



# The UML for Engineering Agent Systems



Michael Papasimeon and Clint Heinze  
Aeronautical & Maritime Research Laboratory  
DSTO

Presented to Agents Victoria - May 2001



## Outline of Talk

- Architectural design [brief] (Clint)
  - The AUML
- Specification – UML and Use Cases (Clint)
- Design and Implementation - Extending UML to Design Jack Agents (Mike)
- Tool Support (Mike)
- UML Symbolology for Agents (Mike/Clint)
- Generalising to Different Types of Agents (Mike/Clint)
  
- Questions

- 
- 
- 

# Engineering Agent Systems

The Unified Modelling Language (UML) is an **obvious** choice



## Why Obvious?

- UML is mainstream and widely adopted
- Good tool support
- Many agent projects are < 20% agent
  - One modelling language for the whole project *seems* like a good idea
- The UML is dynamic and developing fast
  - The AUML is on the way
  - Business modelling looks like agent modelling
- Agent languages (JACK) are extensions of OO languages like (JAVA)
- **Agents Live in Object Oriented Environments**

# Obvious Does Not Equal Useful

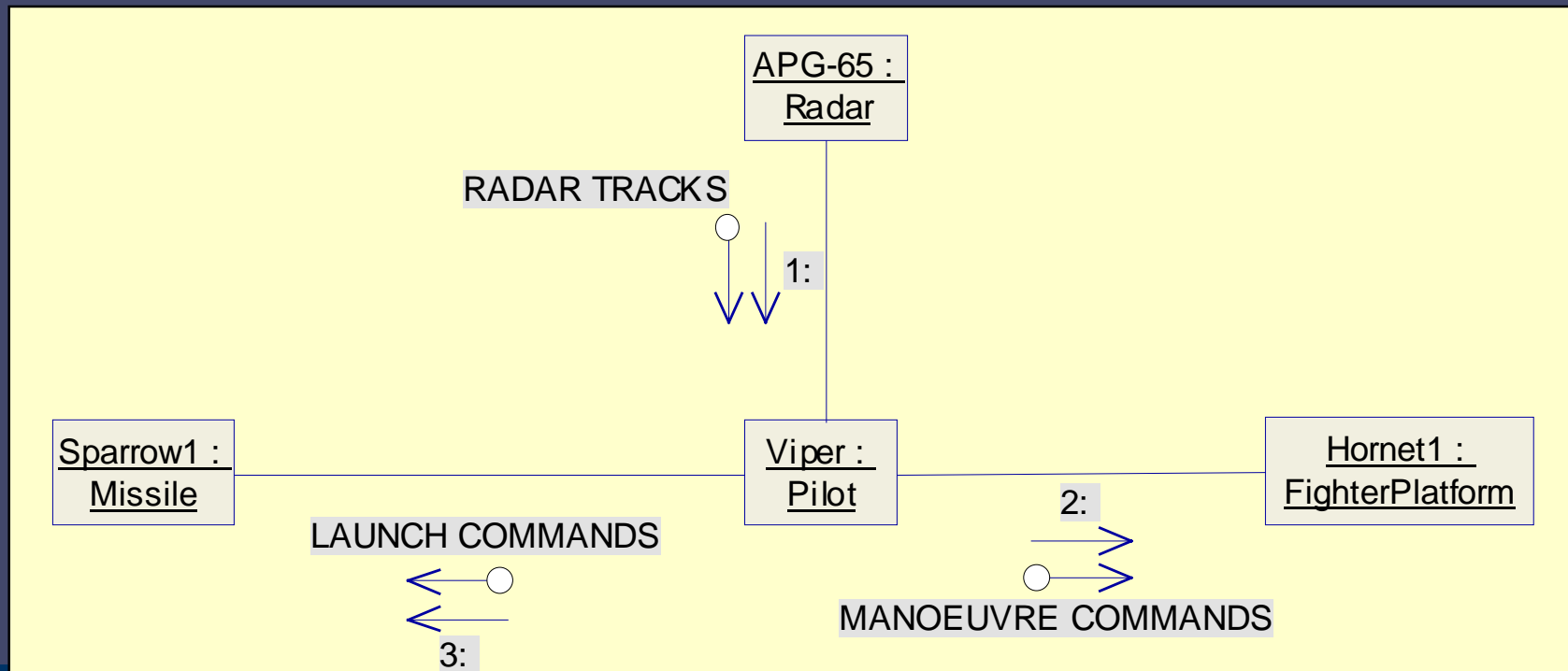
- So let's try it!
- Explore suitability of the UML for the software engineering of agent systems
- Allow the possibility of extensions and modifications to the UML as necessary
- Consider the research and development activities of others
- Many aspects of software engineering
  - Business Modelling
  - Knowledge Management
  - Architectural Design
  - Specification
  - Design and Implementation

# Architectural Design



*Is the UML useful for architectural design  
in agent systems?*

# F/A-18 Hornet Component Interactions

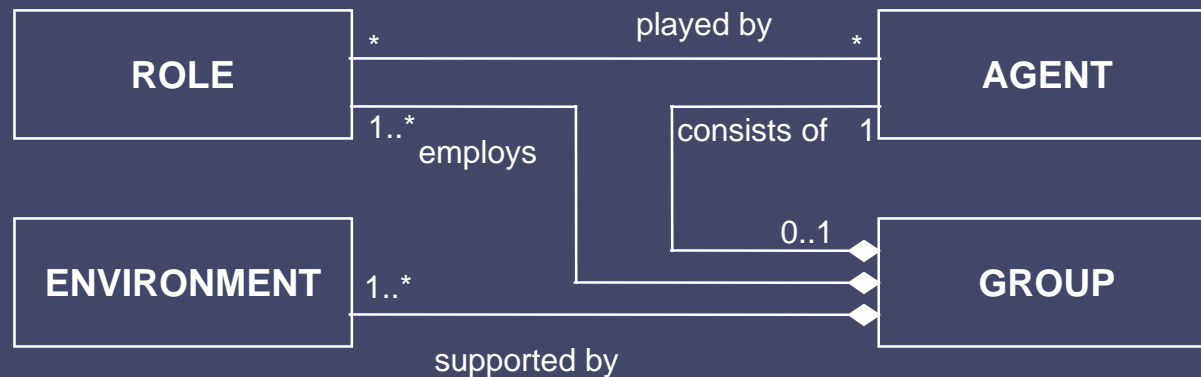


# The AUML



- [www.auuml.org](http://www.auuml.org)
- Small team – James Odell, Van Parunak et. al.
- Focus on agent interactions and architectural issues
- Presentation AOSE 2001 (next week) by Parunak and Odell
  - Representing social structures in UML

[http://www.jamesodell.com/AOSE\\_2001-presentation.pdf](http://www.jamesodell.com/AOSE_2001-presentation.pdf)



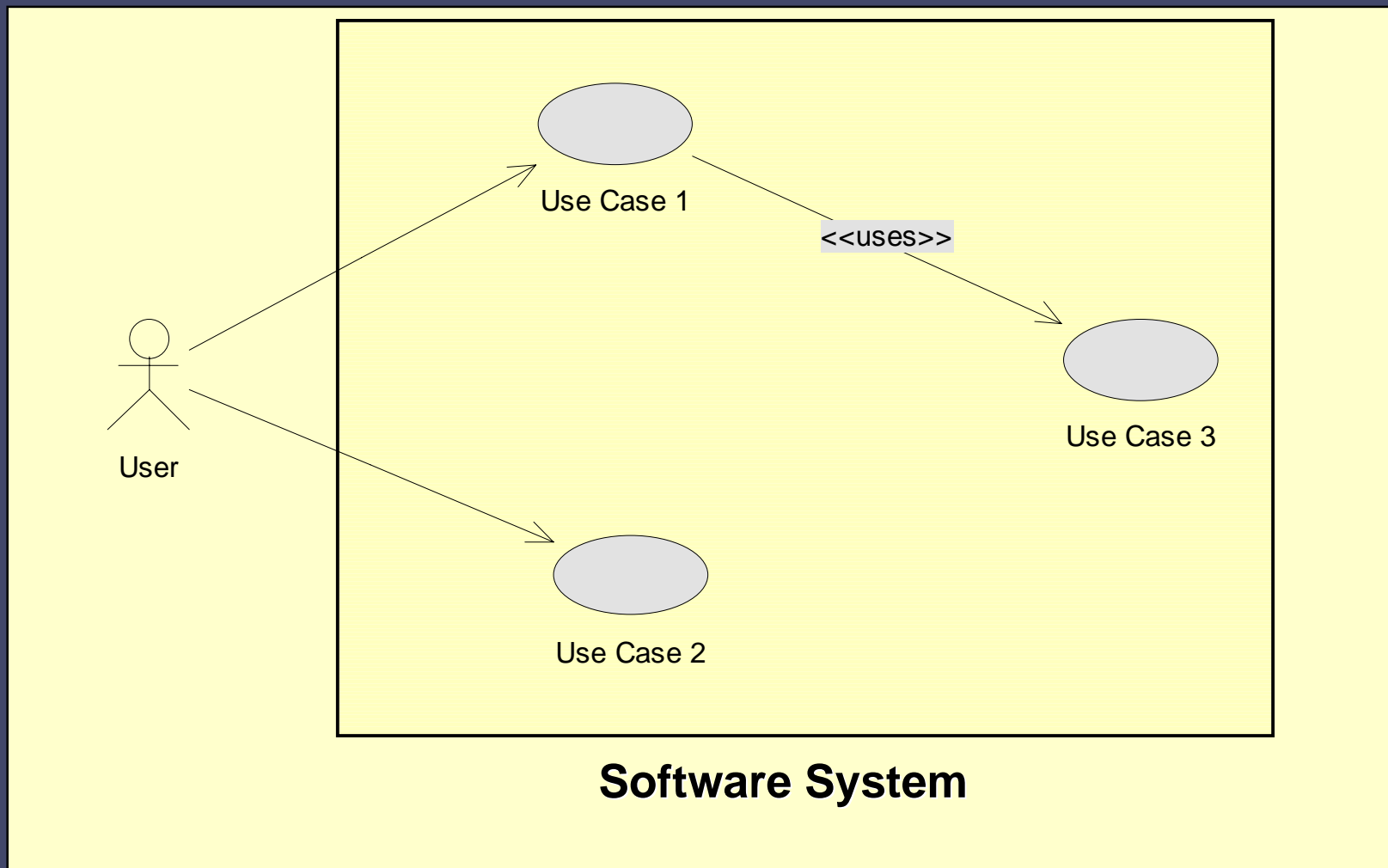
# Specification



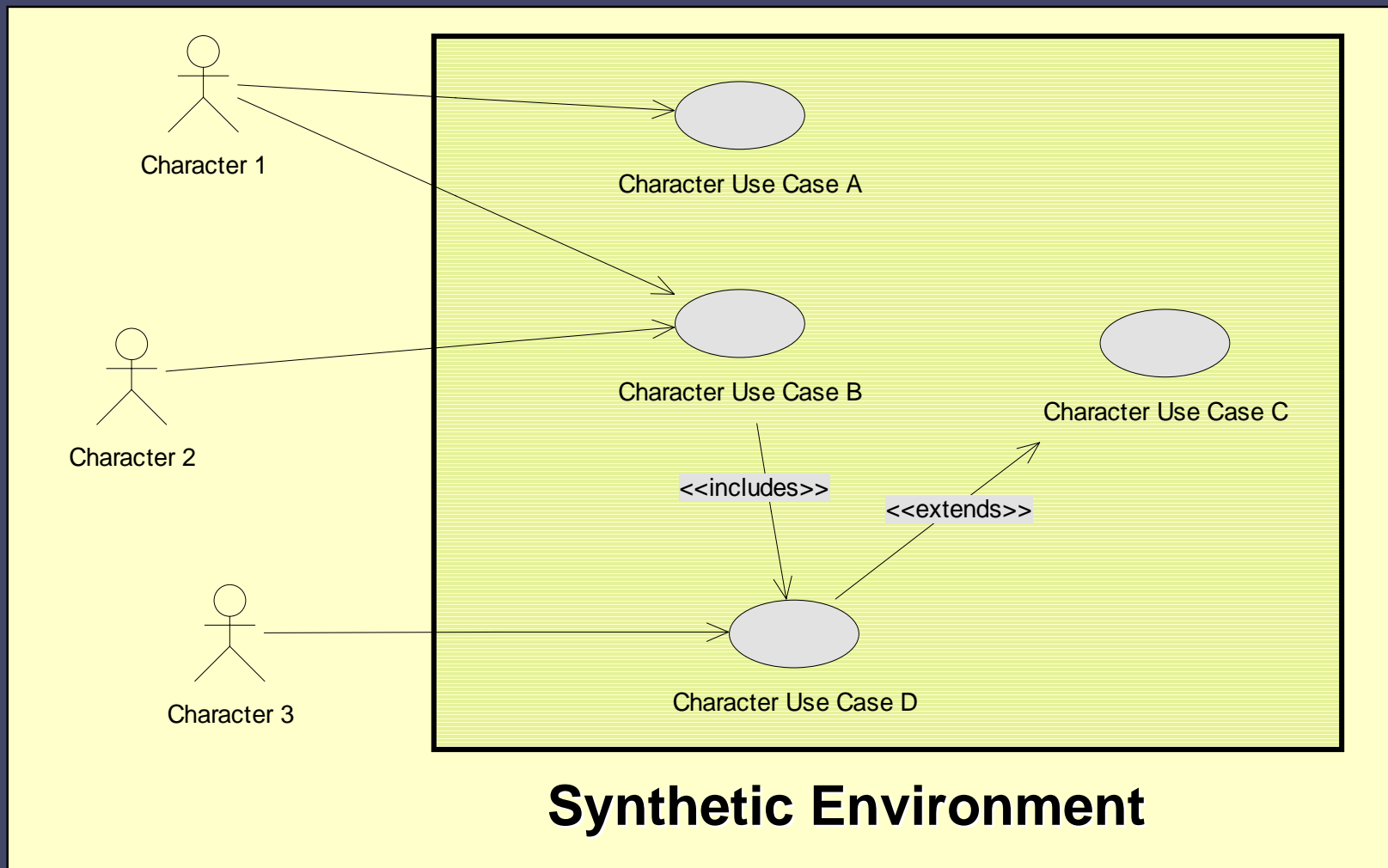
*Can the UML assist the software engineer  
in the specification of an agent system?*

C. Heinze, M. Papisimeon, and S. Goss. Specifying Agent Behaviour With Use Cases. In *Proceedings of Pacific Rim Workshop on Multi-Agents*, 2000.

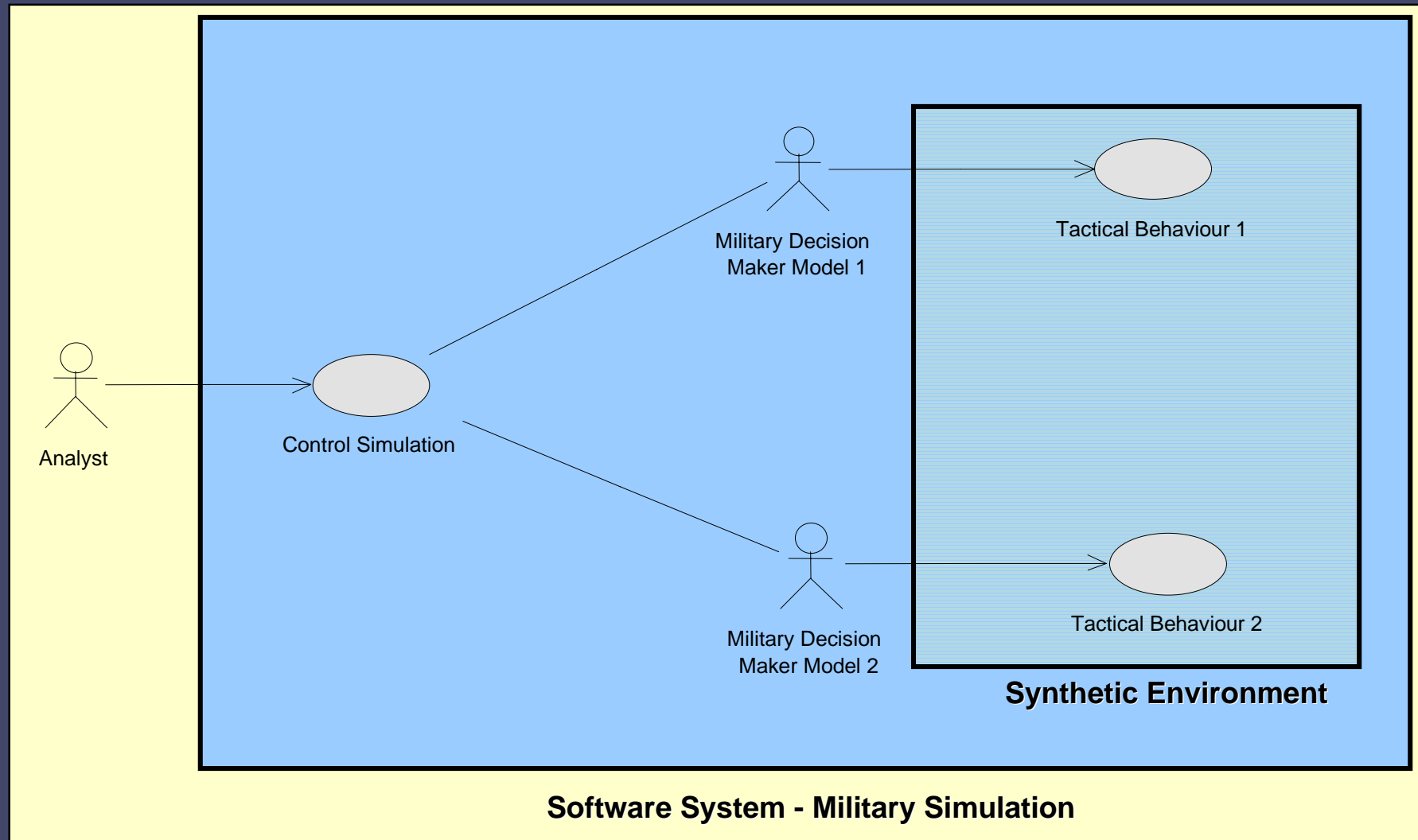
# An Interactive Software System



# Characters Interacting with a Synthetic Environment



# The Systems We Build



# Design and Implementation



***Can the UML assist the software engineer in the detail design and implementation of an agent system?***

M. Papasimeon, C. Heinze. Extending the UML for designing JACK agents. In *Proceedings of Australian Software Engineering Conference (ASWEC) 2001 (to appear)*.

# UML Extensions for JACK

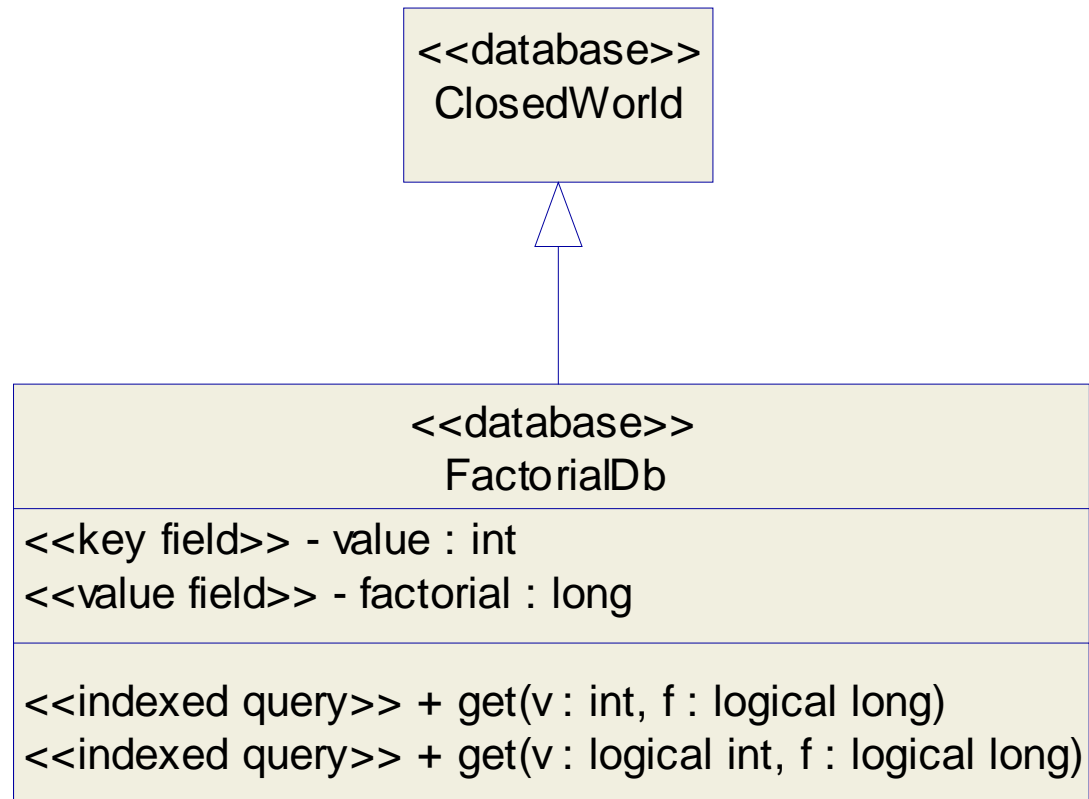
- JACK = Java + Extensions + Agent Architecture
- UML can easily be used to model Java systems
- UML can be extended to support JACK at the detailed design and implementation level using <<stereotypes>>

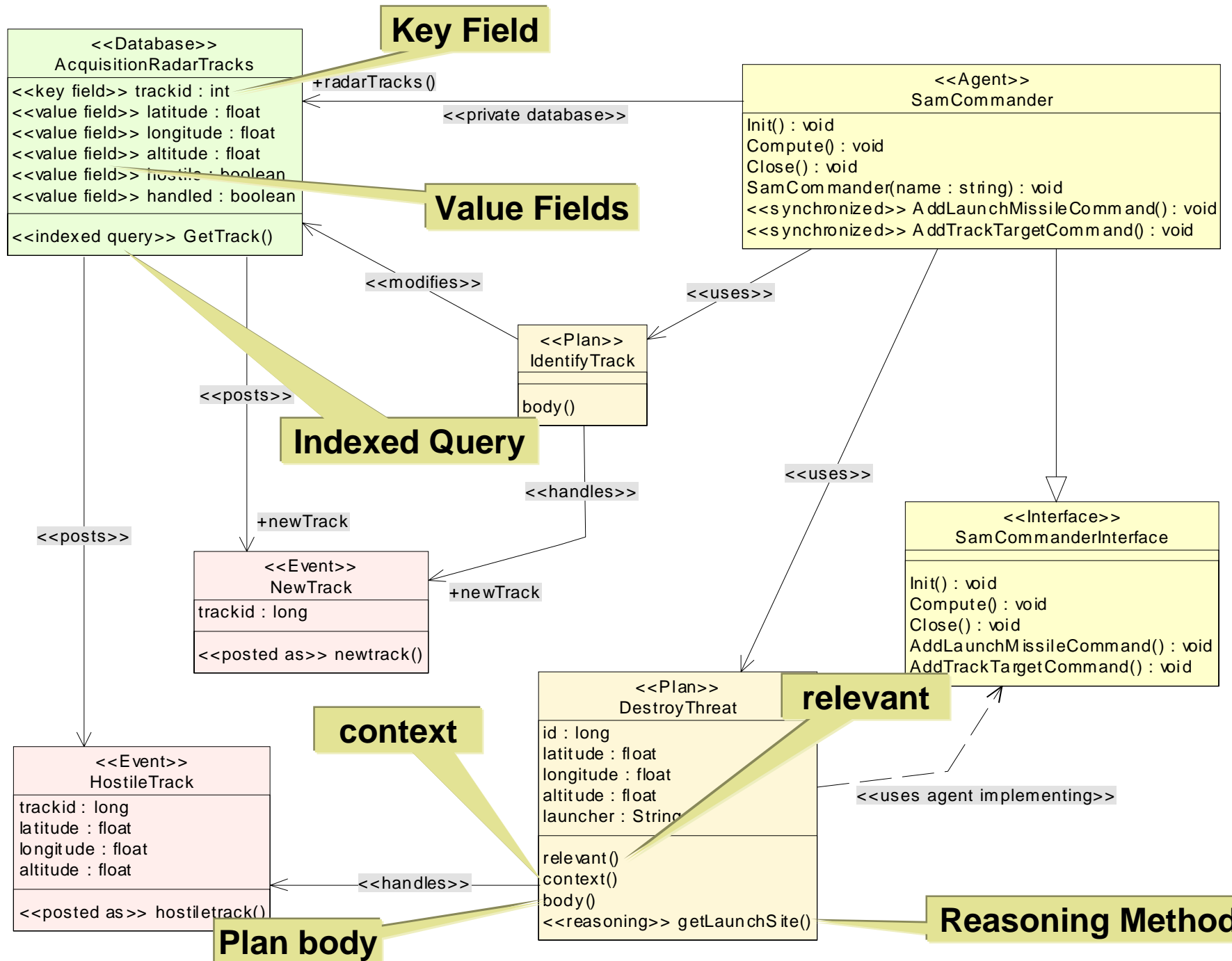
# JACK Agents - Detailed Design

- High Level JACK Stereotypes
  - <<agent>>, <<plan>>, <<event>>, <<database>>, <<capability>>
- Association Stereotypes
  - <<uses plan>>, <<handles event>>, <<posts>>, <<reads database>>
- Low Level Stereotypes
  - <<key field>>, <<value field>>, <<indexed query>>

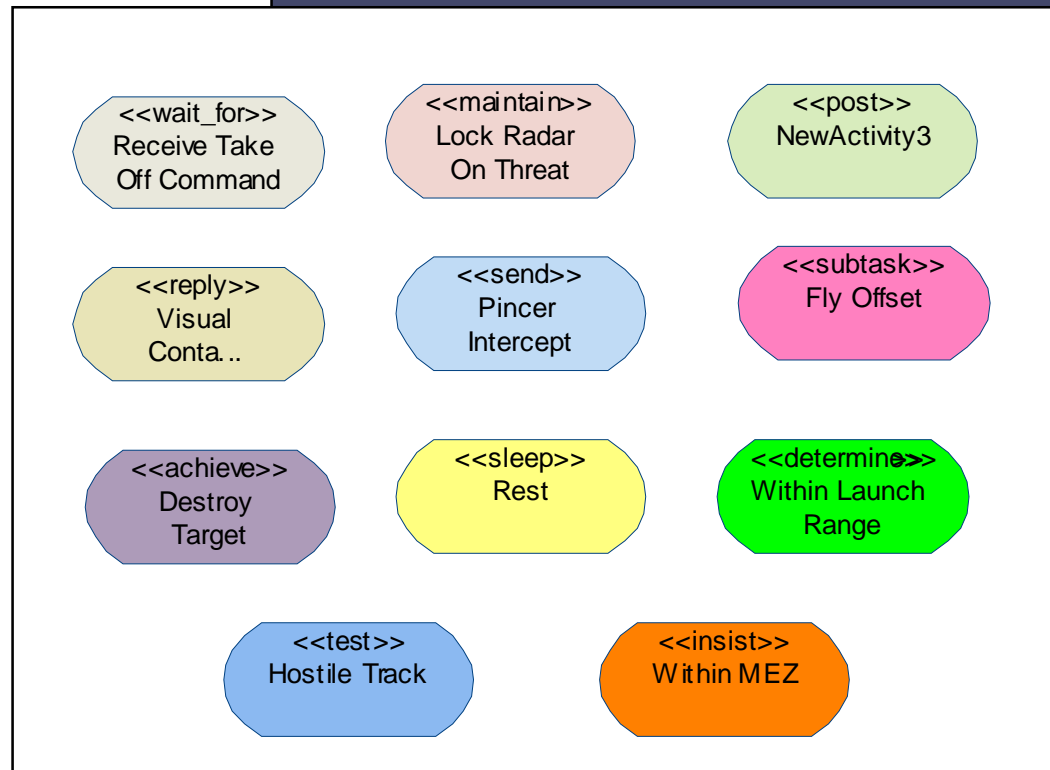
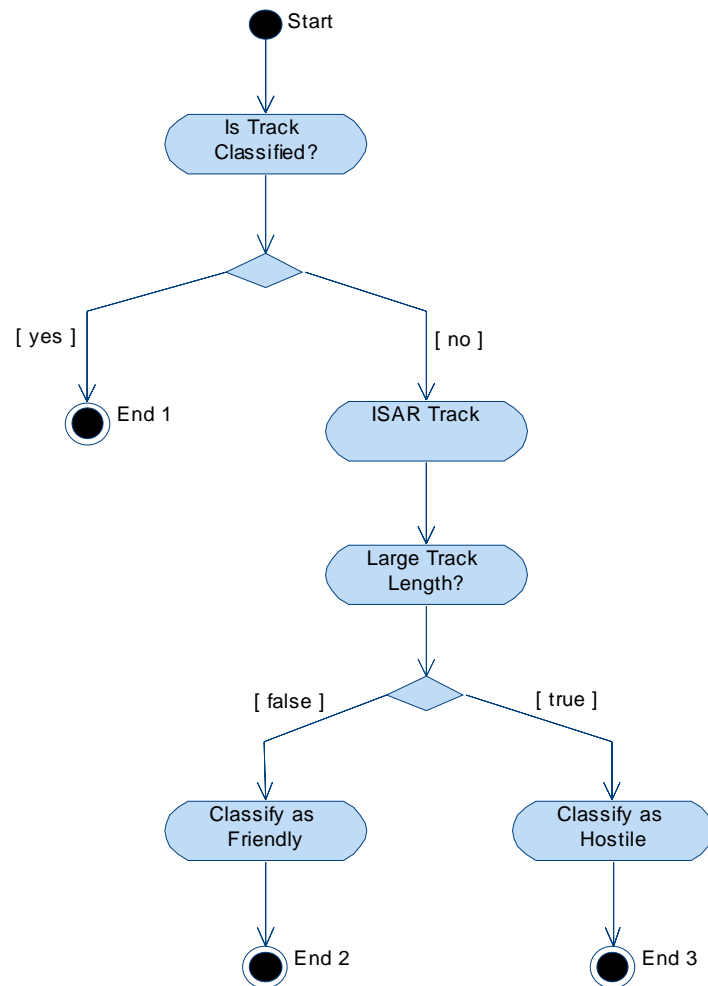
```
database FactorialDb extends ClosedWorld {
  #key field int value;
  #value field long factorial;

  #indexed query get(int v, logical long f);
  #indexed query get(logical int v, logical long f);
}
```

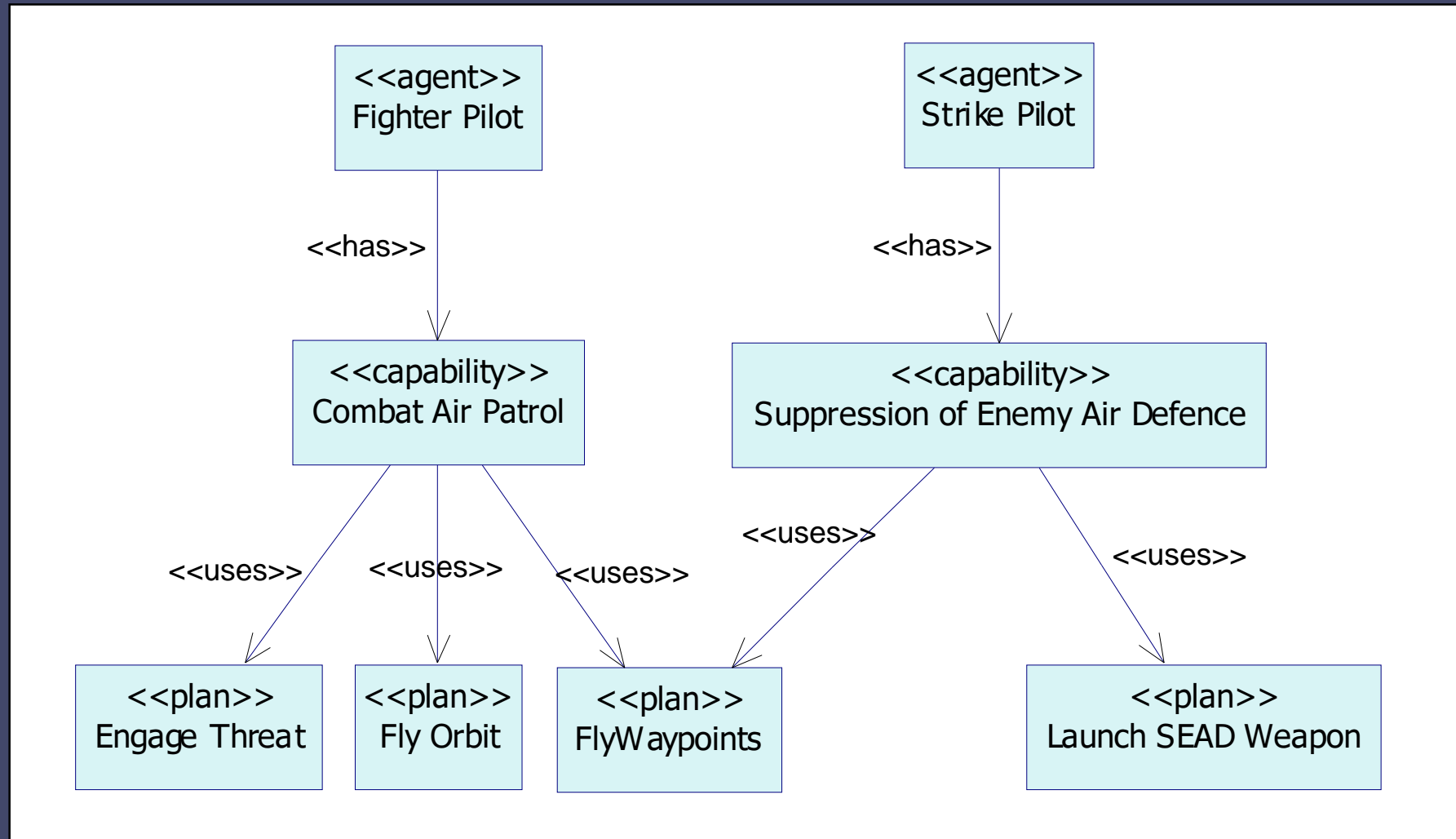


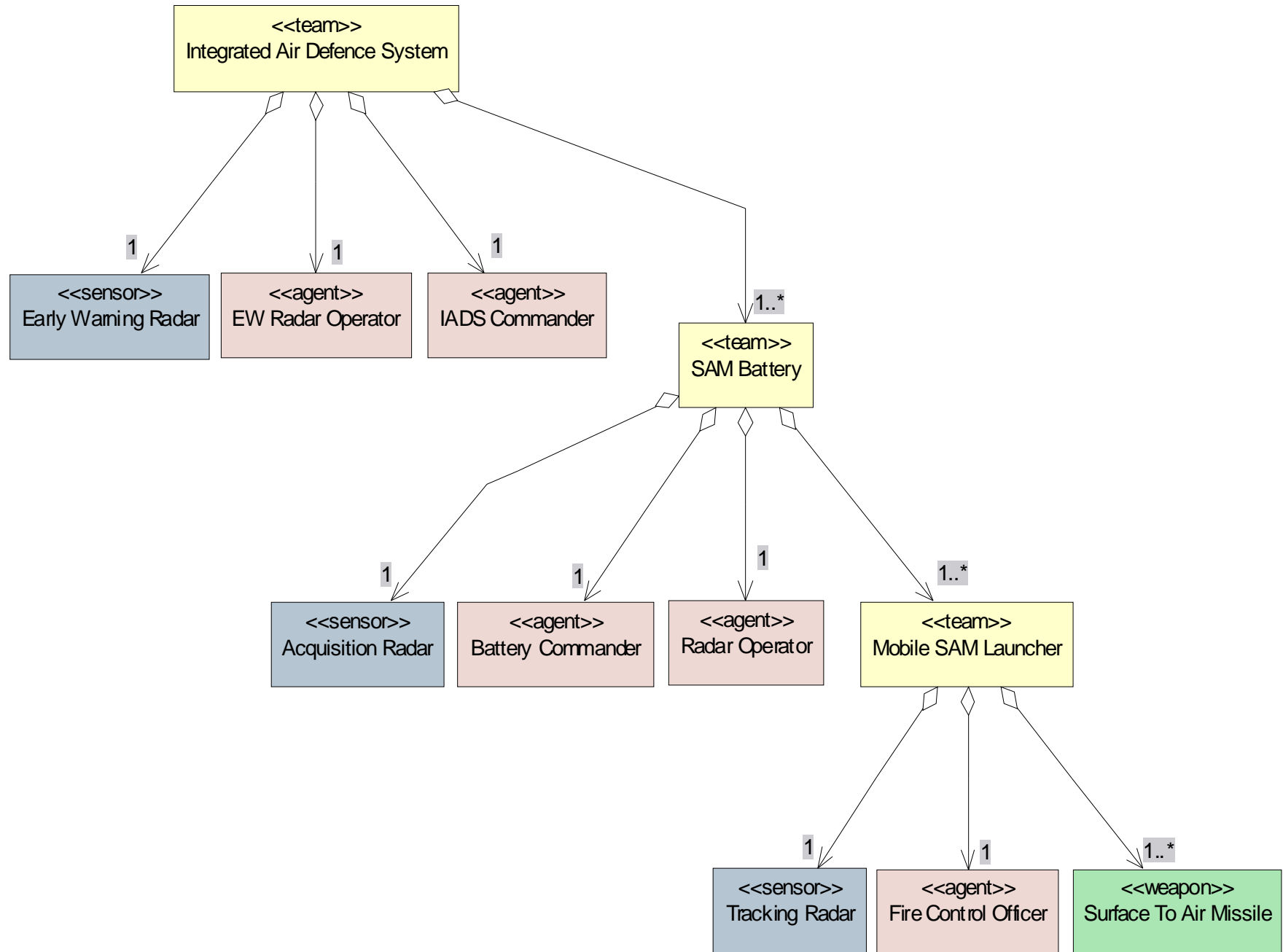


# Plan Body Specified with Activity Diagram



# JACK Capabilities in UML





# UML Tool Support for Jack Extensions





**Class Specification for DestroyTarget**

Components: General | Detail | Operations | Attributes | Relations | IDL | Java | Files

Report File Name: D:\Jack-Agents\SamOperator

Report Type: SAM Operator Jack Agent Design

Report Options:  
 Logical View Report  
 Component View Report

Attributes And Operations Syntax:  
 Use Unified Modeling Language Syntax  
 Use Visual Basic Syntax  
 Use C++ Syntax

Report Options:  
 Include Operations  
 Include Attributes  
 Public Operations And Attributes Only  
 Include Documentation

Buttons: Generate Selected, Generate, Cancel

Buttons: OK, Cancel, Apply, Browse, Help

Class: DestroyTarget  
 Show classes  
Implementation

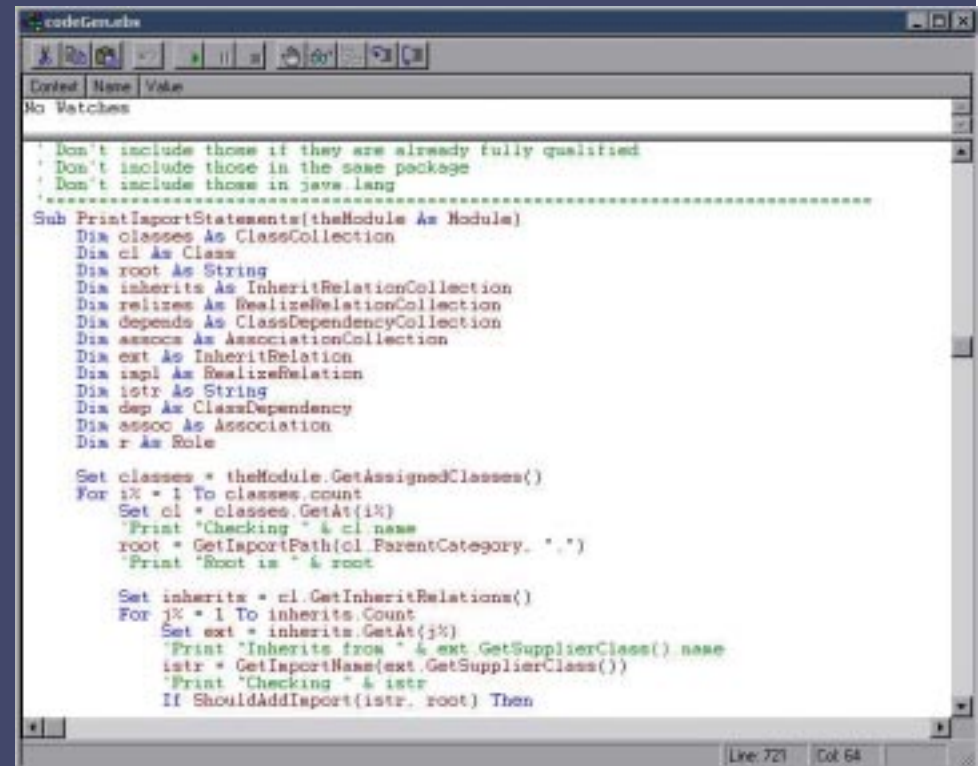
Logical View Hierarchy:  
Use Case View  
Logical View  
Main  
All Associations  
QuickDraw  
QuickDraw2  
QuickDraw3  
Events  
-> <<BDIGoalEvent>>  
-> <<Event>> HostileTr  
-> <<Event>> NewTrac  
Databases  
-> <<Database>> Acq  
-> <<Database>> Laur  
Agents  
-> <<Agent>> SamCom  
Plans  
-> <<Plan>> SelectLau  
-> <<Plan>> IdentityTr  
-> <<Plan>> DestroyTh  
id  
latitude  
longitude  
altitude  
launcher  
relevant  
context  
body  
-> <<reasoning>> getLaunchSite  
Data  
Component View  
Deployment View

HostileTrack  
trackid : long  
latitude : float

This reasoning method obtains the closest launch site for the Sam Commander.

# Jack Code Generation

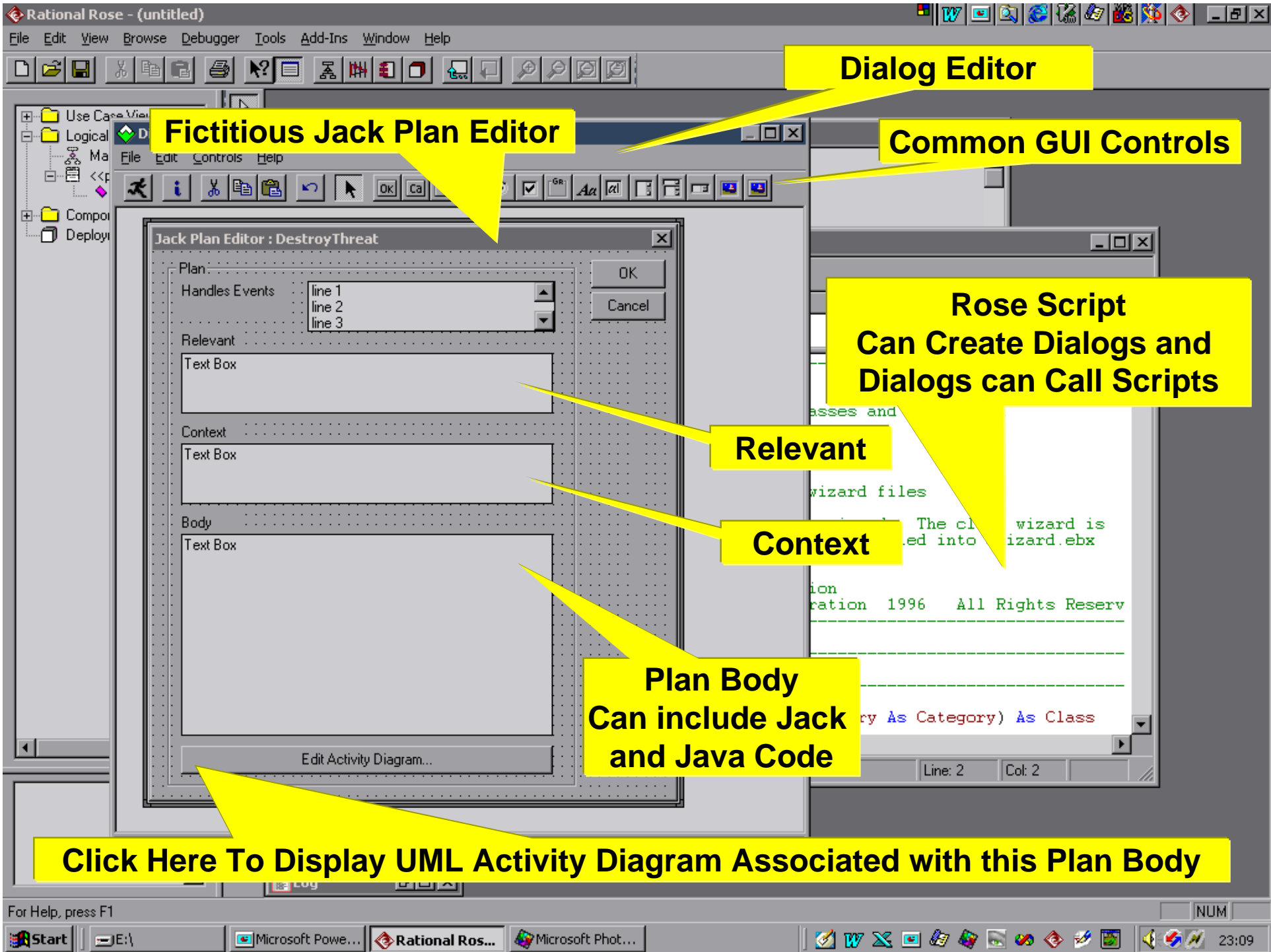
- Can easily write a Jack code generator
- RoseScript = VBScript + REI (VBA)
- Java Code Scripts already available - can be modified
- Jack Script will need to handle stereotypes correctly
- Only stubs in first cut



```
codeGen.vbs
-----
| Control | Name | Value |
|-----|-----|-----|
| No Watches |
|-----|-----|-----|
| Don't include those if they are already fully qualified
| Don't include those in the same package
| Don't include those in java.lang
|-----|-----|-----|
Sub PrintImportStatements(theModule As Module)
  Dim classes As ClassCollection
  Dim cl As Class
  Dim root As String
  Dim inherits As InheritRelationCollection
  Dim realizes As RealizeRelationCollection
  Dim depends As ClassDependencyCollection
  Dim assoc As AssociationCollection
  Dim ext As InheritRelation
  Dim impl As RealizeRelation
  Dim istr As String
  Dim dep As ClassDependency
  Dim assoc As Association
  Dim r As Role

  Set classes = theModule.GetAssignedClasses()
  For iX = 1 To classes.Count
    Set cl = classes.GetAt(iX)
    'Print "Checking " & cl.name
    root = GetImportPath(cl.ParentCategory, ".")
    'Print "Root is " & root

    Set inherits = cl.GetInheritRelations()
    For jX = 1 To inherits.Count
      Set ext = inherits.GetAt(jX)
      'Print "Inherits from " & ext.GetSupplierClass().name
      istr = GetImportName(ext.GetSupplierClass())
      'Print "Checking " & istr
      If ShouldAddImport(istr, root) Then
```



**Dialog Editor**

**Fictitious Jack Plan Editor**

**Common GUI Controls**

**Rose Script  
Can Create Dialogs and  
Dialogs can Call Scripts**

**Relevant**

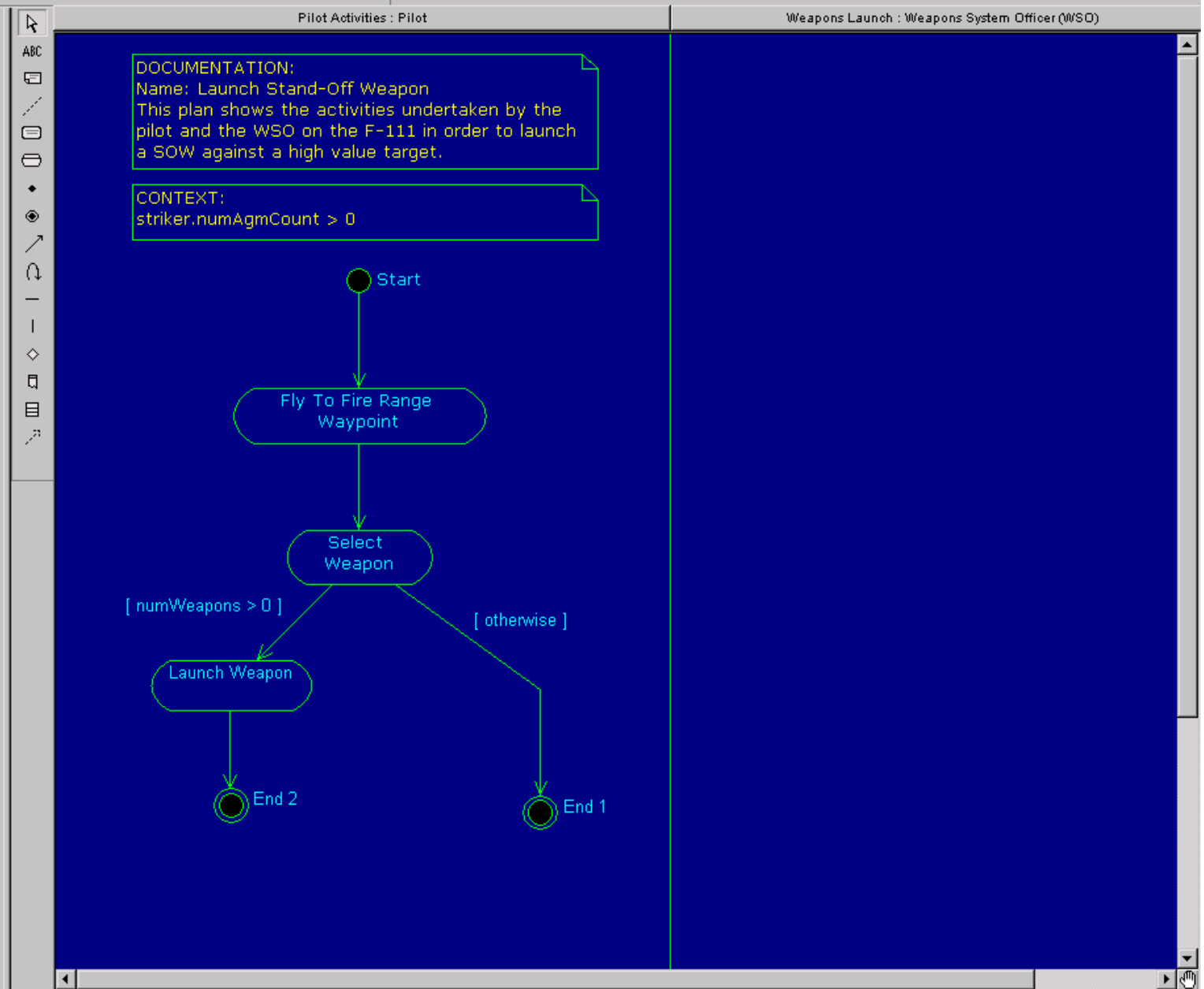
**Context**

**Plan Body  
Can include Jack  
and Java Code**

**Click Here To Display UML Activity Diagram Associated with this Plan Body**



- dMARS\_UML
  - Use Case View
    - Main
      - Electronic Warfare Officer (EWO)
      - Navigator
      - Pilot
      - Weapons System Officer (WSO)
    - State/Activity Model
      - NewDiagram
        - End 1
        - End 2
        - Start
        - Fly To Fire Range Waypoint
        - Launch Weapon
        - Select Weapon
        - NewSwimlane
        - Pilot Activities : Pilot
        - Weapons Launch : Weapons
  - Associations
  - Logical View
    - Main
    - NewClass
    - Associations
  - Component View
  - Deployment View
  - Model Properties



# Reuse and Knowledge Management



*Does the UML support reuse of software components across systems?*

*Does the UML assist with knowledge management?*

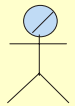
## Ongoing Investigation – Sneak Preview

- Business Modelling
  - Agent Modelling
  - Domain Modelling
  - Knowledge Management
  - Reuse
- 
- At least one tangible output - Agent Symbology

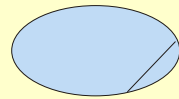
# UML Agent Symbology



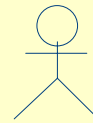
# Use Case Symbols



Business Actor



Business Use Case



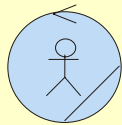
Actor



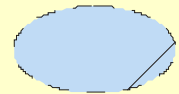
Use Case



Use Case Realisation



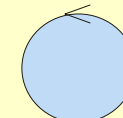
Business Worker



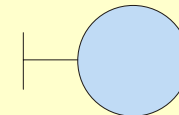
Business Use Case Realisation



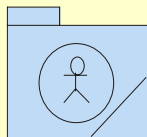
Domain



Control Class



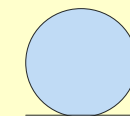
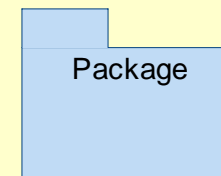
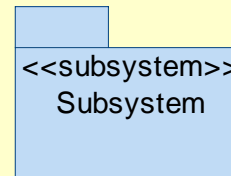
Boundary Class



Organisation Unit

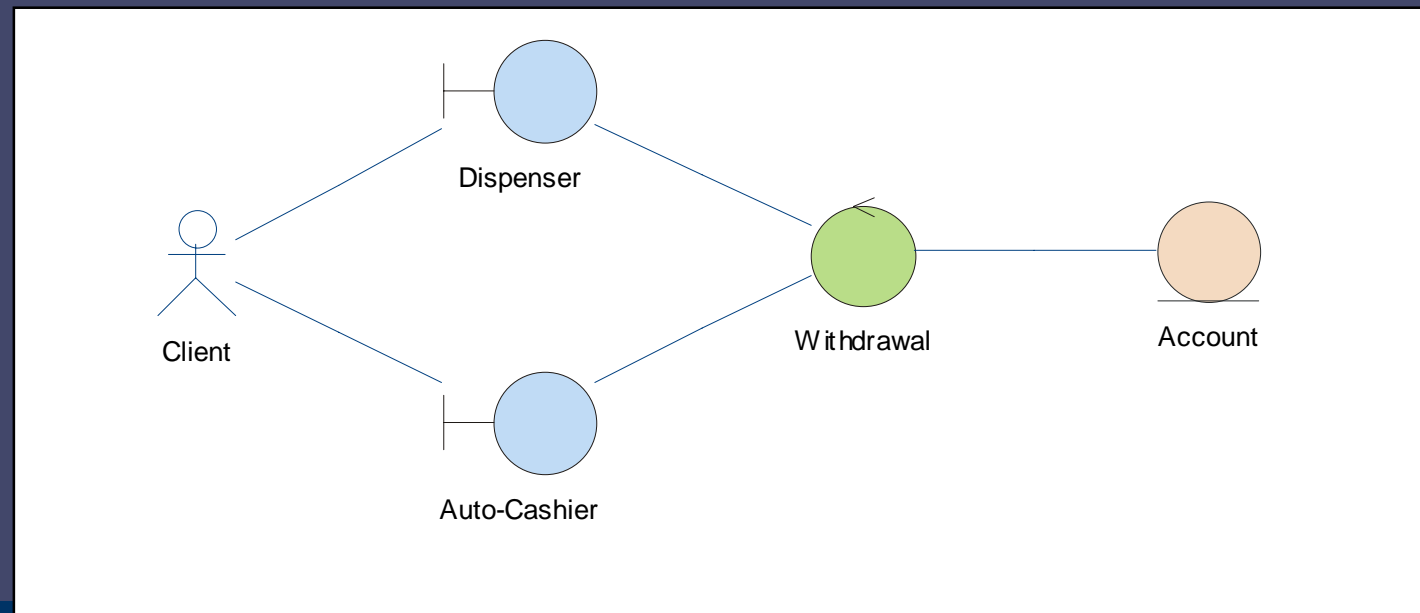
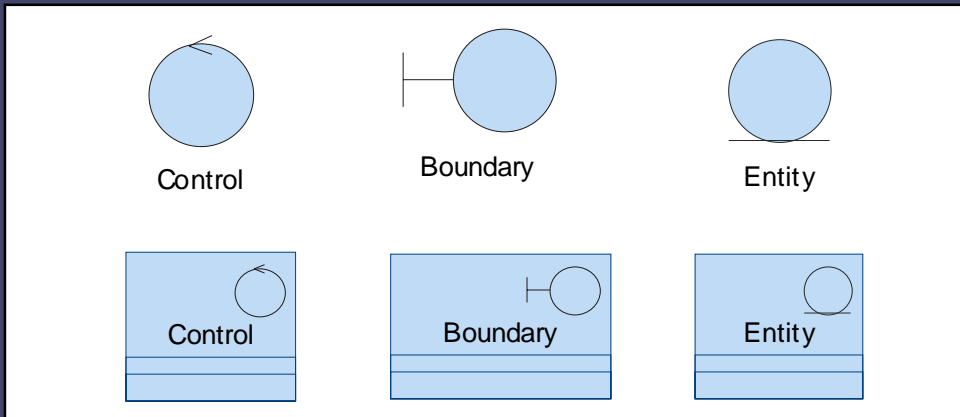


Domain Package

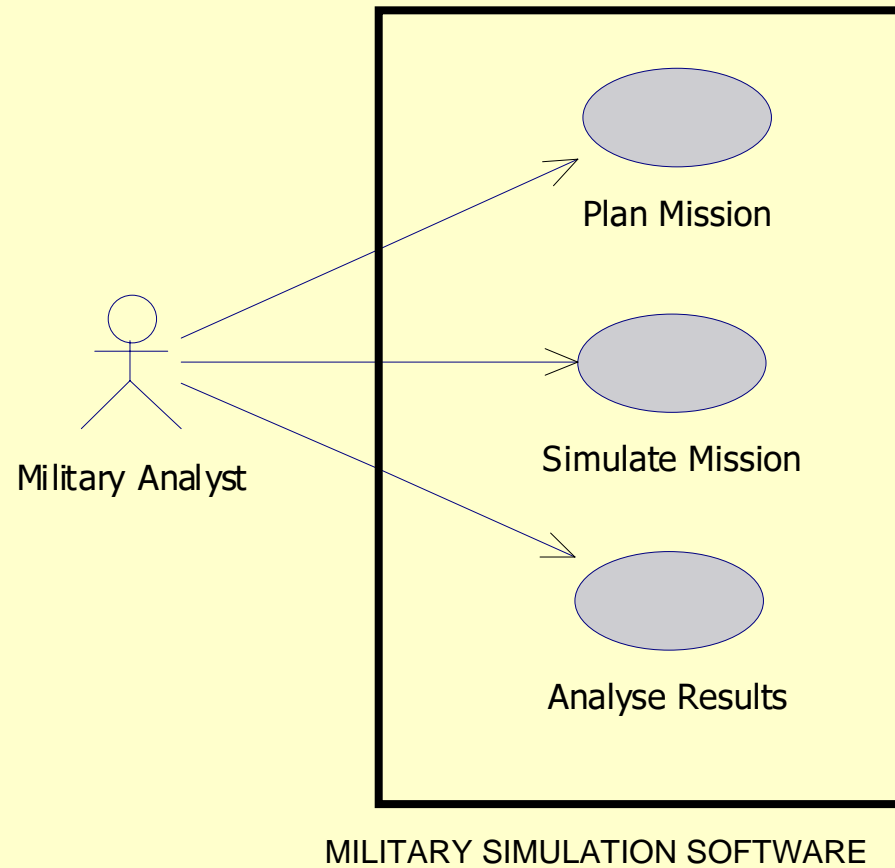


Entity Class

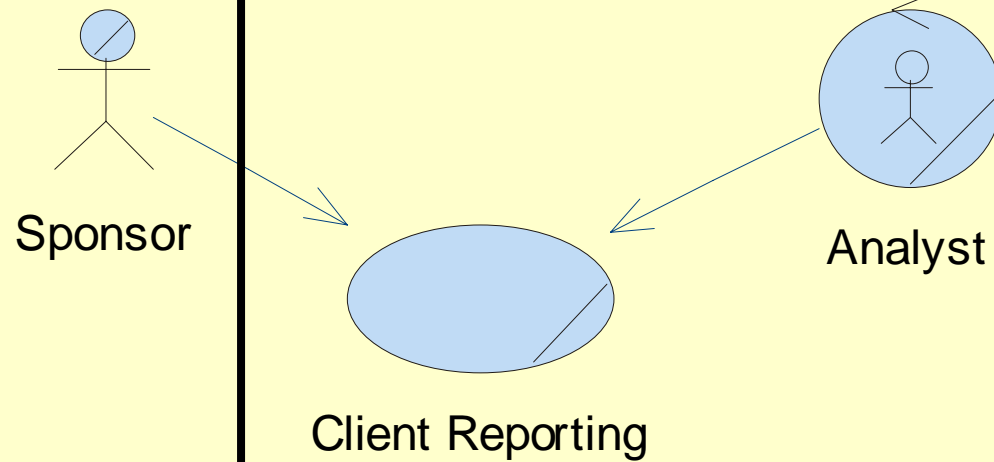
# Analysis Classes (Control, Boundary, Entity)



# Use Case Modelling

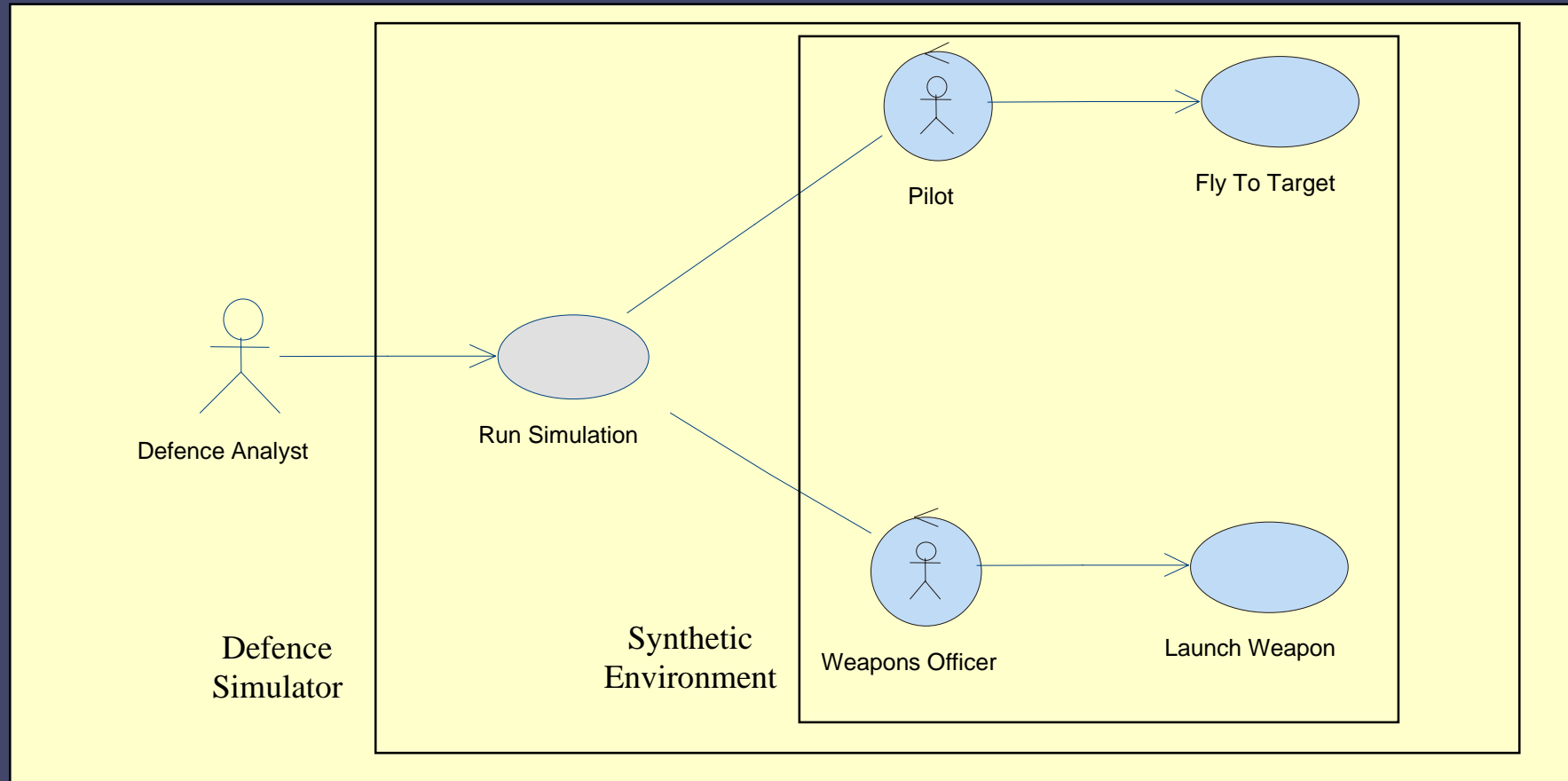


# Business Modelling

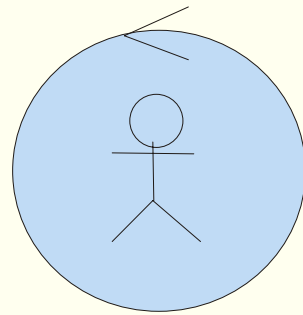


AIR OPERATIONAL ANALYSIS BRANCH

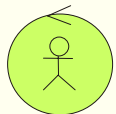
# Agent Use Case Modelling



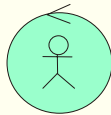
Therefore, the symbol for **Agent** should be...



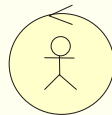
Agent



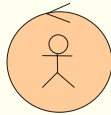
Medical  
Diagnosis  
Agent



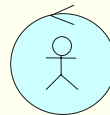
B2B  
Agent



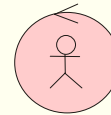
Web  
Spider



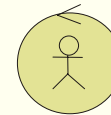
Online  
Shopper



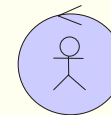
Fighter  
Controller



Quake  
Bot



Pilot



Mobile  
Agent

# Generalising Agent Extensions



*Is there are general set of agent extensions?*

*What is it?*

## Extending the UML for all BDI Agent?

- Beliefs
- Goals and Events
- Plans and Intentions

*Can the UML be extended to accommodate many different types of Agents?*

