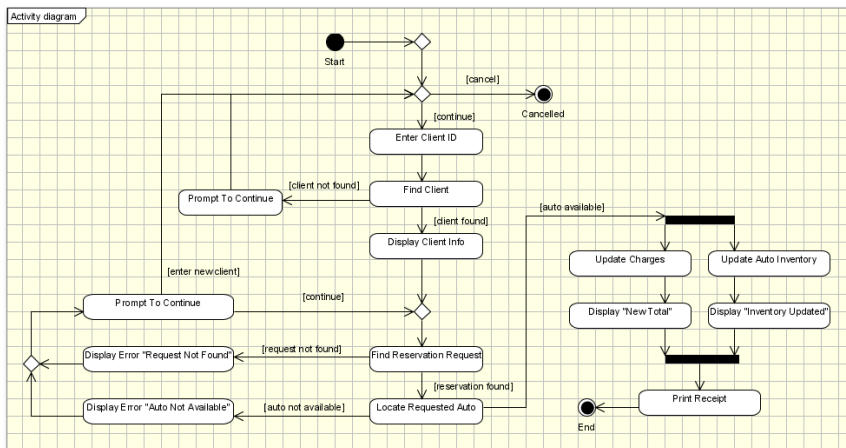


Modelando comportamento baseado em fluxo com Atividades



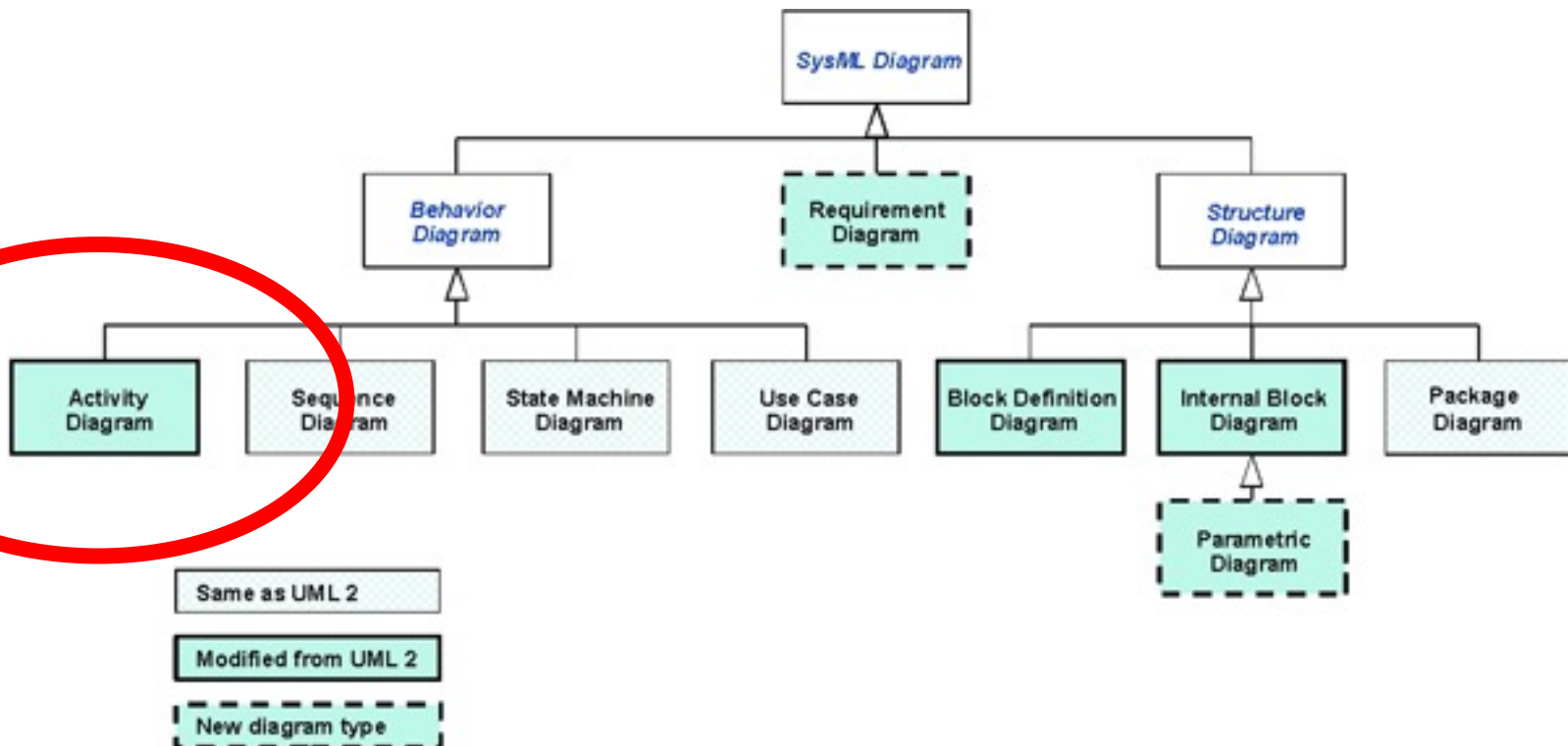
Curso Ford
Prof. Lucas Albertins

Agenda

- ☐ Por que atividades?
- ☐ Visão geral sobre Diagrama de Atividades
- ☐ Fluxos
- ☐ Tipos de Nós
 - ☐ Ação
 - ☐ Controle
 - ☐ Objeto
- ☐ Partições/Raias (*Partitions/Swimlanes*)
- ☐ Relacionando atividades com blocos
- ☐ Uso de diagrama de atividades
- ☐ Conclusões

SysML

■ UML Profile for System Engineering



Por que atividades?

- Mecanismo para descrever fluxos de tarefas
- Necessidade de transformar entradas em saídas
- São semelhantes aos antigos fluxogramas com recursos mais elaborados
- SysML permite relacionar atividades aos blocos

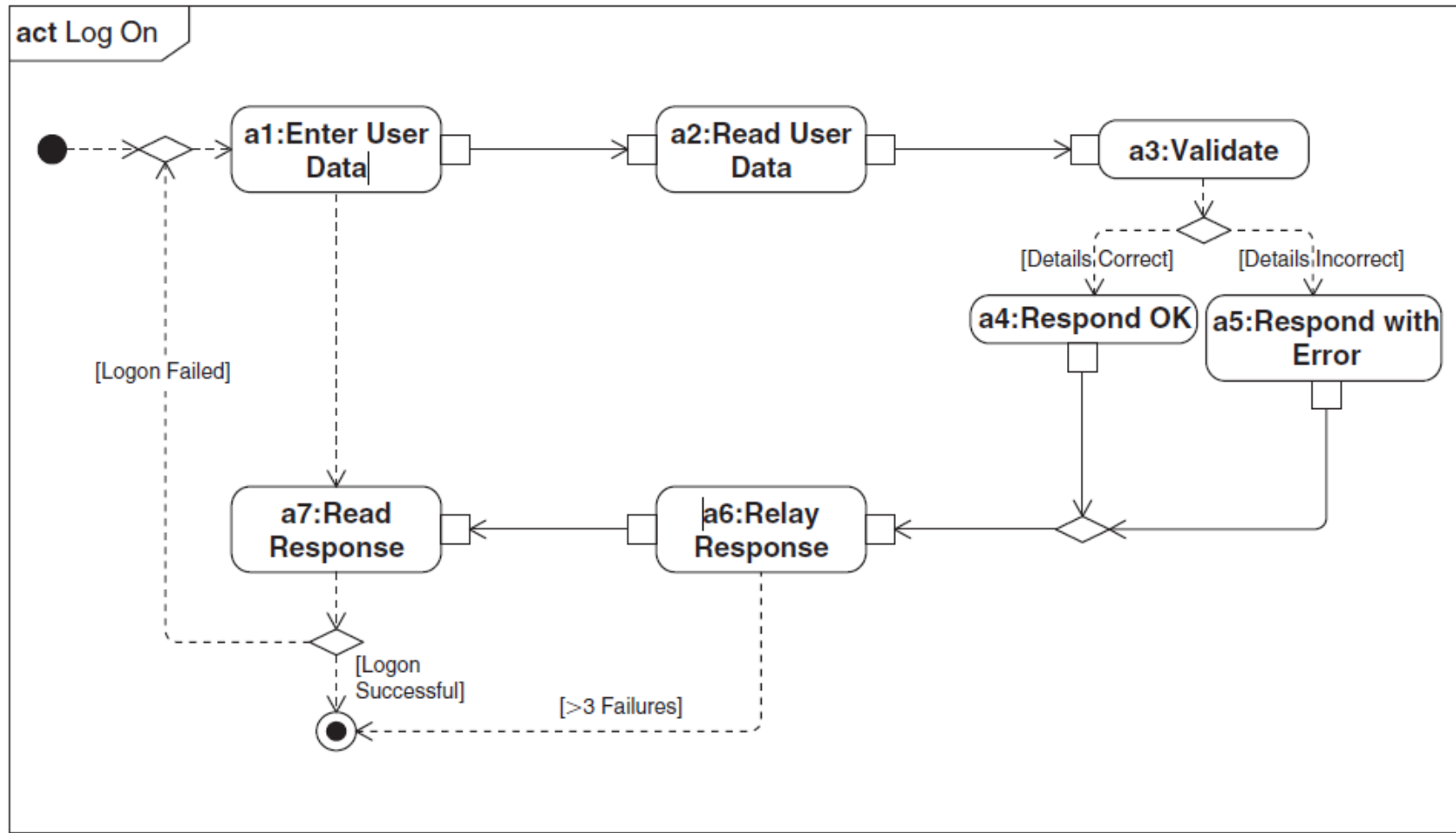
Principais usos de atividades

- Comportamentos envolvendo diferentes blocos
- Comportamento principal de um único bloco
- Como uma operação deve ser realizada
- Ações de uma máquina de estado
- Fluxos de ações relevantes do sistema

Principais elementos de uma atividade

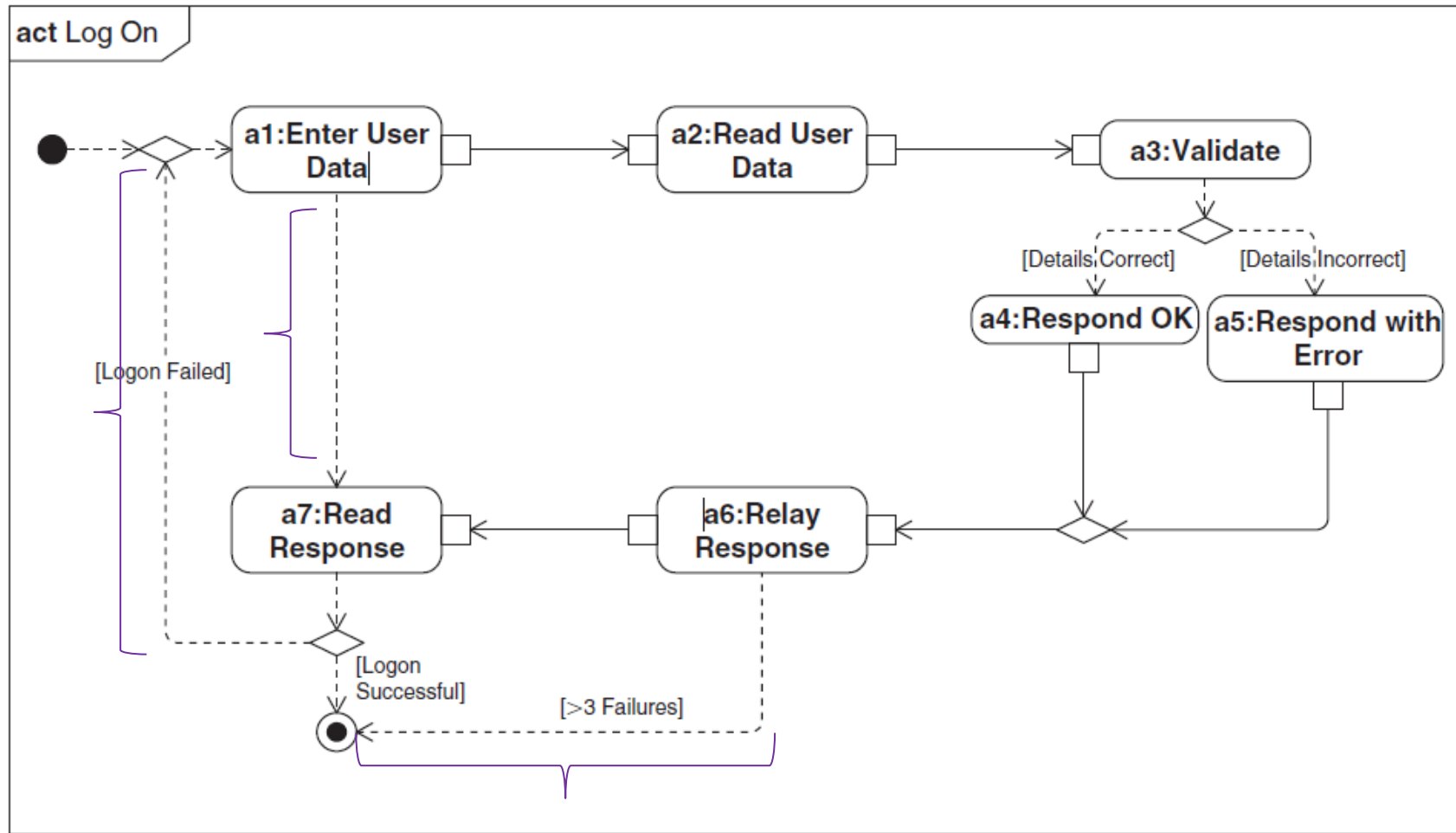
- Fluxo
 - Controle
 - Objeto
- Nós
 - Ação
 - Controle
 - Objeto

Overview Diagrama de Atividades



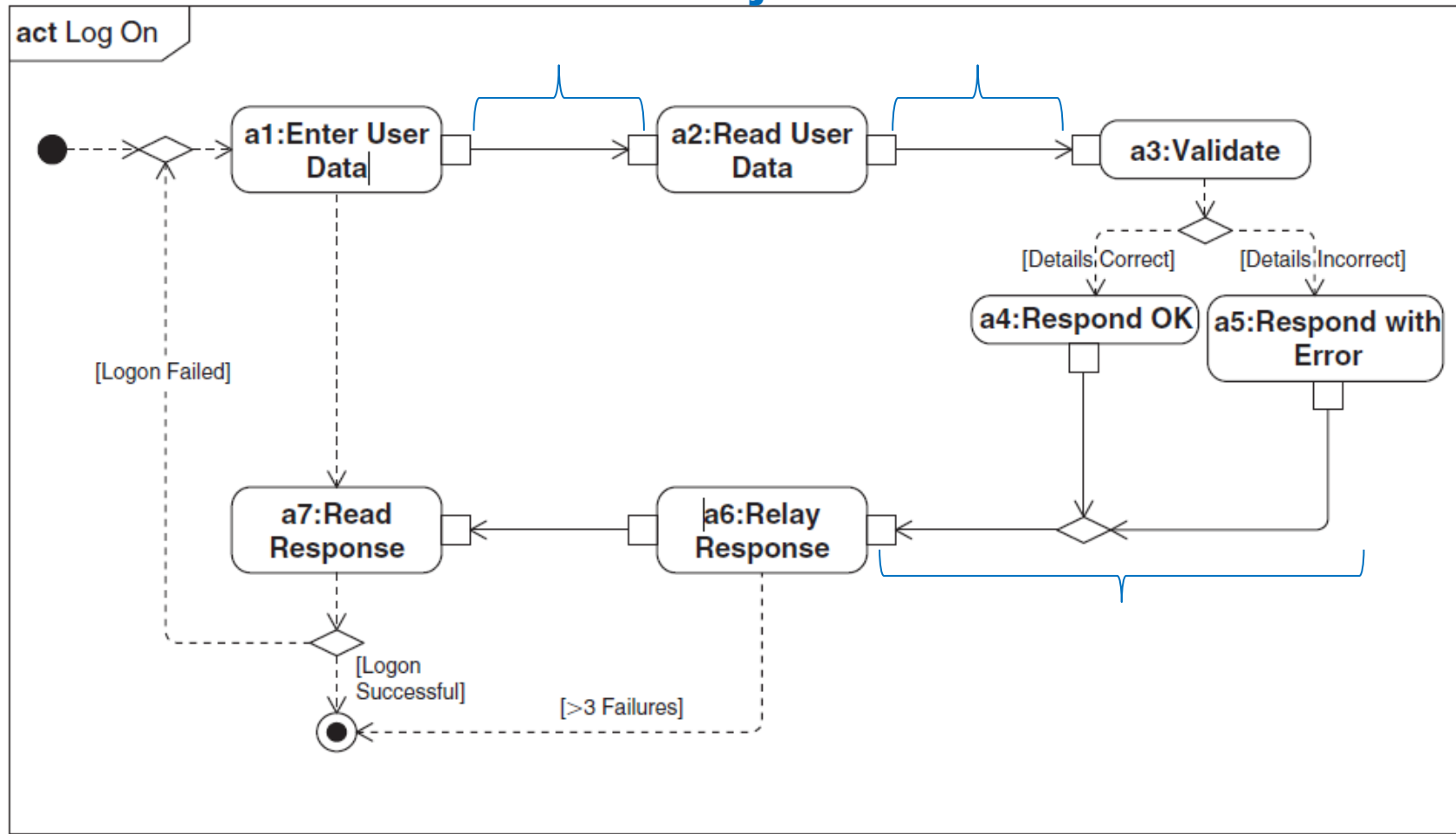
Overview Diagrama de Atividades

Fluxo de Controle



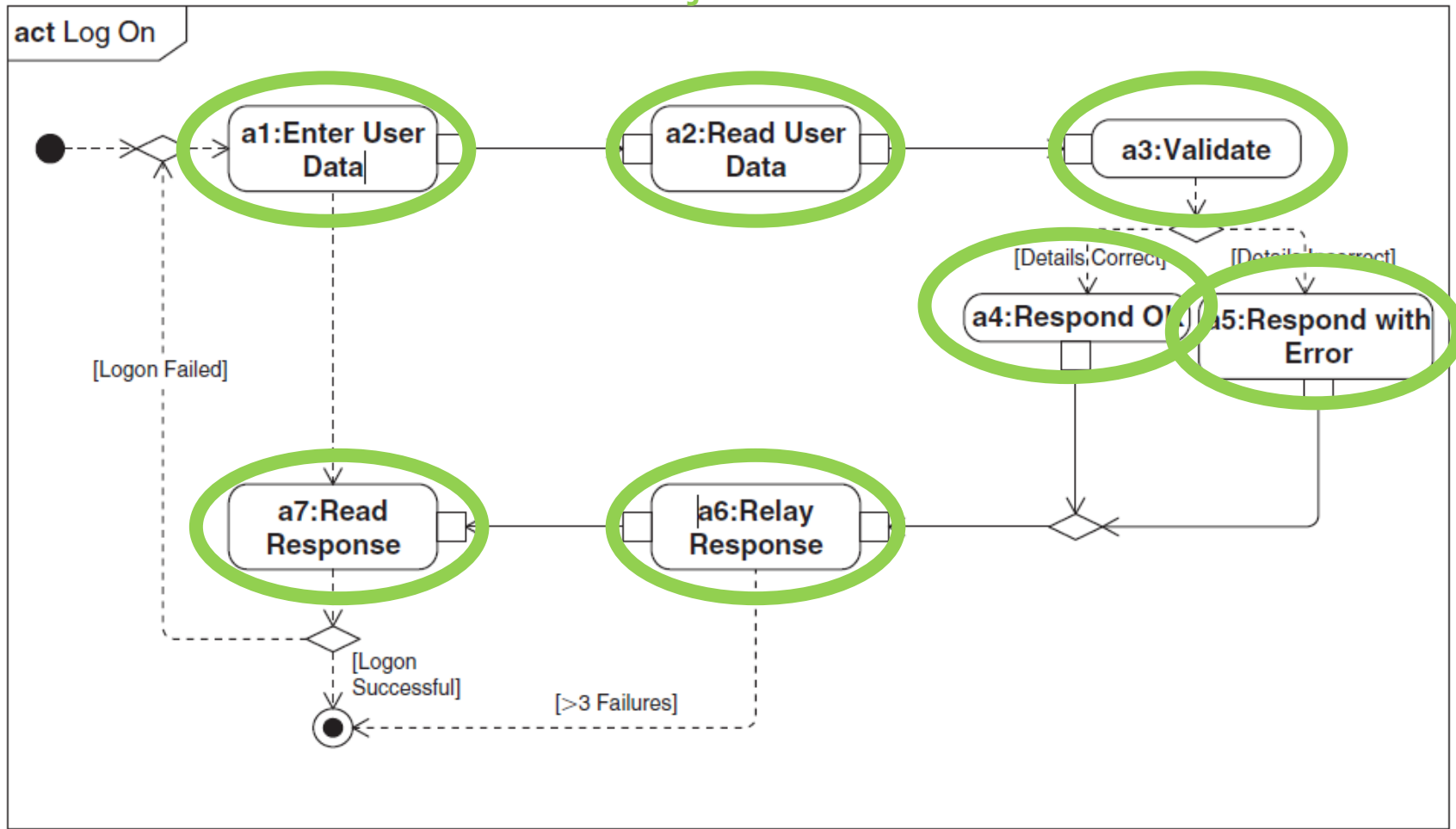
Overview Diagrama de Atividades

Fluxo de Objeto



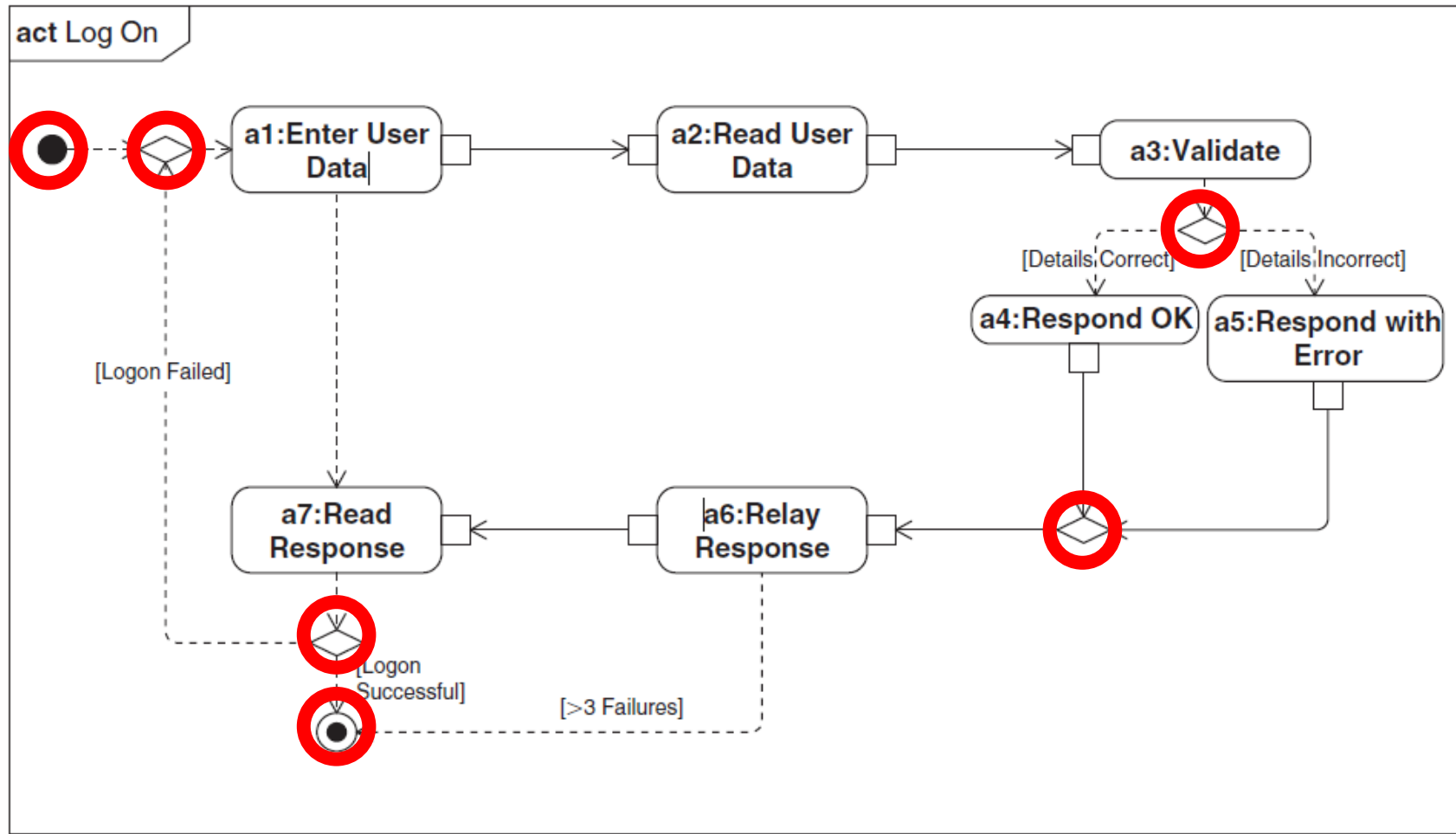
Overview Diagrama de Atividades

Nós de Ação



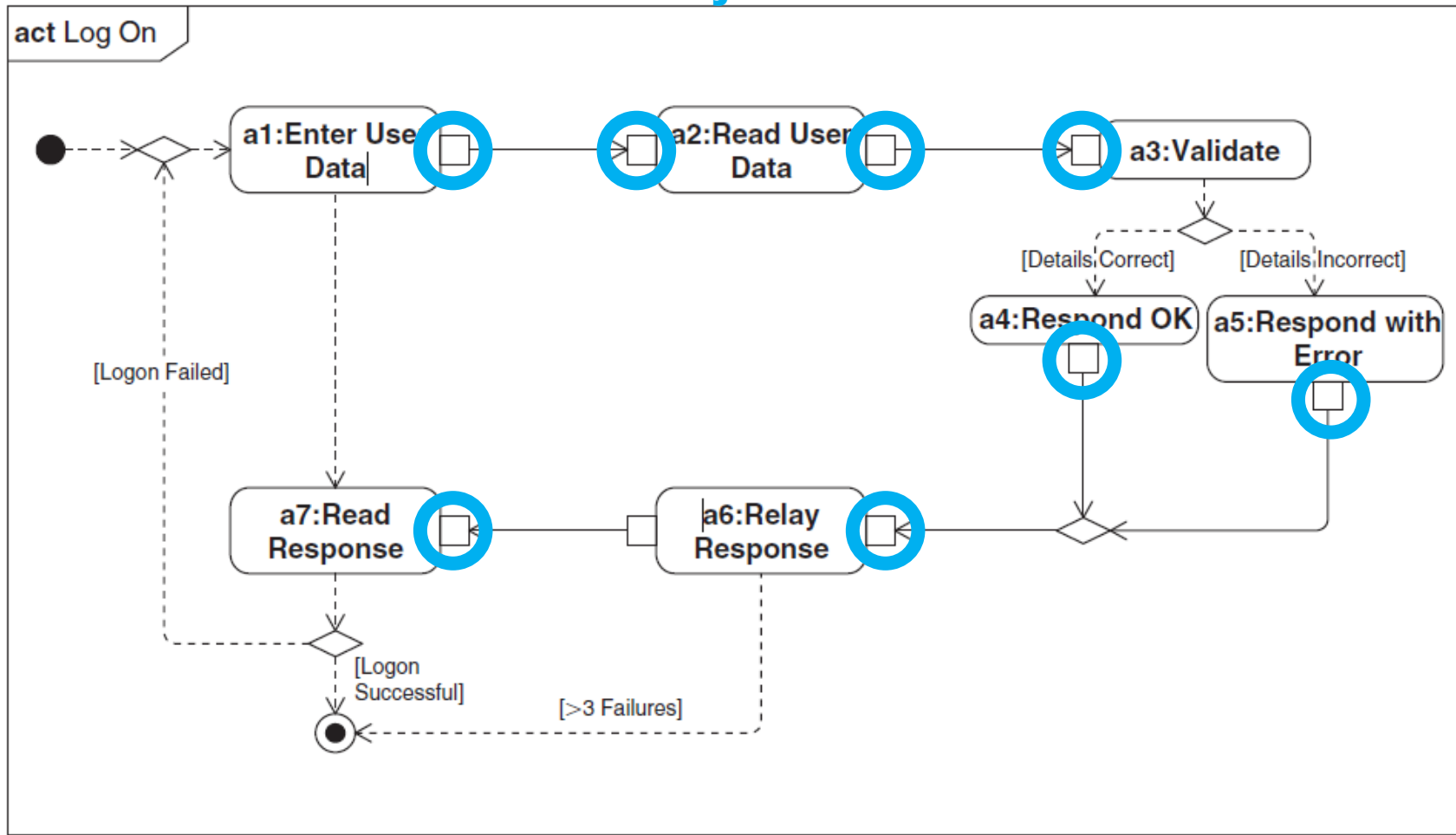
Overview Diagrama de Atividades

Nós de Controle

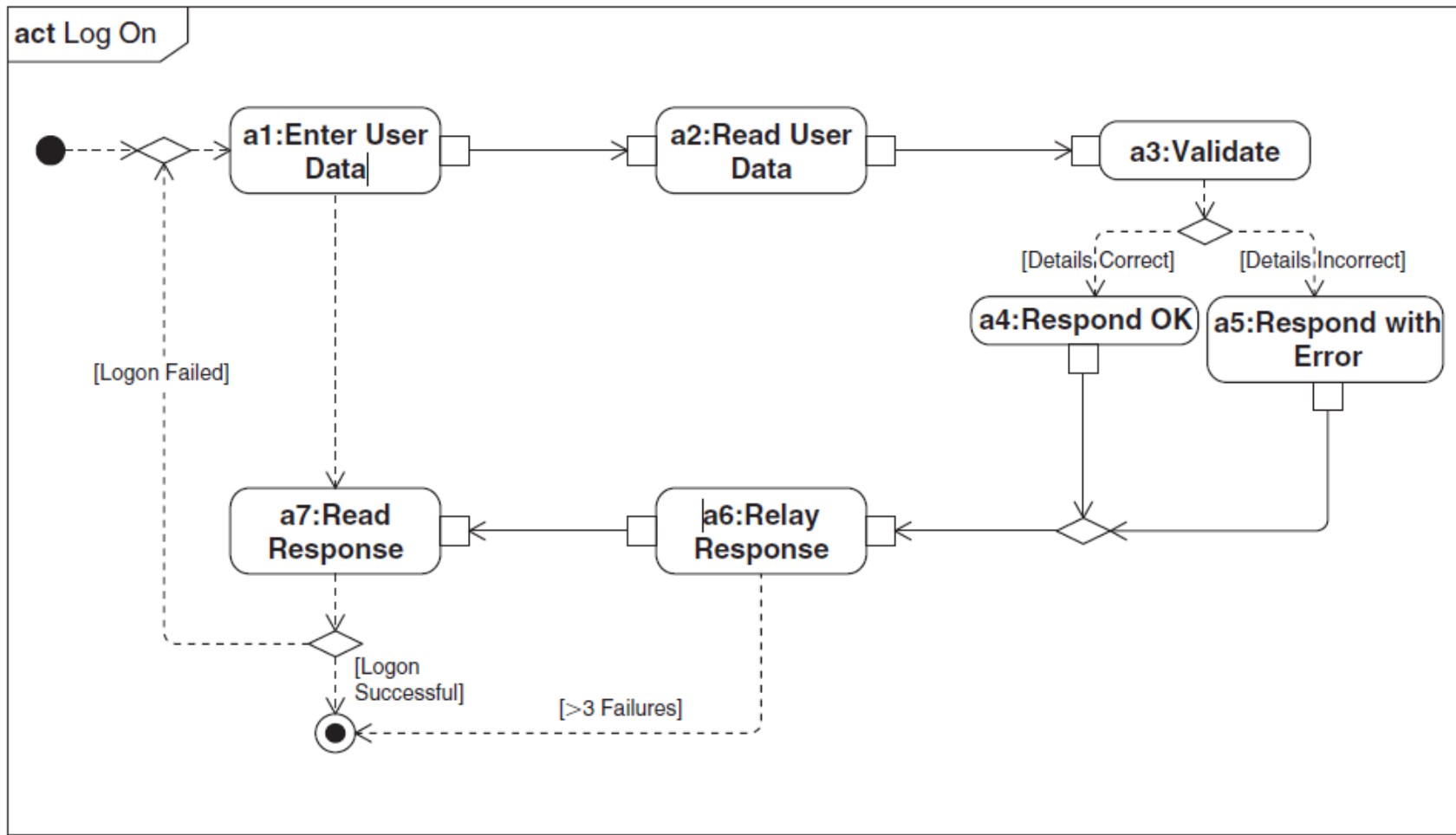


Overview Diagrama de Atividades

Nós de Objeto

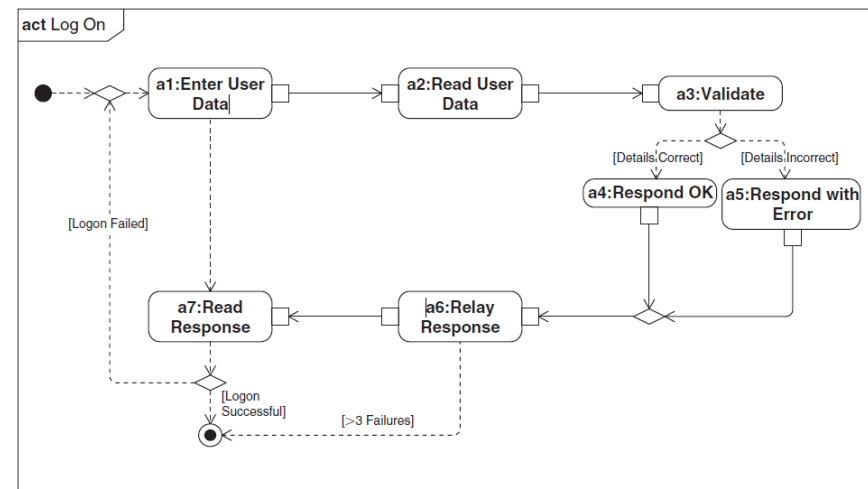


Modelando no MD



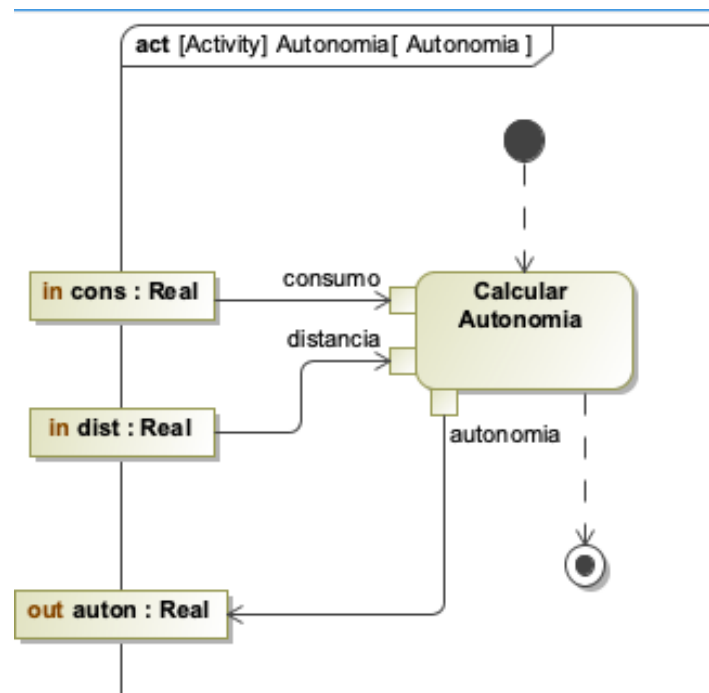
Semântica de uma atividade

- Tokens fluindo ao longo dos fluxos e nós
- Podem ser tokens de controle ou de objeto
- Baseado na semântica de Redes de Petri
- Alguns nós geram, outros consomem tokens, outros fazem ambos
- O token pode ser visto como uma representação de um fluxo ativo dentro da atividade



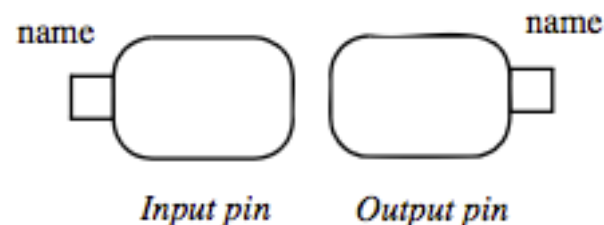
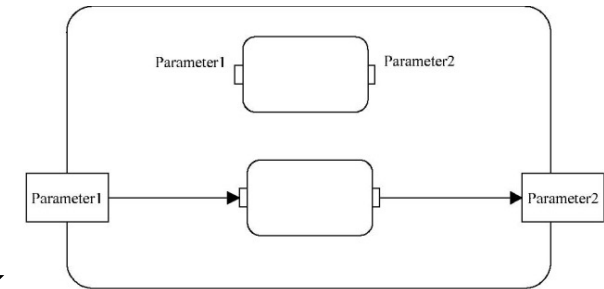
Fluxo de Objeto

- Usados para rotear tokens de objeto que representam informação ou elementos físicos fluindo entre nós de objeto
- Principais nós de objeto são: Parâmetros e Pinos
- Direção do fluxo deve ser compatível do nó de objeto (entrada ou saída) da mesma forma que o tipo do objeto trafegado no fluxo



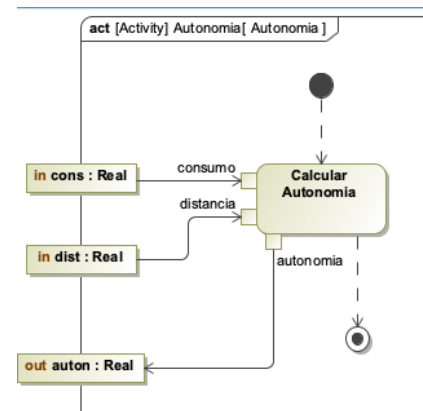
Object Nodes

- Activity Parameters
 - Podem ser de entrada ou saída
 - Devem possuir tipo: Value Type ou Block
 - Multiplicidade determina quantos elementos podem ser consumidos como entrada no início da atividade ou produzidos como saída no final da atividade
 - Propriedade <<stream>> - Pode ficar recebendo entradas ou produzindo saídas (dependendo da direção) enquanto a atividade estiver em execução
- Pins
 - Elementos para representar tráfego de dados em ações
 - Possuem tipos e multiplicidade como os parâmetros



Fluxos de Controle

- Apesar do fluxo de dados impor ordem, as vezes precisamos de regras para controlar fluxo não relacionadas aos objetos
- Fluxo de controle é usado para organizar ordem de execução dos diversos nós através de tokens de controle
- Fluxos de controle podem ser bastante elaborados com o uso de nós de controle
- Nós de controle podem rotear tanto fluxo de controle como fluxo de dados
- Representação gráfica do fluxo de controle é através de arestas tracejadas



Nós de Controle

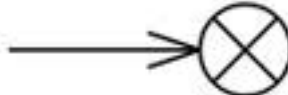
- Existem 7 tipos de nós: alguns com o mesmo elemento gráfico

- InitialNode 

- Quando a atividade inicia, coloca um token é gerado por este nó

- Activity Final 

- Quando um token chega neste nó, ele é consumido e toda a execução da atividade termina

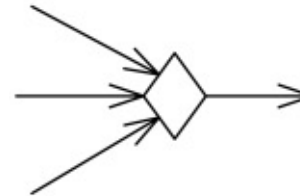
- FlowFinal 

- Tokens que chegam nesse nó são consumidos, mas nenhum efeito é gerado na atividade

Nós de Controle

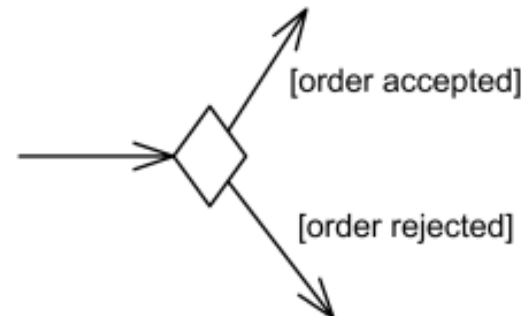
■ MergeNode

- Uma aresta de saída e várias de entrada. Repassa tokens das arestas de entrada na aresta de saída a medida que eles chegam



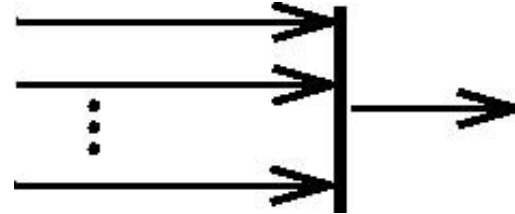
■ DecisionNode

- Uma aresta de entrada e várias de saída. A decisão sobre qual aresta de saída acontece de acordo com as guardas
- Se mais de uma guarda for verdadeira, a decisão sobre qual deve receber o token é não-determinística
- Pode ter uma guarda “else”



Nós de Controle

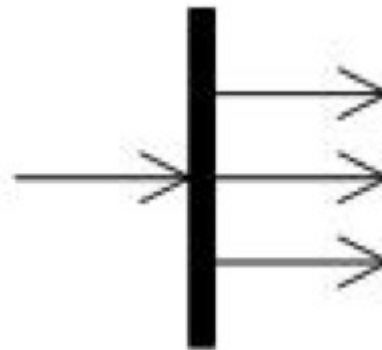
■ JoinNode



- Uma aresta de saída e várias de entrada. A saída só acontece quando todas as entradas são recebidas (sincronização)

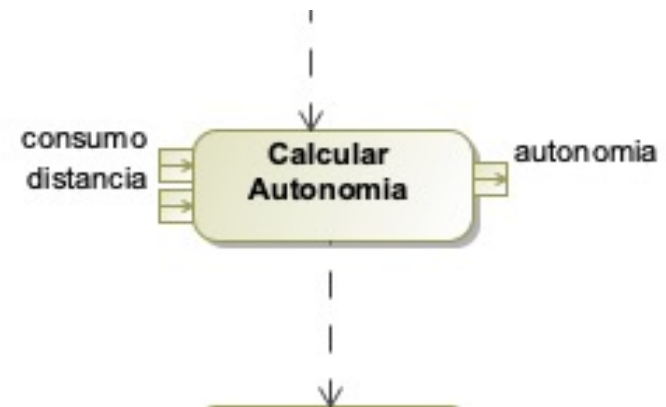
■ ForkNode

- Recebe um token na aresta de entrada e replica-o em cada uma das arestas de saída (gera fluxos concorrentes)



Actions (Ações)

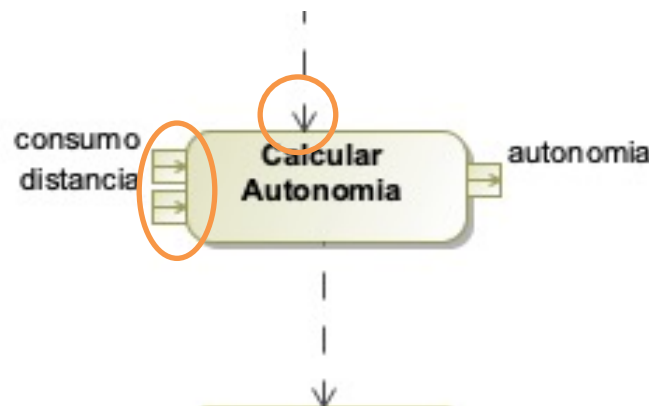
- Elemento mais básico (atômico) para descrição de comportamento
- Podem transformar entradas em saídas através do uso de pinos (pins)
- Símbolo depende do tipo de ação



- Podem ser usados para invocar outras atividades criando uma hierarquia de execução

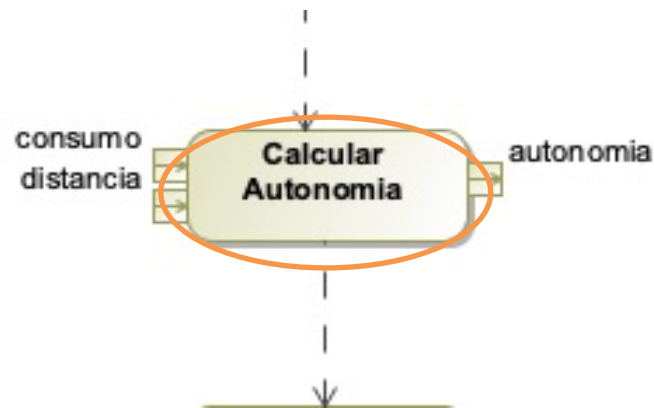
Actions (Ações)

- Semântica geral de execução
 - A atividade dona da ação deve estar em execução
 - O número de tokens em todos os pinos de entrada é igual ou superior ao limite inferior da multiplicidade
 - Um token está disponível em cada aresta de controle de entrada



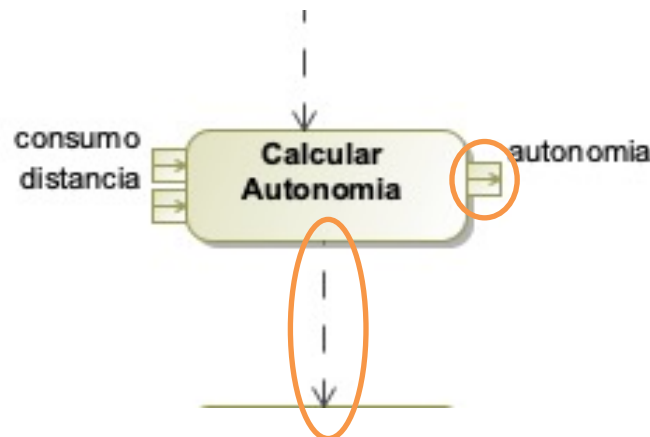
Actions (Ações)

- Semântica geral de execução
 - Em seguida, todos os tokens de entrada são consumidos e o comportamento da ação é executado



Actions (Ações)

- Semântica geral de execução
 - Ao término do seu comportamento, a ação deve disponibilizar tokens nos pinos de saída iguais ou superiores ao limite inferior da multiplicidade dos pinos
 - Também disponibilizar tokens nas suas arestas de controle de saída
 - **OBS:** caso a atividade dona da ação termine, as ações também devem terminar sua execução



Actions (tipos de ações)

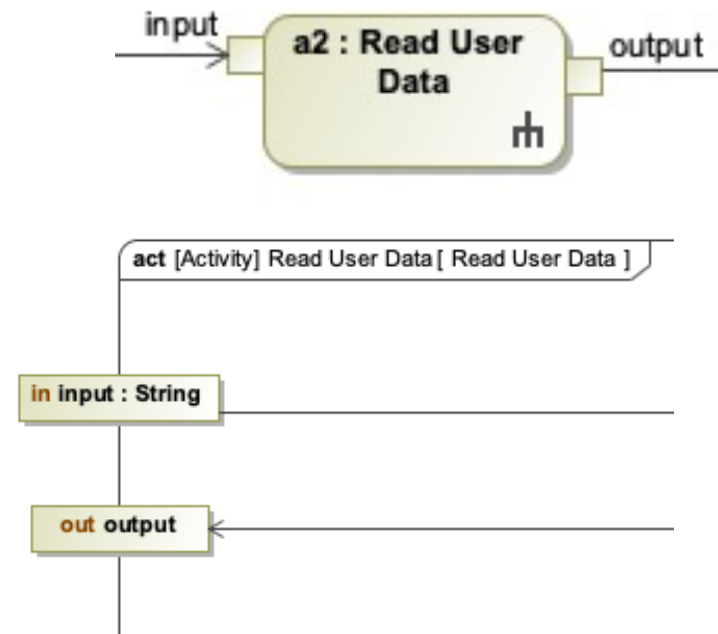
- CallOperationAction
 - Invoca uma operação de um bloco
- OpaqueAction
 - Comportamento descrito utilizando uma linguagem externa a SysML
- SendSignalAction
 - Envio de um sinal (comunicação assíncrona)
- AcceptEventAction
 - Espera por um evento acontecer
- CallBehaviourAction
 - Invoca o comportamento de uma outra atividade
- ValueSpecificationAction
 - Atribui ao pino de saída o valor descrito na ação

Exercício

- Crie uma atividade para representar o comportamento do veículo do momento que ele é ligado até o momento que o motor gera potência. Considere comportamentos como ligar o veículo, passar marcha, e o motor fornecer potência. Lembre-se que a sua atividade deve gerar uma saída relacionada ao torque que é gerado pelo motor, e que para gerar esta saída o motor deve receber como entrada o ar que vem de fora do veículo. Use somente os construtores apresentados até o momento.

Compondo Atividades usando Call Behavior Actions

- Permite que uma atividade invoque outra atividade através de um tipo específico de ação chamada **Call Behavior Action**
- Pinos desta ação devem corresponder aos parâmetros da atividade que ela representa
- Úteis para modularizar comportamentos complexos
- Cria-se uma hierarquia de invocações
- Uma mesma atividade pode ser invocada diversas vezes e de forma concorrente

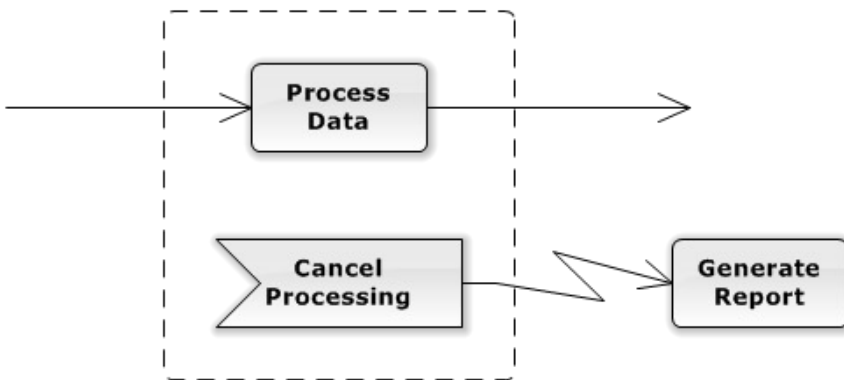


Usando sinais e eventos

- Uma atividade pode esperar por a chegada de um sinal ou evento através de **Accept Event Actions**
- Envio de sinais se dá pela ação **Send Signal Action**
- Atividades (do mesmo bloco ou blocos distintos) podem se comunicar de forma assíncrona por meio de troca de sinais através do uso destas ações
- **Accept Event Actions** podem aceitar outros tipos de eventos (time event e change event)aci
 - Podem não ter nenhuma aresta de entrada
- Os sinais de ambas estas ações podem trafegar através de portas
- Sinais também pode ser enviados e recebidos através de máquinas de estado

Regiões de Interrupção (*Interruptible Regions*)

- Agrupa um conjunto de ações que podem ter suas execuções interrompidas através de uma **aresta de interrupção**
- A **aresta de interrupção** tem a origem dentro da **região de interrupção** e destino fora dela. Ela pode ser tanto de controle quanto de objeto



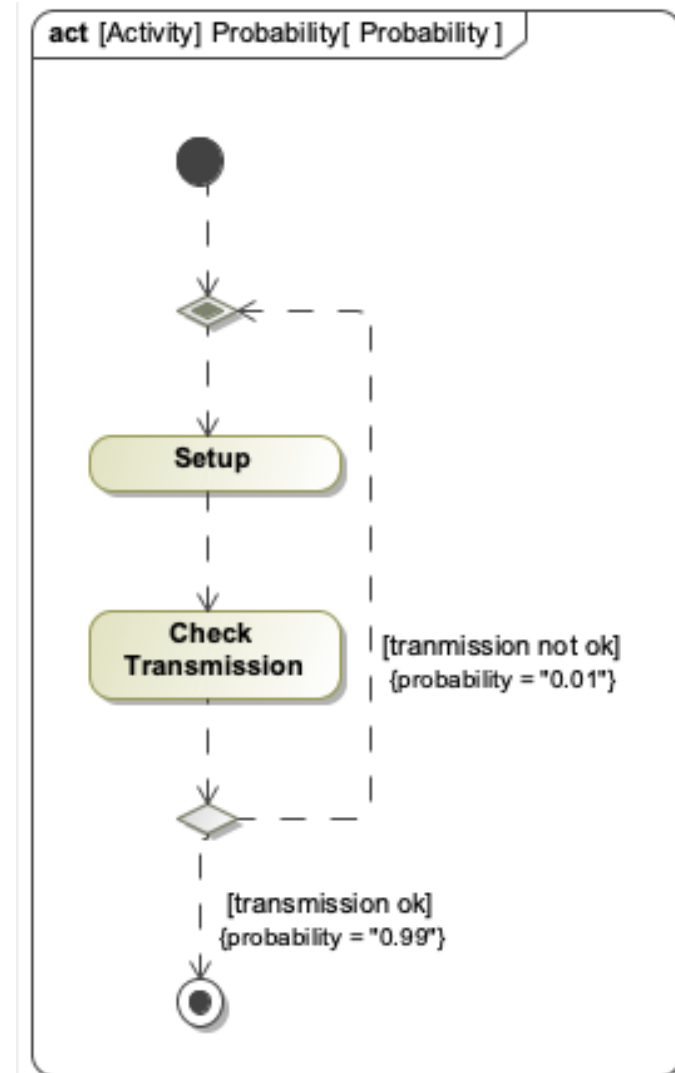
- A região é interrompida quando um token chega na **aresta de interrupção**
- Geralmente a origem da aresta de interrupção é um Accept Event Action que recebe um sinal indicando que a região deve ser interrompida
- Bastante útil para representar tratamento de erros (*exception handlers*)

Nós estruturados

- Permite a construção de fluxos de controle fixos agrupando um conjunto de nós
- Geralmente utilizados quando aquele fluxo pode ser reusado
- Podem definir variáveis e possuir pinos
- Três tipos de nó:
 - **Sequence node:** Executa as ações em sequência
 - **Conditional Node:** Grupos de ações que são executados de acordo com as condições de teste (cláusulas)
 - Ordem pode ser definida para garantir que só uma cláusula seja executada
 - **Loop Node:** Permite a repetição de um grupo de ações enquanto uma condição for satisfeita
 - Possui **setup**, **test** e **body**
 - Semelhante aos comandos de repetição de linguagens de programação

Probabilidades em atividades

- Arestas podem ter marcações de probabilidades
- Indicam a probabilidade de um token passar pela aresta
- Geralmente são usadas em arestas de saída de Decision Nodes, mas podem ser usadas em ações ou outros elementos
- No Magic Draw pode ser utilizado para indicar probabilidade dos caminhos percorridos numa simulação



Exercício

- Atualize a sua atividade criada anteriormente considerando as seguintes características:
 - Defina a ação de fornecer potência como uma Call Behavior Action, e crie a atividade correspondente detalhando seu funcionamento. Por exemplo, o controle da mistura do ar que deve ser passado para o motor, a liberação de combustível, o controle da marcha, os quais devem acontecer antes do torque ser gerado, e ainda uma ação para amplificar o torque.
 - Faça com que a atividade possa ser interrompida a qualquer momento caso o sinal de desligar o carro seja recebido.

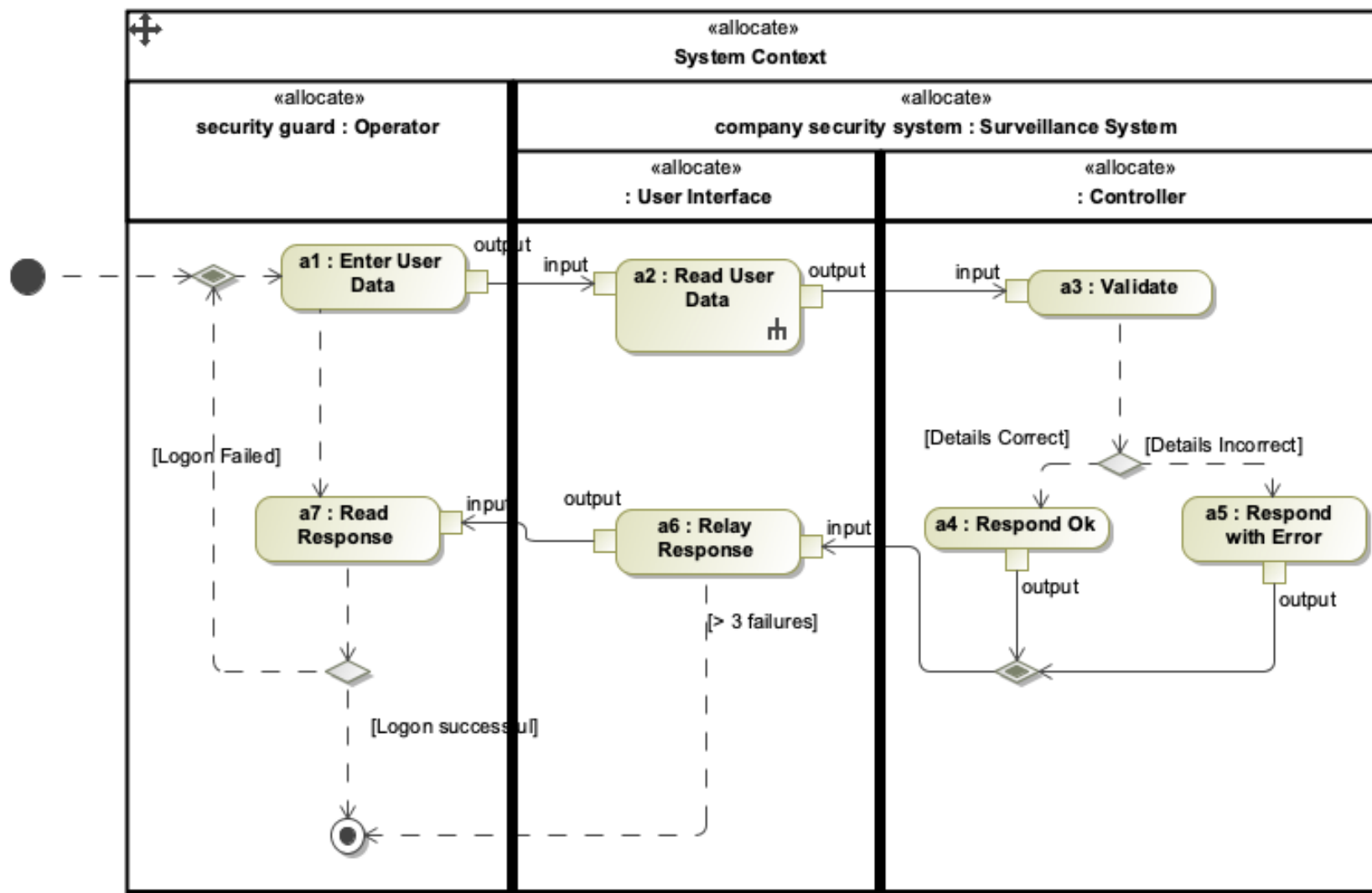
Relacionando atividades a blocos e outros comportamentos

- Três principais formas:
 - Partições
 - Atividade no contexto de um bloco
 - Atividades como comportamento do bloco
 - Atividades como métodos (operações)
 - Atividades com outros comportamentos
 - Geralmente máquinas de estado

Partições (Swimlanes)

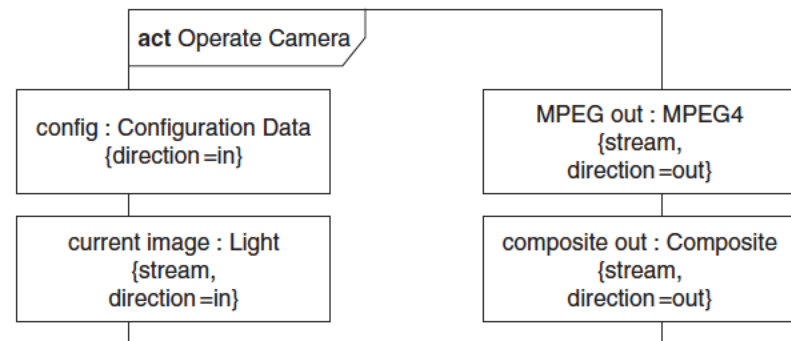
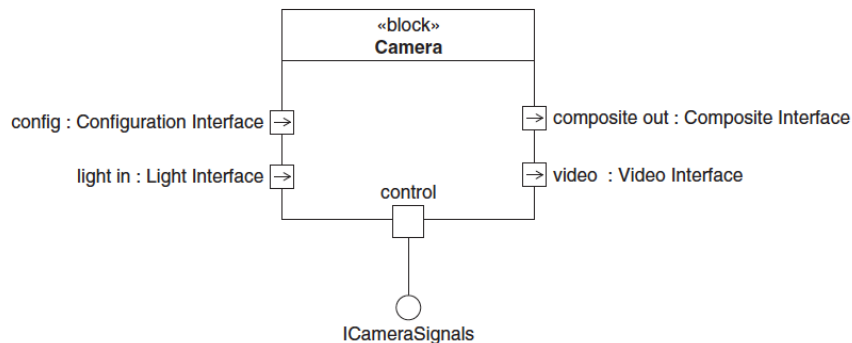
- Uma **partição** atribui responsabilidade (ou um papel) para a execução de um conjunto de nós
- Geralmente a responsabilidade é atribuída a blocos ou partes
- São ilustradas por retângulos que englobam o conjunto de nós dentro da partição e um cabeçalho indicando de quem é a responsabilidade (bloco, parte, ou só um texto livre)
- Podem ser alinhadas horizontalmente ou na verticalmente
- Também pode ter subpartições para representar uma decomposição mais detalhada
 - Por exemplo: um bloco na partição maior e suas partes como subpartições

act [Activity] Log on[Log on partition]



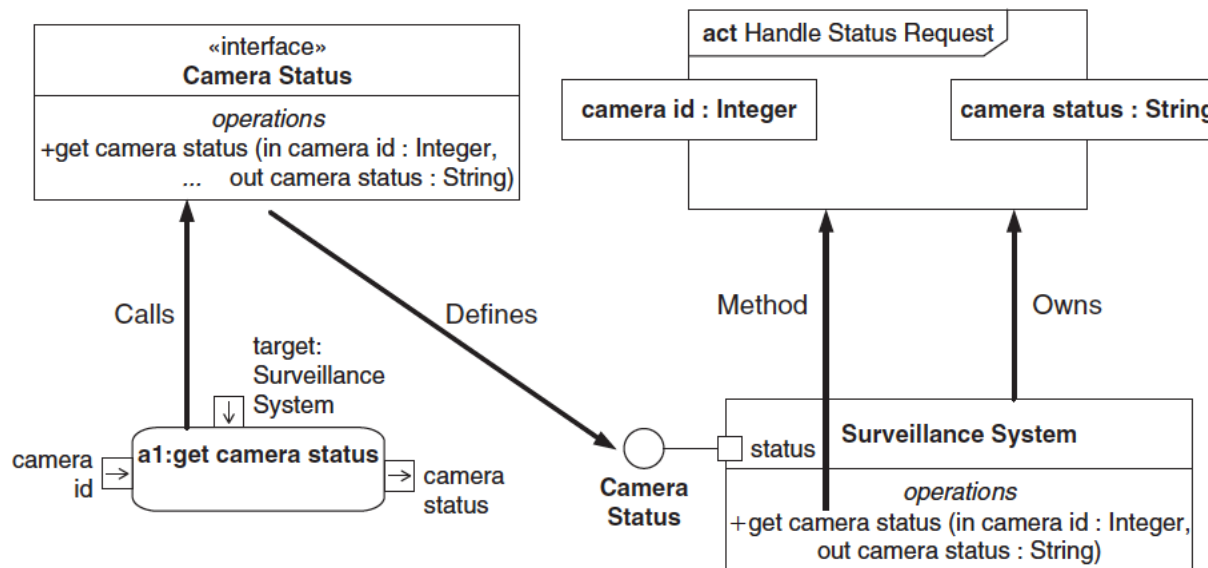
Atividade no contexto de um bloco

- Uma atividade pode representar o comportamento do bloco ou a implementação de um serviço do bloco (métodos)
- Atividade como comportamento do bloco
 - Parâmetros podem ser mapeados para flow properties do bloco ou suas portas (SysML não especifica como)
 - Sinais podem ser recebidos ou comunicados através das portas e expressados, por exemplo, em Accept Event Actions



Atividade no contexto de um bloco

- Atividade como método
 - Precisa ter mesma assinatura (parâmetros, tipos multiplicidades e direções) da operação
 - Operações síncronas vs requisições assíncronas
 - Call Operation Actions podem ser usadas para invocar operações (pinos casam com os parâmetros da operação)
 - Pino adicional para representar o elemento que vai receber a requisição

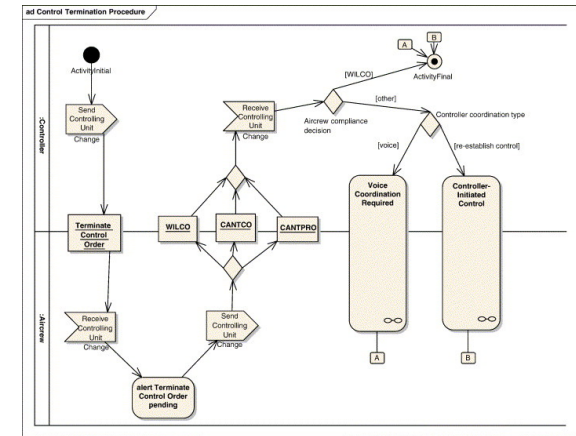


Atividades com outros comportamentos

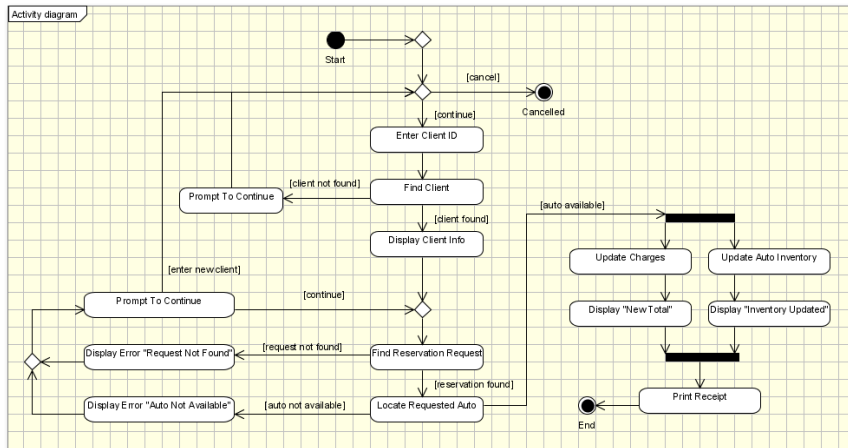
- Uma ação pode ser usada para invocar qualquer tipo de comportamento
- SysML não especifica como o comportamento deve ser disparado (e.g., uma ação disparar uma máquina de estado ou um diagrama de sequência)
- Situações em SM que geralmente disparam atividades:
 - Entrar num estado (entry action)
 - Sair de um estado (exit action)
 - Estar em um estado (do action)
 - Transição entre estado (efeito da transição)

Exercício

- Atualize mais uma vez as suas atividades atribuindo responsabilidades as ações que elas contém utilizando partições



Modelando comportamento baseado em fluxo com Atividades



Curso Ford
Prof. Lucas Albertins