seleção do requisito não será tão simplista.