Perfil:

Emerson Antonio Klisiewicz.

Possui graduação em Ciências da Computação, pela Pontifícia Universidade Católica do Paraná (1994), Especialização em Sistemas de Informações Gerenciais, pela Pontifícia Universidade Católica do Paraná (1995), Especialização em Redes e Sistemas Distribuídos pela Pontifícia Universidade Católica do Paraná (1997), com experiência de mais de 20 anos em áreas de tecnologia de empresas estatais e financeiras. Currículo Lattes:

http://buscatextual.cnpq.br/buscatextual/visualizacv.do?id=K4282839T3

AULA 06 – Diagrama UML

Introdução:

Nessa sexta aula, estaremos estudando sobre os diagramas que a UML nos apresenta e como podemos utiliza-los de forma a melhorar a documentação e qualidade dos softwares e sistemas que estaremos desenvolvendo. A Unified Modelling Language (UML) é uma linguagem ou notação de diagramas para especificar, visualizar e documentar modelos de 'software' orientados por objetos.

Contextualizando:

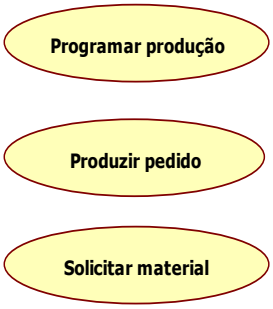
Diagrama de Casos de Uso

Descreve o que o sistema faz do ponto de vista do observador externo. Ajuda a esclarecer os requisitos do sistema e a dividir o desenvolvimento do sistema em tarefas.

Componentes

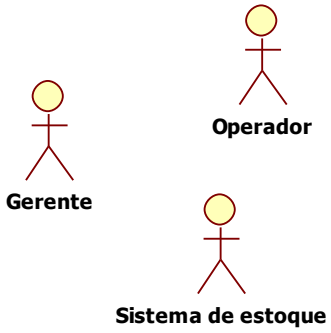
a-) *Caso de uso*

Representa as diferentes funcionalidades que o sistema disponibiliza aos usuários.



b-) *Atores*

Diferentes usuários que operam o sistema. Sistemas externos que interagem com o sistema.



c-) *Associação*

Representa a comunicação entre o ator e o caso de uso. Também existem associações entre casos de usos.

d-) *Relacionamento entre Casos de Uso*

Include, Extend, Generalization.

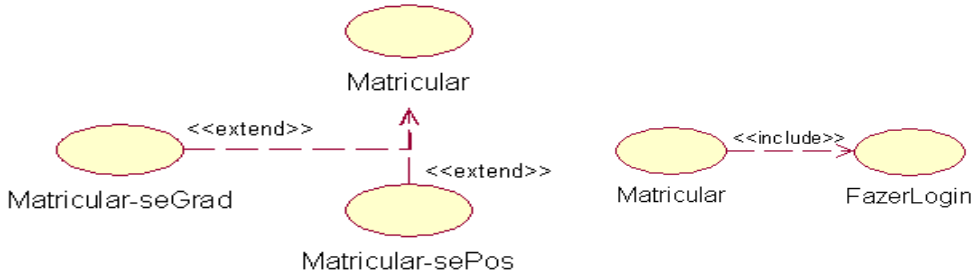
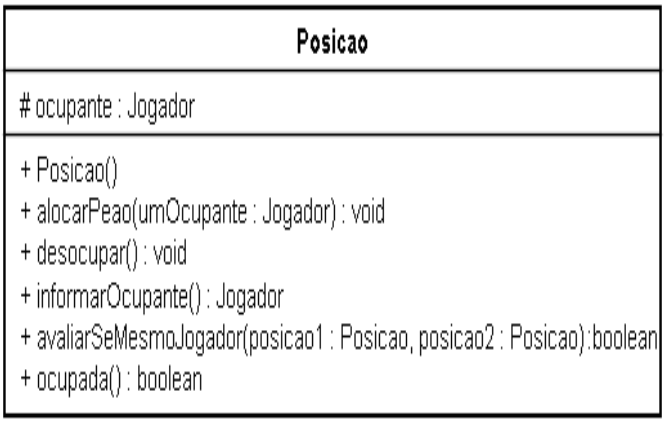


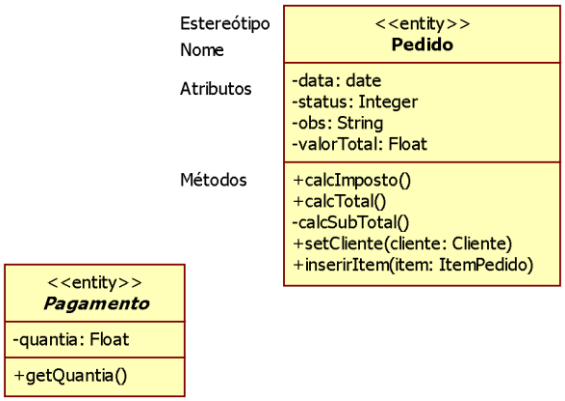
Diagrama de Classe

Diagramas de Classe mostram as diferentes classes que fazem um sistema e como elas se relacionam. Os Diagramas de Classe são chamados diagramas "estáticos" porque mostram as classes, com seus métodos e atributos bem como os relacionamentos estáticos entre elas: quais classes "conhecem" ou quais classes "são parte" de outras classes, mas não mostram a troca de mensagens entre elas.

Componentes

a-) *Classe*





b-) *Relacionamentos*

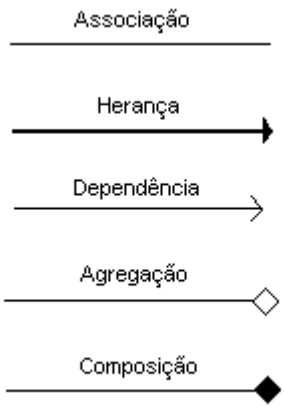
b1-) **Associação** : São relacionamentos estruturais entre instâncias e especificam que objetos de uma classe estão ligados a objetos de outras classes.

b2-) **Generalização** (herança: simples ou composta) - Relacionamento entre um elemento mais geral e um mais específico. Onde o elemento mais específico herda as propriedades e métodos do elemento mais geral.

b3-) **Dependência** - São relacionamentos de utilização no qual uma mudança na especificação de um elemento pode alterar a especificação do elemento dependente. A dependência entre classes indica que os objetos de uma classe usam serviços dos objetos de outra classe.

b4-) **Agregação Regular** - tipo de associação (é parte de, todo/parte) onde o objeto parte é um atributo do todo; onde os objetos partes somente são criados se o todo ao qual estão agregados seja criado. Pedidos é composto por itens de pedidos.

b5-) **Composição** - Relacionamento entre um elemento (o todo) e outros elementos (as partes) onde as partes só podem pertencer ao todo e são criadas e destruídas com ele.



Relacionamentos

c-) *Atributos*

Na UML, atributos são mostrados com pelo menos seu nome, e podem também mostrar seu tipo, valor inicial e outras propriedades. Atributos podem também ser exibidos com sua visibilidade:

+ indica atributos *públicos*
# indica atributos *protegidos*
- indica atributos *privados*

d-) *Operações*

Operações (métodos) também são exibidos com pelo menos seu nome, e podem também mostrar seus parâmetros e valores de retorno. Operações podem, como os Atributos, mostras sua visibilidade:

+ indica operações *públicas*
# indica operações *protegidas*
- indica operações *privadas*

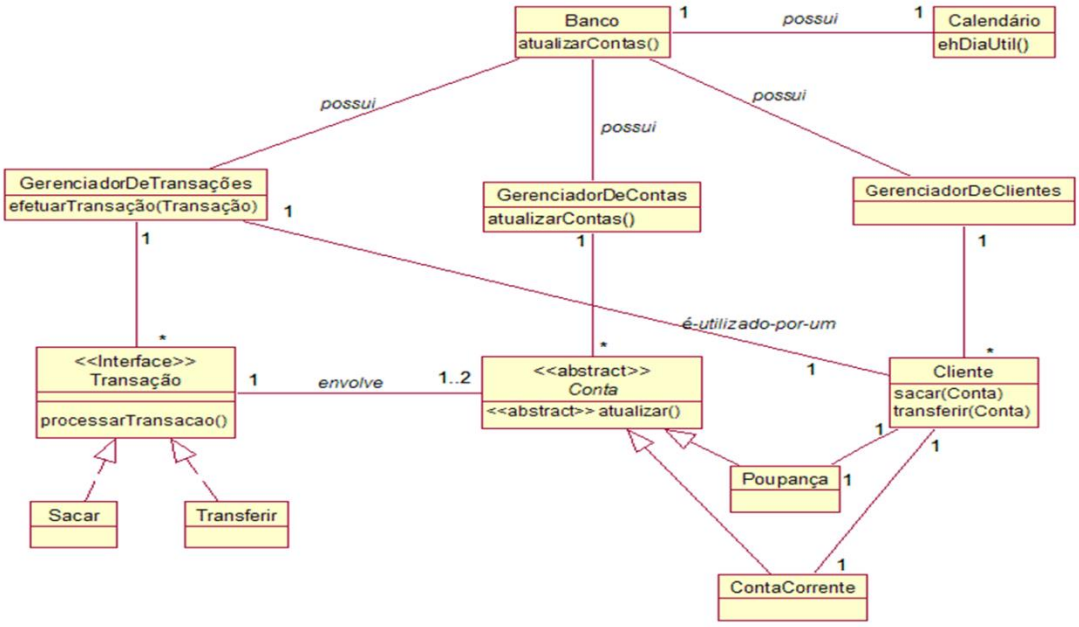
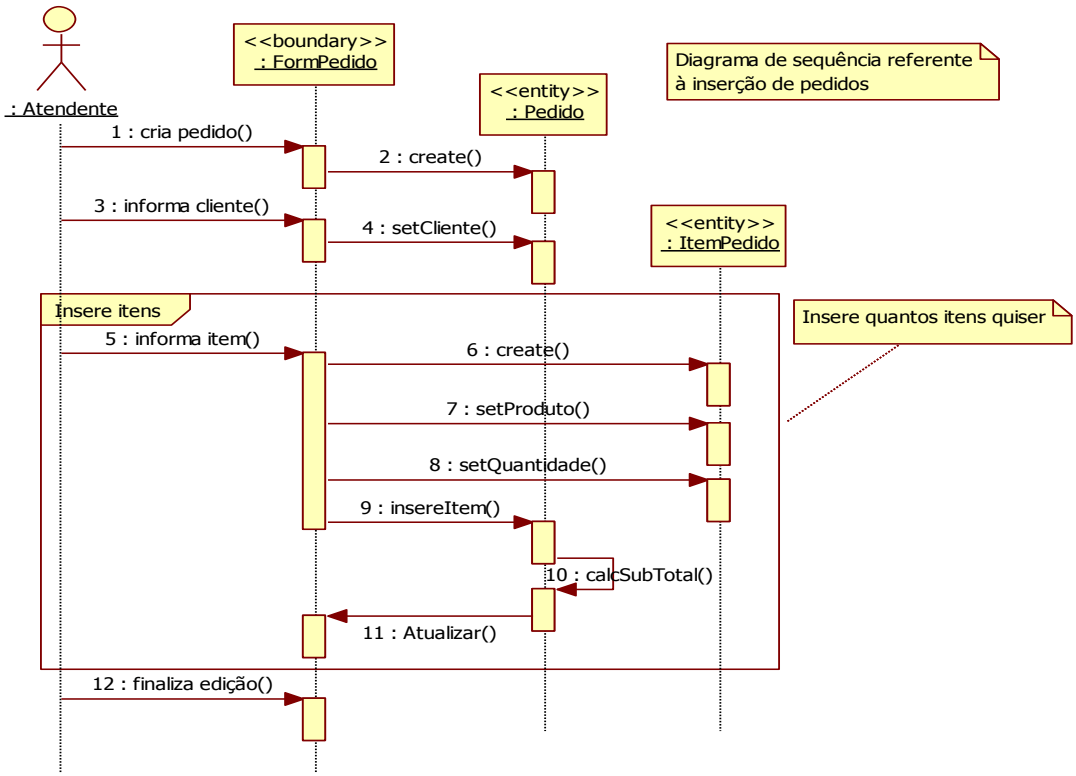


Diagrama de Sequencia

Descreve as interações entre as classes através das trocas de mensagens ao logo do tempo. Mostra um conjunto de objetos, seus relacionamentos e as mensagens que podem ser enviadas entre eles. Diagrama de sequência dá ênfase à sequência de mensagens.



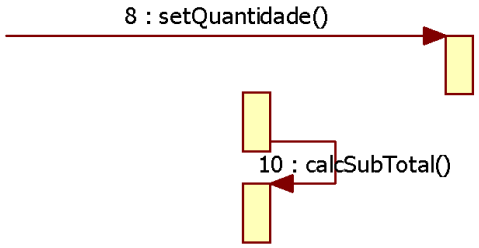
Componentes

a-) *Objetos*

Representa uma instância de uma determinada classe.

b-) *Mensagens*

Representa troca de mensagens entre os objetos.

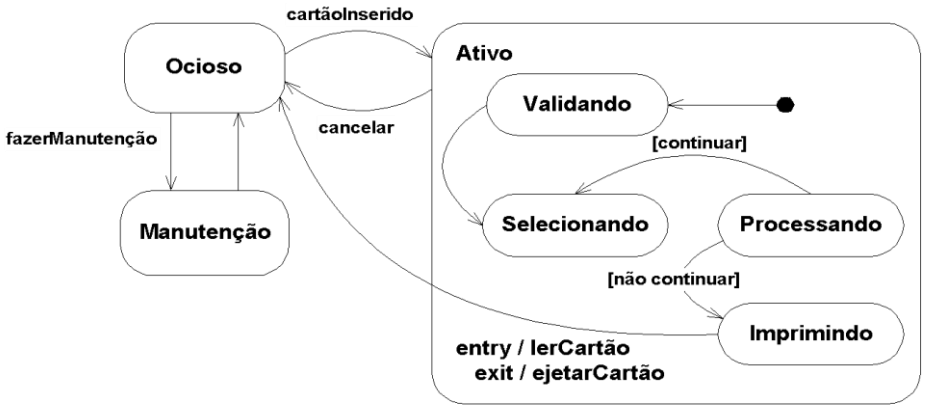


c-) Tipos de mensagens

Quando um objeto envia uma mensagem para outro objeto, o objeto que recebe a mensagem pode enviar outras mensagens e assim por diante, formando uma **sequência** de mensagens. O sequenciamento pode ser procedural, com aninhamento (mensagens **síncronas**) ou plano, sem aninhamento (mensagens **assíncronas**).

Diagrama de Estados

Mostra uma máquina contendo estados, transições, eventos e atividades. Estes diagramas são usados para modelar o comportamento de objetos (com comportamento complexo). Nestes diagramas são modelados os estados em que um objeto pode estar e os eventos que fazem o objeto passar de um estado para outro.



Componentes

- Estado inicial.
- Estado final.
- Estado intermediário.

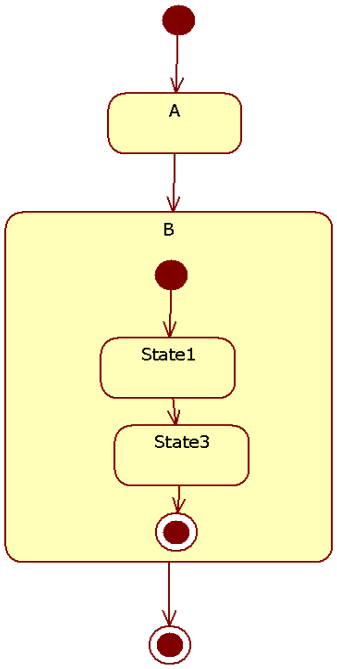
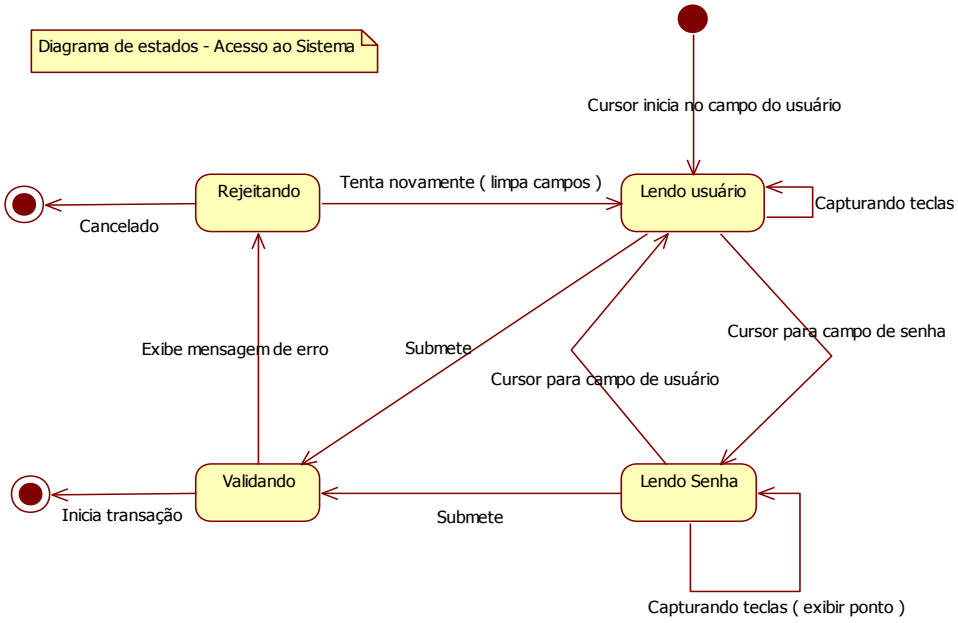


Diagrama completo:



Síntese

O surgimento de sistemas de software complexos resultou na necessidade de reavaliar a forma de desenvolver sistemas. As técnicas têm evoluído de forma impressionante, notavelmente no que tange à modelagem de sistemas.

Um modelo pode ser visto como uma representação idealizada de um sistema a ser construído. Maquetes de edifícios e de aviões e plantas de circuitos eletrônicos

são apenas alguns exemplos de modelos. Uma simplificação da realidade que nos ajuda a entender um problema complexo.

Então a modelagem de sistemas de software consiste na utilização de notações gráficas e textuais com o objetivo de construir modelos que representam as partes essenciais de um sistema, considerando-se diversas perspectivas diferentes e complementares. E a UML nos fornece vários diagramas entre os quais os vistos nessa aula onde podemos de forma segura, com qualidade e padronizada documentar e modelar os sistemas e softwares que iremos desenvolver.