INF016 – Arquitetura de Software

06 - Modelagem

Sandro Santos Andrade
sandroandrade@ifba.edu.br

Instituto Federal de Educação, Ciência e Tecnologia da Bahia
Departamento de Tecnologia Eletro-Eletrônica
Graduação Tecnológica em Análise e Desenvolvimento de Sistemas

Introdução

- Todo *software* possui uma arquitetura, seja ela boa ou não
- Uma vez determinadas as decisões de projeto elas precisam ser registradas
- Decisões de projeto são capturadas em **modelos** e o processo de criação destes modelos é chamado **modelagem**

Modelo Arquitetural: artefato que captura algumas (ou todas as) decisões de projeto que compõem a arquitetura de um sistema

Modelagem Arquitetural: reificação e documentação, através de um modelo arquitetural, das decisões de projeto que compõem a arquitetura de um sistema

Introdução

- Modelos podem capturar as decisões arquiteturais em diferentes níveis de rigor e formalidade
- Modelos possibilitam a discussão, visualização, avaliação e evolução de um arquitetura
- Será discutido como as **notações de modelagem de arquiteturas** são utilizadas para capturar as decisões de projeto

Notação de Modelagem Arquitetural: linguagem ou meios para captura das decisões de projeto

Introdução

- As notações variam desde as mais ricas e ambíguas (como linguagem natural) até aquelas semanticamente restritas e altamente formais (como a Linguagem de Descrição de Arquiteturas RAPIDE)
- Pode-se utilizar uma única notação ou um conjunto formado por diferentes notações (ex: diagramas *UML* em conjunto com descrições em linguagem natural)
- Diversos conceitos de modelagem podem ser capturados, desde elementos arquiteturais básicos (ex: componentes e conectores) até propriedades mais complexas (ex: modelos comportamentais)

Conceitos de Modelagem

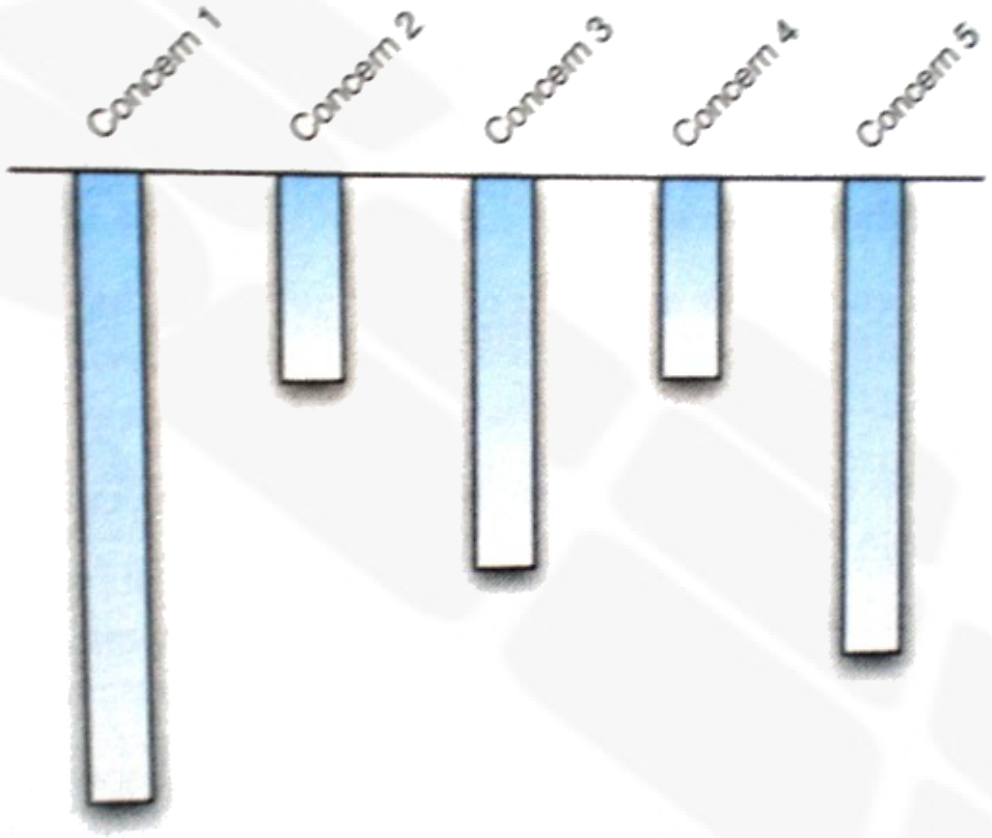
- Quais tipos de “coisas” (conceitos) podem ser modeladas em uma arquitetura ?
- O que é necessário para efetivamente modelar determinado conceito (características notacionais, etc) ?
- Principais conceitos:

1) Modelagem Dirigida a *Stakeholder*
2) Conceitos Arquiteturais Básicos
3) Elementos de um Estilo Arquitetural
4) Aspectos Estáticos e Dinâmicos
5) Aspectos Funcionais e Não-Funcionais

Conceitos de Modelagem
Modelagem Dirigida a *Stakeholder*

- Dentre as principais questões que arquitetos e outros *stakeholders* se deparam, ao modelar arquiteturas, destacam-se:
  1) Quais conceitos e decisões arquiteturais devem ser modelados ?
  2) Em qual nível de detalhe devem ser modelados ?
  3) Qual o nível de rigor ou formalidade necessário ?
- As respostas envolvem uma análise de custo (de criação e manutenção dos modelos) x benefício (de ter-se certos modelos em certas formas ou notações)

Conceitos de Modelagem
Modelagem Dirigida a *Stakeholder*



- Os aspectos mais importantes ou críticos deverão ser aqueles modelados com maiores detalhes e utilizando graus maiores de rigor/formalidade

Conceitos de Modelagem
Modelagem Dirigida a *Stakeholder*

- A modelagem é uma atividade e, como tal, é frequentemente governada por um processo
- Provavelmente será parte de um processo de desenvolvimento de *software* centrado em arquiteturas, de espectro mais amplo
- A própria modelagem, entretanto, pode ser considerada como um sub-processo, que pode variar largamente de projeto para projeto

Conceitos de Modelagem
Modelagem Dirigida a *Stakeholder*

- As atividades básicas de uma modelagem dirigida a *stakeholder* são:

1) Identificação dos aspectos relevantes do *software* a serem modelados
2) Categorização aproximada destes aspectos em relação à sua importância
3) Identificação das metas de modelagem para cada aspecto (comunicação, descoberta de *bugs*, análise de qualidade, geração de outros artefatos, etc)
4) Seleção das notações de modelagem que irão representar cada aspecto em um nível de profundidade que conduza ao atendimento das metas de modelagem
5) Criação dos modelos
6) Uso dos modelos de uma maneira consistente com as metas de modelagem

Conceitos de Modelagem
Modelagem Dirigida a *Stakeholder*

- Tais passos podem ser executados de forma iterativa:
  - No início do projeto provavelmente não se conhece todos os aspectos importantes do sistema, nem todas as metas de modelagem
  - Também não se sabe se tais metas de modelagem podem ser alcançadas utilizando as notações, tecnologias, tempo e dinheiro disponíveis

Conceitos de Modelagem
Conceitos Arquiteturais Básicos

- Elementos centrais de um projeto arquitetural:
  - Componentes
  - Conectores
  - Interfaces
  - Configurações
  - *Rationale*

Conceitos de Modelagem
Conceitos Arquiteturais Básicos

- Elementos centrais de um projeto arquitetural:
  - Componentes:
    - *Building blocks* que encapsulam um sub-conjunto das funcionalidades e dados do sistema e restringem acesso a eles através de interfaces explicitamente definidas
  - Conectores:
    - *Building blocks* que efetivam e regulam as interações entre componentes
  - Interfaces:
    - Pontos de interação de componentes e conectores com o mundo externo (geralmente, outros componentes e conectores)

Conceitos de Modelagem
Conceitos Arquiteturais Básicos

- Elementos centrais de um projeto arquitetural:
  - Configurações:
    - Conjunto de associações específicas entre os componentes e conectores da arquitetura. Tais associações podem ser capturadas por grafos cujos nós representam componentes e conectores e arestas representam suas inter-conexões
  - *Rationale*:
    - Informação que explica porque decisões arquiteturais particulares foram tomadas e qual o propósito de cada elemento arquitetural

Conceitos de Modelagem
Conceitos Arquiteturais Básicos

- Esses conceitos básicos fornecem um ponto de partida para a modelagem arquitetural
- Requerem uma notação capaz de representar grafos de componentes e conectores, preferencialmente com pontos de conexão (interfaces) bem definidos
    - Ex: diagramas simples de retângulos e setas
- Já o *rationale* é mais amorfo, é primariamente utilizado para comunicar informação e intenção entre os *stakeholders*
- O *rationale* geralmente não está descrito explicitamente no sistema construído e sua descrição requer uma notação mais expressiva e menos restrita

Conceitos de Modelagem
Conceitos Arquiteturais Básicos

- Enumerar somente a existência e inter-conexões entre componentes e conectores pode, entretanto, não ser suficiente:
  - Como as funções são particionadas entre os componentes ?
  - Qual a natureza e tipo das interfaces ?
  - Qual o significado, para um componente e um conector, da existência de uma ligação entre eles ?
  - Como todas essas propriedades do sistema evoluem com o decorrer do tempo ?

Conceitos de Modelagem
Conceitos Arquiteturais Básicos

- Enumerar somente a existência e inter-conexões entre componentes e conectores pode, entretanto, não ser suficiente:

  Como as funções são particionadas entre os componentes ?

  Qual a natureza e tipo das interfaces ?

  Qual o significado, para um componente e um conector, da existência de uma ligação entre eles ?

  Como todas essas propriedades do sistema evoluem com o decorrer do tempo ?

  Foco central da modelagem arquitetural

Conceitos de Modelagem
Conceitos Arquiteturais Básicos

- A depender da natureza do sistema sendo desenvolvido a representação destes elementos básicos pode ser mais ou menos difícil
- Aplicações grandes, dinâmicas e distribuídas podem ser bastante complicadas de modelar:
  - Ex: *World Wide Web* – quantidade imensa de componentes e conectores, que surgem e desaparecem a todo momento
  - Possíveis soluções:
    - Modelar somente parte do sistema
    - Modelar configurações específicas que representam *use-cases* esperados
    - Modelar o estilo arquitetural que governa quais elementos podem ser inseridos na arquitetura e como eles podem ser configurados

Conceitos de Modelagem
Elementos de um Estilo Arquitetural

- Relembrando:
  - Um estilo arquitetural é uma coleção de decisões de projeto arquitetural aplicáveis em um determinado contexto de desenvolvimento
  - Os estilos restringem as decisões de projeto àquelas que são específicas de um sistema particular dentro do contexto em questão
  - Os estilos induzem qualidades benefícas no sistema resultante
- Adicionalmente à modelagem dos elementos arquiteturais básicos frequentemente é útil modelar o estilo que governa como estes elementos são usados

Conceitos de Modelagem
Elementos de um Estilo Arquitetural

- Estilos arquiteturais, assim como arquiteturas, são formados por decisões de projeto e, portanto, também podem ser modeladas
- Vantagens da modelagem explícita do estilo arquitetural sendo utilizado:
  - Reduz confusão sobre o que é e não é permitido na arquitetura
  - Ajuda a reduzir o desvio e erosão arquiteturais
  - Torna mais fácil distinguir se uma decisão específica de projeto foi tomada para se adequar a uma restrição estilística ou por alguma outra razão
  - Ajuda a guiar a evolução da arquitetura
  - Pode ser mais viável e útil do que a modelagem dos elementos básicos (ex: sistemas grandes e dinâmicos)
  - Podem ser reutilizados em diversos projetos
  - Representam local único para captura dos *cross-cutting concerns* e *rationale* de uma arquitetura

Conceitos de Modelagem
Elementos de um Estilo Arquitetural

- Os tipos de decisões de um estilo arquitetural são geralmente mais abstratos ou genéricos do que os de uma arquitetura:
  - Elementos Específicos
  - Tipos dos Componentes, Conectores e Interfaces
  - Restrições de Interação
  - Restrições Comportamentais
  - Restrições de Concorrência

Conceitos de Modelagem
Elementos de um Estilo Arquitetural

- Os tipos de decisões de um estilo arquitetural são geralmente mais abstratos ou genéricos do que os de uma arquitetura:
  - Elementos Específicos:
    - O estilo pode prescrever que componentes, conectores e interfaces particulares sejam incluidos na arquitetura ou utilizados em situações específicas. Isto pode ser facilitado por uma abordagem de modelagem que suporte *templates* ou modelos-base que podem ser então elaborados
  - Tipos dos Componentes, Conectores e Interfaces:
    - Tipos específicos de elementos podem ser permitidos, requeridos ou proibidos na arquitetura. Muitas abordagens de modelagem são acompanhadas de um sistema de tipos

Conceitos de Modelagem
Elementos de um Estilo Arquitetural

- Os tipos de decisões de um estilo arquitetural são geralmente mais abstratos ou genéricos do que os de uma arquitetura:
    - Restrições de Interação:
        - Podem ser temporais: “os componentes que iniciam a comunicação devem chamar o método *init()* antes de qualquer outro”
        - Podem ser topológicas: “Somente componentes da camada *client* podem invocar componentes da camada *server*”
        - Podem especificar protocolo de interação particulares, ou por nome (ex: FTP, HTTP) ou por especificação (ex: CSP e diagramas de sequência)
        - Notações que suportam a definição dessas restrições geralmente utilizam algum tipo de lógica (de primeira ordem, temporal, etc)

Conceitos de Modelagem
Elementos de um Estilo Arquitetural

- Os tipos de decisões de um estilo arquitetural são geralmente mais abstratos ou genéricos do que os de uma arquitetura:
  - Restrições Comportamentais:
    - Podem variar desde regras simples até especificações comportamentais completas expressas, por exemplo, em autômatos finitos
    - Notações que suportam a definição de restrições comportamentais também utilizam algum tipo de lógica ou outros modelos formais (como autômatos)
  - Restrições de Concorrência:
    - Indicam quais elementos realizam sua função de forma concorrente e como eles sincronizam o acesso a recursos compartilhados
    - Notações que suportam a definição de tais restrições utilizam modelos comportamentais formais e podem também incluir técnicas para modelagem temporal, tais como diagramas de sequência e de transição de estados

Conceitos de Modelagem
Aspectos Estáticos e Dinâmicos

- Aspectos estáticos de um sistema são aqueles que não envolvem o comportamento do sistema durante sua execução (mais fáceis de modelar)
  - Ex: topologias de componentes e conectores, atribuição de componentes e conectores a *hosts*, configurações de rede e dos *hosts* e mapeamento de elementos arquiteturais em artefatos de código ou binários
- Aspectos dinâmicos envolvem o comportamento em *run-time* do sistema (mais difíceis de modelar)
  - Ex: modelos comportamentais, modelos de interação e modelos de fluxo de dados

Conceitos de Modelagem
Aspectos Estáticos e Dinâmicos

- A distinção entre estático x dinâmico nem sempre é clara, entretanto:
  - Ex: modificações dinâmicas na estrutura do sistema devido à ocorrência de falhas, ao uso de conectores flexíveis ou dinamismo arquitetural
  - Neste caso o modelo captura tanto aspectos estáticos quanto dinâmicos. Ex: uma topologia base acompanhada de um conjunto limitado de modificações passíveis de ocorrer em *run-time*
- Há uma diferença entre:
  - Modelar **aspectos** estáticos ou dinâmicos de um sistema
  - Utilizar **modelos** estáticos ou dinâmicos

Conceitos de Modelagem
Aspectos Estáticos e Dinâmicos

- Um modelo de um **aspecto** dinâmico descreve como o sistema muda à medida em que executa
- Um **modelo** dinâmico é aquele que possui a capacidade de se auto-modificar
- Não é necessário utilizar um modelo dinâmico para capturar um aspecto dinâmico do sistema (ex: *statechart*)
- Se este *statechart*, entretanto, fosse conectado a um sistema em execução de modo que o estado atual fosse identificado teria-se então um modelo dinâmico

Conceitos de Modelagem
Aspectos Estáticos e Dinâmicos

- Modelos dinâmicos são mais difíceis de suportar pois:
  - Devem ser incorporados ao sistema em modo *read-write* – a execução do sistema pode modificar o modelo
  - Requer ferramenta de suporte para manter o modelo e o sistema sempre sincronizados e consistentes
  - Devem ser armazenados com uma notação *machine-readable* e *machine-writable*
  - Devem ser apropriadamente mapeados aos artefatos de implementação
  - Visualizações dos modelos podem ser adequadamente dinâmicas, refletindo mudanças se possível *on-the-fly*

Conceitos de Modelagem
Aspectos Funcionais e Não-Funcionais

- Aspectos funcionais estão relacionados a **o que** o sistema faz
  - Descritos de forma declarativa (sentenças sujeito-verbo). Ex: “o **sistema imprime** registros médicos”
- Aspectos não-funcionais estão relacionados a **como** o sistema executa suas funções
  - Descritos adicionando-se advérbios às sentenças anteriores. Ex: “o sistema imprime registros médicos **rapidamente e confidencialmente**”

Conceitos de Modelagem
Aspectos Funcionais e Não-Funcionais

- Aspectos funcionais são geralmente mais concretos e, portanto, mais fáceis de modelar e de serem descritos com maior formalidade e rigor:
    - Podem capturar os serviços disponibilizados por diferentes componentes e conectores e as inter-conexões que possibilitam as funções gerais do sistema
    - Podem capturar o comportamento de componentes, conectores ou sub-sistemas
    - Podem ser estáticos ou dinâmicos
- A maioria das notações capturam aspectos funcionais, diferindo em quais aspectos do sistema podem ser descritos e como
- A semântica subjacente à notação irá determinar os tipos de investigação e análise que poderão ser realizadas

Conceitos de Modelagem
Aspectos Funcionais e Não-Funcionais

- Aspectos não-funcionais tendem a ser qualitativos e subjetivos
- Podem ser mais informais e menos rigorosos que os funcionais, mas isso não significa que não devem ser capturados
- Frequentemente, os aspectos funcionais são desenvolvidos especificamente para alcançar os objetivos não-funcionais. Ex:
    - Um modelo não-funcional pode descrever simplesmente que o componente de processamento da folha de pagamento seja rápido
    - O modelo funcional informa que este componente utiliza *cache* e processamento local – duas estratégias funcionais que ajudam a atingir o aspecto não-funcional

Conceitos de Modelagem
Aspectos Funcionais e Não-Funcionais

- Assim como o *rationale* aspectos não-funcionais são difíceis de se modelar com rigor
- Geralmente utiliza-se notações expressivas e livres de forma como linguagem natural
- Entretanto, é útil adotar abordagem com suporte à rastreabilidade, permitindo que os arquitetos mapeiem propriedades não-funcionais em decisões funcionais

Conceitos de Modelagem
Ambiguidade, Exatidão e Precisão

- Arquiteturas são abstrações de sistemas: capturam seus aspectos e estados nominais mais importantes enquanto descartam outros menos usuais
- Uma arquitetura não tem o objetivo de ser uma implementação completa do sistema
- Dessa forma, as notações utilizadas para capturá-las não precisam ser totalmente não-ambíguas, exatas e precisas

Conceitos de Modelagem
Ambiguidade, Exatidão e Precisão

- Ambiguidade (*ambiguity*)

> Um modelo é **ambíguo** (*ambiguous*) se ele é passível de mais de uma interpretação

- Interpretações conflitantes geram mal-entendidos, *bugs* e erros
- A principal causa da ambiguidade é a incompletude:
  - Aspectos que não foram especificados (não-principais) abrem caminho para diferentes suposições por diferentes *stakeholders*

Conceitos de Modelagem
Ambiguidade, Exatidão e Precisão

- Arquiteturas são necessariamente incompletas: refletem as decisões **principais**, não **toda** decisão
- Por isso, geralmente é impossível eliminar completamente a ambiguidade. É necessário chegar em um acordo:
  - Permitir que certos aspectos do sistema sejam ambíguos com o consentimento dos *stakeholders*
  - Avalia-se a arquitetura, identificando e documentando os aspectos ambíguos, até que ela esteja boa o suficiente
  - Decisões restantes podem ser revistas em uma atividade futura do desenvolvimento

Conceitos de Modelagem
Ambiguidade, Exatidão e Precisão

- Exatidão (*accuracy*) e Precisão (*precision*)

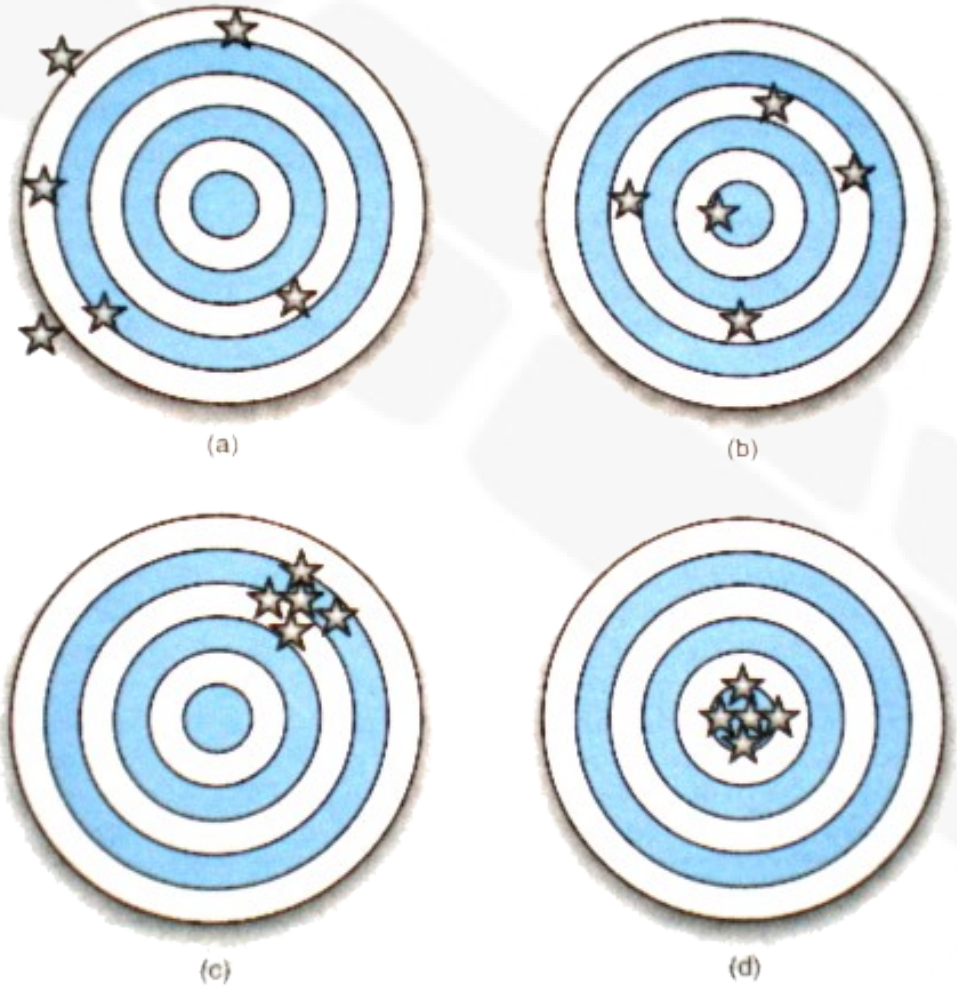
Um modelo é **exato** (*accurate*) se ele é correto, está em conformidade com os fatos ou se desvia de sua corretude dentro de limites aceitáveis

Um modelo é **preciso** (*precise*) se ele é específico e detalhado

  - Um modelo é exato se ele contém informações corretas sobre o sistema sendo modelado
  - Um modelo é preciso se ele contém informações suficientemente detalhadas sobre o sistema

Conceitos de Modelagem
Ambiguidade, Exatidão e Precisão

- Exatidão (*accuracy*) e Precisão (*precision*)



Conceitos de Modelagem
Ambiguidade, Exatidão e Precisão

- Geralmente deve-se priorizar a exatidão à precisão
- Precisão e completude são geralmente melhoradas nas fases posteriores de projeto detalhado e implementação
- Notações e abordagens de modelagem têm impacto direto na ambiguidade, exatidão e precisão:
  - Algumas são propositadamente ambíguas
  - Outras são baseadas em alguma semântica formal, com interpretações geralmente disponibilizadas por ferramentas de visualização e análise

Modelagem Complexa
Múltiplos Conteúdos – Multiplas *views*

- Modelos arquiteturais são artefatos complexos: capturam todas as decisões que são importantes para uma variedade de *stakeholders*
- Diferentes aspectos do mesmo conceitos são capturados simultaneamente. Ex: as inter-conexões, comportamento e histórico de versões de um componente
- Lidar com todos os aspectos de uma vez é inviável: *stakeholders* desejam interagir com as partes mais importantes de acordo com a sua perspectiva

Modelagem Complexa
Múltiplos Conteúdos – Multiplas *views*

- *Views* e *Viewpoints*:

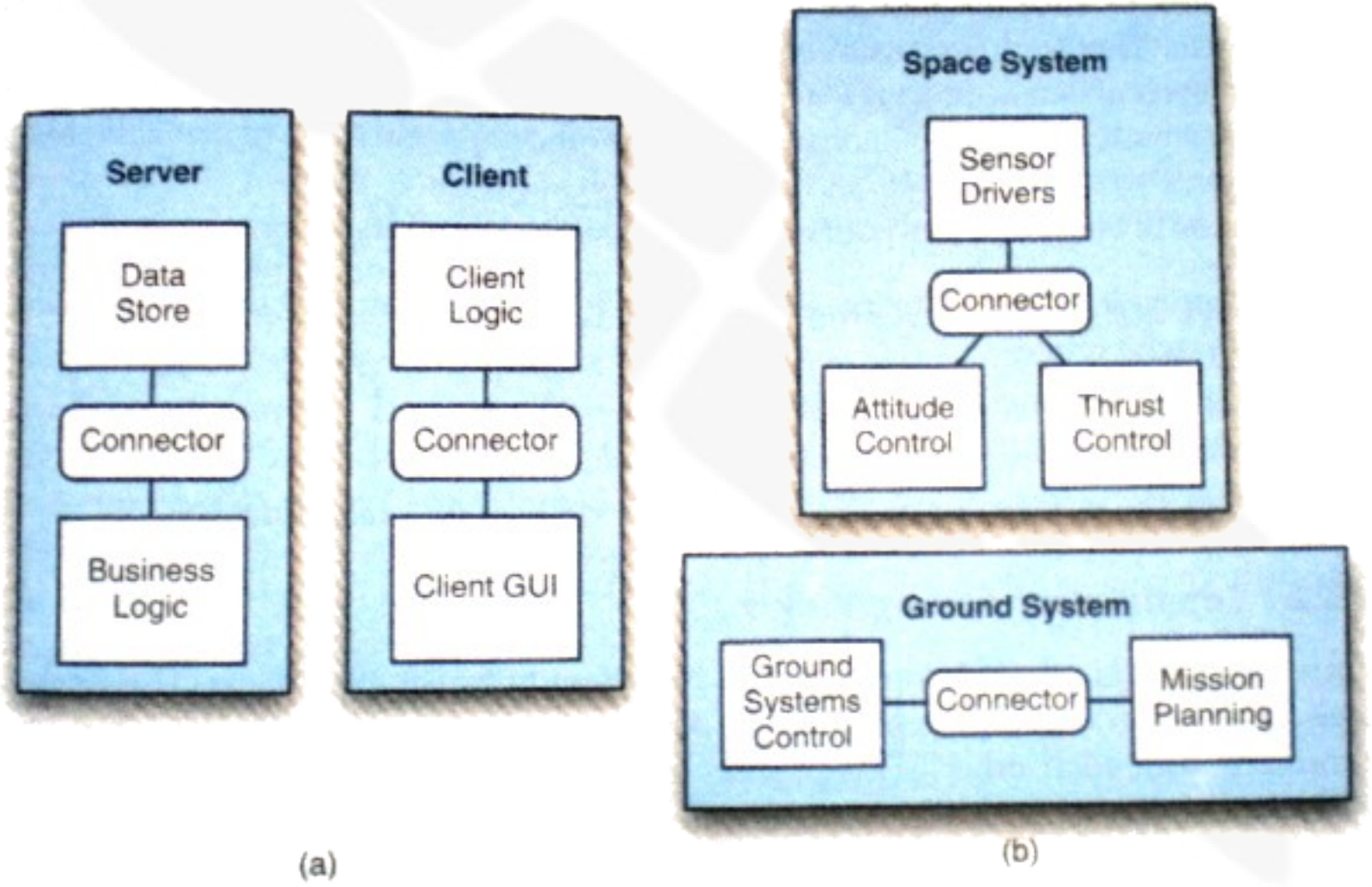
Uma ***view*** é um conjunto de decisões de projeto relacionadas por um interesse comum (ou conjunto de interesses)

Um ***viewpoint*** define uma perspectiva a partir da qual uma *view* é obtida

  - Uma *view* é uma instância de um *viewpoint* para um sistema específico
  - Um *viewpoint* é um filtro de informação e a *view* é o que é visto quando olha-se para o sistema através deste filtro

Modelagem Complexa
Múltiplos Conteúdos – Multiplas *views*

- *Deployment View*:



Modelagem Complexa
Múltiplos Conteúdos – Multiplas *views*

- Exemplos de *viewpoints* frequentemente capturados:
  - *Logical Viewpoint*: captura as entidades lógicas (frequentemente *software*) do sistema e como elas são inter-conectadas
  - *Physical Viewpoint*: captura as entidades físicas (frequentemente *hardware*) do sistema e como elas são inter-conectadas
  - *Deployment Viewpoint*: captura como as entidade lógicas são mapeadas em entidades físicas
  - *Concurrency Viewpoint*: captura como a concorrência e *multi-threading* são gerenciados pelo sistema
  - *Behavioral Viewpoint*: captura o comportamento esperado do sistema (ou de partes dele)

Modelagem Complexa
Múltiplos Conteúdos – Multiplas *views*

- É possível ainda que múltiplas *views* sejam obtidas a partir do mesmo *viewpoint*, com diferentes níveis de detalhe
- O uso de *views* e *viewpoints* é importante pois:
  - Disponibilizam uma forma de limitar a informação apresentada a um sub-conjunto cognitivamente gerenciável da arquitetura
  - Exibem, de forma simultânea, conceitos relacionados
  - Podem ser adaptados às necessidades dos *stakeholders*
  - Podem ser utilizados para exibir os mesmos dados em diferentes níveis de abstração

Modelagem Complexa
Múltiplos Conteúdos – Multiplas *views*

- Consistência entre *views*:
  - Como saber se duas ou mais *views* que exibem informações relacionadas são consistentes entre si ?
  - *Views* são consistentes se as decisões de projeto que elas contêm são compatíveis
  - Uma inconsistência ocorre quando duas *views* afirmam decisões que não podem ser simultaneamente verdadeiras
  - Possíveis tipos de inconsistência:
    - Inconsistência Direta
    - Inconsistência de Refinamento
    - Inconsistência de Aspecto Estático x Aspecto Dinâmico
    - Inconsistência entre Aspectos Dinâmicos
    - Inconsistêncua de Aspecto Funcional x Aspecto Não-Funcional

Modelagem Complexa
Múltiplos Conteúdos – Multiplas *views*

- Inconsistência Direta:
  - Ocorre quando duas *views* afirmam proposições diretamente contraditórias
  - Ex: “o sistema executa em dois *hosts*” x “o sistema executa em três *hosts*”
  - Podem ser detectadas por mecanismos automáticos que empregam restrições e regras apropriadas

Modelagem Complexa
Múltiplos Conteúdos – Multiplas *views*

- Inconsistência de Refinamento:
  - Ocorre quando duas *views* do mesmo sistema mas em diferentes níveis de detalhe afirmam proposições contraditórias
  - Ex: uma *view* estrutural de nível mais alto contém um componente que não está presente na *view* estrutural das sub-arquiteturas
  - Podem ser automaticamente detectadas com regras de consistência adequadas, desde que ambas as *views* contenham informações suficientes para entender o relacionamento entre elas

Modelagem Complexa
Múltiplos Conteúdos – Multiplas *views*

- Inconsistência de Aspecto Estático x Aspecto Dinâmico:
  - Ocorre quando uma *view* de um aspecto estático está em conflito com uma *view* de um aspecto dinâmico
  - Ex: um diagrama descrevendo uma sequência de mensagens (aspecto dinâmico) pode descrever um componente que não está presente na *view* estrutural (aspecto estático)
  - São mais difíceis de detectar de forma automática

Modelagem Complexa
Múltiplos Conteúdos – Multiplas *views*

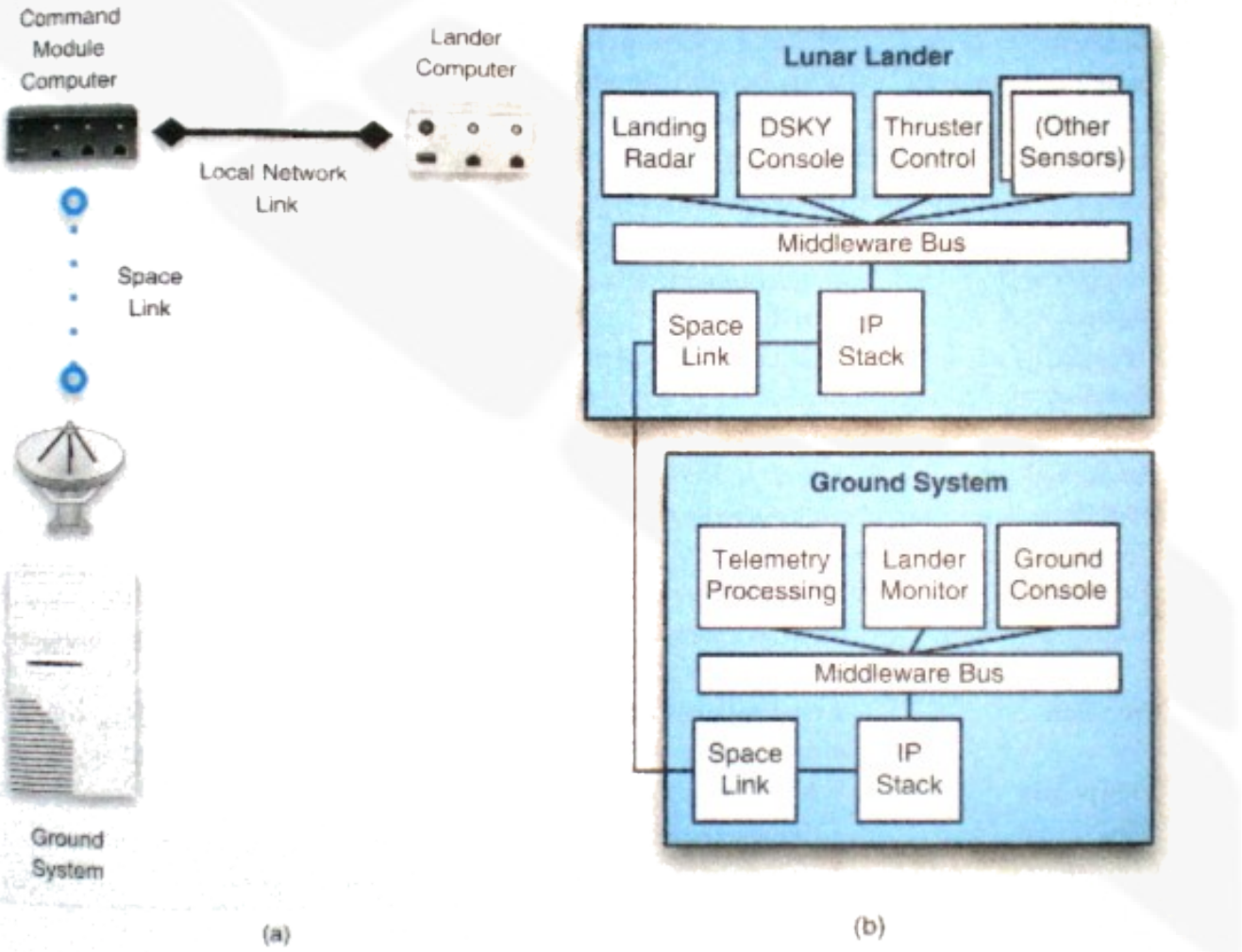
- Inconsistência entre Aspectos Dinâmicos:
  - Ocorre quando duas *views* de aspectos dinâmicos do sistema estão em conflito
  - Ex: um diagrama descrevendo uma sequência de mensagens pode descrever uma interação específica entre componentes que não é permitida pelas especificações comportamentais contidas nos *statecharts* destes componentes
  - São extremamente difíceis de serem automaticamente detectadas, pois requer exploração de estados ou simulações extensivas

Modelagem Complexa
Múltiplos Conteúdos – Multiplas *views*

- Inconsistência de Aspecto Funcional x Aspecto Não-Funcional:
  - Ocorre quando uma propriedade não-funcional de um sistema, descrita em uma *view* não-funcional, não é atendida pelo projeto descrito nas *views* funcionais
  - Ex: uma *view* não-funcional de um sistema *client-server* descreve que o sistema deve ser robusto, porém a *view* física do sistema exibe somente um servidor, ou seja, nenhum evidência de mecanismos para tratamento de falhas
  - São as mais difíceis de detectar devido à natureza genérica e abstrata das propriedades não-funcionais

Modelagem Complexa
Múltiplos Conteúdos – Multiplas *views*

- Inconsistência Direta:



Técnicas Específicas de Modelagem

- Arquitetos têm à sua disposição um conjunto de notações e técnicas para modelar diferentes aspectos de uma arquitetura
- Ao mesmo tempo em que algumas abordagens de modelagem arquitetural são bastante amplas e aplicáveis a uma série de sistemas em diversos domínios é importante não se ater dogmaticamente a uma única técnica

Técnicas Específicas de Modelagem

- Técnicas genéricas:
  - Linguagem Natural
  - Gráficos Informais
  - *Unified Modeling Language* (*UML*)
- *Architecture Description Languages* (*ADLs*):
  - 1ª geração:
    - *Darwin*
    - *Rapide*
    - *Wright*
  - Específicas de Domínio ou de Estilo:
    - Koala
    - Weaves
    - AADL (*Architecture Analysis and Design Language*)

Técnicas Específicas de Modelagem
Técnicas Genéricas

- Linguagem Natural:

Escopo e Propósito: descrever conceitos arbitrários utilizando vocabulário extensivo porém informal
Elementos Básicos: qualquer conceito necessário, porém sem suporte especial a nenhum deles
Estilo: conceitos estilísticos podem ser descritos utilizando uma linguagem mais genérica
Aspectos Estáticos e Dinâmicos: tanto aspectos estáticos quanto dinâmicos podem ser modelados
Modelagem Dinâmica: modelos podem ser modificados por humanos mas não há uma forma viável de relacionar estes modelos à implementação do sistema
Aspectos Não-Funcionais: disponibilidade de vocabulário extensivo para descrição de aspectos não-funcionais. Não há suporte para verificação destes aspectos
Ambiguidade: é provavelmente a notação mais ambígua. *Templates* e dicionários bem definidos podem ajudar a reduzir a ambiguidade
Exatidão: a corretude pode ser avaliada através de revisões e inspeções manuais
Precisão: textos adicionais podem ser utilizados para descrever, em maior detalhe, qualquer aspecto da arquitetura
Viewspoints: todos, porém sem suporte especifico a nenhum deles
Consistência entre *Views*: pode ser avaliada através de revisões e inspeções manuais

Técnicas Específicas de Modelagem
Técnicas Genéricas

- Linguagem Natural (pouso lunar):

A aplicação Pouso Lunar consiste em três componentes: **data store**, **calculation** e **user interface**

O papel do componente **data store** é armazenar e permitir que outros componentes acessem a altitude, velocidade e nível de combustível da nave, bem como o tempo atual de simulação

O papel do componente **calculation** é, ao receber uma determinada taxa de queima de combustível, recuperar os valores atuais de altitude, velocidade e nível de combustível, atualizá-los em relação à taxa de queima recebida e armazená-los de volta. Ele também recupera, incrementa e armazena de volta o tempo atual de simulação. É também responsável por notificar o componente que o invocou sobre o término da simulação, bem como seu estado (pouso com sucesso, nave destruída, etc)

O papel do componente **user interface** é exibir o *status* atual da nave utilizando informações obtidas tanto do **data store** quanto do **calculation**. Enquanto a simulação está em execução ele recebe a nova taxa de queima de combustível do usuário e invoca o componente **calculation**

Técnicas Específicas de Modelagem
Técnicas Genéricas

- Linguagem Natural (resumo):
  - Deve ser utilizada como um auxiliar de linguagens mais rigorosas e formais para aqueles aspectos da arquitetura onde o formalismo é inviável ou desnecessário
  - É particularmente eficiente para especificar propriedades não-funcionais

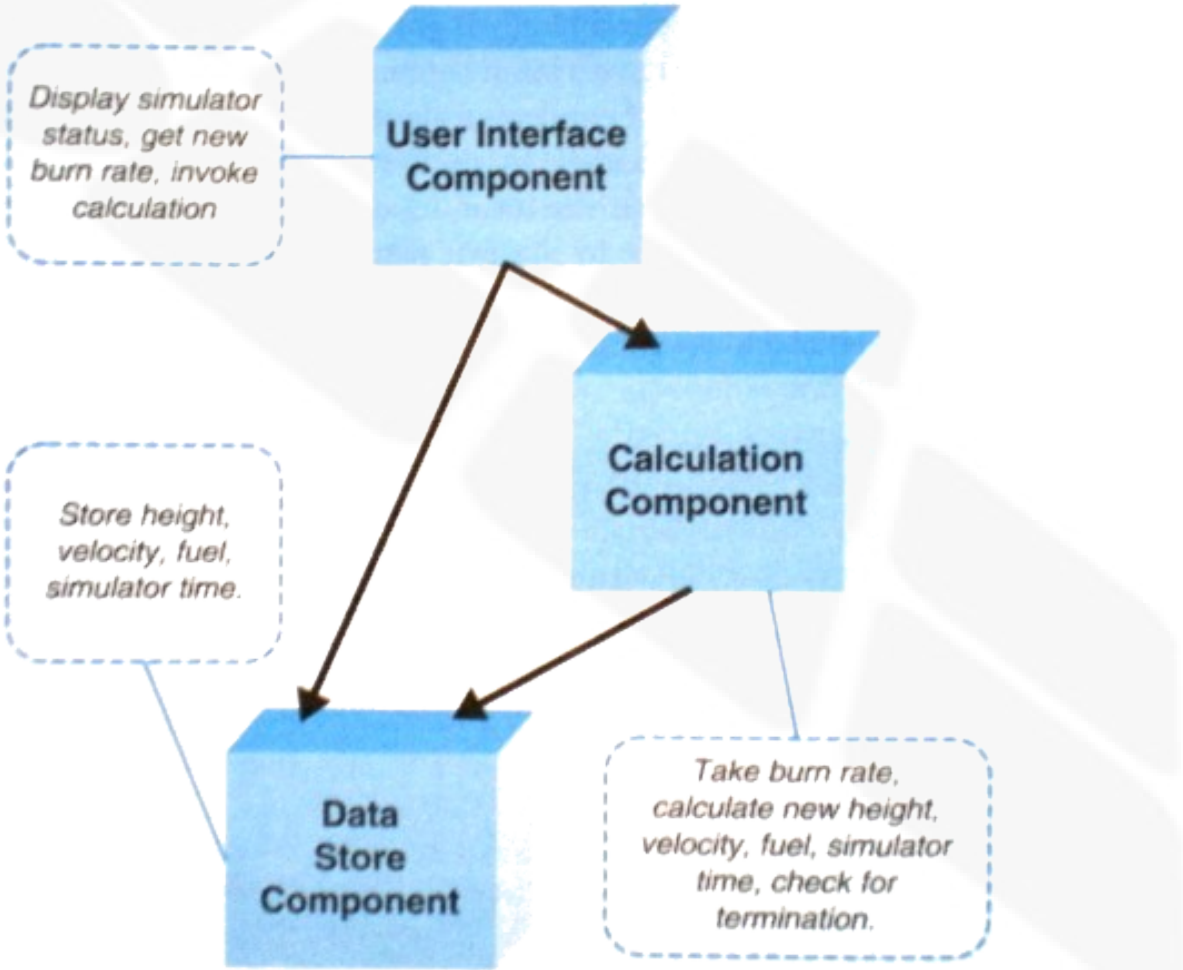
Técnicas Específicas de Modelagem
Técnicas Genéricas

- Gráficos Informais:

Escopo e Propósito: diagramas arbitrários compostos de elementos gráficos e textuais
Elementos Básicos: formas geométricas, *splines*, texto, *clip-art*, etc
Estilo: genérico, sem suporte
Aspectos Estáticos e Dinâmicos: devido à ausência de semântica arquitetural não existe a noção de aspectos estáticos e dinâmicos do sistema
Modelagem Dinâmica: os formatos são geralmente proprietários e difíceis de modificar fora do seu ambiente nativo. Alguns ambientes, entretanto, expõem interfaces para manipulação (ex: MS COM)
Aspectos Não-Funcionais: podem ser modelados através de decorações em linguagem natural
Ambiguidade: pode ser controlada através de um dicionário ou vocabulário simbólico controlado
Exatidão: a corretude pode ser avaliada através de revisões e inspeções manuais
Precisão: pode-se escolher o nível de detalhe, entretanto limita-se à quantidade de informação que cabe em um diagrama ou *slide*
Viewspoints: devido à ausência de semântica arquitetural não há suporte direto a múltiplos *viewpoints*. Em teoria, um modelo pode exibir qualquer *viewpoint*
Consistência entre *Views*: pode ser avaliada através de revisões e inspeções manuais

Técnicas Específicas de Modelagem
Técnicas Genéricas

- Gráficos Informais (pouso lunar):



Técnicas Específicas de Modelagem
Técnicas Genéricas

- Gráficos Informais (resumo):
  - São sedutores pois tornam fácil a criação de diagramas
  - Entretanto, deve-se ter cuidado ao utilizar gráficos informais como parte de qualquer descrição oficial de uma arquitetura:
    - Podem ser interessantes somente para uma descrição inicial da arquitetura (rascunho)

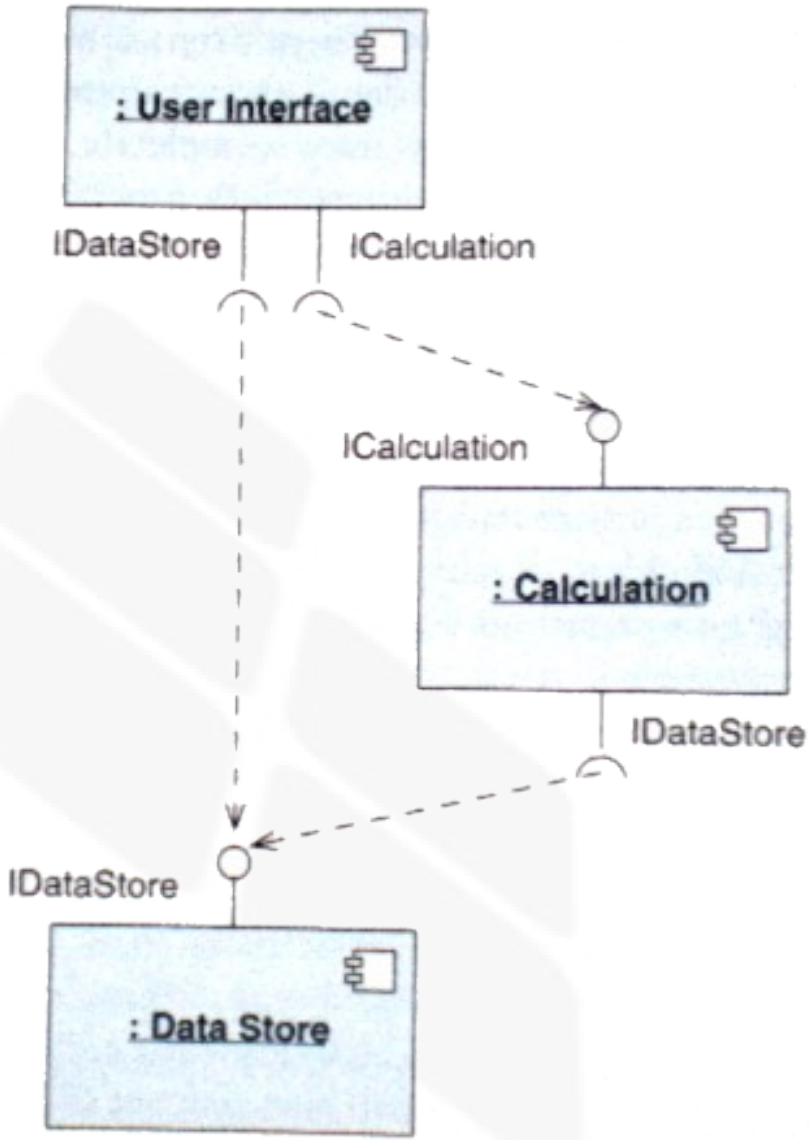
Técnicas Específicas de Modelagem
Técnicas Genéricas

- *Unified Modeling Language* (*UML*):

Escopo e Propósito: capturar as decisões de projeto de um *software* utilizando até treze tipos diferentes de diagramas
Elementos Básicos: classes, associações, estados, atividades, nós *composite*, restrições (*OCL*), etc
Estilo: restrições estilísticas podem ser expressas em *OCL* ou em modelos parciais em um dos diversos *viewpoints*
Aspectos Estáticos e Dinâmicos: inclui um conjunto de diagramas tanto para aspectos estáticos (de classes, de objetos, de pacotes) quanto dinâmicos (de estados, de atividades)
Modelagem Dinâmica: depende do ambiente de modelagem, não ocorre frequentemente na prática
Aspectos Não-Funcionais: não há suporte direto, somente anotações textuais
Ambiguidade: elementos *UML* podem significar coisas diferentes em diferentes contextos. Pode ser reduzida com o uso de *profiles UML*, incluindo *stereotypes*, *tagged values* e restrições
Exatidão: nenhum suporte além de checagem de restrições *OCL* e regras básicas de boa formação
Precisão: pode-se escolher o nível de detalhe desejado, com grande flexibilidade
Viewspoints: cada tipo de diagrama representa no mínimo um *viewpoint*
Consistência entre *Views*: pouco suporte – somente o uso limitado de restrições *OCL*

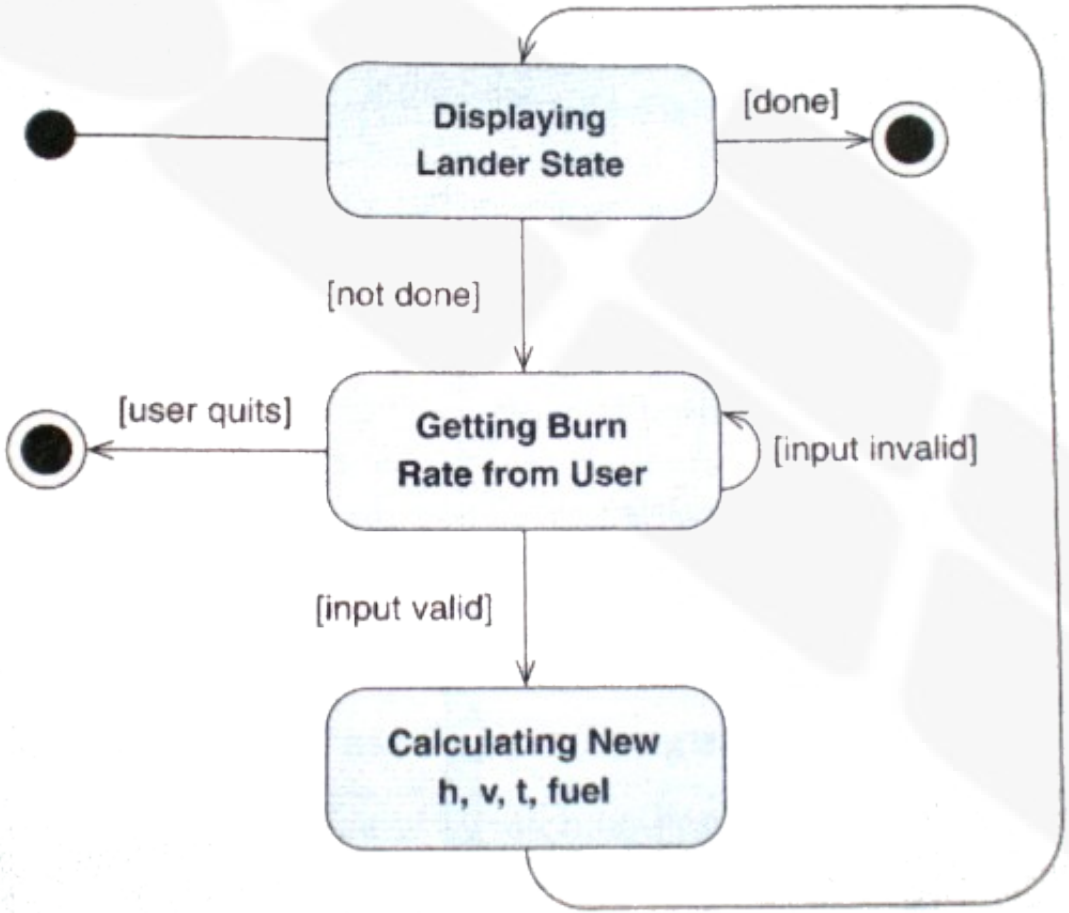
Técnicas Específicas de Modelagem
Técnicas Genéricas

- *Unified Modeling Language* (Pouso Lunar):



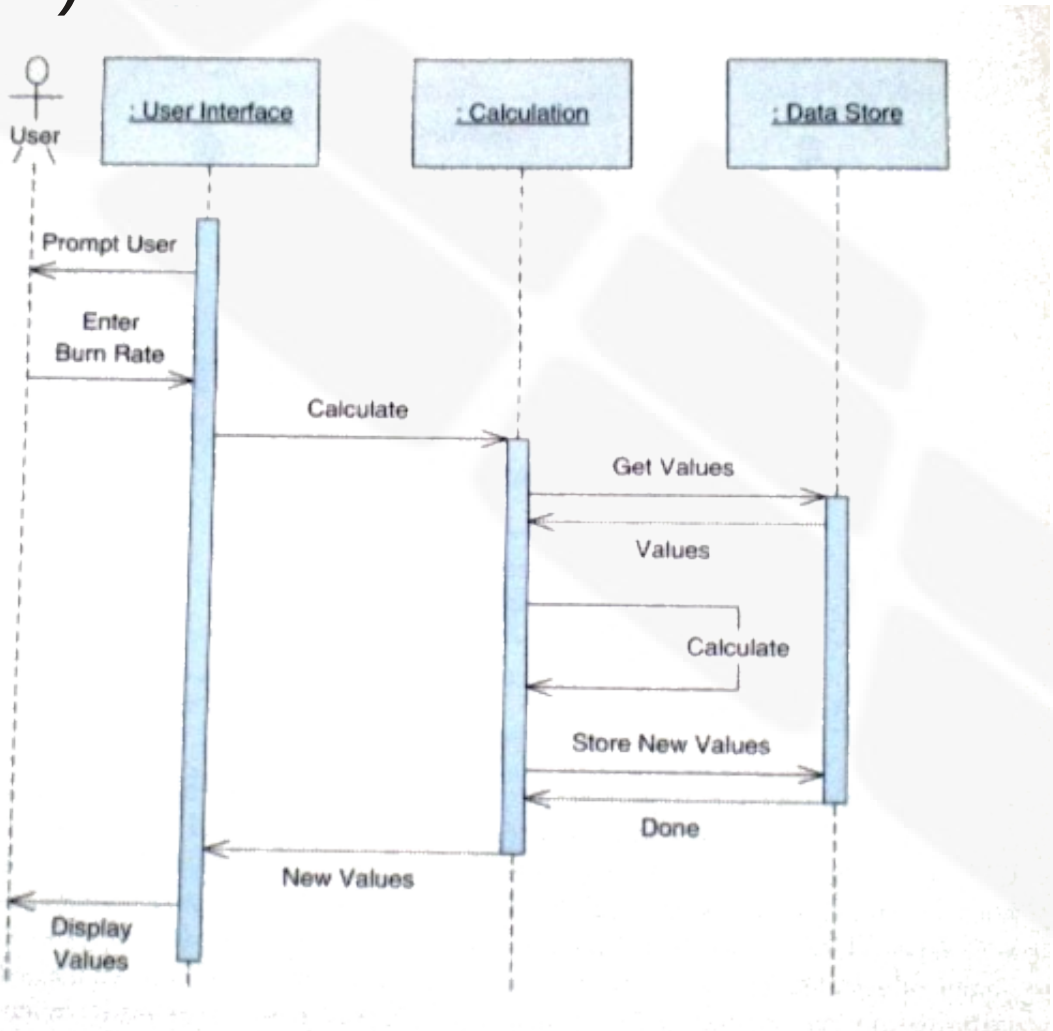
Técnicas Específicas de Modelagem
Técnicas Genéricas

- *Unified Modeling Language* (Pouso Lunar):



Técnicas Específicas de Modelagem
Técnicas Genéricas

- *Unified Modeling Language* (Pouso Lunar):



Técnicas Específicas de Modelagem
Técnicas Genéricas

- *Unified Modeling Language* (resumo):
  - Notação sintaticamente rica com suporte extensivo de ferramentas
  - Superior a notações somente baseadas em símbolos, como os gráficos informais
  - Entretanto, sua ambiguidade propositada a respeito do significado da maioria dos seus símbolos a deixa aberta a abusos
  - *Profiles* devem ser desenvolvidos e utilizados para garantir uma modelagem consistente, embora isso não garanta interpretações não-ambíguas

Técnicas Específicas de Modelagem
Architecture Description Languages (ADLs)

- 1ª geração:
  - *Darwin*
  - *Rapide*
  - *Wright*

Técnicas Específicas de Modelagem
Architecture Description Languages (ADLs)

- 1ª geração - *Darwin*:
  - *ADL* de propósito geral para especificação da estrutura de sistemas formados por componentes que se comunicam através de interfaces explícitas
  - Inclui uma representação textual canônica onde componentes e suas inter-conexões são descritos
  - Inclui também uma visualização gráfica associada
  - Não há a noção de conectores explícitos, mas um componente que facilita as interações pode ser interpretado como um conector
  - Componentes exportam um conjunto de serviços (interfaces) *required* e *provided*, genericamente denominados *ports*
  - Suporta composição hierárquica

Técnicas Específicas de Modelagem
Architecture Description Languages (ADLs)

- 1ª geração - *Darwin*:

Escopo e Propósito: estruturas de sistemas distribuídos que se comunicam através de interfaces
Elementos Básicos: componentes, interfaces (*required* e *provided*), ligações e composições hierárquicas
Estilo: suporte limitado através do uso de construtores parametrizados
Aspectos Estáticos e Dinâmicos: suporte a *views* estruturais estáticas, suporte adicional a arquiteturas dinâmicas através de *lazy/dynamic instantiation/binding*
Modelagem Dinâmica: não disponível
Aspectos Não-Funcionais: não disponível
Ambiguidade: a composição e uso dos elementos são bem definidos. Seu significado externo está sujeito à interpretação dos *stakeholders*
Exatidão: pode ser modelado em *pi-calculus*, possibilitando a análise de consistência interna
Precisão: detalhes limitados a elementos estruturais e suas inter-conexões
Viewspoints: estruturais e de implantação através do uso de composição hierárquica
Consistência entre *Views*: não disponível

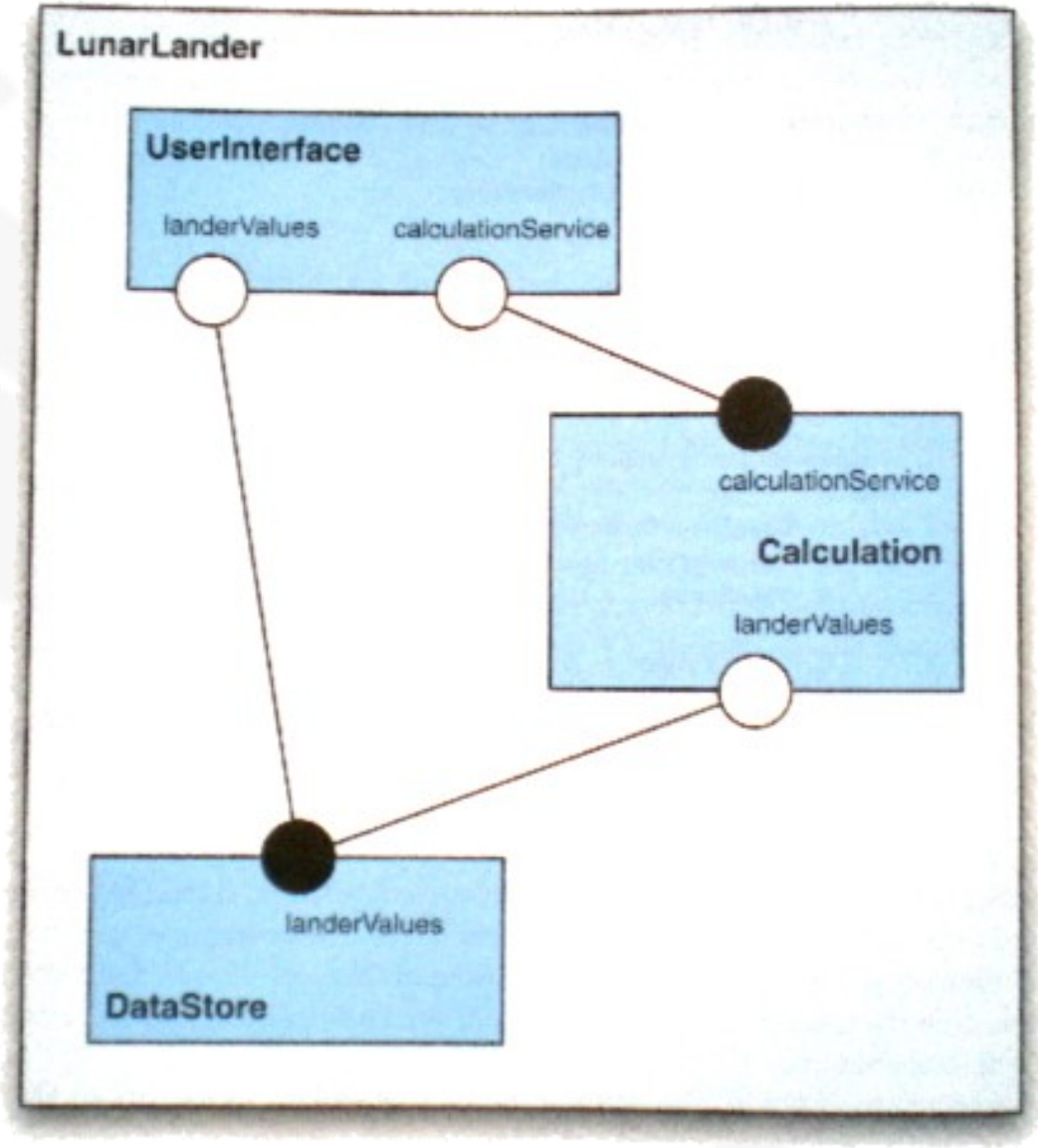
Técnicas Específicas de Modelagem
Architecture Description Languages (ADLs)

- 1ª geração – *Darwin* (pouso lunar):

```
component DataStore{
    provide landerValues;
}

component Calculation{
    require landerValues;
    provide calculationService;
}

component UserInterface{
    require calculationService;
    require landerValues;
}

component LunarLander{
inst
    U: UserInterface;
    C: Calculation;
    D: DataStore;
bind
    C.landerValues -- D.landerValues;
    U.landerValues -- D.landerValues;
    U.calculationService -- C.calculationService;
}
```

Técnicas Específicas de Modelagem
Architecture Description Languages (ADLs)

- 1ª geração – *Darwin* (pouso lunar):



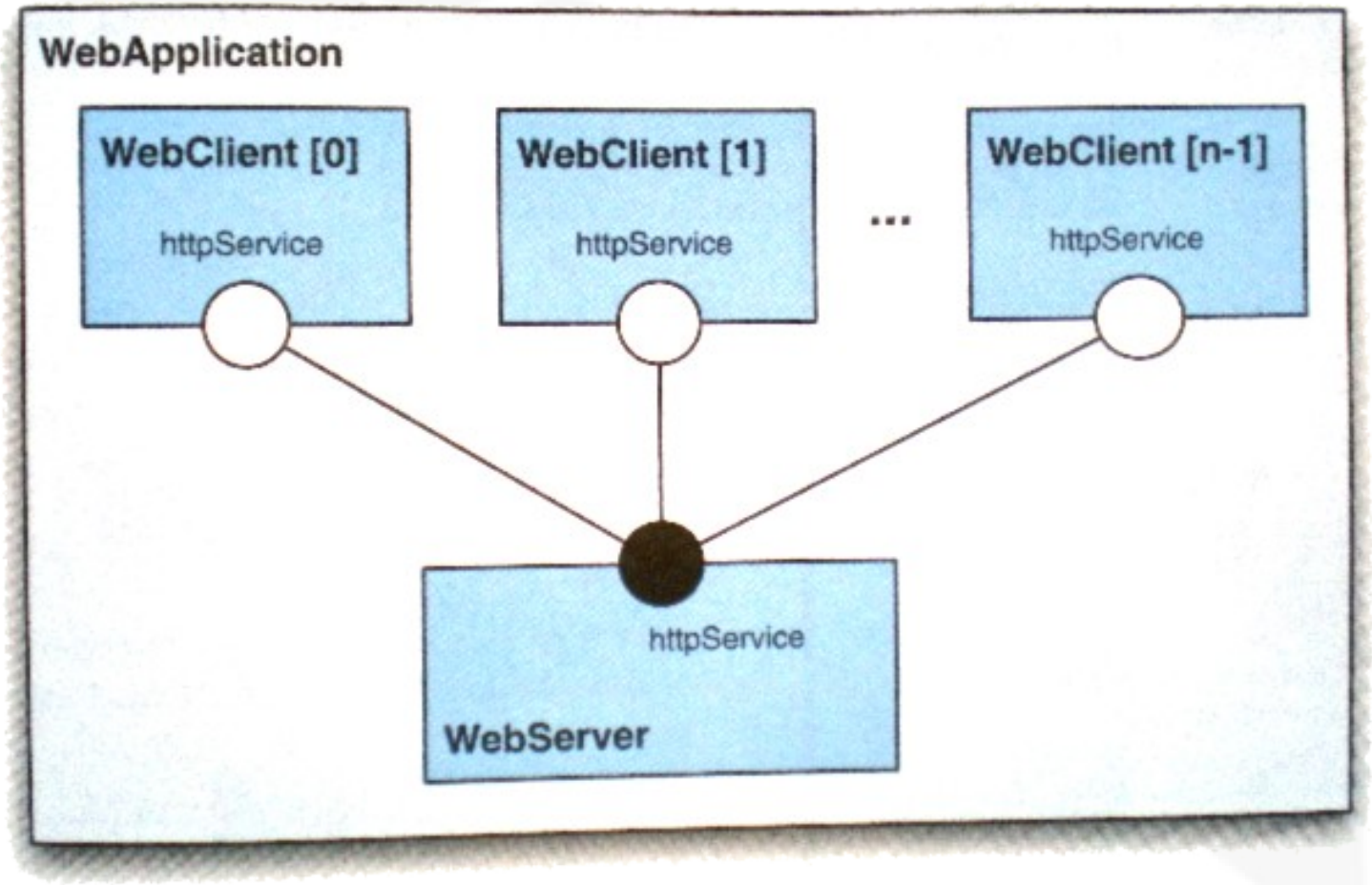
Técnicas Específicas de Modelagem
Architecture Description Languages (ADLs)

- 1ª geração – *Darwin* (aplicação web):

```
component WebServer{
    provide httpService;
}

component WebClient{
    require httpService;
}

component WebApplication(int numClients){
    inst S: WebServer;
    array C[numClients]: WebClient;
    forall k:0..numClients-1{
        inst C[k] @ k;
        bind C[k].httpService -- S.httpService;
    }
}
```

Técnicas Específicas de Modelagem
Architecture Description Languages (ADLs)

- 1ª geração – *Darwin* (aplicação web):



Técnicas Específicas de Modelagem
Architecture Description Languages (ADLs)

- 1ª geração – *Rapide*:

Escopo e Propósito: interações entre componentes, descritas como conjuntos parcialmente ordenados de eventos
Elementos Básicos: arquiteturas (estruturas), interfaces (componentes), ações (mensagens/eventos) e operações descrevendo como as ações se relacionam
Estilo: não disponível
Aspectos Estáticos e Dinâmicos: inter-conexões entre componentes capturam a estrutura estática, ações e comportamentos capturam aspectos dinâmicos
Modelagem Dinâmica: os modelos não mudam em *run-time*, embora algumas ferramentas disponibilizam capacidades limitadas de animação
Aspectos Não-Funcionais: não disponível
Ambiguidade: as semânticas dos comportamentos são bem definidos
Exatidão: simulador produz resultados que podem ser examinados + restrições Rapide
Precisão: inter-conexões e comportamentos podem ser modelados em detalhe
Viewspoints: um único *viewpoint* estrutural-comportamental
Consistência entre *Views*: verificação automática não disponível. Inspeções podem ser usadas

Técnicas Específicas de Modelagem
Architecture Description Languages (ADLs)

- 1ª geração – *Rapide* (pouso lunar):

```
type DataStore is interface
  action in  SetValues();
         out NotifyNewValues();
  behavior
  begin
         SetValues => NotifyNewValues();;
end DataStore;

type Calculation is interface
  action in  SetBurnRate();
         out DoSetValues();
  behavior
         action CalcNewState();
  begin
         SetBurnRate => CalcNewState(); DoSetValues();;
end Calculation;
```

Técnicas Específicas de Modelagem
Architecture Description Languages (ADLs)

- 1ª geração – *Rapide* (pouso lunar):

```
type Player is interface
  action out DoSetBurnRate();
         in  NotifyNewValues();
  behavior
         TurnsRemaining : var integer := 1;
         action UpdateStatusDisplay();
         action Done();
  begin
         (start or UpdateStatusDisplay) where \
             ($TurnsRemaining > 0) => \
             if ( $TurnsRemaining > 0 ) then \
                 TurnsRemaining := $TurnsRemaining - 1; \
                 DoSetBurnRate(); \
             end if;;
         NotifyNewValues => UpdateStatusDisplay();;
         UpdateStatusDisplay where $TurnsRemaining == 0 \
             => Done();;
end UserInterface;
```

Técnicas Específicas de Modelagem
Architecture Description Languages (ADLs)

- 1ª geração – *Rapide* (pouso lunar):

```
architecture lander() is
  P1, P2 : Player;
  C : Calculation;
  D : DataStore;
connect
  P1.DoSetBurnRate to C.SetBurnRate;
  P2.DoSetBurnRate to C.SetBurnRate;
  C.DoSetValues to D.SetValues;
  D.NotifyNewValues to P1.NotifyNewValues();
  D.NotifyNewValues to P2.NotifyNewValues();
end LunarLander;
```

Técnicas Específicas de Modelagem
Architecture Description Languages (ADLs)

- 1ª geração – *Rapide* (resumo):
  - Aborda aspectos dinâmicos da arquitetura de forma direta
  - Disponibiliza ferramenta de apoio à simulação destes aspectos
  - Entretanto, a notação é arcaica e com uma alta curva de aprendizado
  - Não possui recursos para garantir que as implementações sejam consistentes com as especificações

Técnicas Específicas de Modelagem
Architecture Description Languages (ADLs)

- 1ª geração – *Wright*:

Escopo e Propósito: estruturas, comportamentos e estilos de sistemas
Elementos Básicos: componentes, conectores, *ports* e *roles* (interfaces), *attachments* e estilos
Estilo: suportado através de predicados aplicados a modelos de instância
Aspectos Estáticos e Dinâmicos: modelos estruturais estáticos são anotados com especificações comportamentais que descrevem como componentes e conectores devem se comportar e interagir
Modelagem Dinâmica: não disponível
Aspectos Não-Funcionais: não disponível
Ambiguidade: especificações em *Wright* podem ser traduzidas em *CSP*, porém o significado externo dos componentes e conectores não é especificado
Exatidão: com as traduções para *CSP* pode-se realizar verificação automática de consistência estrutural e *deadlocks*
Precisão: pode descrever estruturas e seu comportamento em grande detalhe – especificações comportamentais formais são necessárias para a realização de análises
Viewspoints: estrutural, comportamental e de estilo
Consistência entre *Views*: pode ser determinada com o uso de um avaliador *CSP*

Técnicas Específicas de Modelagem
Architecture Description Languages (ADLs)

- 1ª geração – *Wright* (pouso lunar):

```
Component DataStore
  Port getValues (behavior specification)
  Port storeValues (behavior specification)
  Computation (behavior specification)

Component Calculation
  Port getValues (behavior specification)
  Port storeValues (behavior specification)
  Port calculate (behavior specification)
  Computation (behavior specification)

Component UserInterface
  Port getValues (behavior specification)
  Port calculate (behavior specification)
  Computation (behavior specification)

Connector Call
  Role Caller = call → return → Caller[] §
  Role Callee = call → return → Callee[] §
    Caller.call → Callee.call → Glue
  Glue = []Callee.return → Caller.return → Glue
    [] §
```

Técnicas Específicas de Modelagem
Architecture Description Languages (ADLs)

- 1ª geração – *Wright* (pouso lunar):

```
Configuration LunarLander
  Instances
    DS : DataStore
    C : Calculation
    UI : UserInterface
    CtoUIgetValues, CtoUIstoreValues, UItoC, UItoDS : Call

  Attachments
    C.getValues as CtoUIgetValues.Caller
    DS.getValues as CtoUIgetValues.Callee

    C.storeValues as CtoUIstoreValues.Caller
    DS.storeValues as CtoUIstoreValues.Callee

    UI.calculate as UItoC.Caller
    C.calulate as UItoC.Callee

    UI.getValues as UItoDS.Caller
    DS.getValues as UItoDS.Callee
End LunarLander.
```

Técnicas Específicas de Modelagem
Architecture Description Languages (ADLs)

- 1ª geração – *Wright* (resumo):
  - Alta curva de aprendizado e alto *overhead* cognitivo
  - Capacidades poderosas de análise
  - Pode valer a pena em sistemas *safety-critical*, que necessitam de modelagem extensivamente formal
  - Não possui suporte ao refinamento de especificações arquiteturais em artefatos de implementação

Técnicas Específicas de Modelagem
Architecture Description Languages (ADLs)

- Específicas de Domínio ou de Estilo:
  - Koala
  - Weaves
  - AADL (*Architecture Analysis and Design Language*)

Técnicas Específicas de Modelagem
Architecture Description Languages (ADLs)

- Específicas de Domínio ou de Estilo - *Koala*:

Escopo e Propósito: capturar a estrutura, configuração e interfaces de componentes no domínio de sistemas embarcados para dispositivos de eletrônica de consumo
Elementos Básicos: componentes, interfaces e construtores para pontos específicos de variação (*diversity interfaces*, *switches* e *multiplexers*)
Estilo: Koala captura arquiteturas de produtos relacionados definindo, portanto, um estilo arquitetural
Aspectos Estáticos e Dinâmicos: somente estrutura estática e interfaces são modelados
Modelagem Dinâmica: embora os pontos de variação sejam definidos estaticamente na arquitetura a seleção das variantes pode mudar em *run-time*
Aspectos Não-Funcionais: não são modelados explicitamente
Ambiguidade: elementos Koala são mapeados em implementações de forma estreita e concreta
Exatidão: modelos possuem regras e padrões de inter-conexão bem definidos. Erros são relativamente fáceis de identificar procurando por violações de padrões ou erros de compilação
Precisão: configuração bem definida, porém outros aspectos não são especificados
Viewspoints: estrutural com pontos específicos de variação
Consistência entre *Views*: em geral, não inclui múltiplas *views*

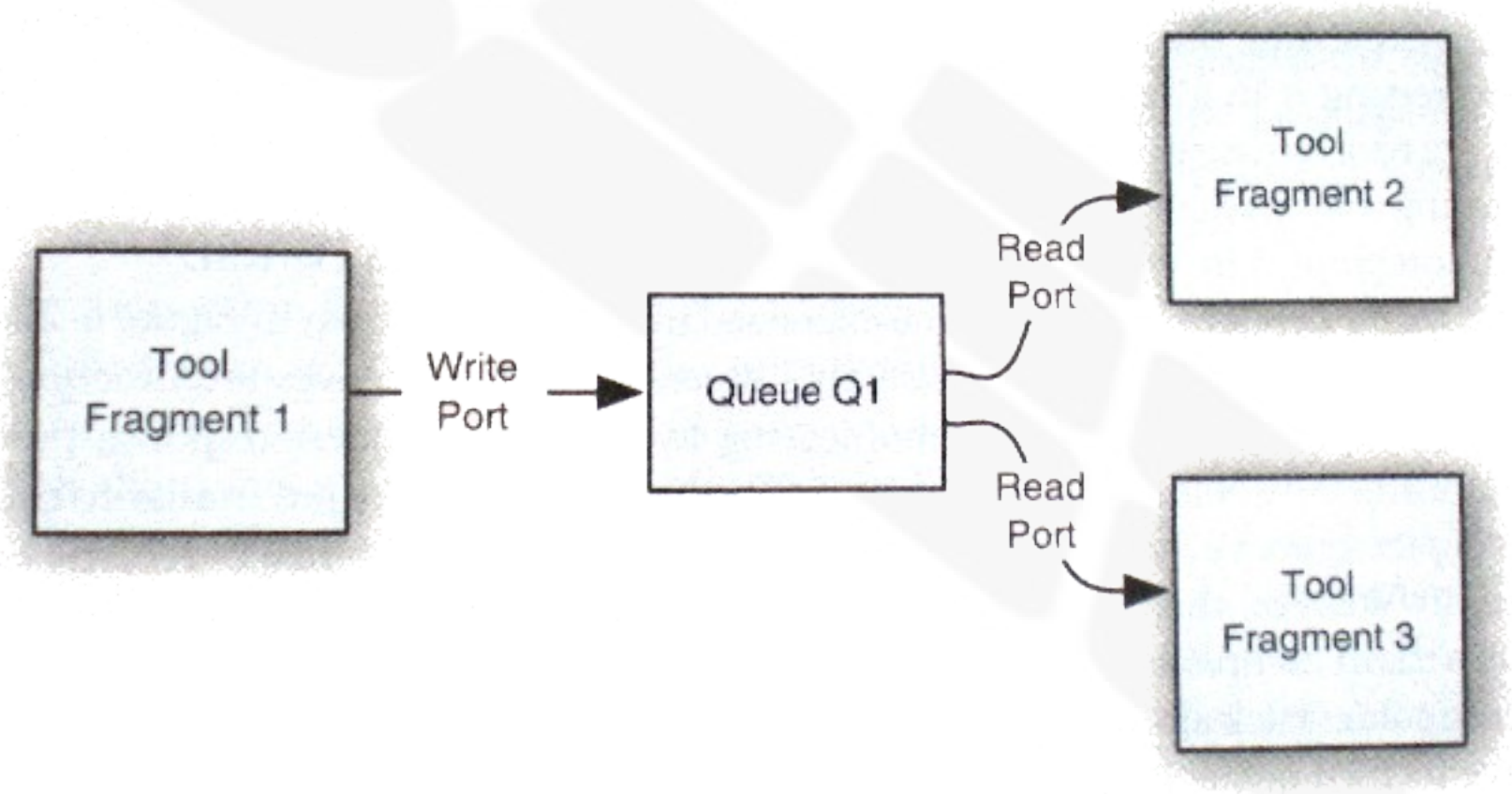
Técnicas Específicas de Modelagem
Architecture Description Languages (ADLs)

- Específicas de Domínio ou de Estilo - *Weaves*:

Escopo e Propósito: capturar a estrutura e configuração de componentes em arquiteturas em conformidade com o estilo arquitetural *Weaves*
Elementos Básicos: componentes, conectores (*queues*) e inter-conexões direcionadas
Estilo: estilo arquitetural *Weaves*, com restrições implícitas no modelo
Aspectos Estáticos e Dinâmicos: somente estrutura estática é modelada
Modelagem Dinâmica: embora sem suporte direto, existe clara correspondência entre arquitetura e implementação. Mudanças em um podem ser rastreadas e aplicadas em outro
Aspectos Não-Funcionais: induz certas propriedades não-funcionais, porém não são explicitamente modeladas
Ambiguidade: os significados dos elementos do *Weaves* são bem especificados e não-ambíguos
Exatidão: erros estão limitados a ligações quebradas e problemas de inter-conexão. É fácil determinar se implementações correspondem a modelos *Weaves*
Precisão: a configuração de componentes e conectores é bem definida, mas outros aspectos não são identificados
Viewspoints: estrutural
Consistência entre *Views*: em geral, não inclui múltiplas *views*

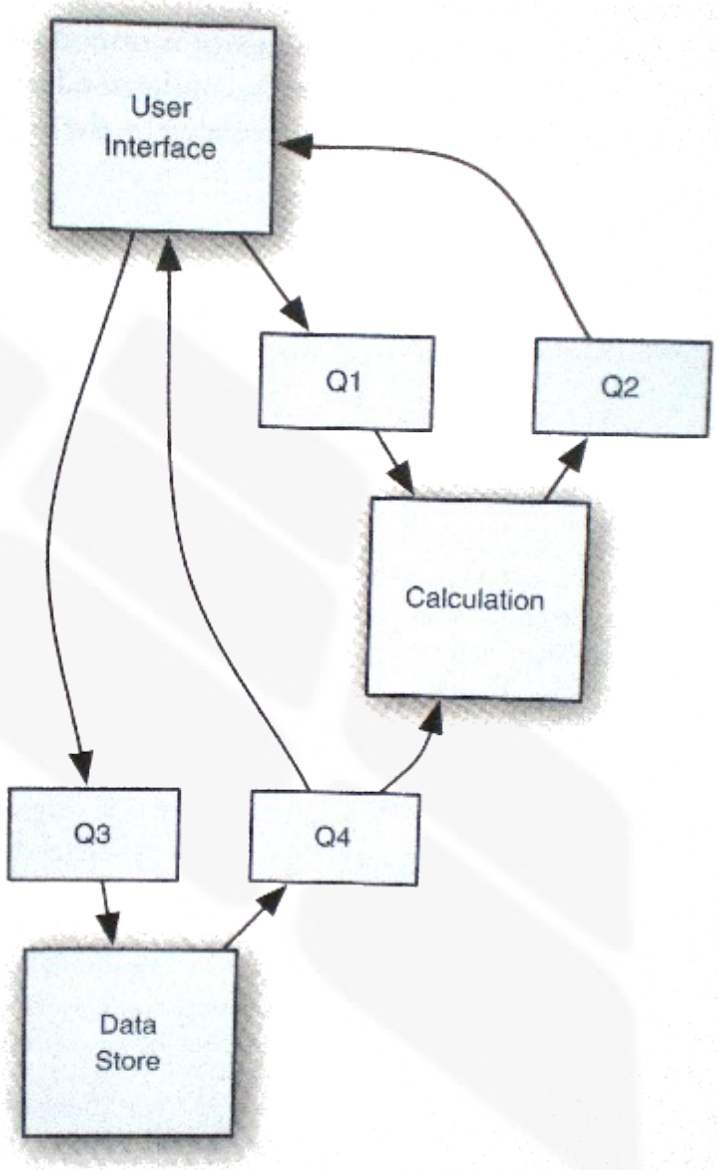
Técnicas Específicas de Modelagem
Architecture Description Languages (ADLs)

- Específicas de Domínio ou de Estilo – *Weaves*:
  - Construtor básico



Técnicas Específicas de Modelagem
Architecture Description Languages (ADLs)

- Específicas de Domínio ou de Estilo – *Weaves* (pouso lunar):



Técnicas Específicas de Modelagem
Architecture Description Languages (ADLs)

- Específicas de Domínio ou de Estilo – AADL:

Escopo e Propósito: modelos multi-nível de elementos de *hardware* e *software* inter-conectados
Elementos Básicos: elementos de *hardware* e *software* (redes, barramentos, processos, *threads* etc)
Estilo: não há suporte explícito
Aspectos Estáticos e Dinâmicos: primariamente aspectos estáticos, mas propriedades podem capturar alguns aspectos dinâmicos
Modelagem Dinâmica: não há suporte explícito
Aspectos Não-Funcionais: propriedades definidas pelo usuário podem capturar aspectos não-funcionais, porém estes não podem ser automaticamente analisados
Ambiguidade: elementos possuem um correspondente no mundo real
Exatidão: propriedades estruturais podem ser automaticamente analisadas a partir de modelos suficientemente anotados
Precisão: propriedades especificam características com detalhes suficientes para serem analisadas
Viewspoints: *viewpoints* de *hardware* e *software* inter-conectados: sistema, processador, processos, *threads*, rede, etc
Consistência entre *Views*: o *Open Source AADL Tool Environment* (OSATE) inclui vários *plugins* para verificações de consistência

Técnicas Específicas de Modelagem
Architecture Description Languages (ADLs)

- Específicas de Domínio ou de Estilo – AADL (resumo):
  - Notação de alto custo e alto valor
  - É produto de um significativo esforço de pensamento e desenvolvimento
  - Possibilita que análises não triviais sejam realizadas
  - Especificações são complexas e extensas, mesmo para sistemas pequenos
  - Pode valer a pena no domínio específico de sistemas embarcados de tempo-real

INF016 – Arquitetura de Software

06 - Modelagem

Sandro Santos Andrade
sandroandrade@ifba.edu.br

Instituto Federal de Educação, Ciência e Tecnologia da Bahia
Departamento de Tecnologia Eletro-Eletrônica
Graduação Tecnológica em Análise e Desenvolvimento de Sistemas