

SysEng 6542

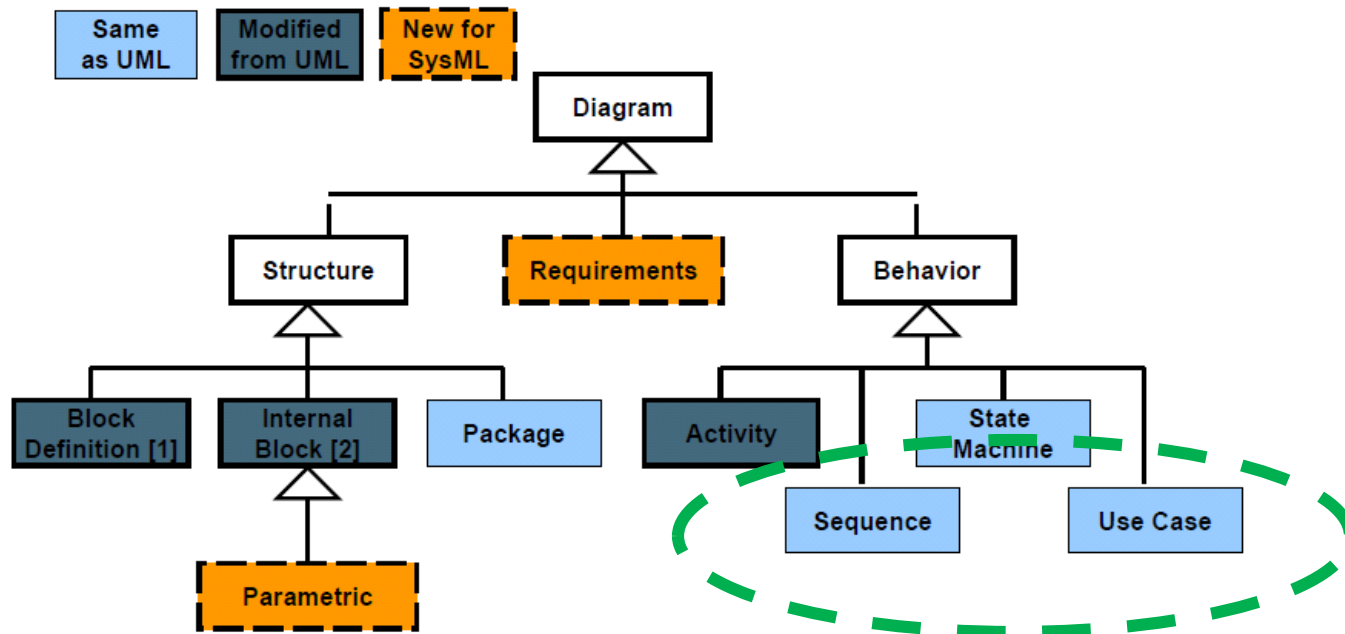
Model Based Systems Engineering

Modeling Behavior – Part 2

Dr Quoc Do

Scope

SysML Taxonomy of Diagrams



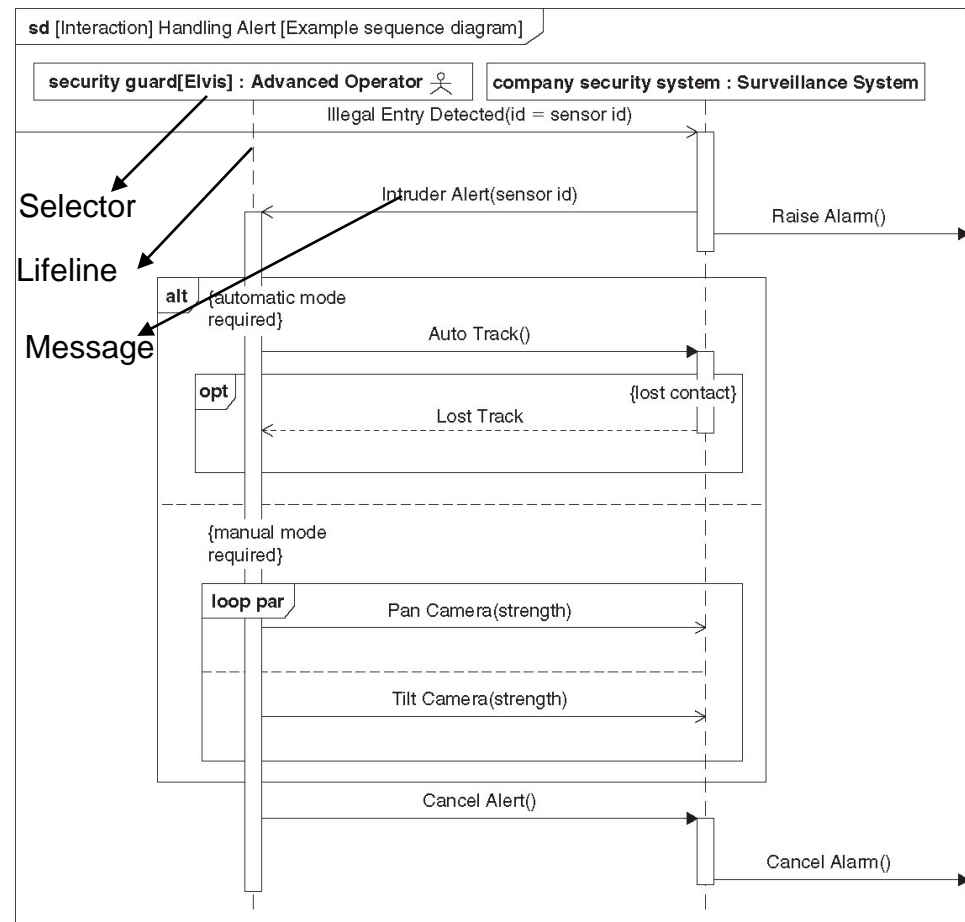
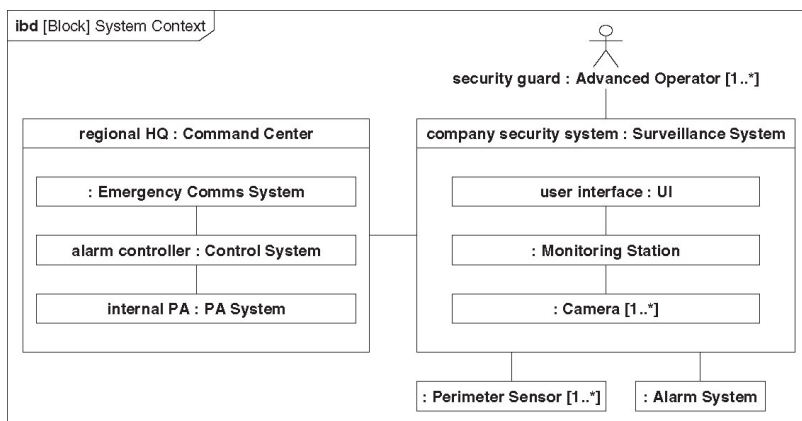
Sequence Diagram

- UML Behavior Diagram
- Represents Message-based behavior and interaction between system components and any externals (actors, environment, etc.)
- Only represent interactions, so no model element type necessary

Sequence Diagram

An example

- Interactions take place in the context of a block. The example show interaction within a *System Context* block.

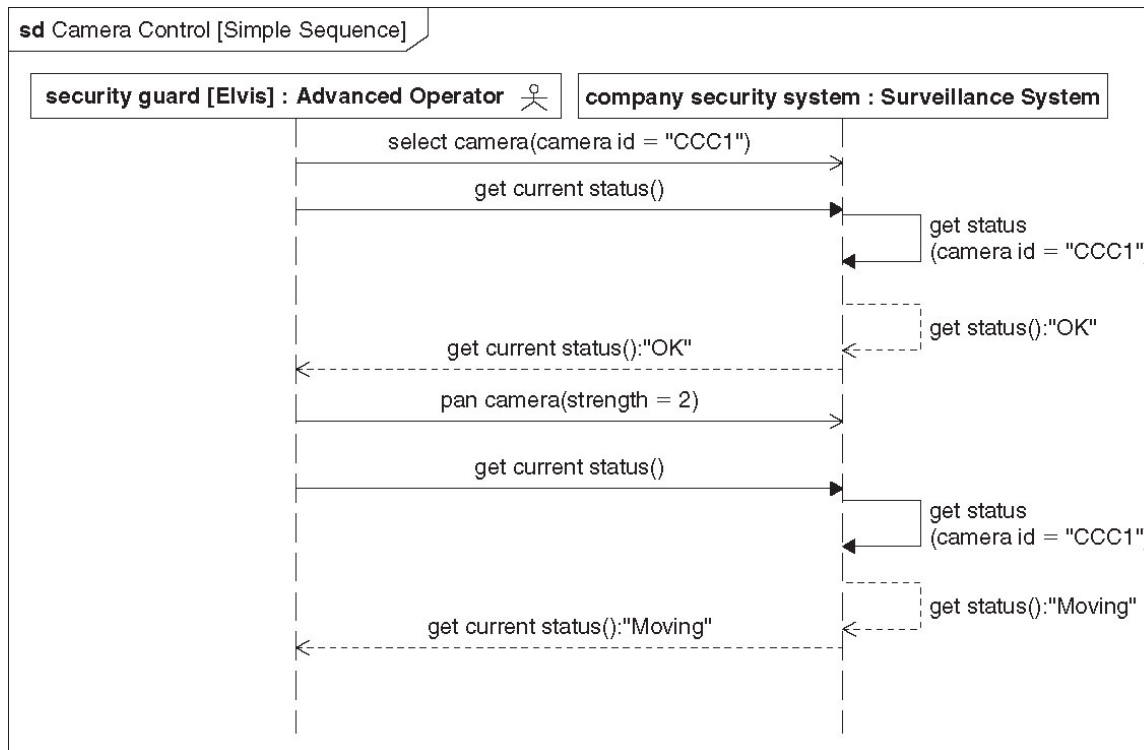


Events and Occurrences

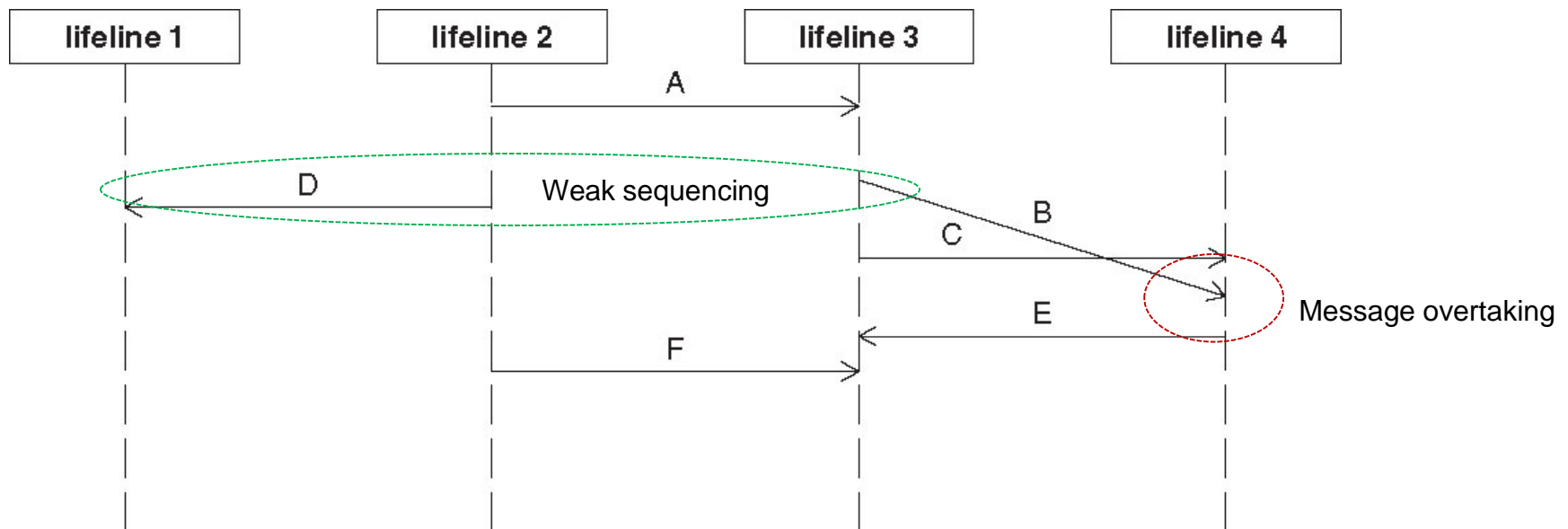
- Events – an ordered list of things that happen along a lifeline
- Occurrences – instances of events during interaction
- Trace – an interaction to validate an ordered set of occurrences in time
- There are three categories of events:
 - The sending and receiving of messages;
 - The start and completion of execution of actions and behavior; and
 - Creation and destruction of instances

Synchronous/Asynchronous

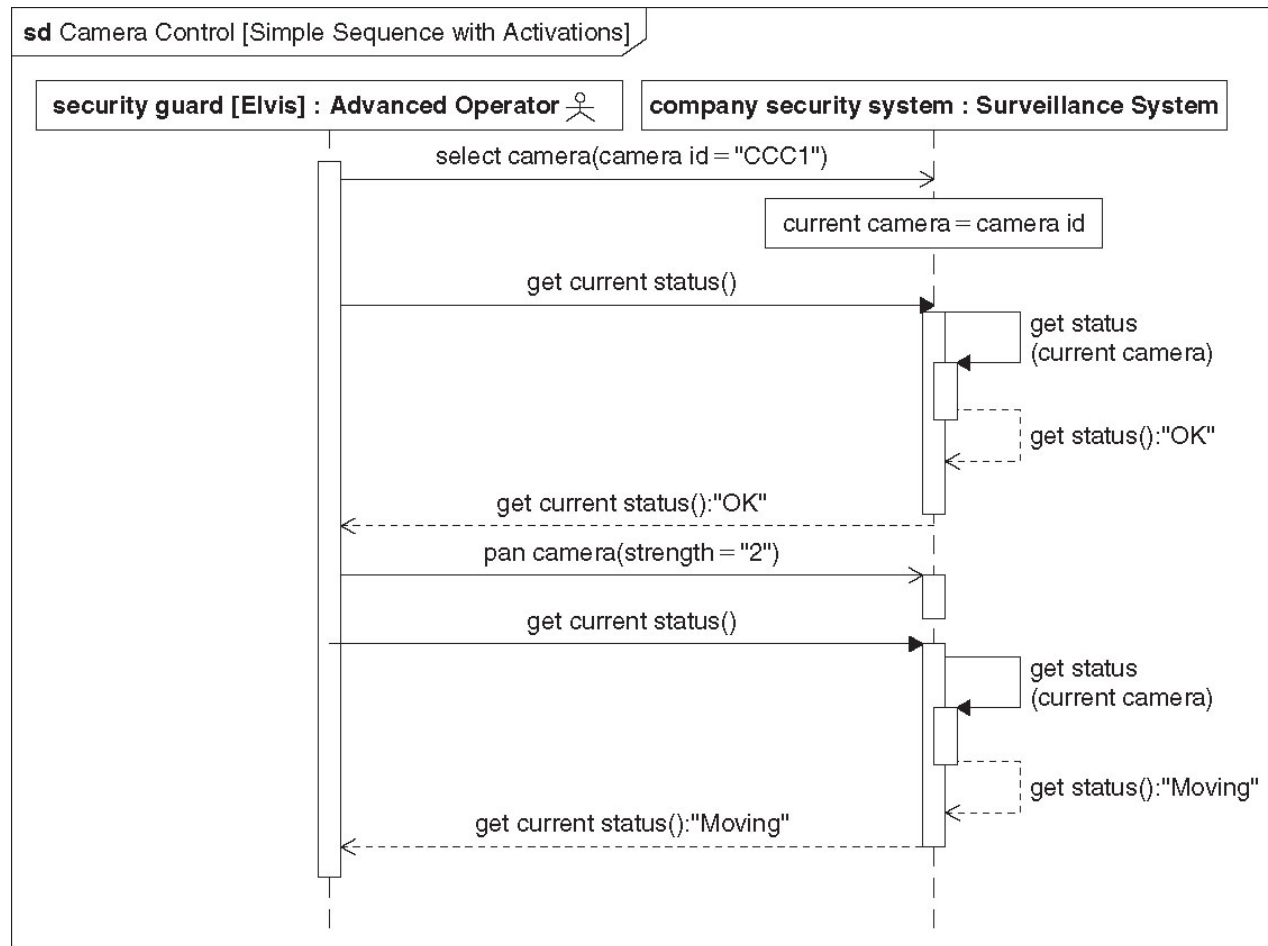
- Synchronous – wait for a response (closed arrowhead)
- Asynchronous – send message and continue (open arrowhead)
- Reply – dashed line & open arrowhead



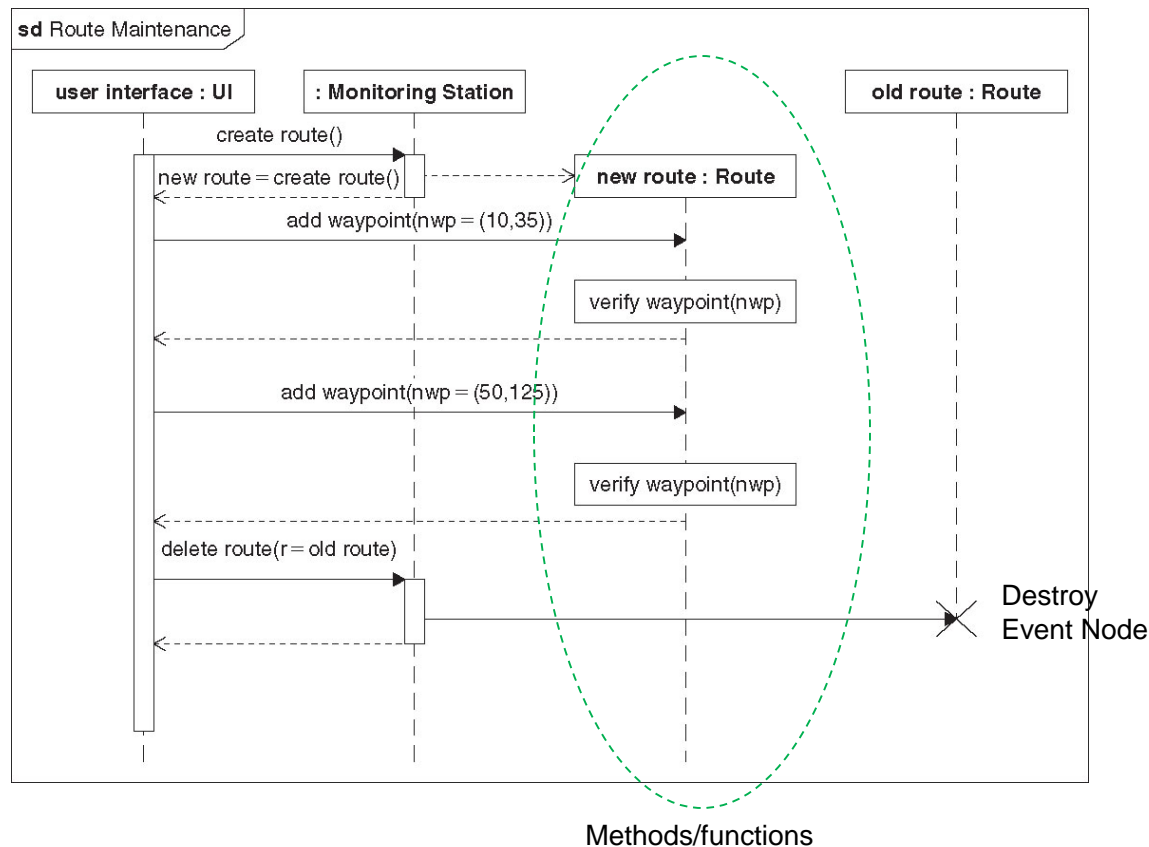
Weak Sequencing



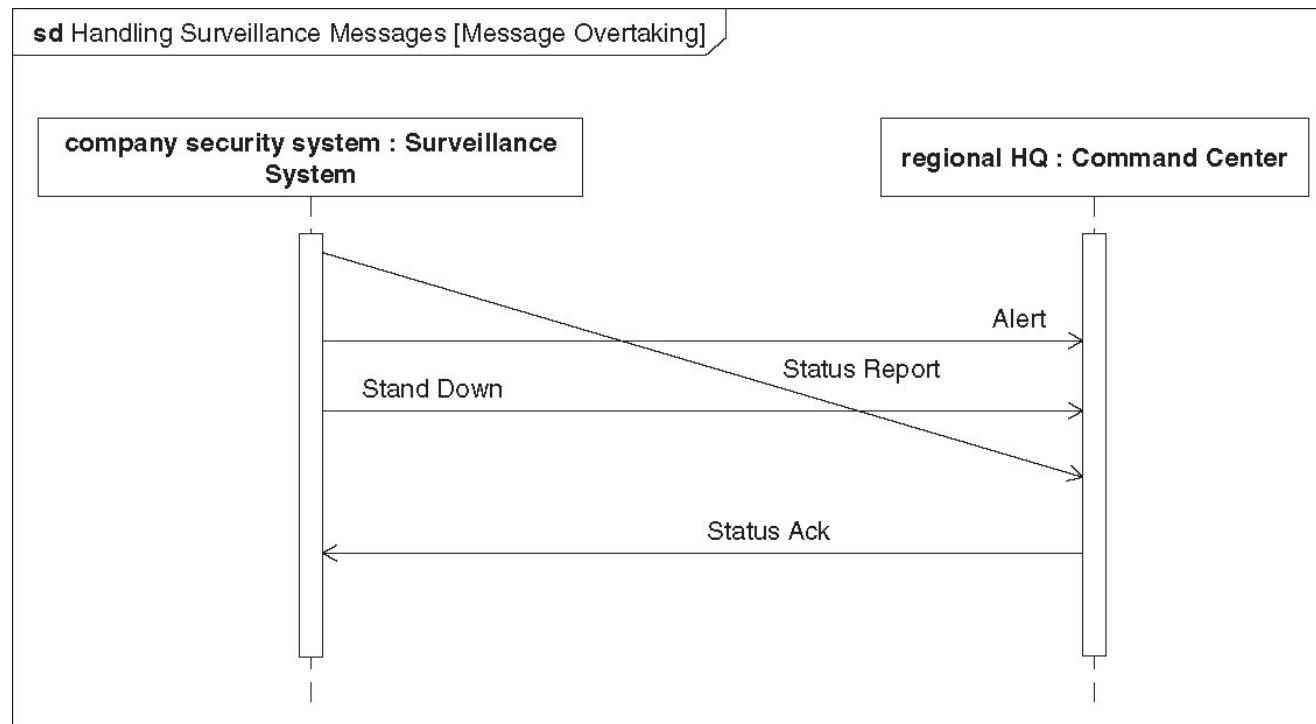
Message can Trigger Execution



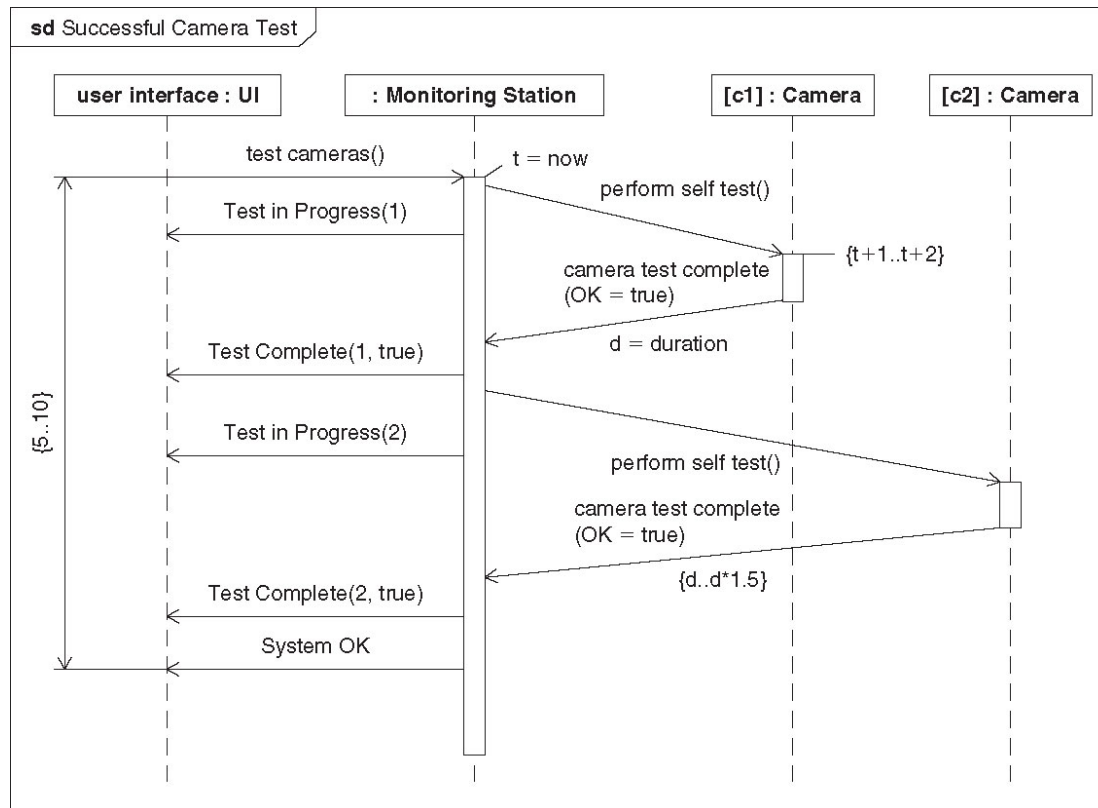
Create and Destroy Messages



Message Overtaking



Time on a Sequence Diagram

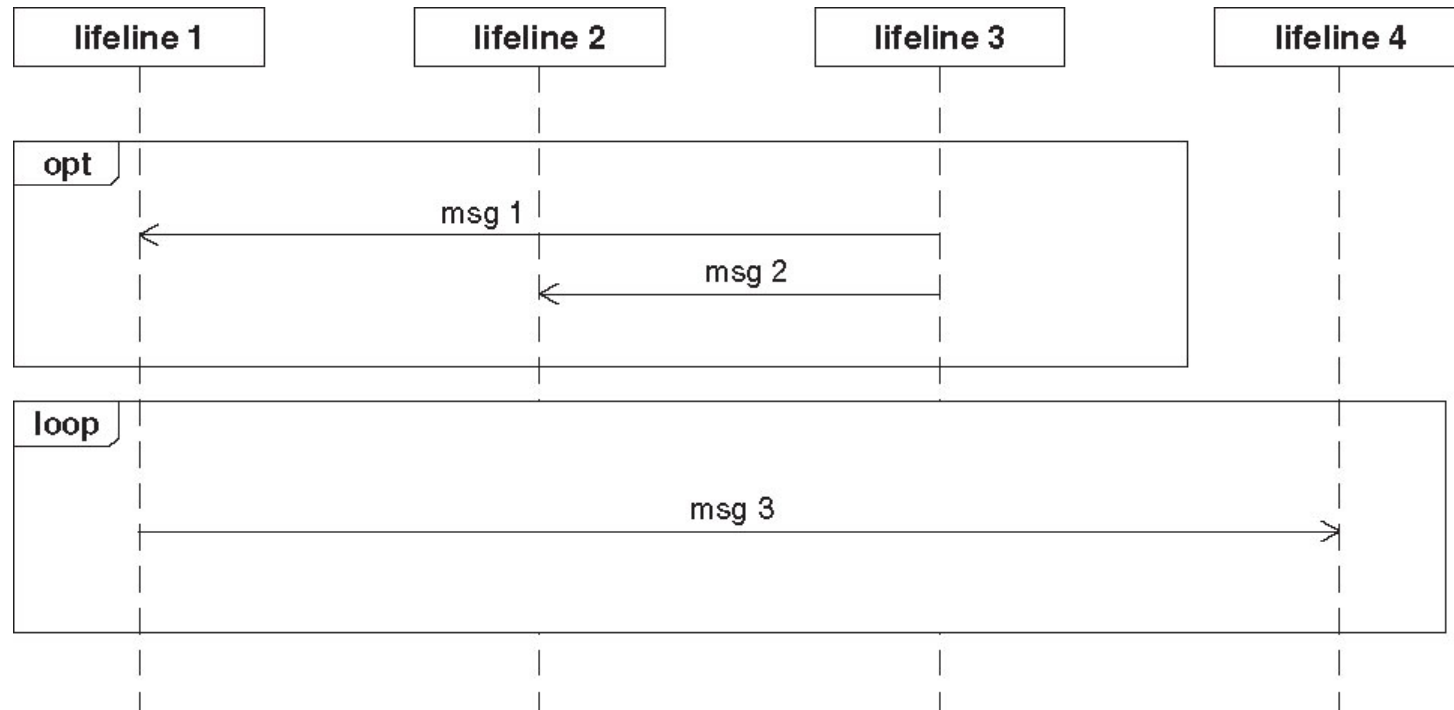


Modelling Complex Scenarios

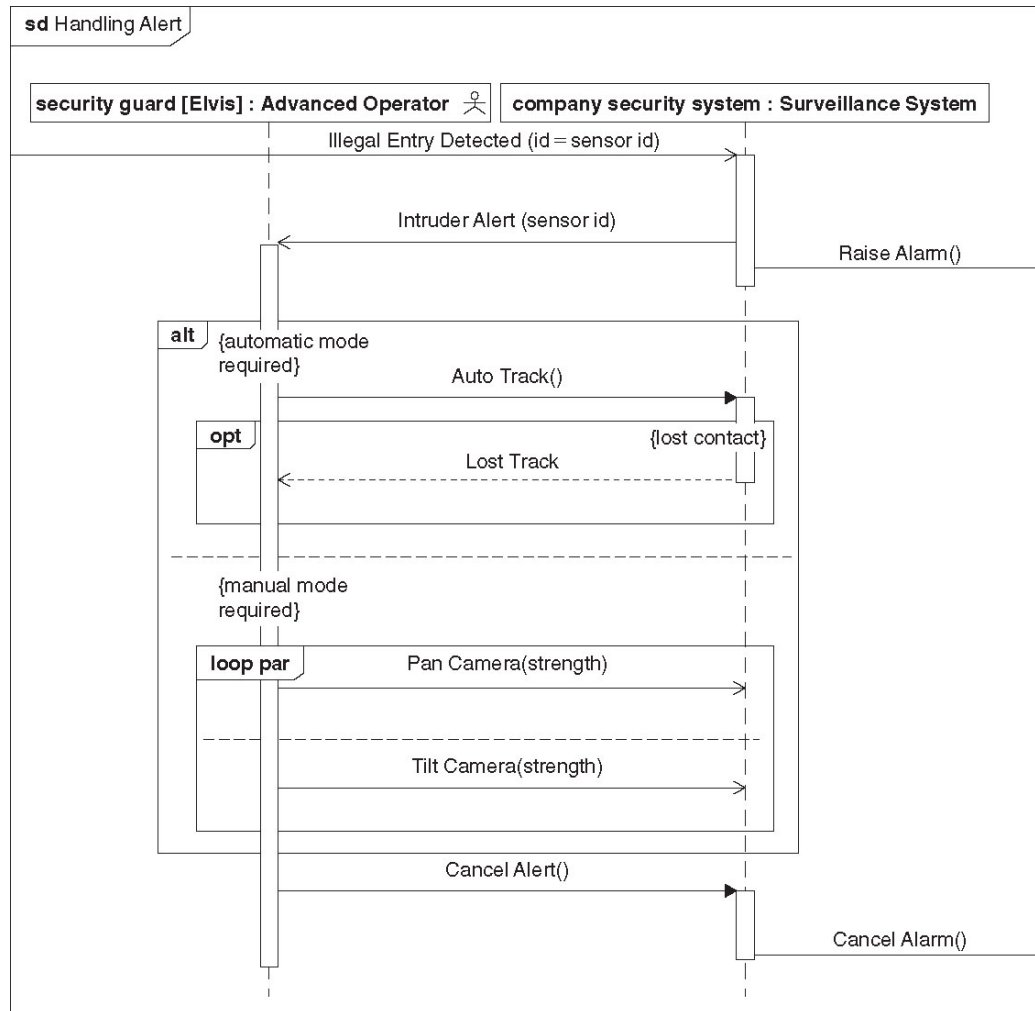
Interaction Operators

- Seq – Weak sequencing
- Par – Parallel, each following seq
- Alt/else – One selected based on guard. Has a choice between fragments
- Opt – unary operator (go/no-go)
- Loop – repeat fragment until constraint is met

Lifelines obscured if not used



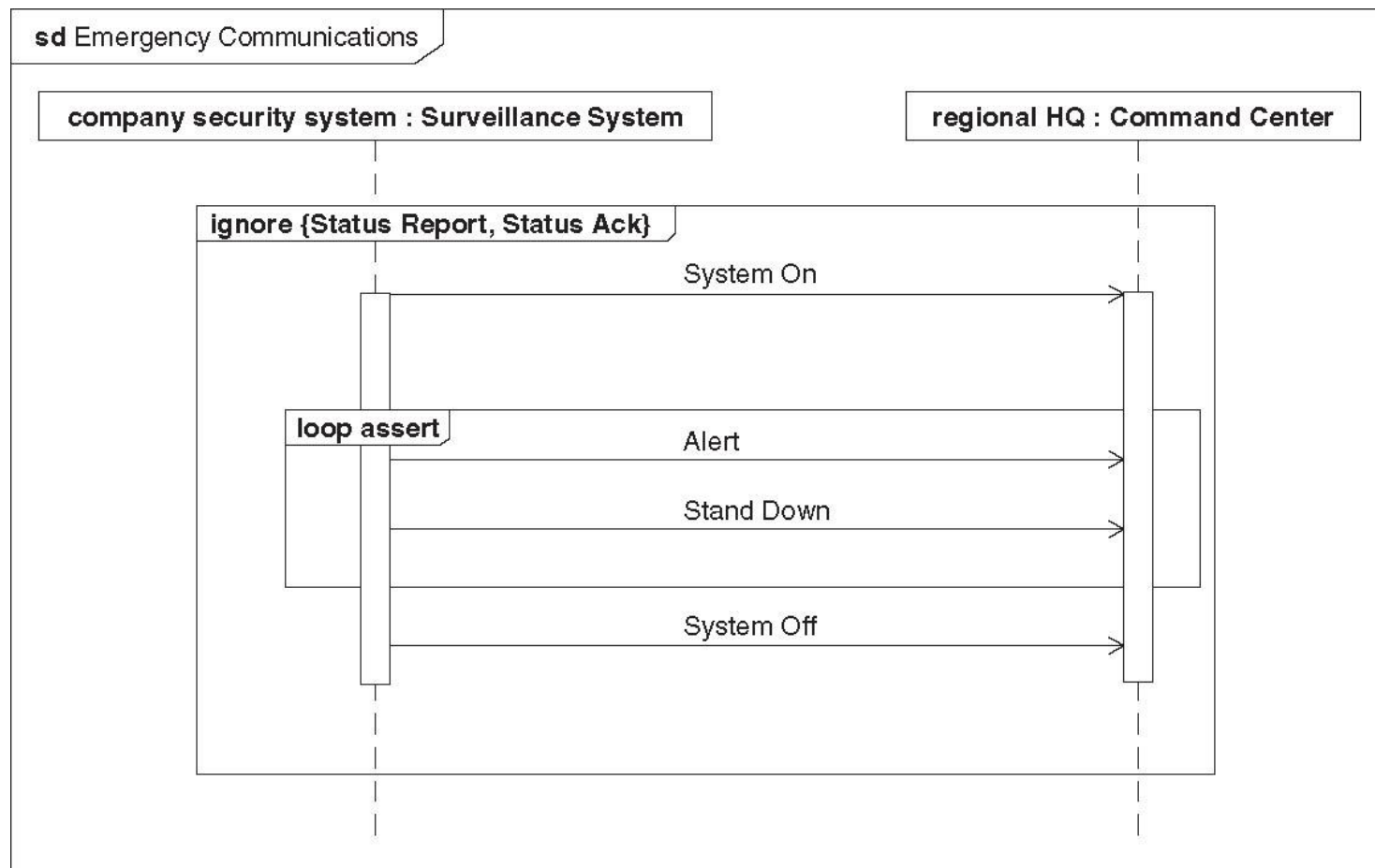
Sequence Diagram



More Interaction Operators

- Strict – like seq, but also affects receives
- Break – if satisfied, operand is executed instead of remainder of fragment
- Critical – indicates that operand must be performed with no interleaving
- Consider – only use messages from a specified set of operations/signals
- Ignore – do not consider messages from a specified set of operations/signals
- Assert – overrides consider and ignore operators within the assert's operand

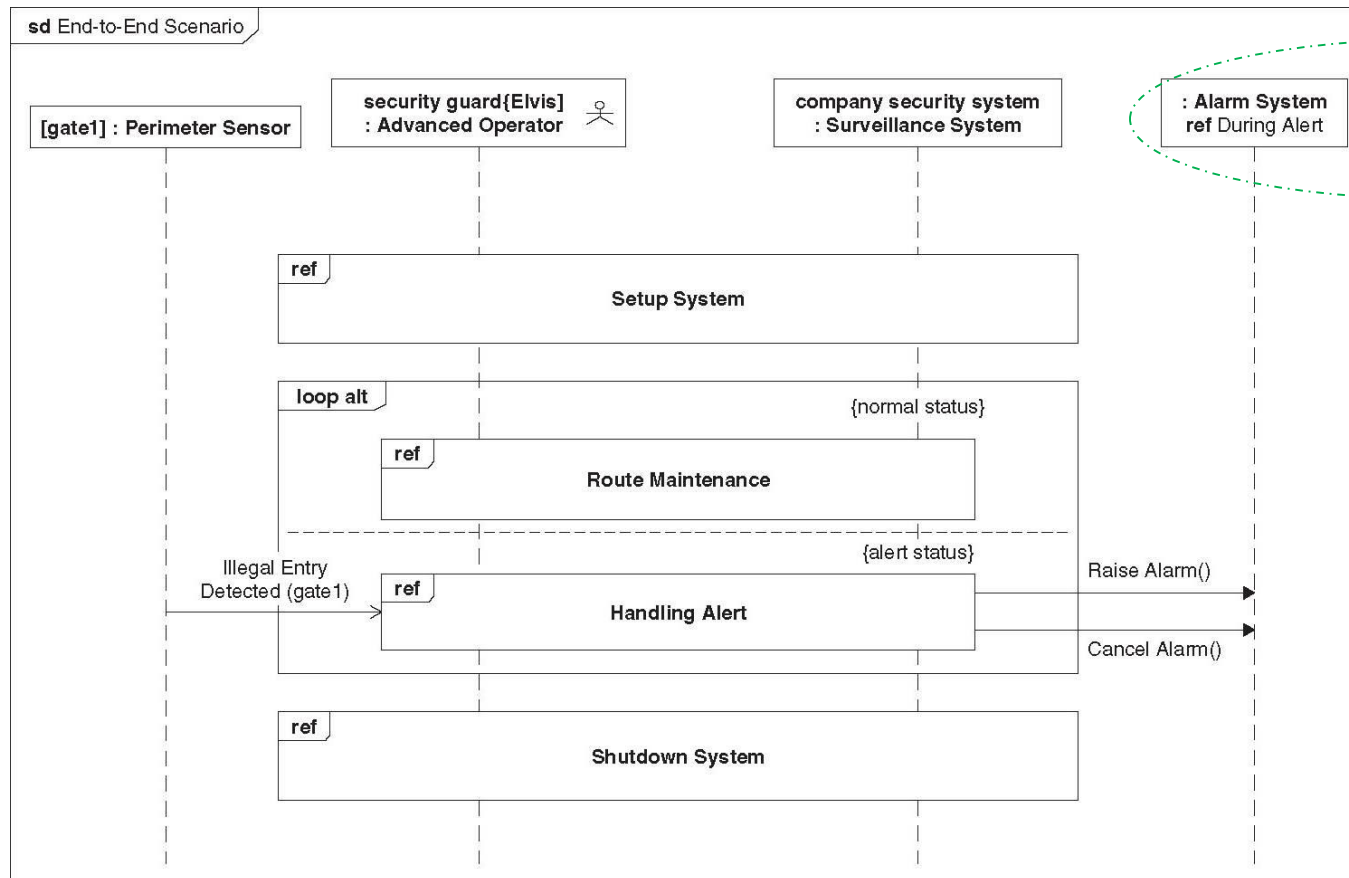
Consider/Ignore/Assert



Interaction References

- Interactions can reference previously defined interactions (interaction use) for:
 - Reuse; and
 - Scalability
- Gates used to show message exchange
 - Formal – gate on the called interaction; and
 - Actual – gate on the calling interaction
- Reference operands denoted as **ref**

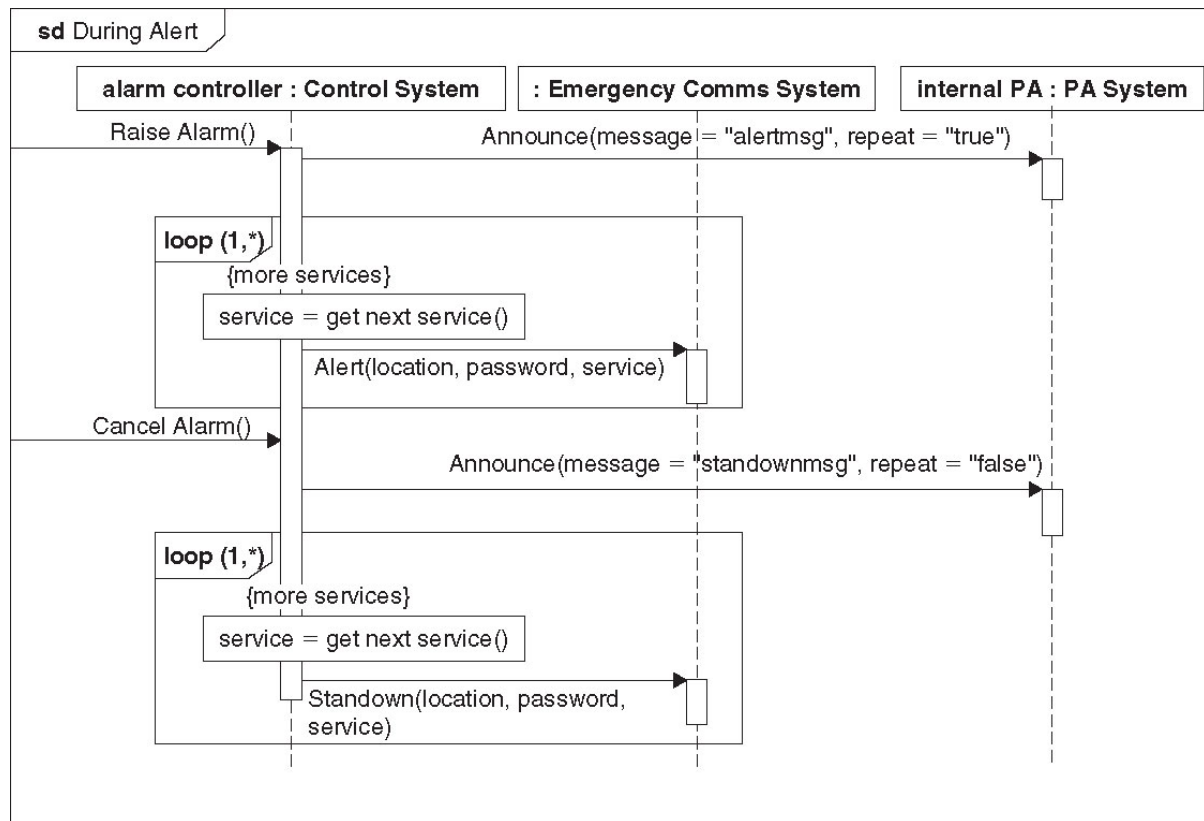
Reference Usage



See its
decomposition
on the next slide

Reference Usage

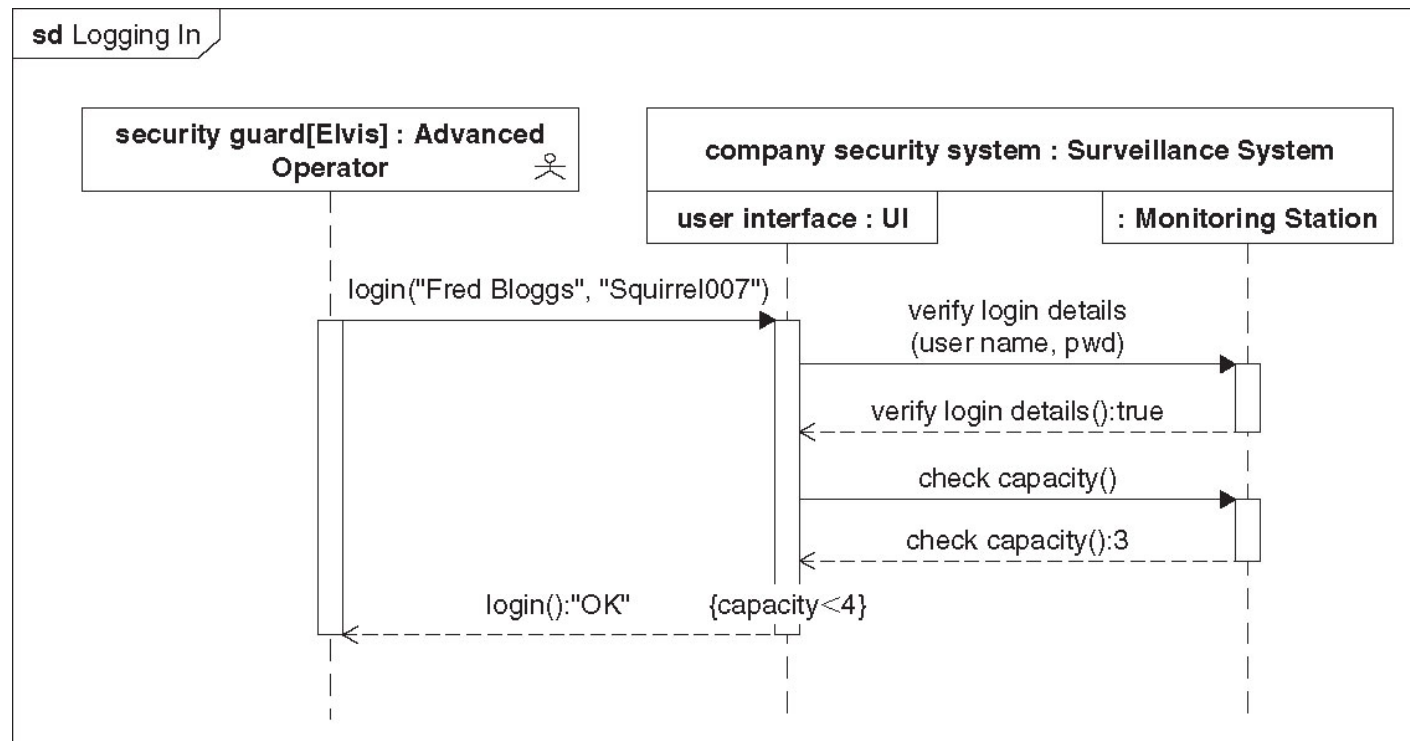
Interactions within the Alarm System



Reference Usage

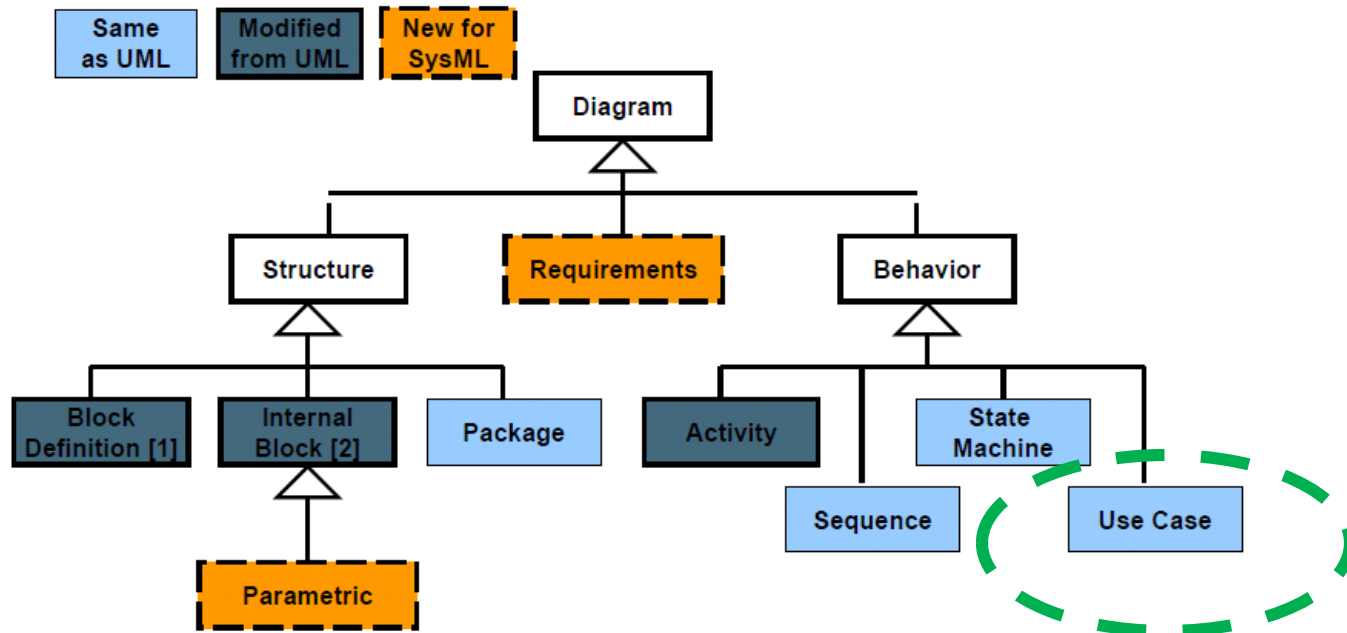
Nesting of Lifeline Decomposition

Example – A *security guard* wishes to log into a *company security system*



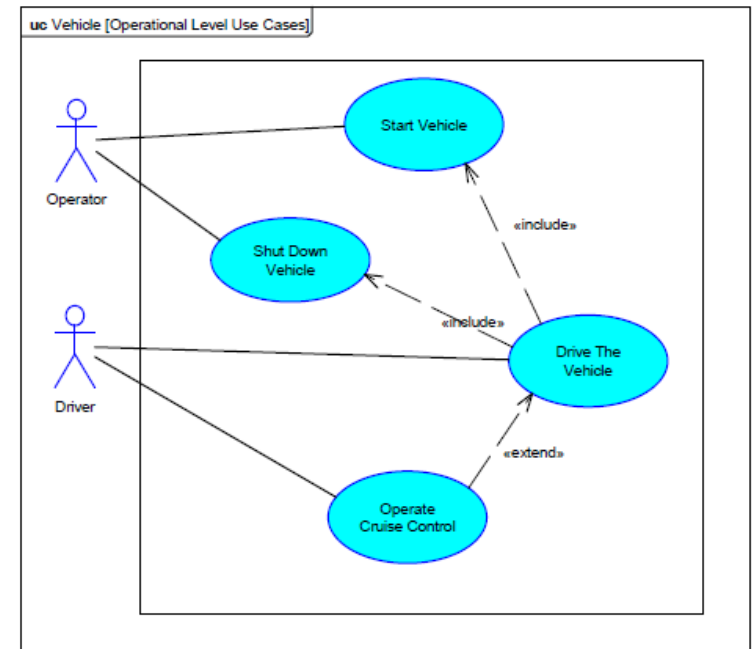
Use Case

SysML Taxonomy of Diagrams

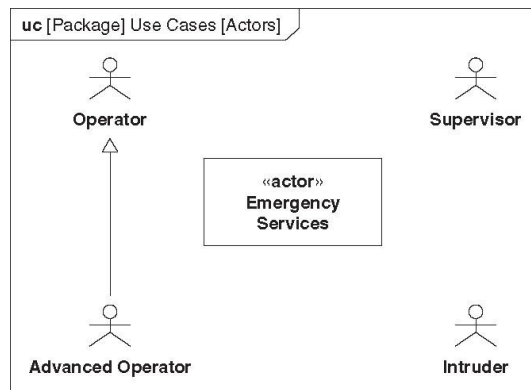
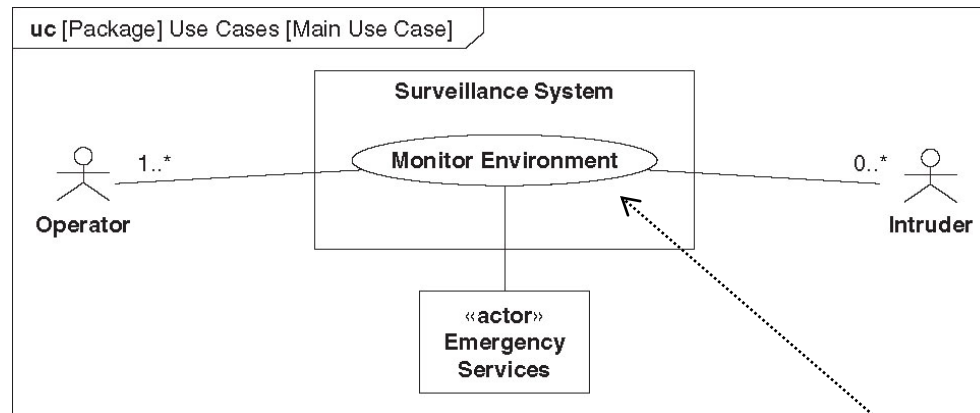


Use Case Diagram

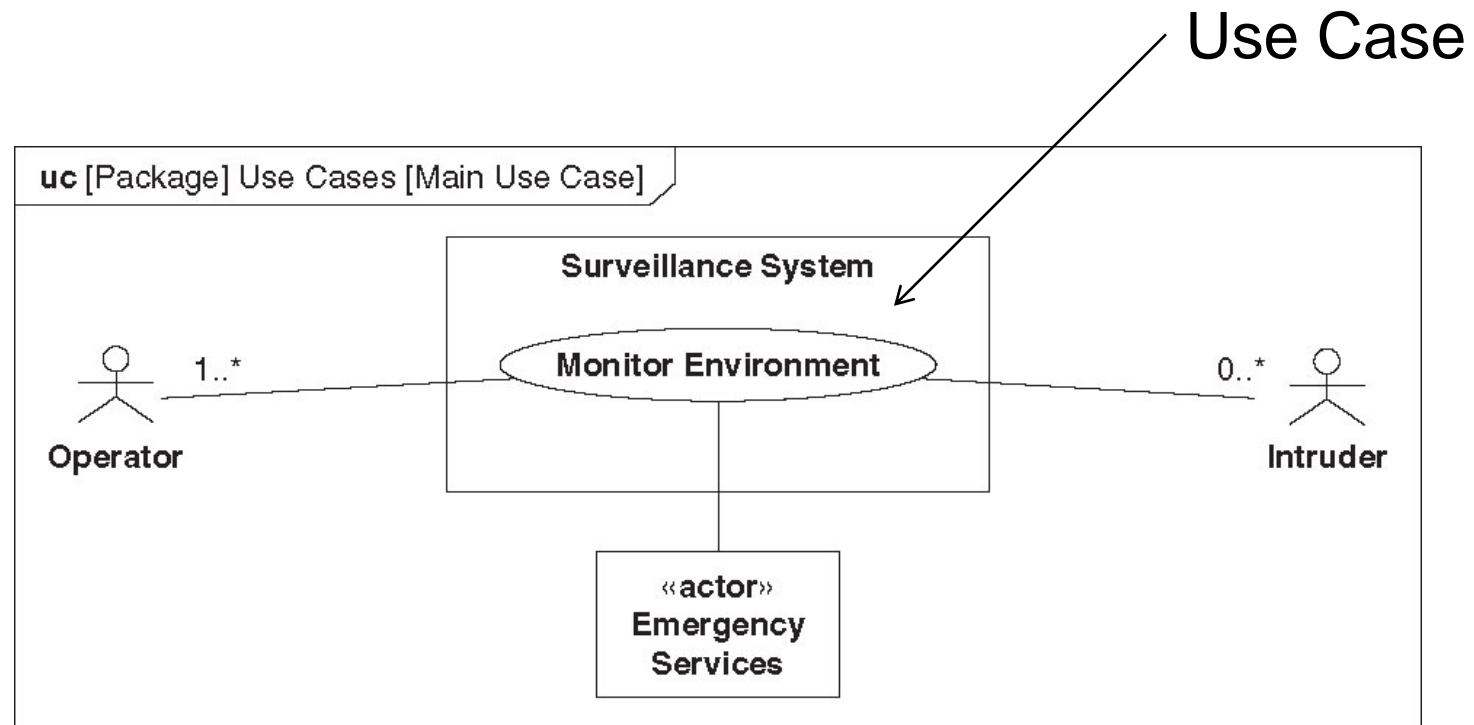
- UML Behavior Diagram
- Provides a means for describing functionality in terms of system usage by actors
- Typically used only at high levels
- Actors represent any external system that participates in the use of the system (human, organization, etc.)
- Typically shown as a stick figure with a name underneath



Use Case Diagram

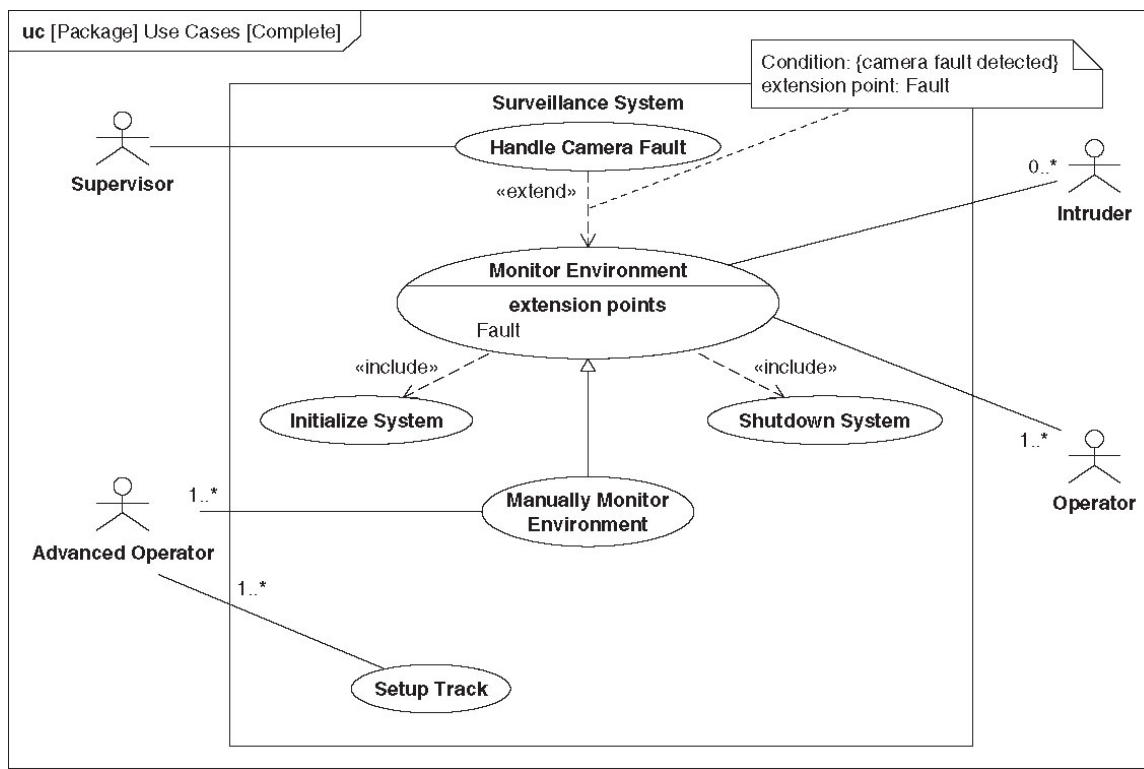


Use Case Diagram



Use Case Relationships

- Inclusion – allows a base use case to include the functionality of another use case as part of its overall functionality when performed.
- Extension – a fragment of functionality that describes an exceptional behavior
 - Must specify extension point



Use Case Description

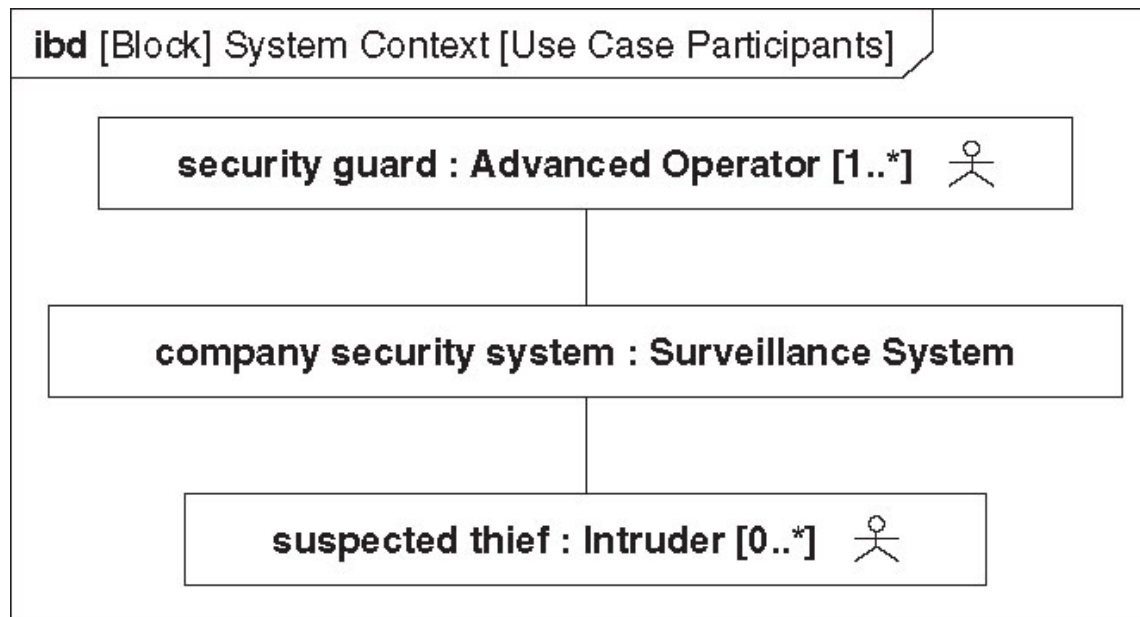
- Text based document to support use case definition
 - Preconditions: must be met to begin
 - Postconditions: must exist when completed
 - Primary flow: most likely scenario(s)
 - Alternate/exception flows: other scenario(s)

Elaborating Use Cases

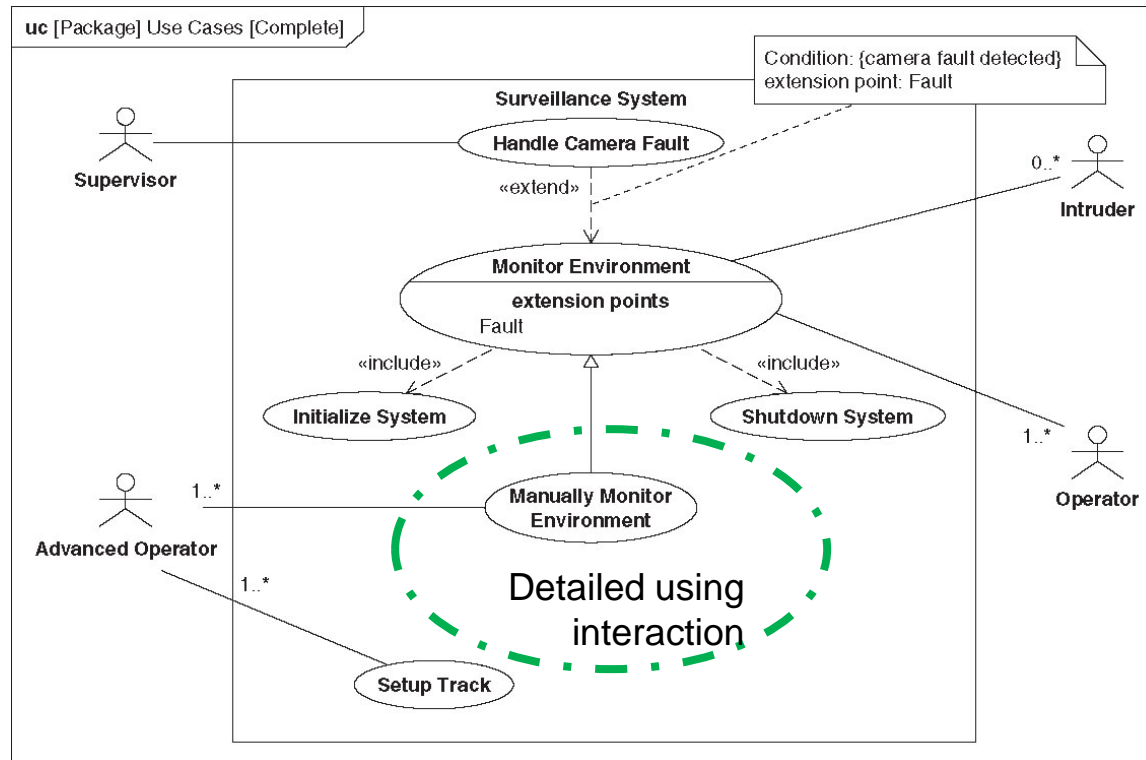
- Detailed Definition of a Use Case can be modelled with interaction, activities or state machines:
 - **Interactions** are useful where a scenarios is largely message-based;
 - **Activities** are useful where the scenarios include considerable control logic, flow of i/p and o/p or algorithm that transform data;
 - **State machines** are useful when the interaction between the actors and the subject is asynchronous (event-based), not easily represented by an ordered sequence

Context Diagram

Essential to start detailed modelling with a Context Diagram

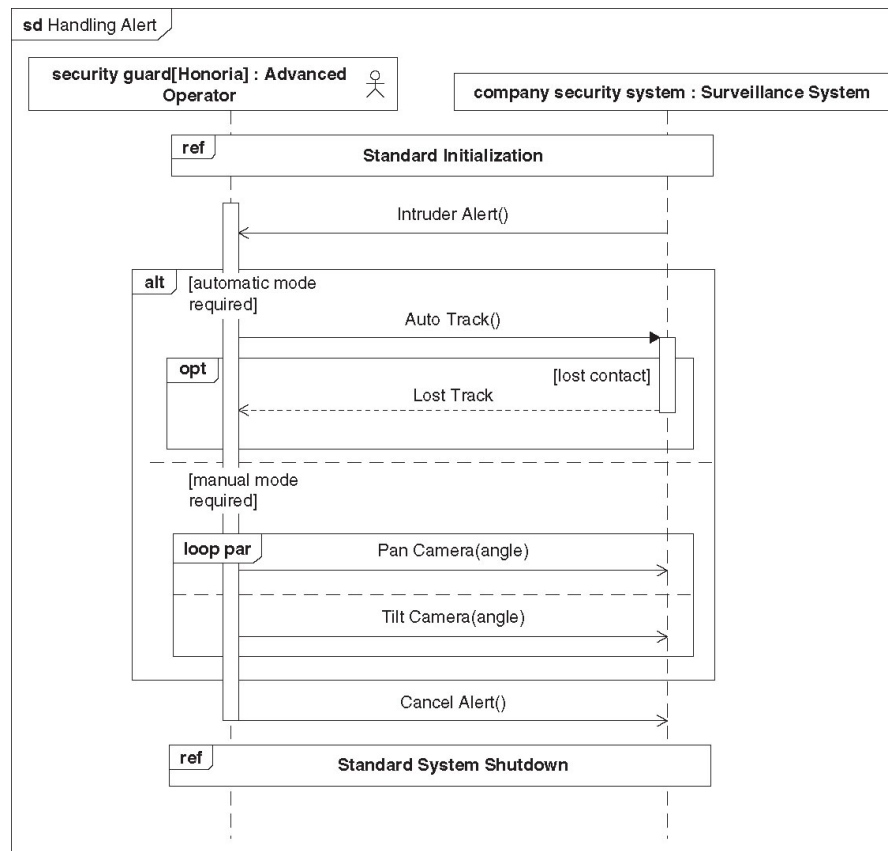


High-Level Use Case



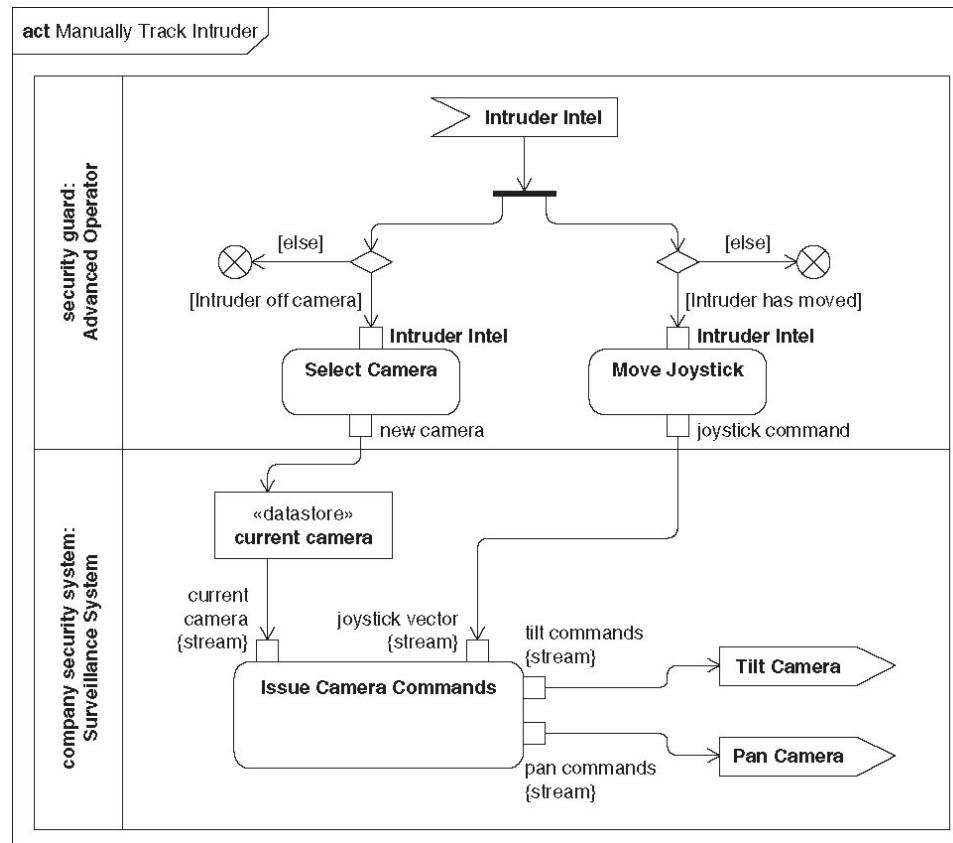
Detailed modelling of Use Case using Interaction

- Handling Alert of the *Manually Monitor Environment* use case



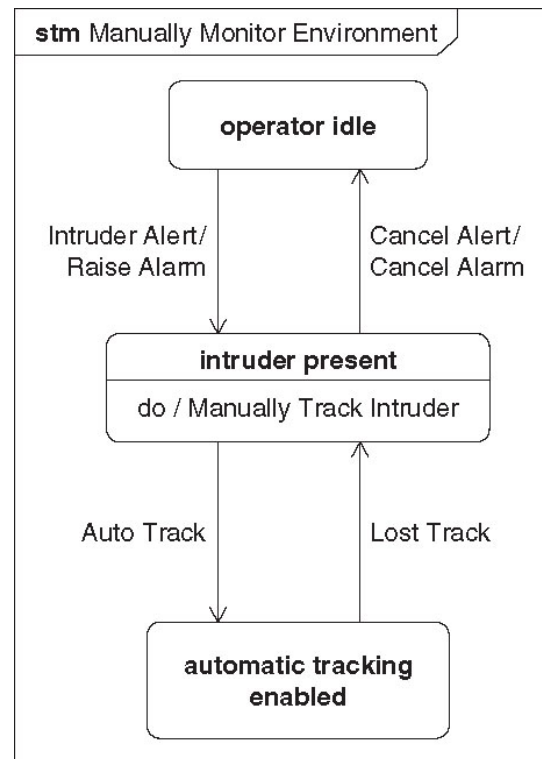
Detailed modelling of Use Case using Activity

- Manual Track Intruder* activity of the *Manually Monitor Environment* use case



Detailed modelling of Use Case using State Machines

- Key states in the *Manually Monitor Environment* use case are: *operator idle*, *intruder present*, *automatic tracking enabled*
- Focus on states rather than messages.



Program Completed

Missouri University of Science &
Technology