Centro de Informática UFPE

Universidade Federal de Pernambuco

Organizando elementos SysML em Pacotes

Curso Ford

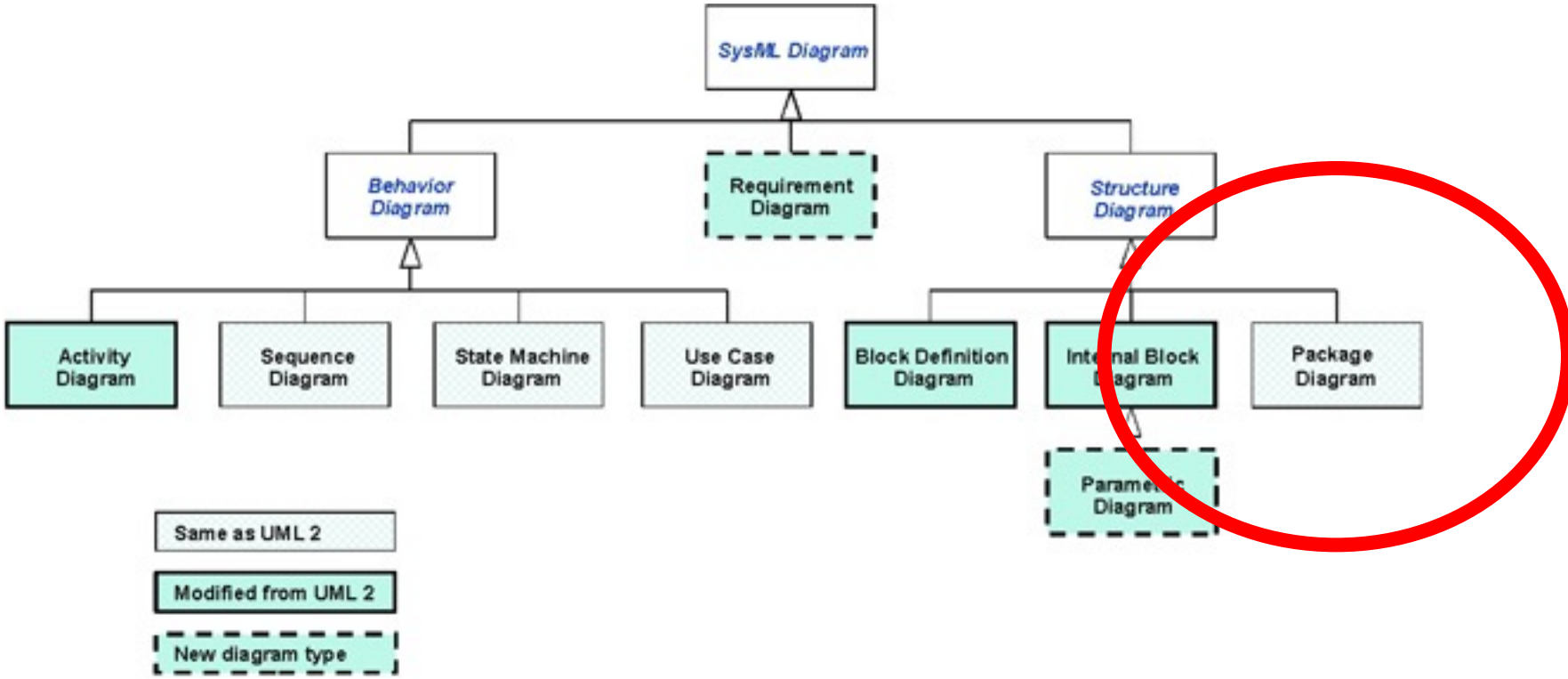
Prof. Lucas Albertins

Agenda

- Pacotes
- O diagrama de pacotes
- Definindo Pacotes
- Organizando uma hierarquia de pacotes
- Exibindo elementos dentro de pacotes
- Pacotes como resolução de nomes (namespaces)
- Importando elementos em pacotes
- Dependências entre elementos do pacote
- Exercício

SysML

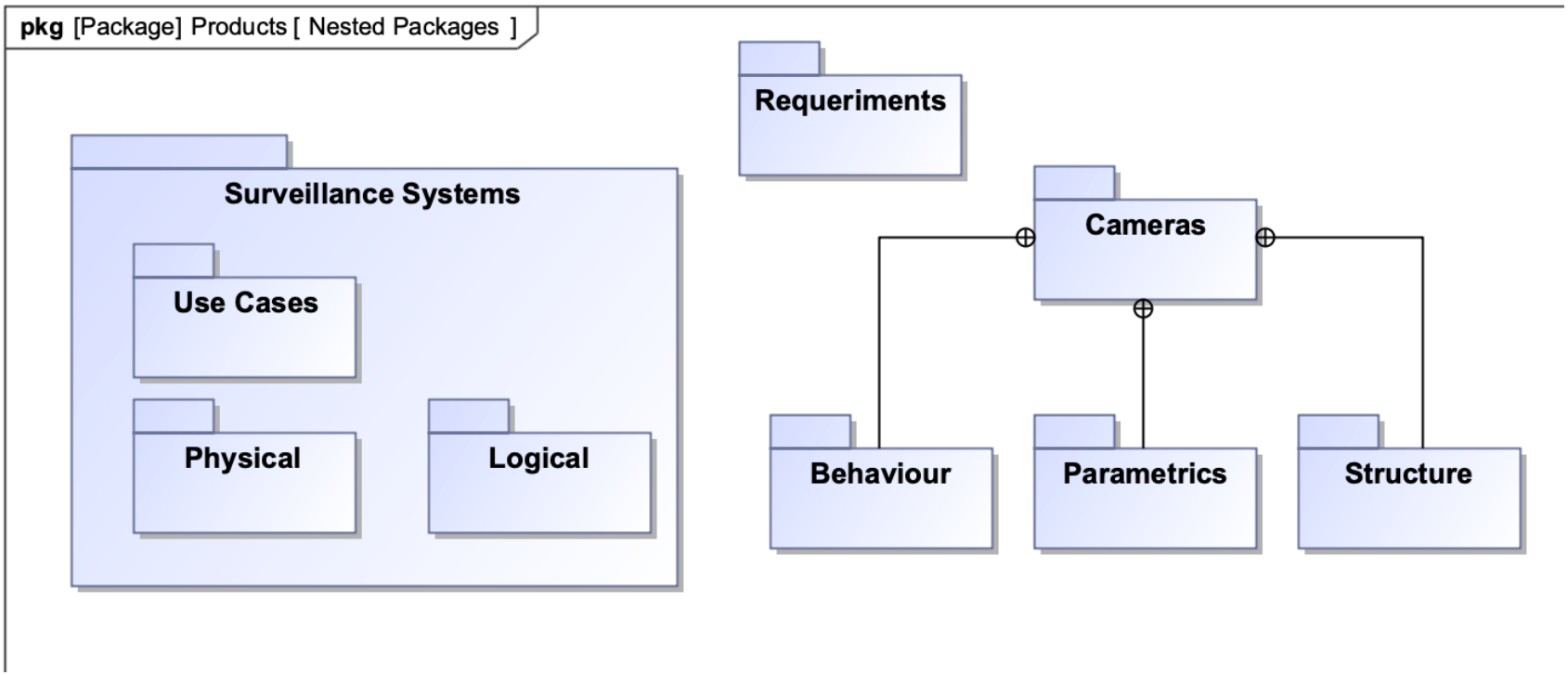
- **UML Profile for System Engineering**



Pacotes

- Usados para organizar o modelo
  - Agrupam elementos do modelo em um *namespace*
  - Representados em uma ferramenta de navegação
  - Suportam o gerenciamento de configurações
- Modelos podem ser organizados de diferentes maneiras
  - Hierarquia do Sistema (e.g., empresa, Sistema, componente)
  - Tipo de diagrama (e.g., requisites, casos de uso, componentes)
- Diagramas de Pacote fornecem uma representação gráfica da organização do modelo e/ou do conteúdo do pacote

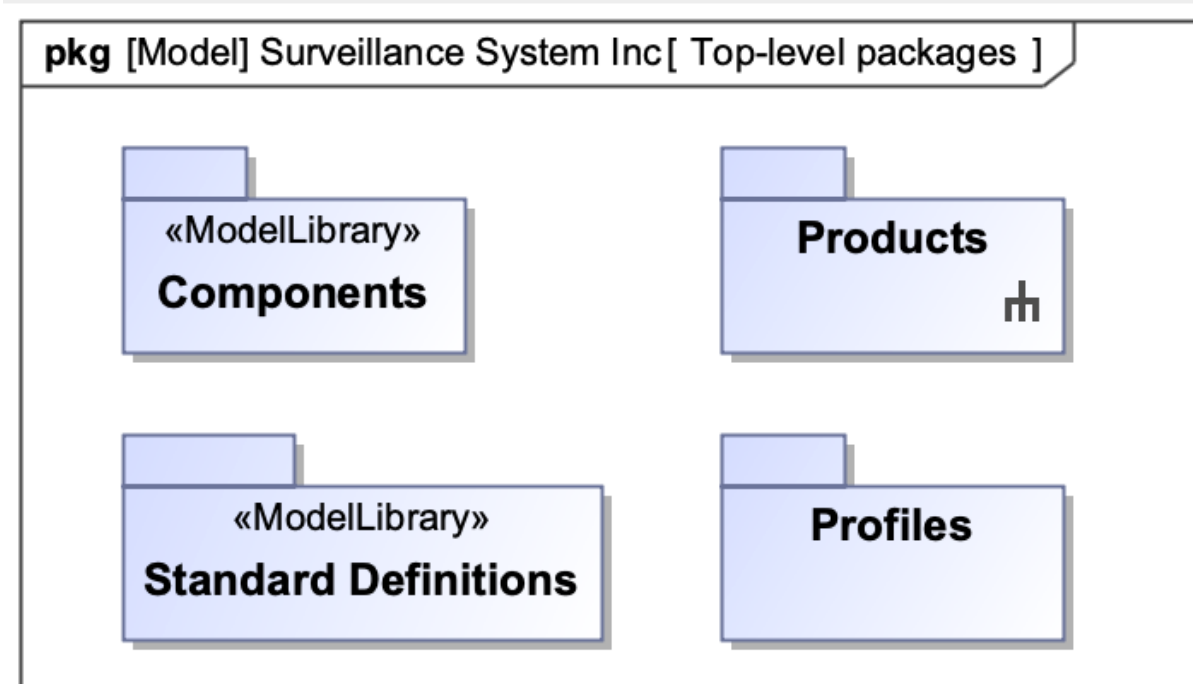
O diagrama de pacotes



Tipos de pacotes

- Pacote (*Package)*
  - Contêiner para elementos do modelo, os quais estão contidos em exatamente um pacote
  - Quando o pacote é deletado/copiado, o mesmo acontece com os seus elementos
- Modelo (*Model)*
  - Geralmente pacote no topo da hierarquia que contém a descrição completa de um sistema ou domínio a ser modelado
- *Model Library*
  - Pacote cujo propósito focal é o seu reuso em outros pacotes
  - Utiliza-se o estereótipo «modelLibrary»
- *View*
  - Fornece perspectivas adicionais sobre o modelo usando diferentes princípios organizacionais

Tipos de Pacotes

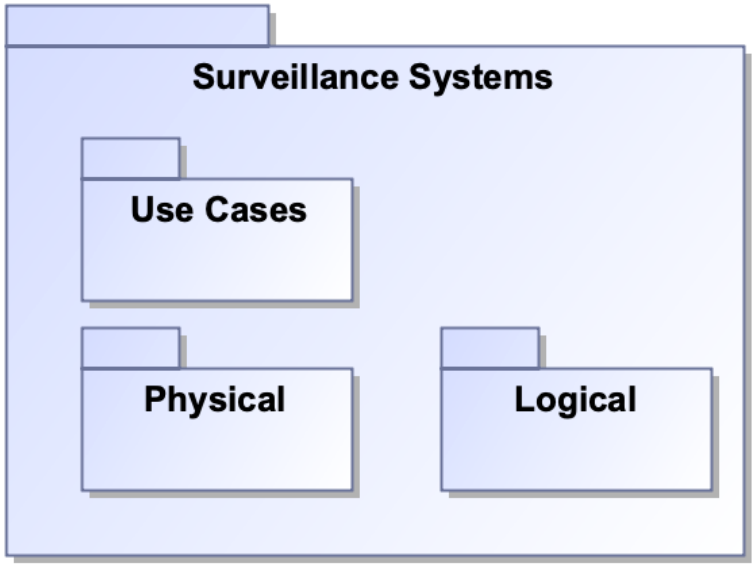


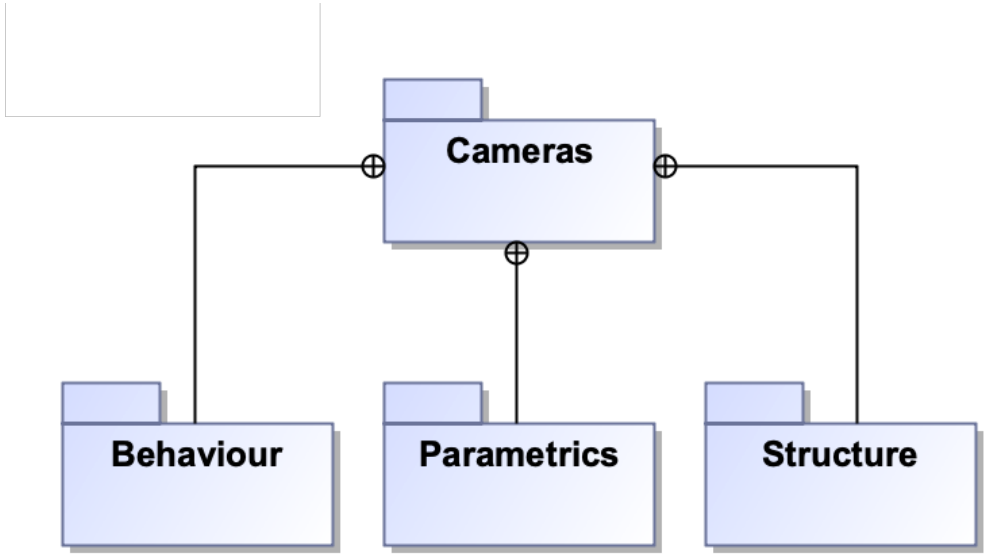
Princípios para organização de Pacotes

- Hierarquia de Sistema: nível de sistema, elemento, componente, etc.
- Processo de ciclo de vida: requisitos, análise, projeto, ...
- Equipe de trabalho: Time de requisitos, Time de Integração, Time de Testes
- Tipo dos elementos de modelo: requisitos, comportamento, estrutura
- Elementos do modelo mais prováveis de mudar em conjunto ou co-evoluir
- Elementos do modelo organizados para suportar o reuso (model libraries)
- Algum critério de agrupamento lógico ou de coesão (baseado em possíveis partições do modelo)
- Combinação dos princípios anteriores

Organizando o pacote em hierarquias

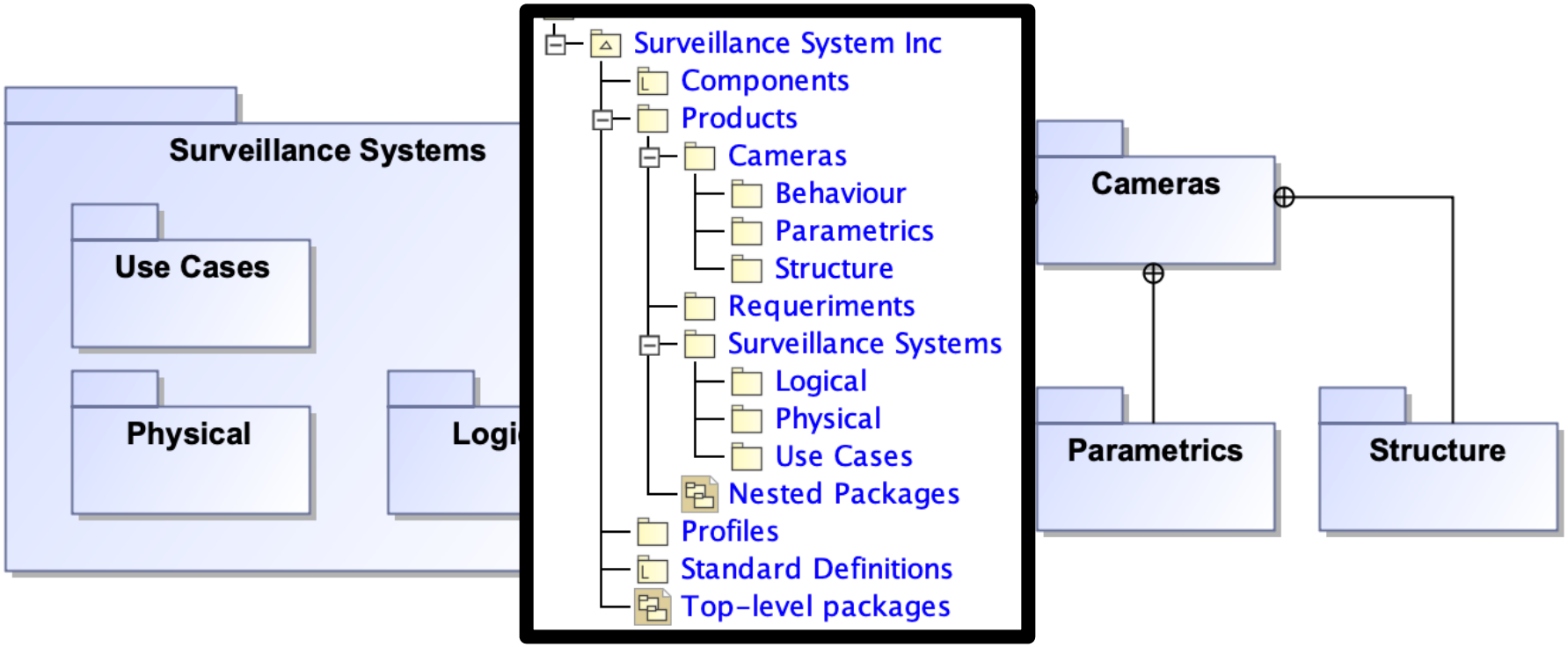
- Relacionamento de composição (*containment*)



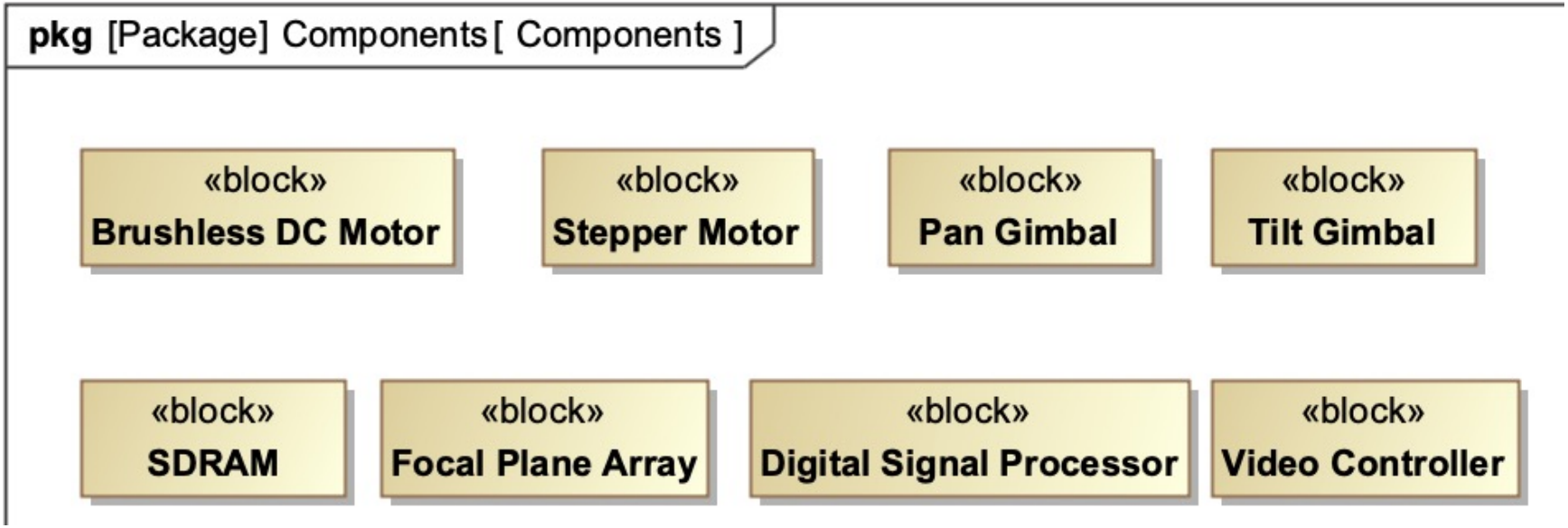


Organizando o pacote em hierarquias

- Relacionamento de composição (*containment*)

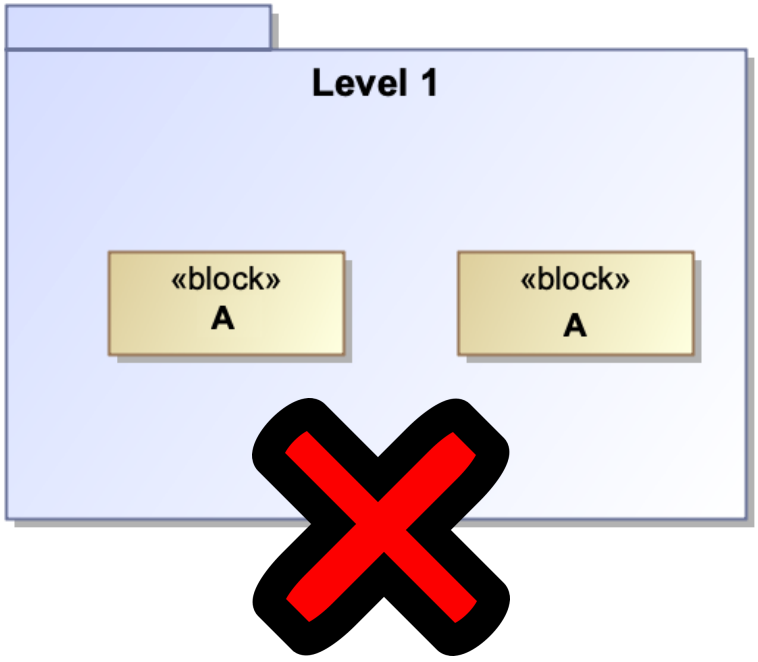


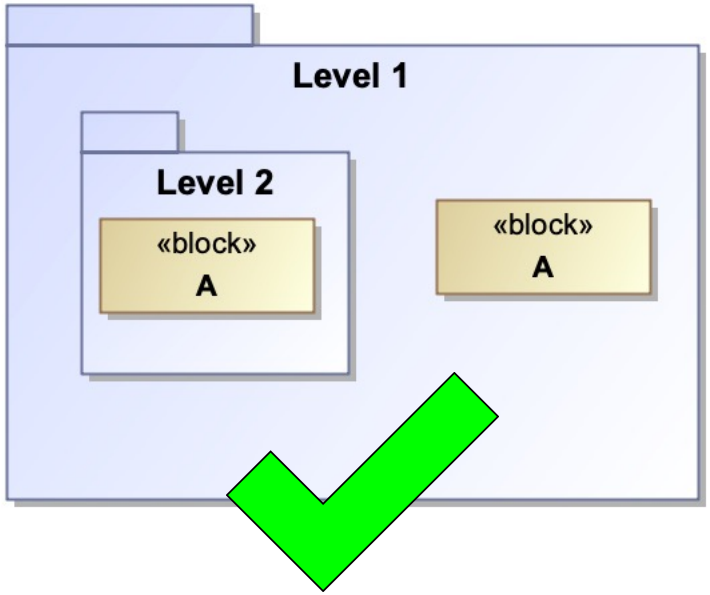
Exibindo elementos dentro de pacotes



Pacotes como resolução de nomes (*namespaces*)

- O nome do pacote server para identificar todos os elementos que estão contidos nele
- Cada elemento deve ter um nome único (considerando a hierarquia de pacotes em que ele esteja inserido)



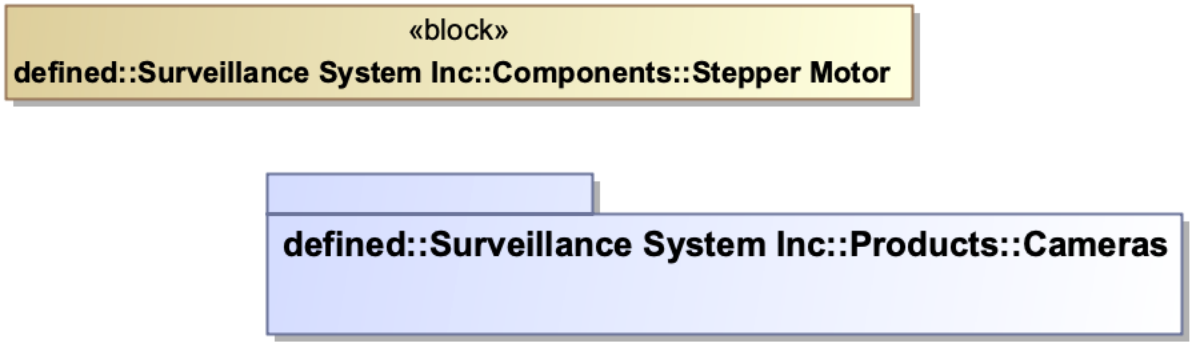


Nome qualificado

Level 1:: A
Level 1:: Level 2:: A

Pacotes como resolução de nomes (*namespaces*)

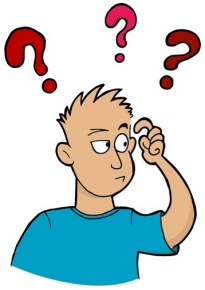
- O nome do pacote server para identificar todos os elementos que estão contidos nele
- Cada elemento deve ter um nome único (considerando a hierarquia de pacotes em que ele esteja inserido)



Importando elementos em pacotes

- As vezes um diagrama precisa referenciar elementos de diversos outros pacotes. Usar nome qualificado para todos não é a melhor alternativa
- Relacionamento *import* traz um elemento ou coleção de elementos de um *namespace* origem para outro
- Elementos importados não precisam ser identificados pelo nome qualificado
- Dois tipos de *import:*
  - *Package import:* importa todos os elementos do pacote
  - *Element import:* importa somente um elemento de um pacote

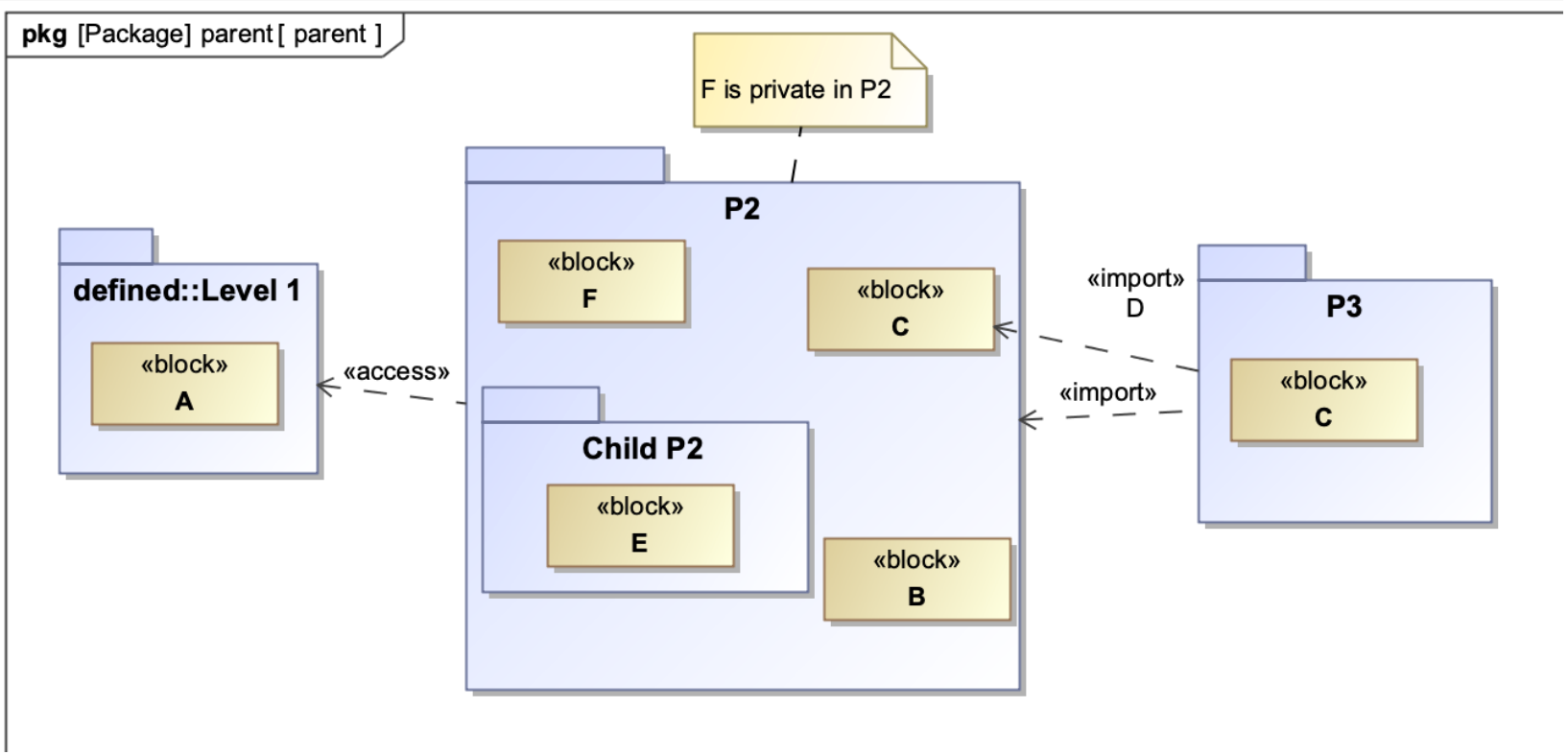
- Qual a motivação para os dois tipos?



Importando elementos em pacotes

- Todos os elementos com nomes dentro de um *namespace,* importados ou não, são chamados de **membros**
- **Membros** têm visibilidade: **pública** (padrão) ou **privada**
- Uma importação de pacote só traz os elementos públicos
- No entanto, podemos informar no relacionamento de importação se os nomes importados do pacote origem devem ser públicos ou privados dentro do pacote destino

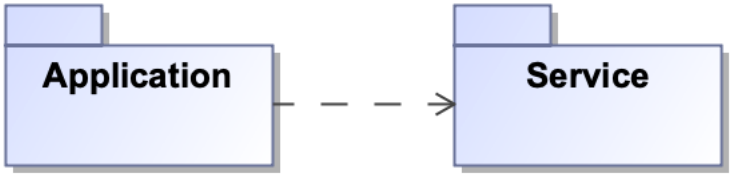
Importando elementos em pacotes



- O que está acontecendo entre os pacotes acima?

Dependências entre elementos do pacote

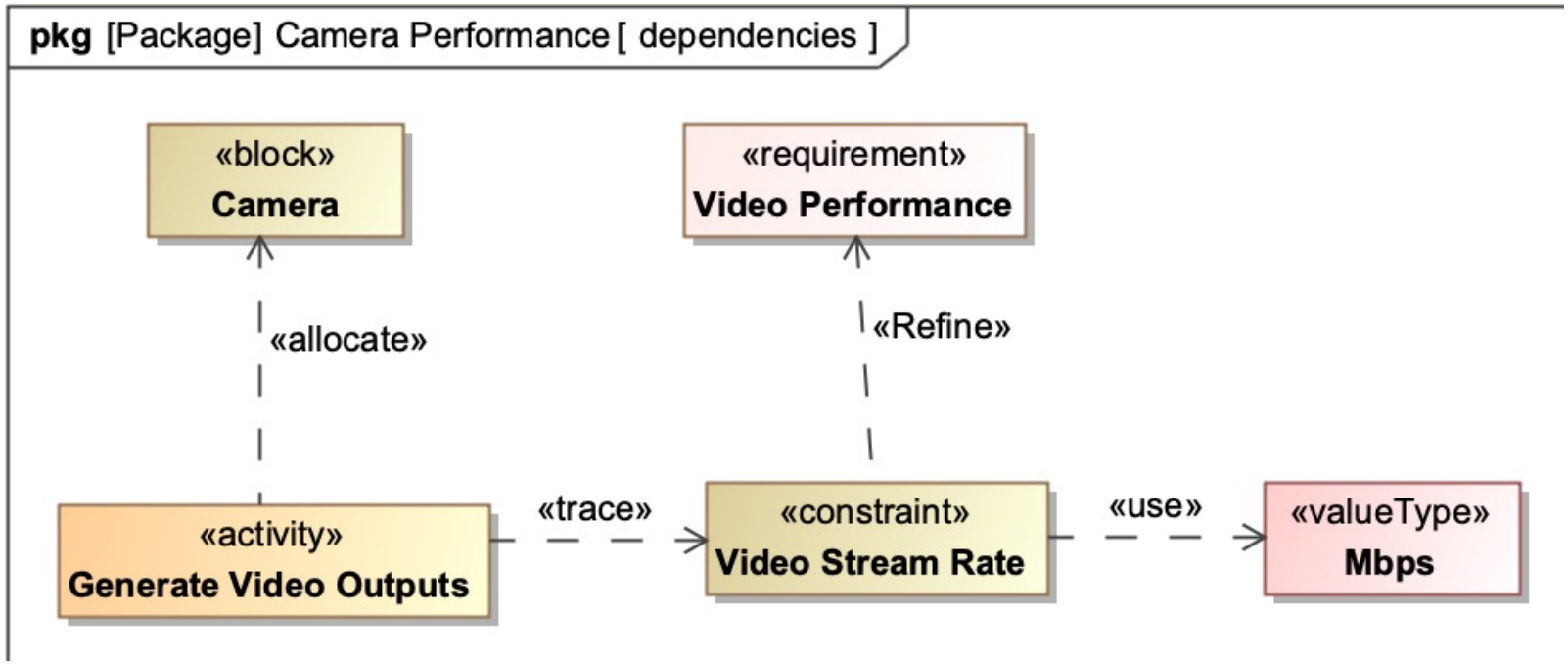
- Uma **dependência** é um relacionamento para indicar que uma mudança no elemento destino afeta o elemento origem
- A **dependência** entre pacotes indica que o conteúdo de um pacote depende do conteúdo do outro
  - Ex: um pacote que ilustra a camada de aplicação de um software depende do pacote da camada de serviços



Dependências entre elementos do pacote

- Uma **dependência** é um relacionamento mais abstrato (genérico) e pode ser substituído por relacionamentos mais concretos ao longo da evolução dos modelos
- Outros tipos de dependência são:
  - **Use:** origem usa destino
  - **Refine:** quando a origem for mais detalhada em termos do elemento destino
  - **Realization:** quando a origem realiza a especificação descrita no elemento destino. Por exemplo, uma implementação que realiza uma interface.
  - **Trace:** Quando queremos indicar rastreabilidade entre elementos. Geralmente usada no contexto de requisitos
  - **Allocate:** um elemento origem é alocado no elemento destino

Dependências entre elementos do pacote



Exercício

1. Crie a seguinte estrutura em pacotes no pacote “ex01”:
   1. Um pacote P1 que contém três blocos B1, B2 e B3 todos com visibilidade pública, e um pacote P4 com visibilidade privada
   2. Um pacote P2 que contém um pacote B1 e dois blocos B2 e B4. O pacote P2 importa com visibilidade pública o pacote P1
   3. Ilustre todos os elementos de P2 em um diagrama de pacotes mostrando seus nomes qualificados
2. Organize os diferentes elementos listados no pacote “ex02” em uma estrutura de pacotes. Em seguida, crie um diagrama de pacotes que ilustra esta estrutura. Por último crie um diagrama em pacotes que ilustra possíveis relacionamentos entre os pacotes criados.
3. Crie um diagrama de pacotes para organizar o domínio automobilístico. Quais pacotes você definiria? Como eles estariam relacionados?

Centro de Informática UFPE

Universidade Federal de Pernambuco

Organizando elementos SysML em Pacotes

Curso Ford

Prof. Lucas Albertins