MDA at the MoD

Welcome

Chris Raistrick
chris.raistrick@abstractsolutions.co.uk

MINISTRY OF DEFENCE

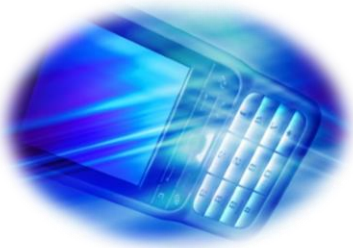
Agenda

- Abstract Solutions and Military OSAs
- The MoD Requirement
- MDA for LOSA
- Model Verification and Deployment
- Summary

Abstract Solutions

Formerly Kennedy Carter

Since 1989, specialists in helping organisations adopt



MDD for Systems and Software

through

Training – Consulting - Mentoring

MDA with UML for Open System Architectures

MDA has been adopted for a number of military architectures…

US OSD UCS Architecture

Mandated by DoD for all UAVs over 20 pounds
Evolution of:

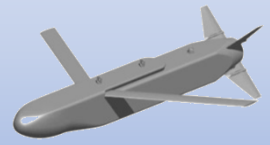
- NIAG SG97 (ALWI)
- NIAG SG125 (UAS Architecture)

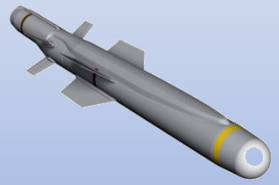
www.ucsarchitecture.org

Deployed in US

Complex Weapons OSA

Standardised architecture for all stages of kill chain.





Generic Vehicle Architecture / LOSA

MINISTRY OF DEFENCE

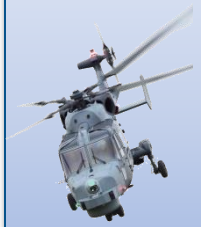
For platform and equipment interoperability

Deployed in UK. Adopted by NATO
Now being extended for soldiers and bases.

Weapon Integration UK

WIUK
WEAPONS INTEGRATION UK

PnP Architecture for weaponised aircraft
Evolution of NATO ALWI Architecture

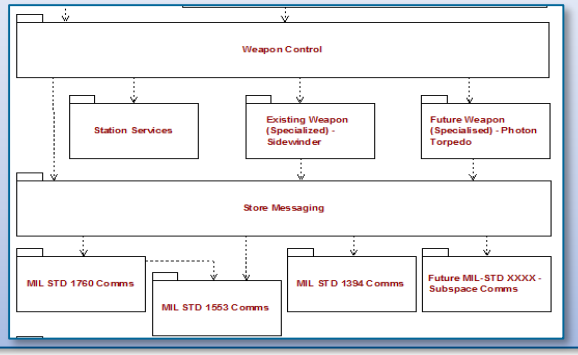


Deployed in UK and US

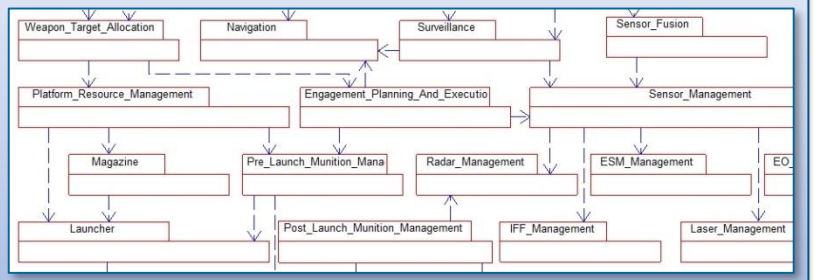
MDA with UML for Open System Architectures

These military open system architectures embody the key principles of Model Driven Architecture (MDA):

US OSD UCS Architecture



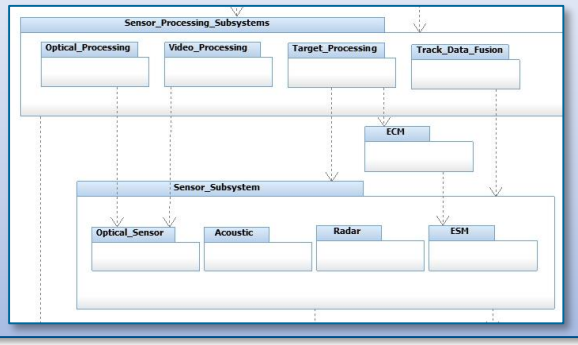
Complex Weapons OSA



- **Domain based partitioning**
- **Platform independent models**
- **Automatically generated platform-specific artefacts**

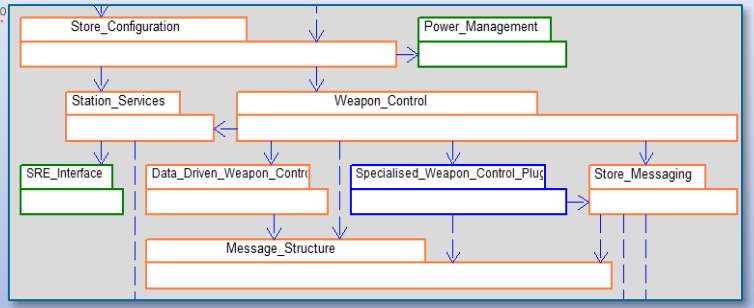
Generic Vehicle Architecture / LOSA

MINISTRY OF DEFENCE



Weapon Integration UK

WIUK WEAPONS INTEGRATIO



Abstract Solutions

MDA at the MoD

The MoD Requirement

MINISTRY OF DEFENCE

Chris Raistrick
chris.raistrick@abstractsolutions.co.uk

Requirements for LOSA

Affordability

- To minimise the cost of upgrades and UORs

Flexibility & Portability

- To allow deployment on different installations (base, vehicle, soldier)

Managed Complexity and Open Standards

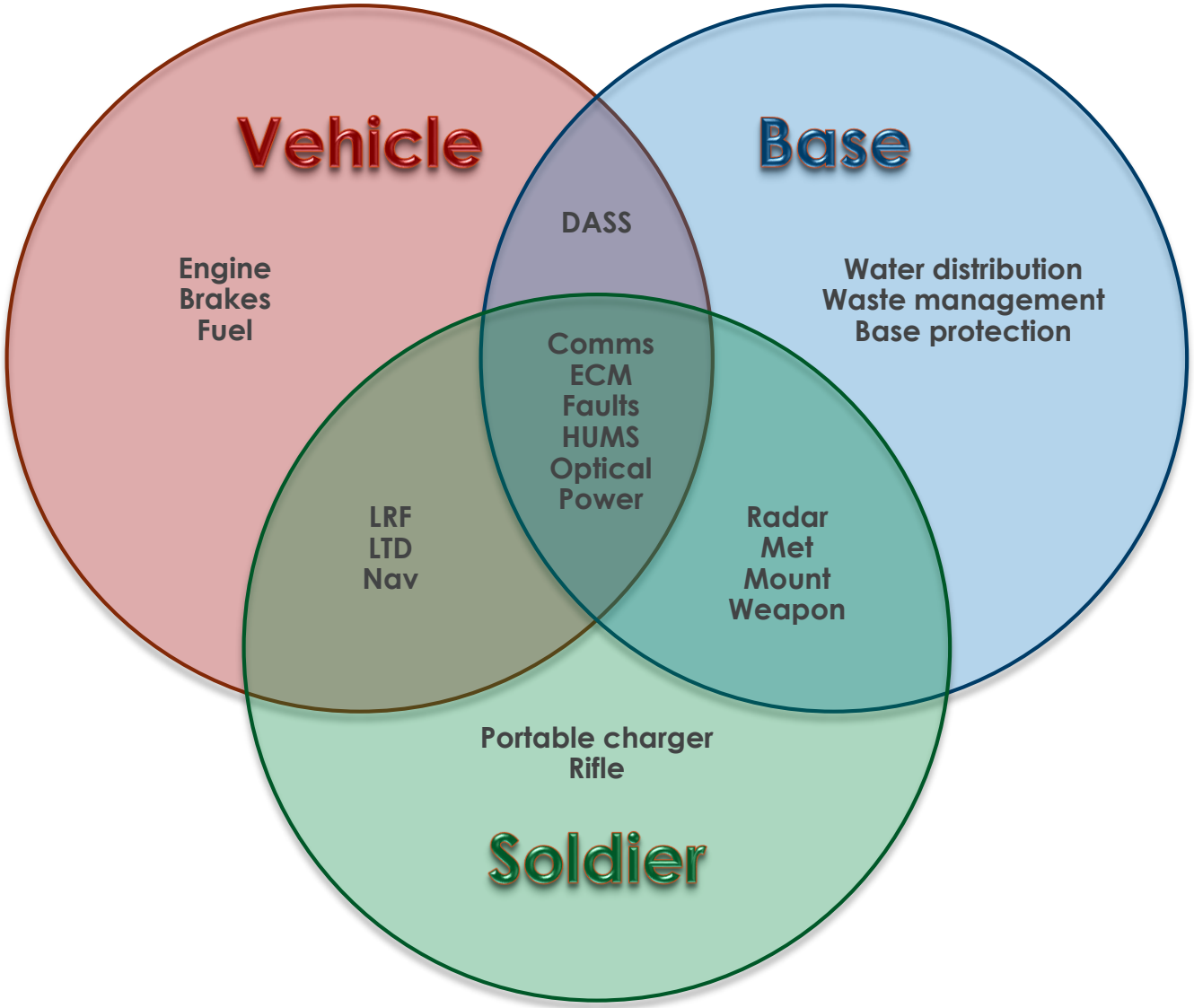
- To facilitate universal understanding and reduce integration risk for systems with complex and rapidly evolving requirements

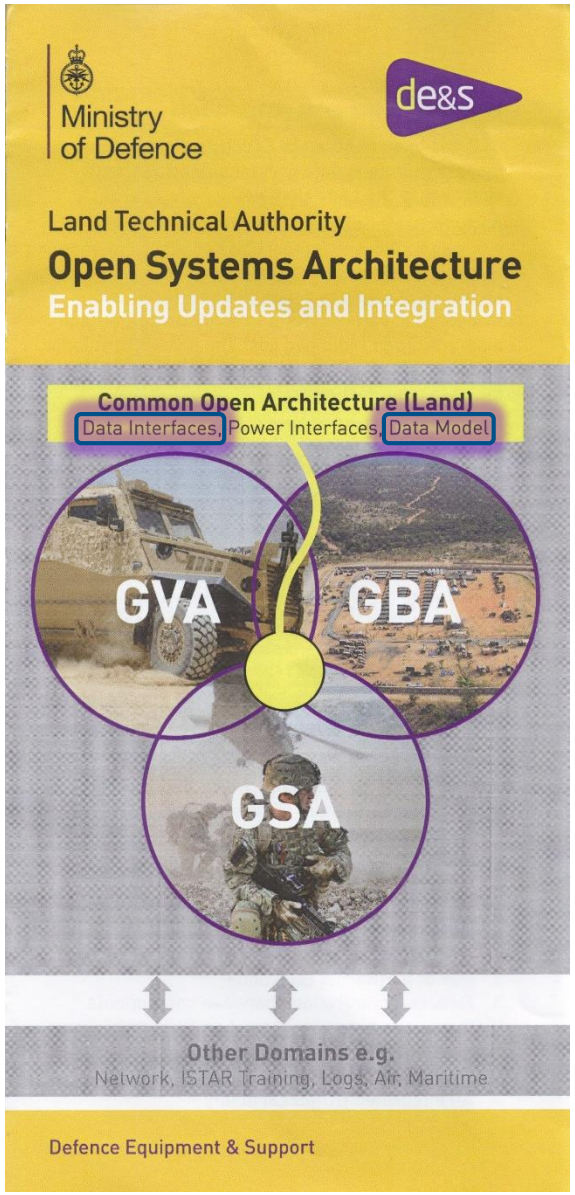
Longevity and Scalability

- In keeping with the long lifetimes of the deployed systems

The Requirement for Reusable Modules

Most modules are applicable to multiple installation types…





The Requirement for Portable Modules

Middleware and messaging technologies:

- **evolve and are superseded over time**
- **vary from system to system**

To enable easy adoption of different software technologies, the generic architecture model must be:

- **independent of middleware and software architecture**

- **independent of messaging technologies and message passing strategies**

IDL

The Java Message Service

A Platform Independent Model
is a long-life reusable asset

MDA at the MoD

MDA for LOSA

MINISTRY OF DEFENCE

Chris Raistrick
chris.raistrick@abstractsolutions.co.uk

Model Driven Architecture Principles

The OMG Model Driven Architecture embeds three key principles:

- **Domain Partitioning of the System**
- **Platform Independent Modelling of each Domain**
- **Automated Generation of the Platform Specific artefacts**

These principles are designed to achieve specific goals:

- **Component Reuse through pollution control**
- **Model Longevity through platform independence**
- **Easier Upgrades through extensible layered architecture and data driven models**

Important Abbreviations

see
omg.org/mda

The process for system development, known as
Model Driven Architecture (MDA)
involves building
Platform-Independent Models (PIMs)
from which we derive
Platform-Specific Models (PSMs)
and/or
Platform-Specific Implementations (PSIs).

The models are represented using the notation known as the
Unified Modeling Language (UML).
Both the MDA process and the UML notation
are owned by the non-profit consortium known as the
Object Management Group (OMG).

Platform Independent Model

A Platform Independent Model (PIM) is a technology agnostic model, which fully specifies the capabilities and data provided by the system, but suppresses detail concerning:

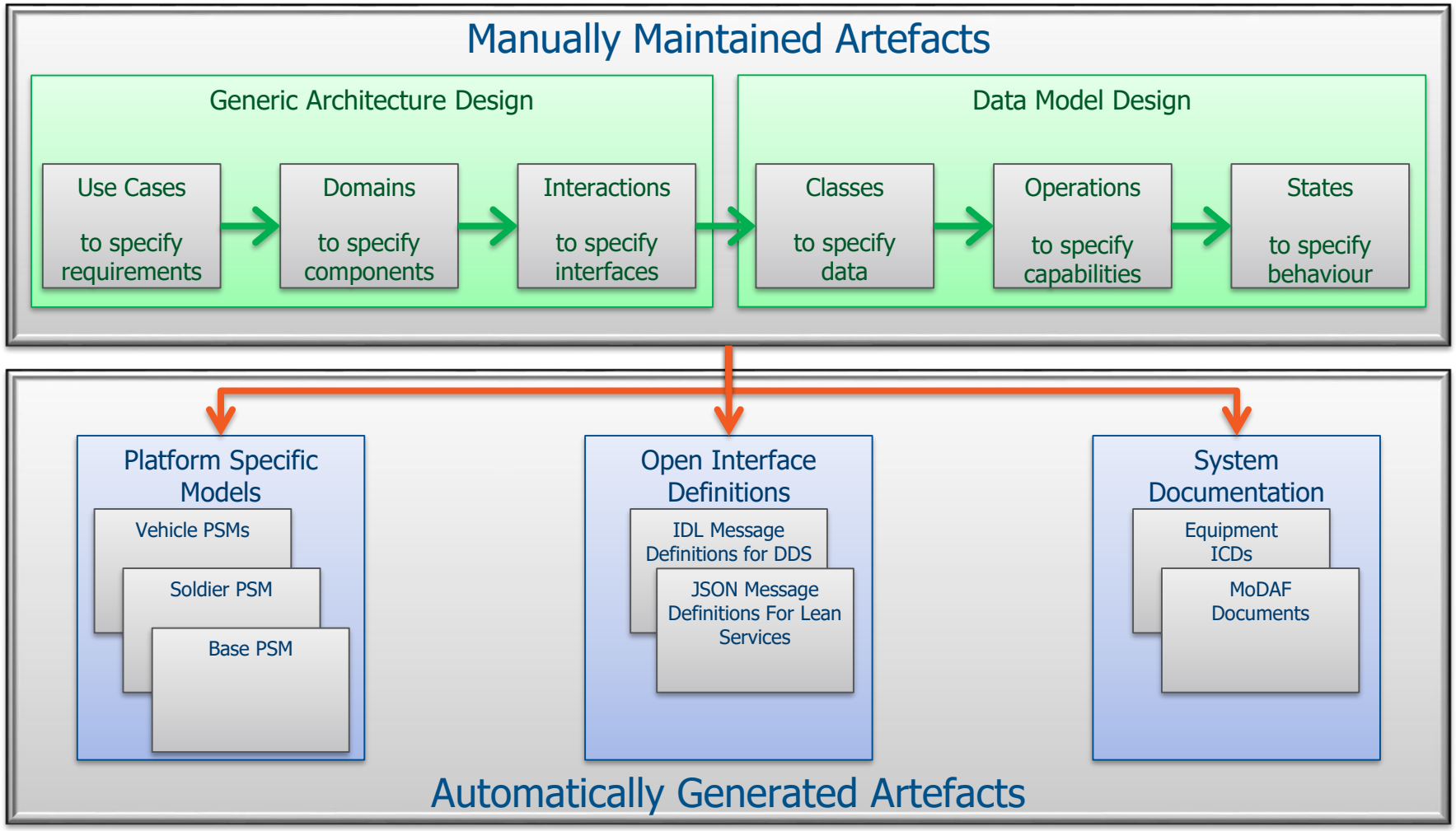
- **Hardware architecture**
- **Software architecture**
- **Middleware**
- **Messaging technology**

"Platform" in this context refers to the "execution platform"…
…although we aim to be vehicle platform independent too.

A PIM is therefore much simpler than a Platform-Specific Model (PSM)

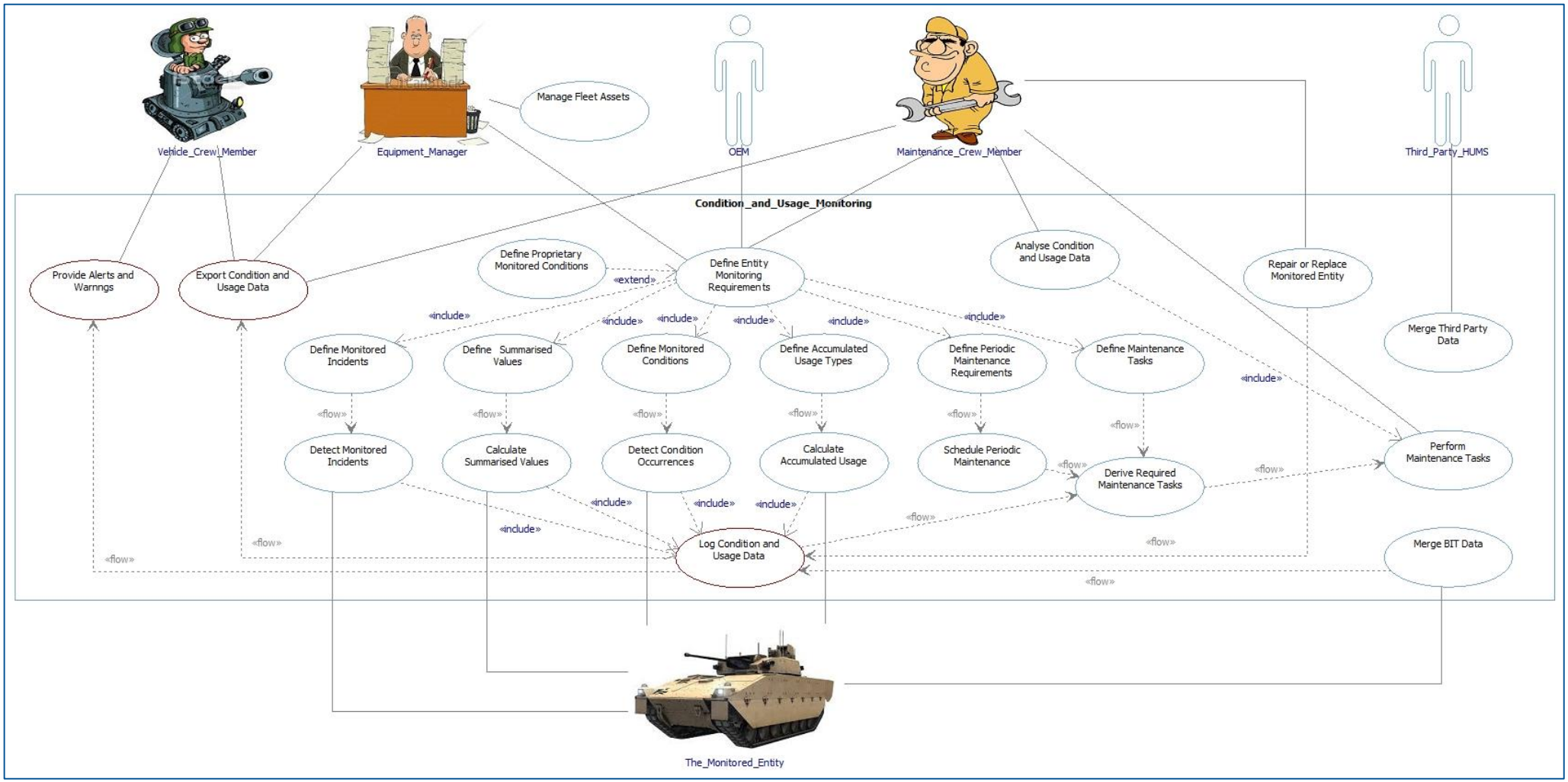
The MDA System for Building Systems

- Use case analysis to specify capabilities
- Reusable domain-based platform independent models
- Translation of model onto multiple platforms



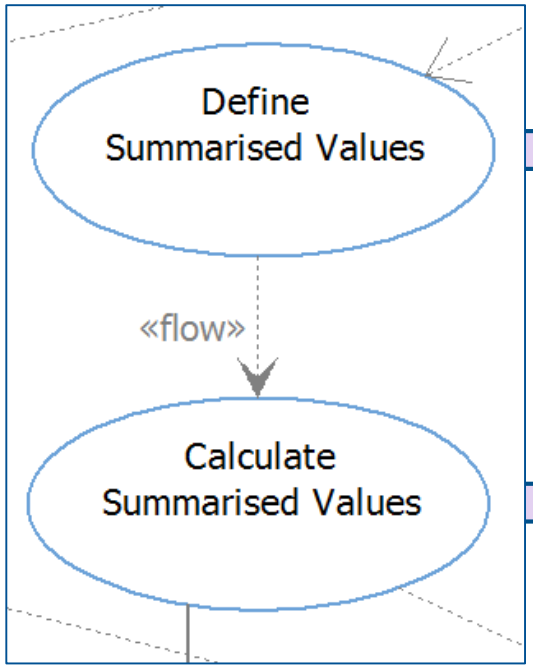
Usage and Condition Monitoring Use Case Model

Use cases specify the capabilities to be provided by the system, and how they interact with humans (e.g. Crew Member) and equipment (e.g. Monitored Entity)



Usage and Condition Monitoring Class Model

The use case model provides the basis for building the PIM…



Note that configuration data shown in compartment 4 is not part of the model

Monitored_Entity_Usage_Specification
usageThresholdTypeRhpString
monitoredEntityTypeName:RhpString
usageUnitType:UsageUnitType
Engine / Engine_Running / miles
Headlight / Headlight_On / hours

Monitored_Entity_Specification
monitoredEntityTypeName:RhpString
Engine
Headlight

Monitored_Characteristic_Specification
historySizeRhpInteger
logOnlyWhenExceedenceInProgress:RhpBool...
monitoredCharacteristicNameRhpString
monitoredEntityTypeName:RhpString
pollingIntervalInSeconds:RhpInteger
Engine / RPM / 10
Engine / Oil_Temp / 60

Summarised_Value_During_Usage_Period_Specification
monitoredEntityTypeName:RhpString
summarisedCharacteristicName:RhpString
summarisedValueType:RhpString
usageTypeName:UsageUnitType
Engine_Running / Engine_RPM / Max

Monitored_Repeated_Incident_Specification
actionTypeName:RhpString
incidentTypeName:RhpString
logOccurrences:RhpBoolean
noOfRepeatsForAction:RhpInteger
Lightning / 10 / Check_Lightning_Conductor

Actual_Repeated_Incident_Occurrence
incidentDateTime:DateTimeType

Actual_Summarised_Value_During_Period
monitoredEntityId:RhpInteger
summarisedCharacteristicName:RhpString
summarisedValue:RhpReal
summarisedValueType:RhpString
The_Engine / RPM / Max / 6500

Actual_Repeated_Incident
countSinceLastTrigger:RhpInteger
resetCountSinceLastTrigger():void

Monitored_Entity_Usage_Period
endDateTime:DateTimeType
startDateTime:DateTimeType
usageCount:RhpReal
usageTypeName:RhpString
setEndUsagePeriod()
startUsagePeriod():void
The_Engine / Engine_Running / 14Jan2015-09:00:00 / 14Jan2015-10:30:00 / 20.0
Left_Headlight / Headlight_On / 15Jan2015-19:00:00 / 15Jan2015-21:00:00 / 2.0

Monitored_Entity
monitoredEntityId:RhpString
monitoredEntityTypeName:RhpString
exportHUMSData():void
importHUMSDataForReplacement():void
mergeThirdPartyHUMSData():void
resetHUMSDataAfterMaintenance():void
Engine / The_Engine
Headlight / Left_Headlight
Headlight / Right_Headlight

Monitored_Characteristic_Value
dateTimeLogged:DateTimeType
monitoredCharacteristicNameRhpString
recordedValue:RhpReal
recordCharacteristicValue():void
Engine / RPM / 14Jan2015-09:00:10 / 2500.0
Engine / RPM / 14Jan2015-09:30:50 / 5006.0
Engine / Oil_Temp / 14Jan2015-09:30:50 / 58.0

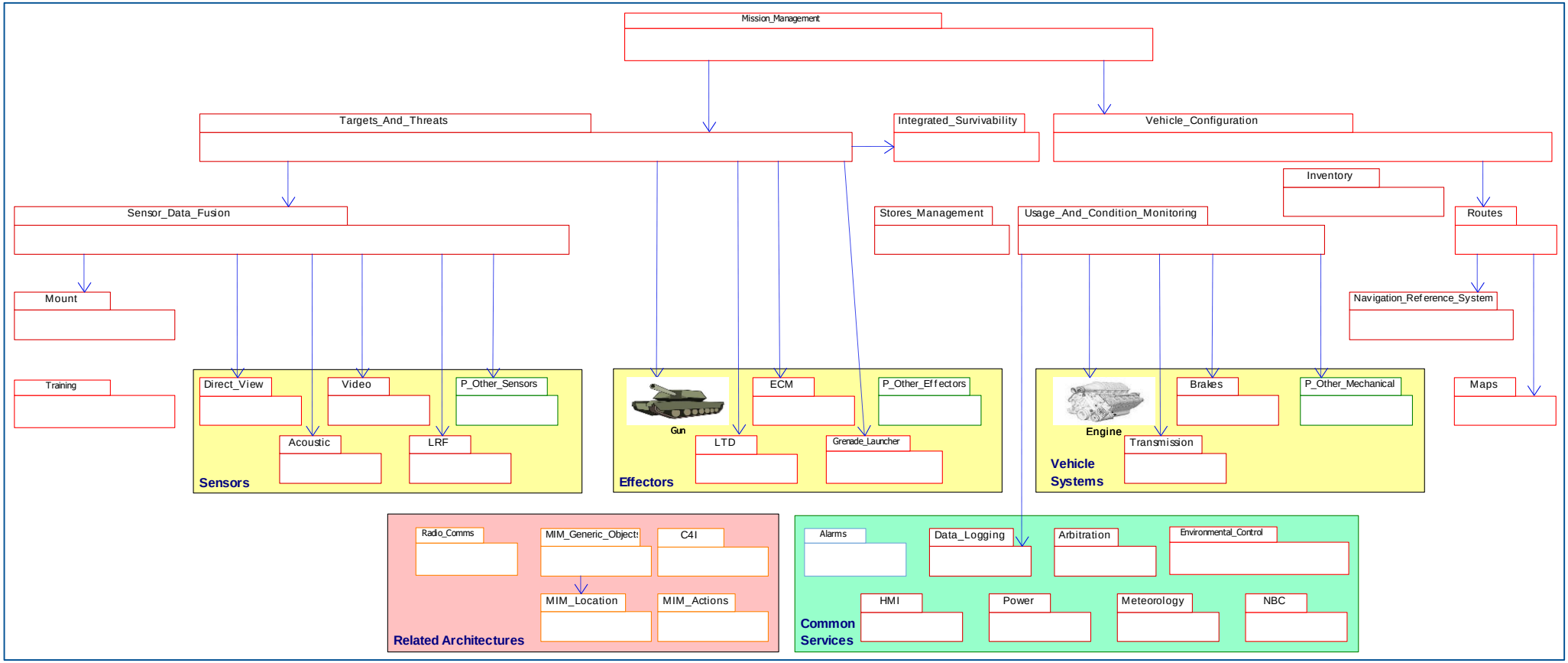
Domain Model to Provide Architectural Overview

Domains embody the subject matters, or areas of expertise, in our system

Layered data driven domains are easy to extend with new capabilities, and protect against change

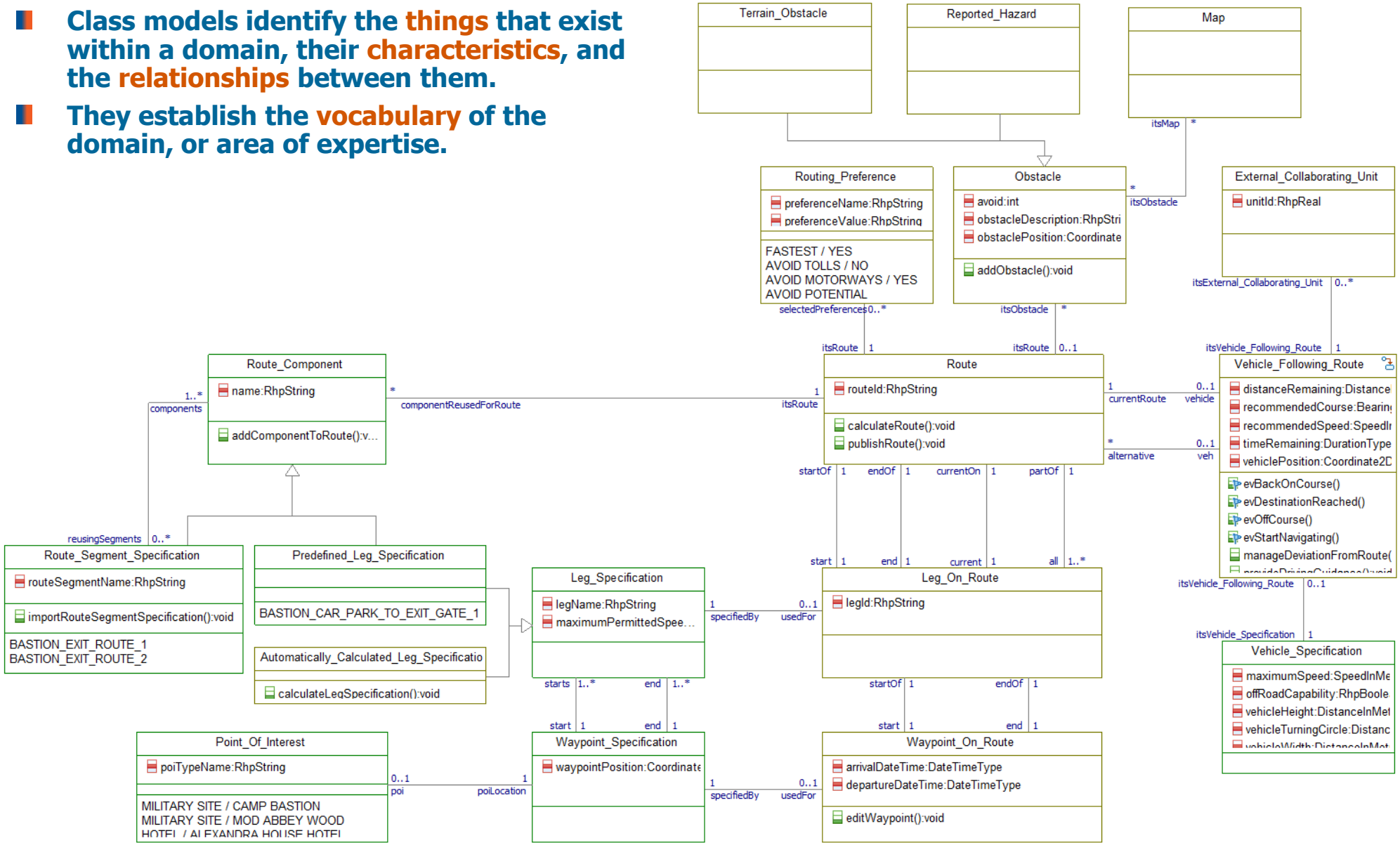
For each domain we build a Platform Independent Model (PIM) that is technology agnostic

The PIMs can then be reused and deployed on any current or future execution platform



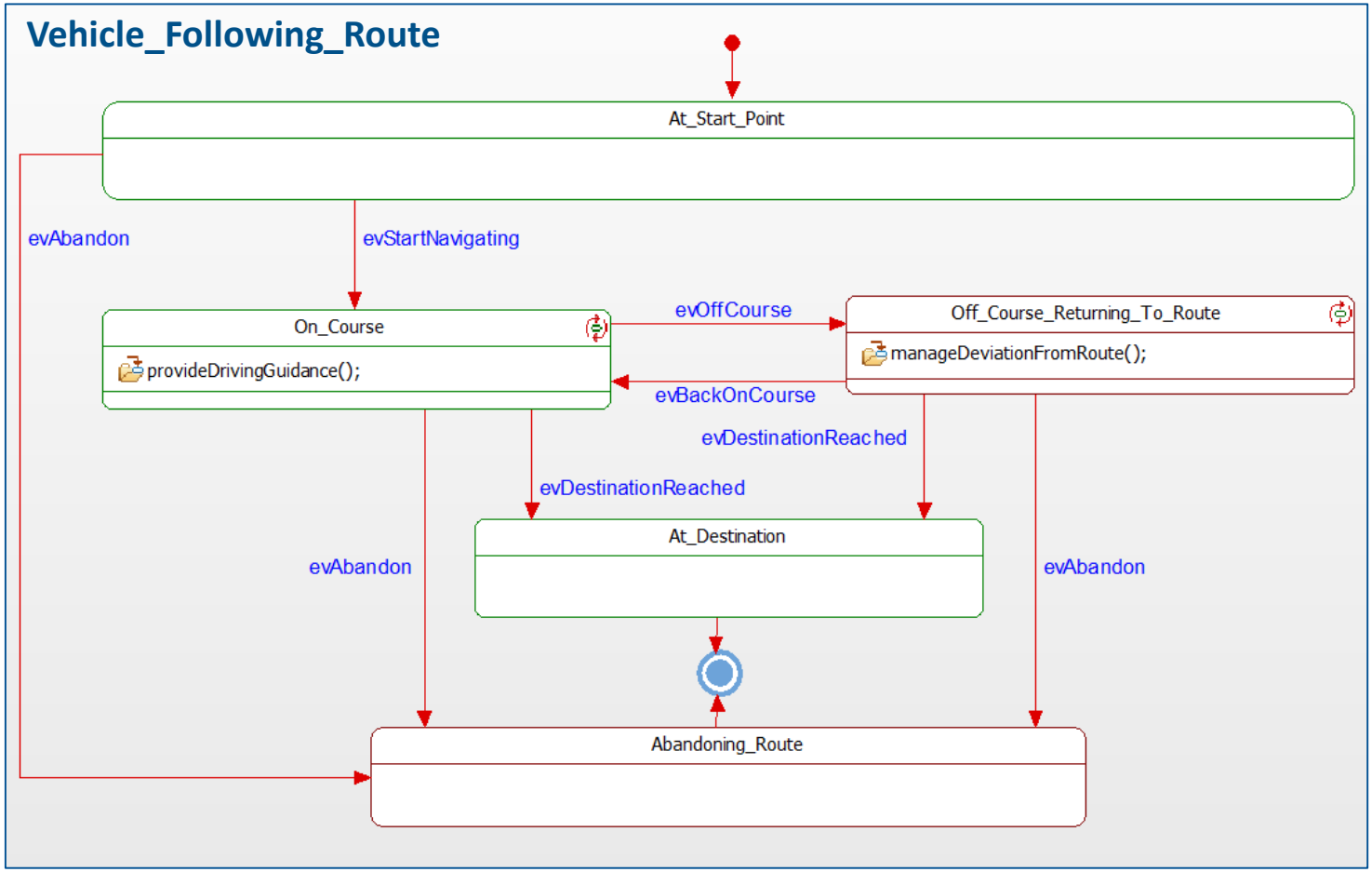
Class Model

- **Class models identify the things that exist within a domain, their characteristics, and the relationships between them.**
- **They establish the vocabulary of the domain, or area of expertise.**



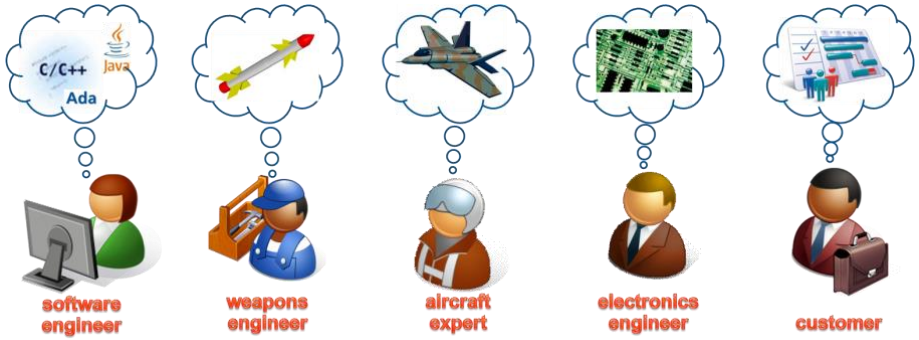
State Model

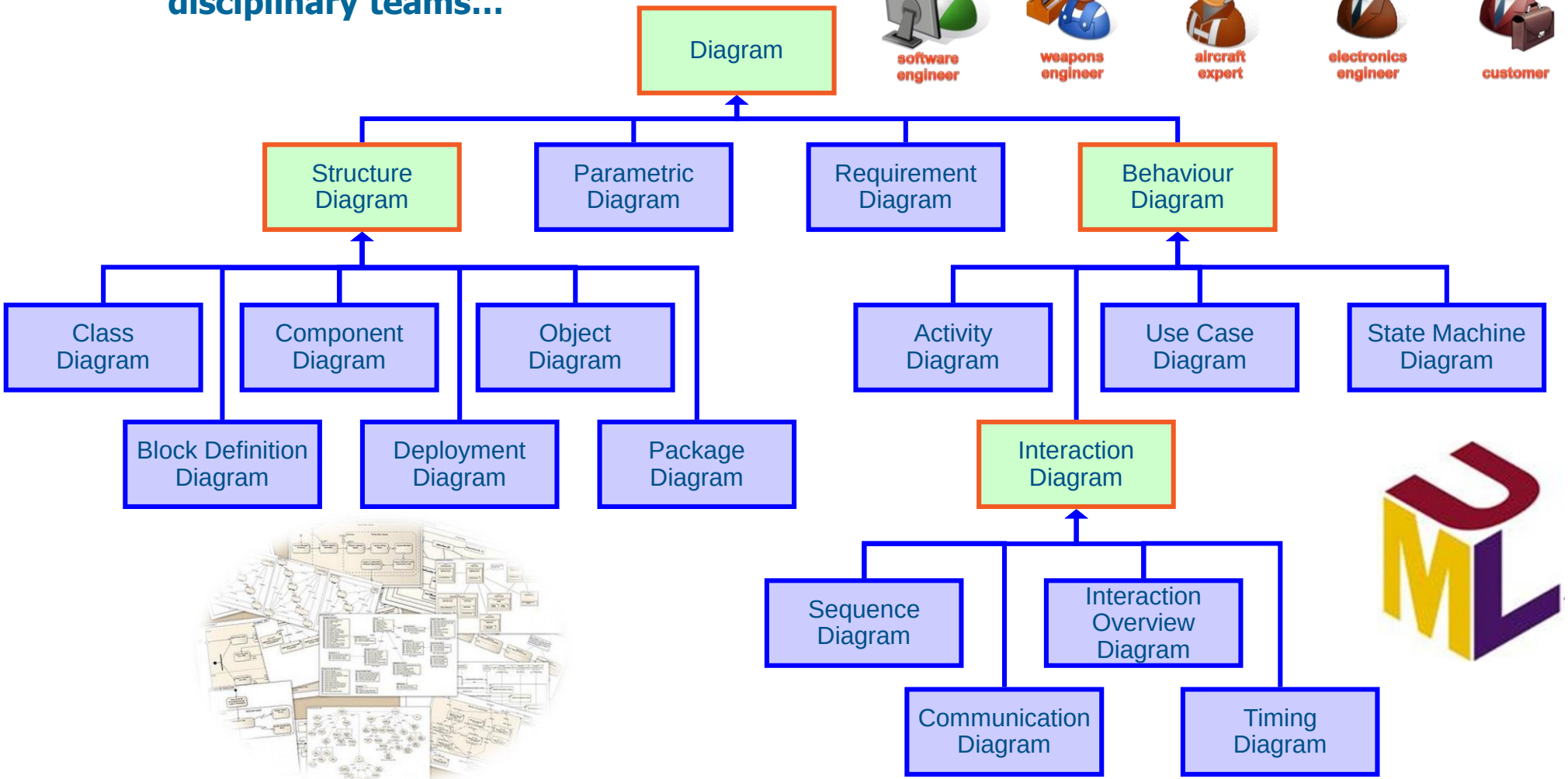
- State models capture:
  - Rules and policies defining when specific messages are valid
  - System and equipment modes
  - Behaviour (if desired)



Pitfall: Use of Complex Modelling Notations

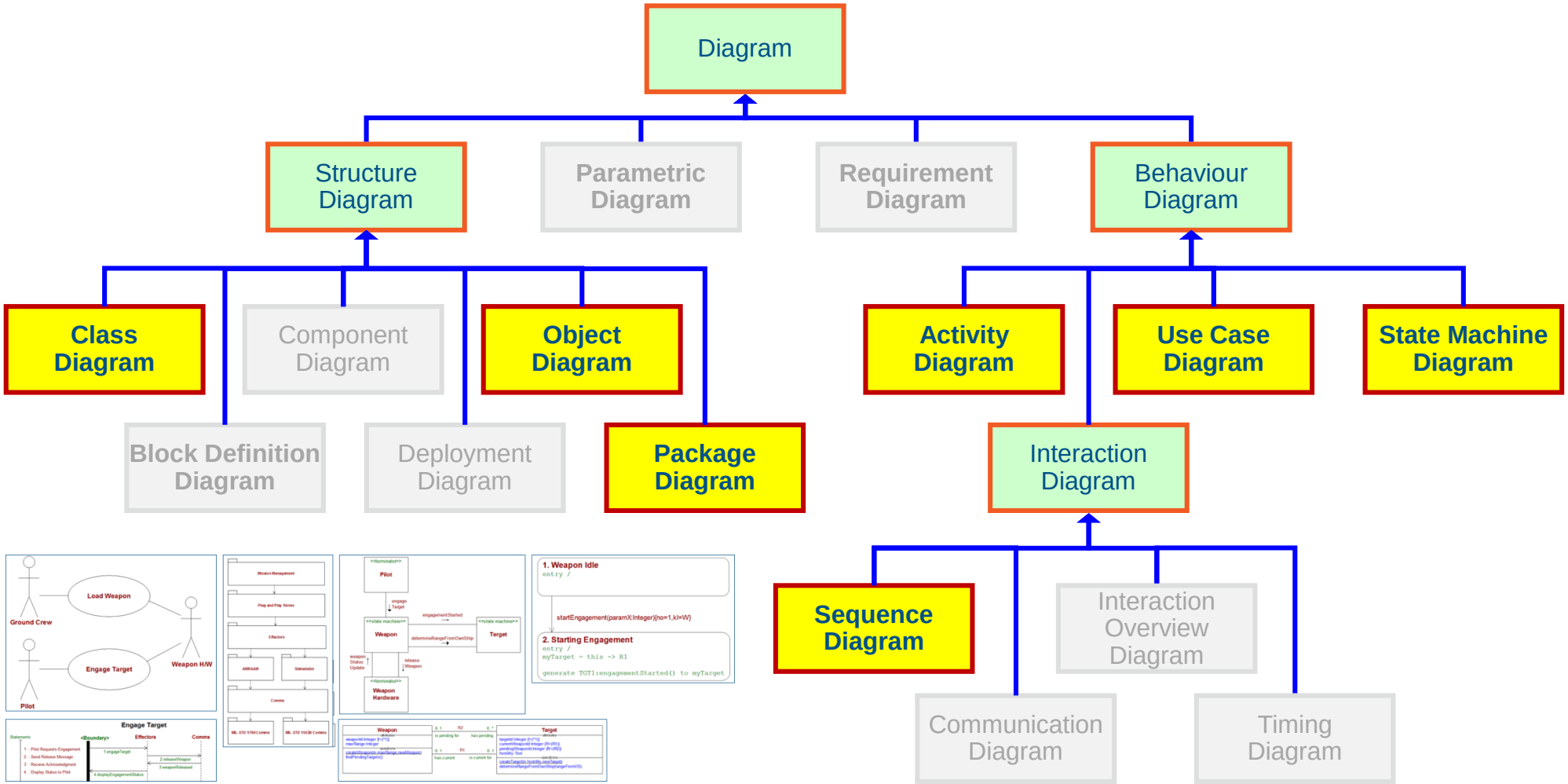
UML & SysML have evolved into a set of complex notations…
…that can create a barrier to communication in multi-disciplinary teams…





Use a Simple Modelling Notation

The primary purpose of platform independent modelling is to formalise the knowledge in the heads of a variety of subject matter experts…
…complexity is the biggest threat to collaboration…
…and use of a simple subset of UML will facilitate universal understanding



MDA at the MoD

Domain Partitioning

MINISTRY OF DEFENCE

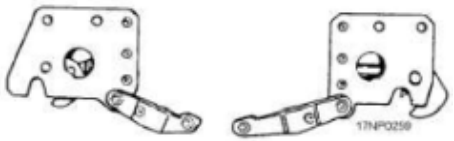
Chris Raistrick
chris.raistrick@abstractsolutions.co.uk

A Stores Management Computer

The stores management computer in a military aircraft performs a number of functions…

Manage aircraft C of G as weapons are released

Control clamps to release weapons



Provide Power and Comms via MIL-STD-1760 and MIL-STD-1553 Interfaces

Provide a display of current weapon loadout

Ensure launch safety by monitoring safety settings

Pitfall: Pollution as a Cause of Component Complexity

Models polluted with multiple unrelated concerns become unnecessarily complex…

I progress through a defined **launch sequence**

I check the **aircraft is safely balanced** after release

I release myself by opening **electrically actuated clamps**

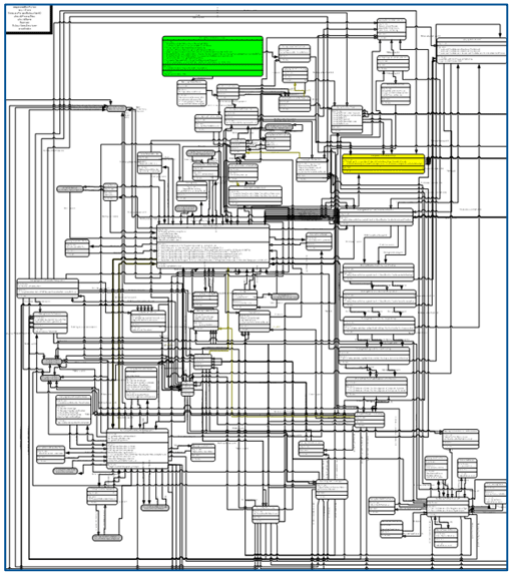
I check all **safety settings** are correct before release

I render myself as an **icon with colour** to denote status

I communicate across a MIL-STD-1553 **comms bus**

I save my state in a **DDS topic**

I am running on Greenhills INTEGRITY-178B **RTOS**

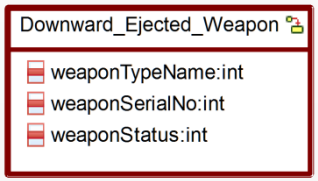


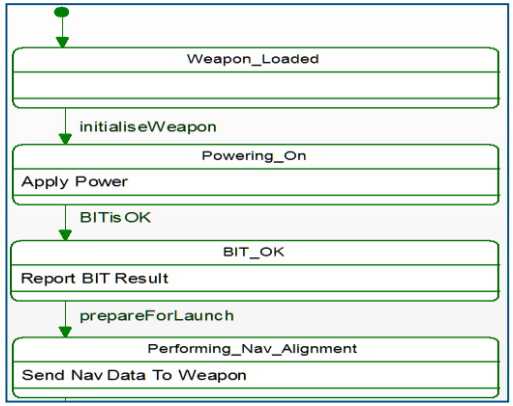
Weapon class
without domain partitioning is:

- **Complex**
- **Hard to understand**
- **Expensive to change & retest**
- **Hard to reuse**
- **Hard to port**

Separation of concerns gives highly cohesive, loosely coupled domains… which are much simpler, and highly reusable…

I must progress through a defined **launch sequence**





Weapon class
with domain partitioning is:

- **Simple**
- **Easy to understand**
- **Cheap to change & retest**
- **Easy to reuse**
- **Easy to port**

Achieve Pollution Control through Domains

- **Strategic reuse through Pollution Control**
- **Highly cohesive, loosely coupled domains**
- **Separation of concerns makes each domain much simpler…**
  …and is the key to reuse
- **Platform Independent Models become long-life corporate assets, making accumulated IP accessible and reusable**

I affect the balance of the aircraft

Store_Configuration::Actual_Store

I must check my safety settings before I am released

Station_Services::Actual_Unit

I must progress through a defined launch sequence

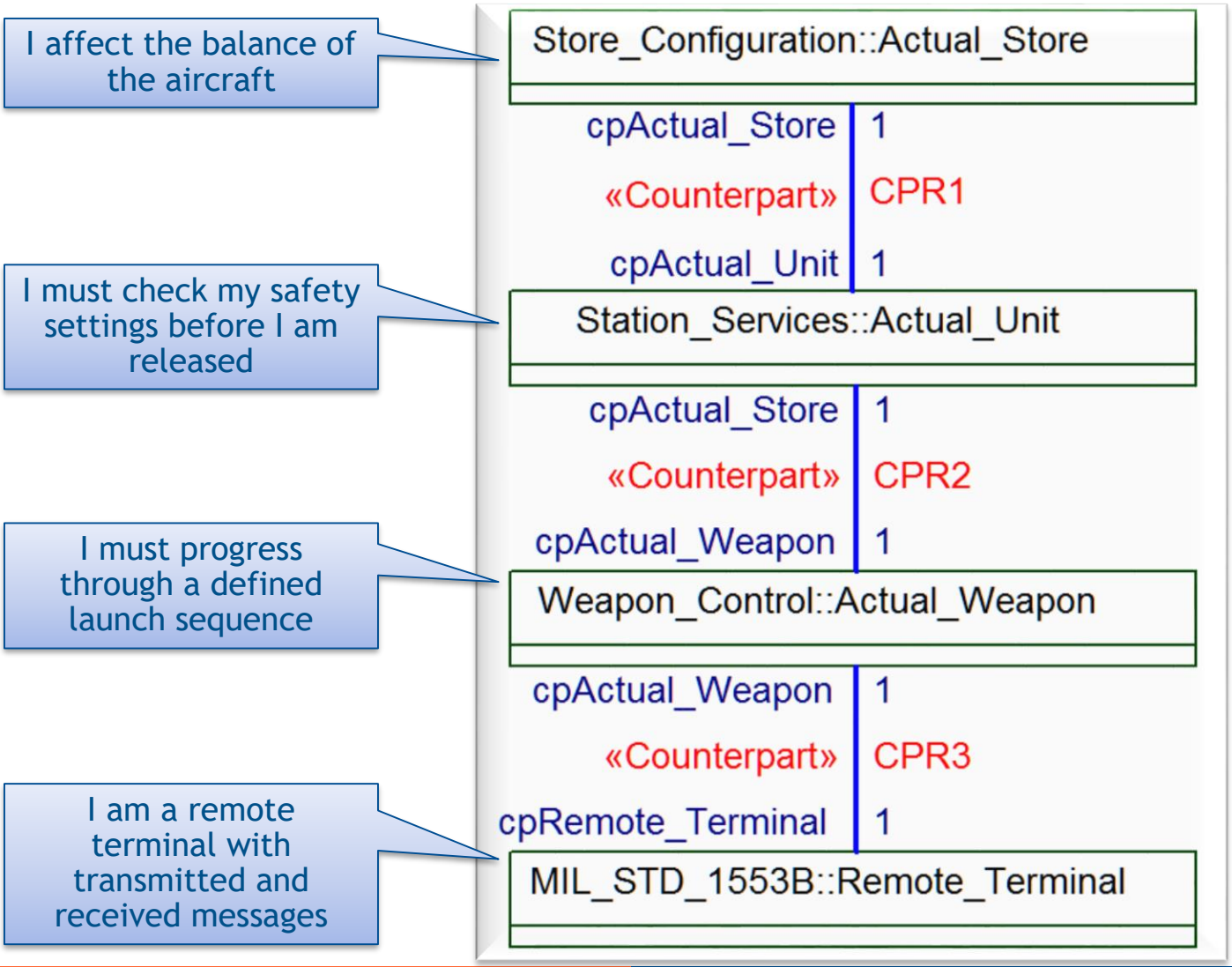
Weapon_Control::Actual_Weapon

I am a remote terminal with transmitted and received messages

MIL_STD_1553B::Remote_Terminal

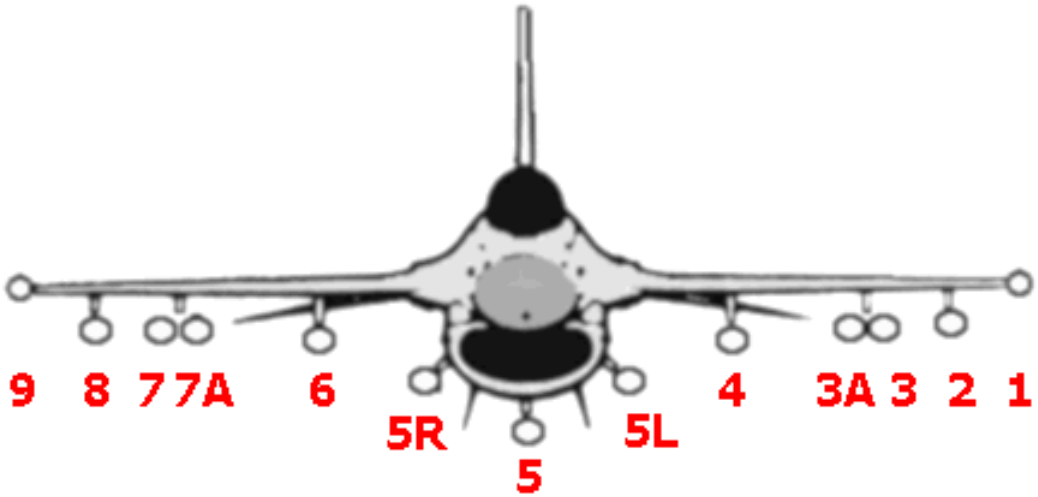
Example WIUK Counterpart Associations

Each real-world thing can be represented as different abstractions in various domains…the counterpart classes, linked via counterpart associations



Data Driven PIMs for Vehicle and Equipment Independence

The table below illustrates the possible F-16 Stores Configurations



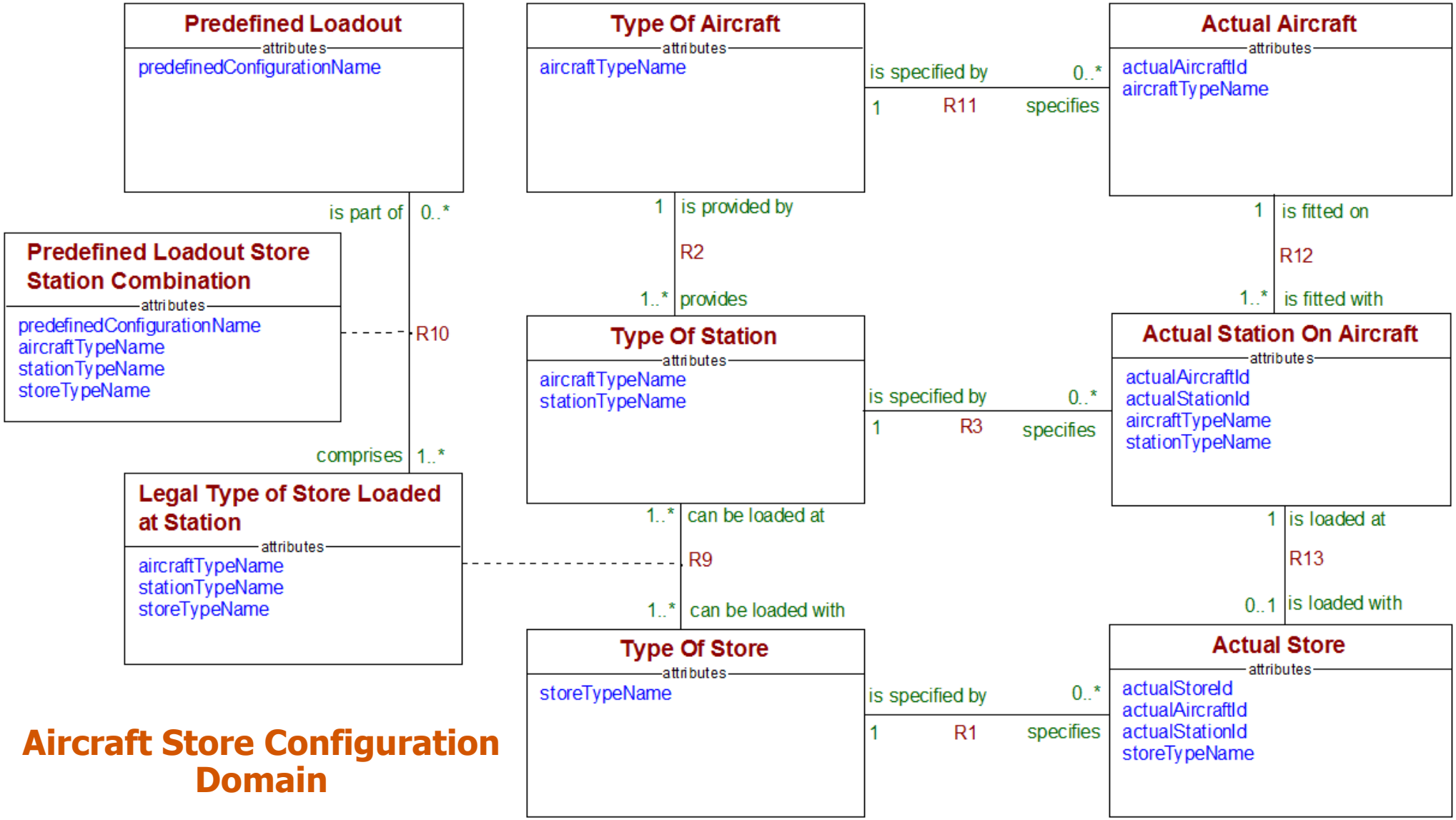
Areas of change are:

- Aircraft type
- Station (rail) configuration
- Store types
- Station-store combinations
- Mission types

...so these must not be "hard-wired" into the PIM

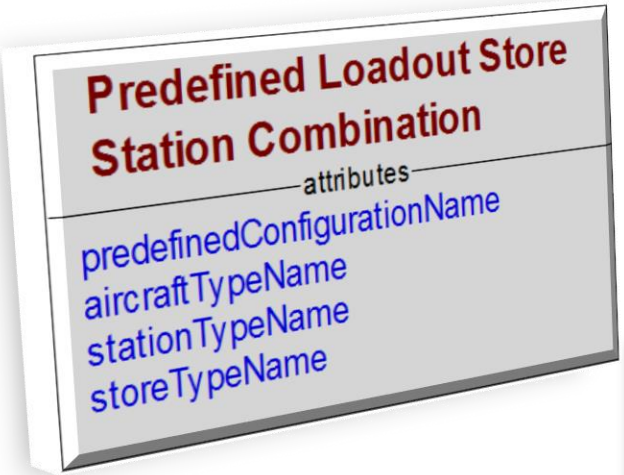
F-16 Rail Stores Loadings	Right Wing						Center						Left Wing
Rail ID	9	8	7	7 a	6	5 R	5	5 L	4	3 a	3	2	1
Defensive Counterair	AMRAAM	AMRAAM	Sidewinder		370g Tank				370g Tank		Sidewinder	AMRAAM	AMRAAM
Interdiction 1	AMRAAM		GBU24		370g Tank		LANTIRN		370g Tank		GBU24		AMRAAM
Interdiction 2	Sidewinder		AGM65		370g Tank		ECM Pod		370g Tank		AGM65		Sidewinder
Suppress Enemy Air Defense	Sidewinder		Harm		370g Tank		LANTIRN		370g Tank		Harm		Sidewinder

Data Driven PIM



Aircraft Store Configuration Domain

Data Driven Store Configuration Domain



The model is instantiated with objects for a specific aircraft using data held in tables…

5: F16 Predefined Loadouts (Predefined Loadout Store Station Combination)

	predefinedConfigurationName	aircraftTypeName	stationTypeName	storeTypeName
1	DEFENSIVE COUNTERAIR	F16	9	AMRAAM
2	DEFENSIVE COUNTERAIR	F16	8	AMRAAM
3	DEFENSIVE COUNTERAIR	F16	7	SIDEWINDER
4	DEFENSIVE COUNTERAIR	F16	6	370G TANK
5	DEFENSIVE COUNTERAIR	F16	4	LANTIRN
6	DEFENSIVE COUNTERAIR	F16	3	370G TANK
7	DEFENSIVE COUNTERAIR	F16	2	SIDEWINDER
8	DEFENSIVE COUNTERAIR	F16	1	AMRAAM
9	INTERDICTION 1	F16	9	AMRAAM

…allowing new weapon types and configurations to be added without changing any code…

…and allowing the model to be reused on any aircraft platform

F-16 Rail Stores Loadings	Right Wing						Center						Left Wing
Rail ID	9	8	7	7 a	6	5 R	5	5 L	4	3 a	3	2	1
Defensive Counterair	AMRAAM	AMRAAM	Sidewinder		370g Tank				370g Tank		Sidewinder	AMRAAM	AMRAAM
Interdiction 1	AMRAAM		GBU24		370g Tank		LANTIRN		370g Tank		GBU24		AMRAAM
Interdiction 2	Sidewinder		AGM65		370g Tank		ECM Pod		370g Tank		AGM65		Sidewinder
Suppress Enemy Air Defense	Sidewinder		Harm		370g Tank		LANTIRN		370g Tank		Harm		Sidewinder

MDA at the MoD

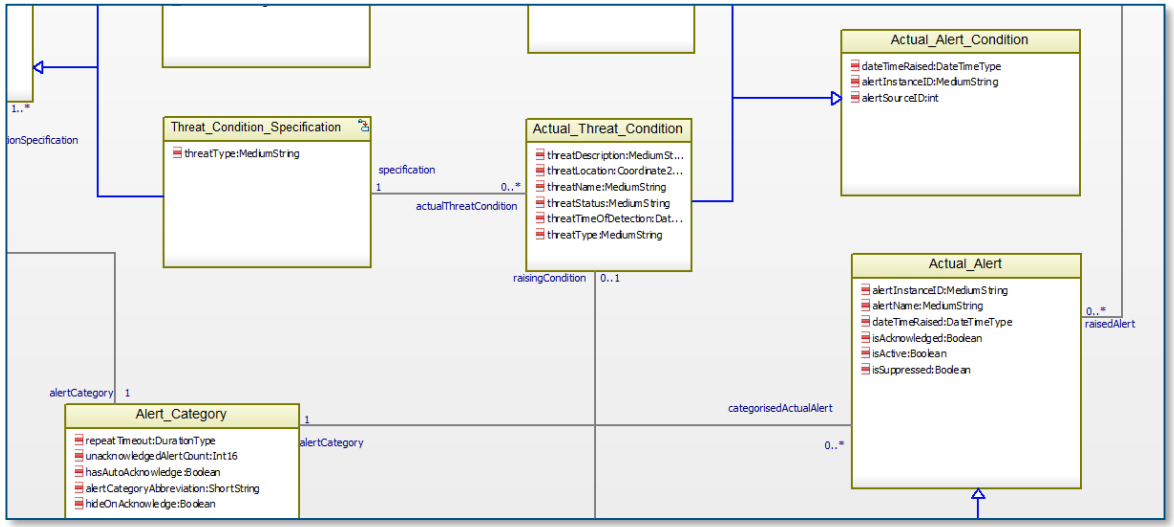
Verification through Model Animation

MINISTRY OF DEFENCE

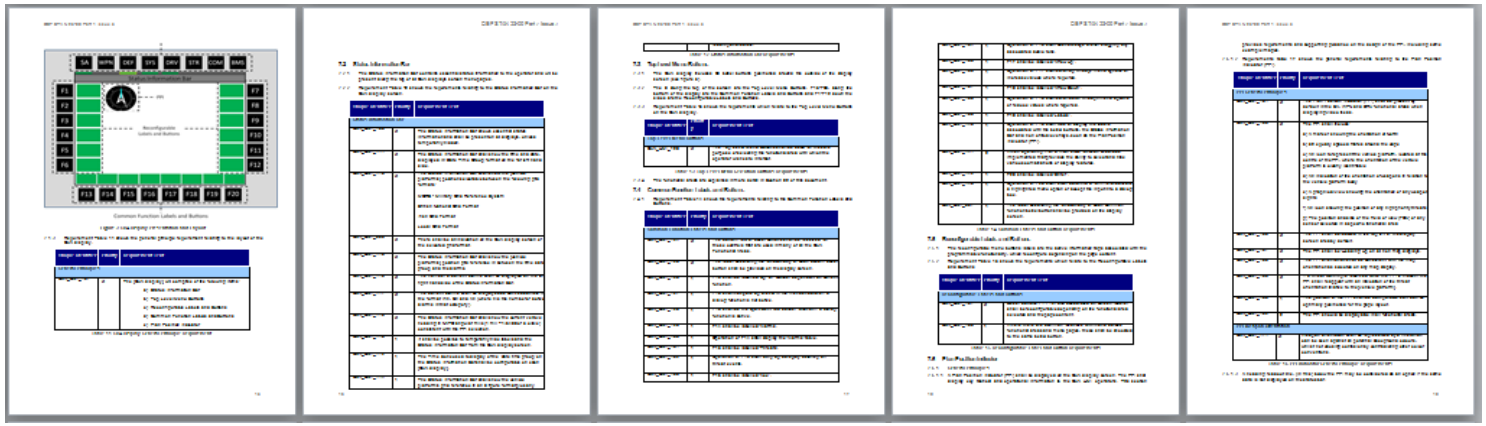
Chris Raistrick
chris.raistrick@abstractsolutions.co.uk

Purpose of Platform Independent Model Animation for Alerts

- **Demonstrate the viability of the LOSA Modules**



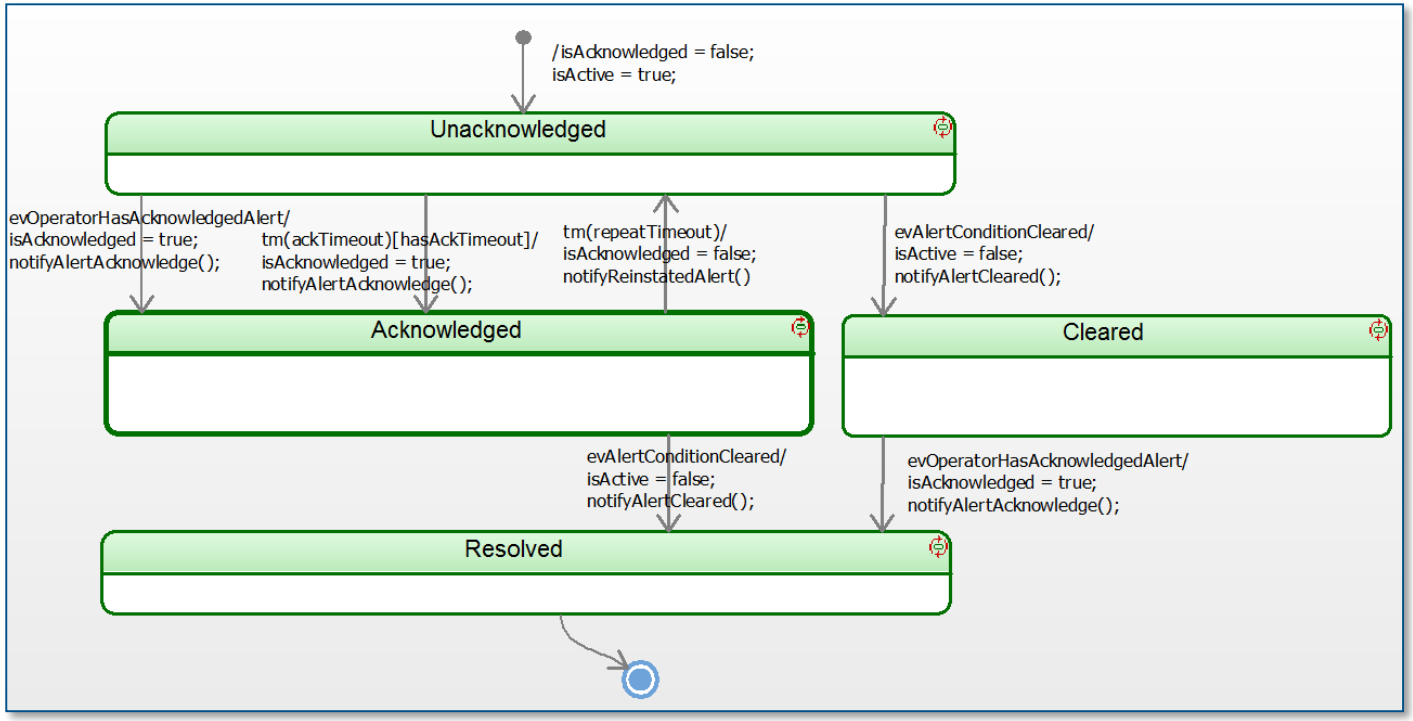
- **Understand in detail the requirements set out in DEF STAN 23-009 and explore issues for clarification**



Executable Models

GVA Alerts Model

- GVA Alerts Class Model contains behaviour specified with state machines and action code



- Model remains uncontaminated with knowledge of User Interface policy

Model Execution: Simulated User Interface

Warning Tone is sounding

Summary of Active Alerts

Unacknowledged Alert shown in Popup

Historic List of all Alerts

Warning Tone | Caution Tone | Threat Tone

F15 | F16

Warnings: 1 Cautions: 0 Advisories: 0 Threats: 0

(U) W:Engine:Oil:High:Pressure

Time	Act	Ack	Alert
13:38:14	C	A	W:Engine:Oil:Low:Pressure
13:38:19	C	A	W:Engine:Water:Low:Temperature
13:38:26	C	A	T:Enemy Force:TankDivision1:MBT:Contact
13:38:34	C	A	W:Engine:Water:High:Temperature
13:38:38	A	U	W:Engine:Oil:High:Pressure

Overridden

W-EngineOilPressureLow	Raise	Clear	Override	Unoverride
W-Eng				
W-Eng				
W-EngineWaterTemperatureHigh	Raise	Clear	Override	Unoverride
C-CameraPowerStatusOff(3)	Raise	Clear	Override	Unoverride
C-CameraPowerStatusOff(5)	Raise	Clear	Override	Unoverride
A-CommsLinkStatusOff	Raise	Clear	Override	Unoverride
T:EnemyForce:TankDivision1	Raise	Clear		

Crew Role: Commander

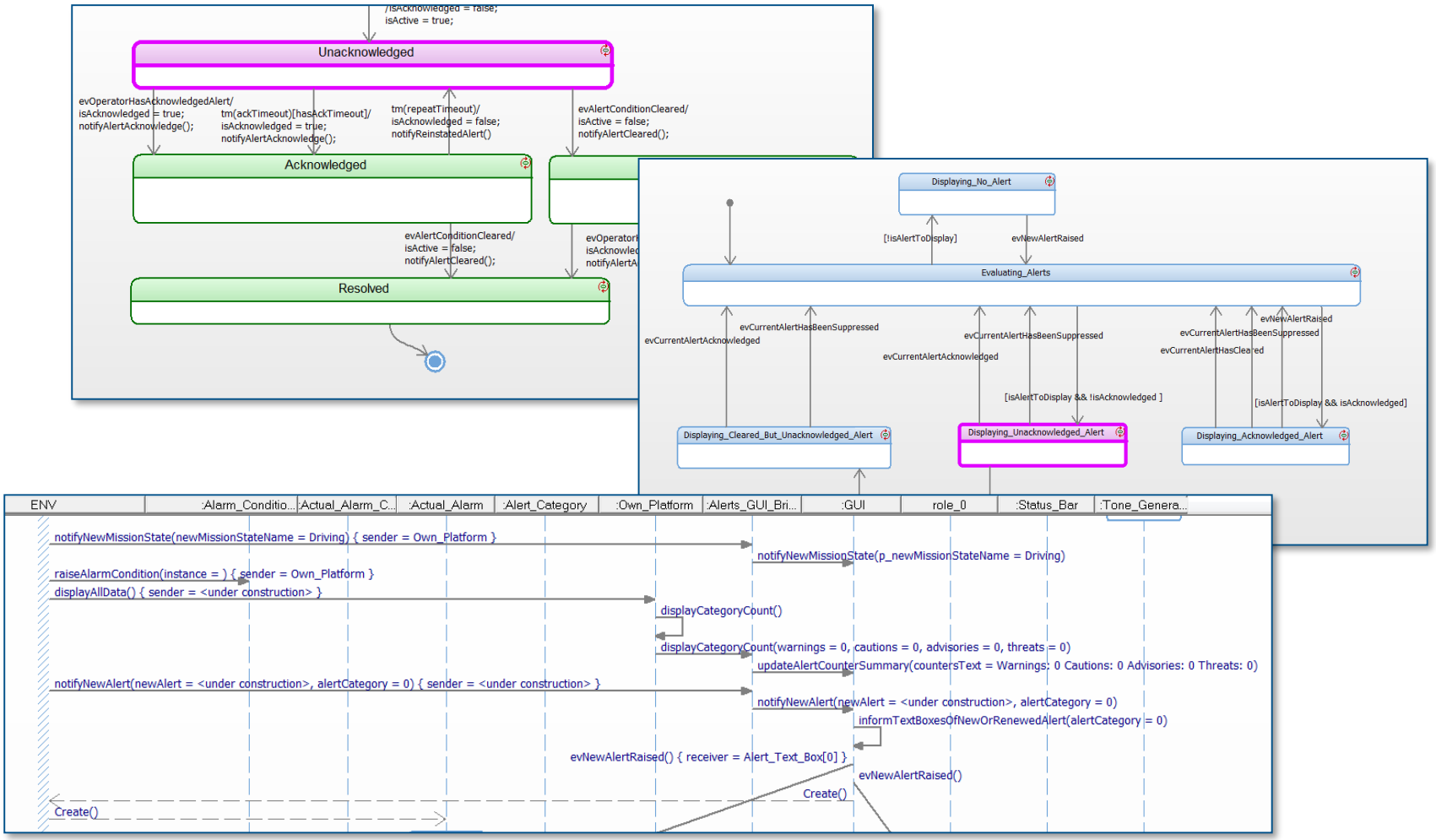
Commander | Gunner | Driver

Mission State: Driving

Driving | Engagement | Recon

Model Animation Outputs

As well as the Panel Diagram, the model animation can be reviewed against expected behaviour during and after runtime…



Results of Alerts Model Animation

- GVA Alerts class model shown to be capable of representing the required data
- 18 Separate issues with the requirements have been identified
  - Missing requirements
  - Inconsistent terminology
  - Unclear requirements
  - Inconsistent requirements
  - Undefined terms
- In addition, a number of dynamic human factors problems identified
- The model implements a number of (documented) decisions in order to deal with the gaps in requirements but now serves as an unambiguous statement of requirement

MDA at the MoD

Deployment through Model Translation

MINISTRY OF DEFENCE

Chris Raistrick
chris.raistrick@abstractsolutions.co.uk

Pitfall: Use of Manually Intensive Processes

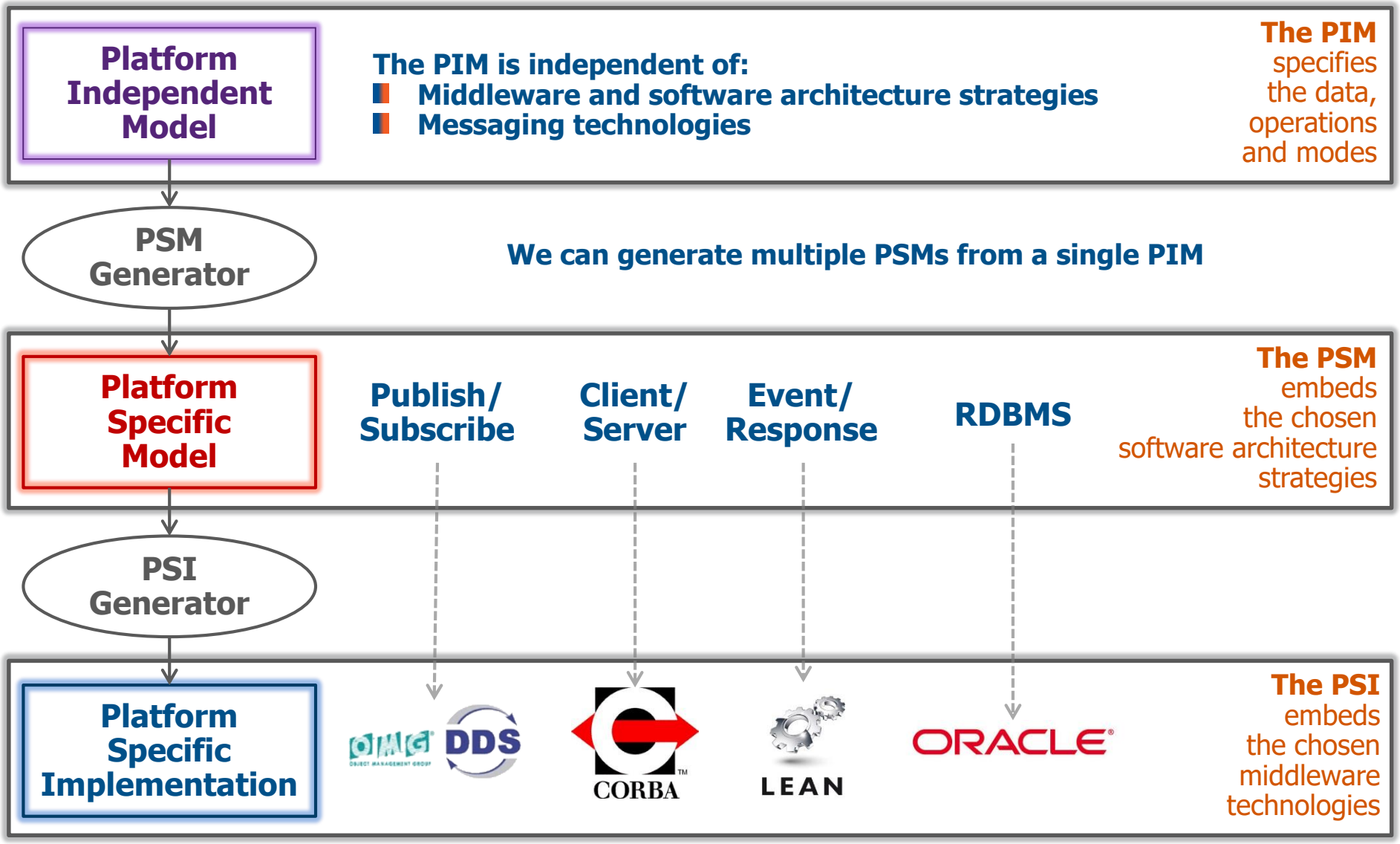
Before 1800

- Hand crafting by skilled practitioners
- Idiosyncratic design strategies
- Premium is on the practitioner

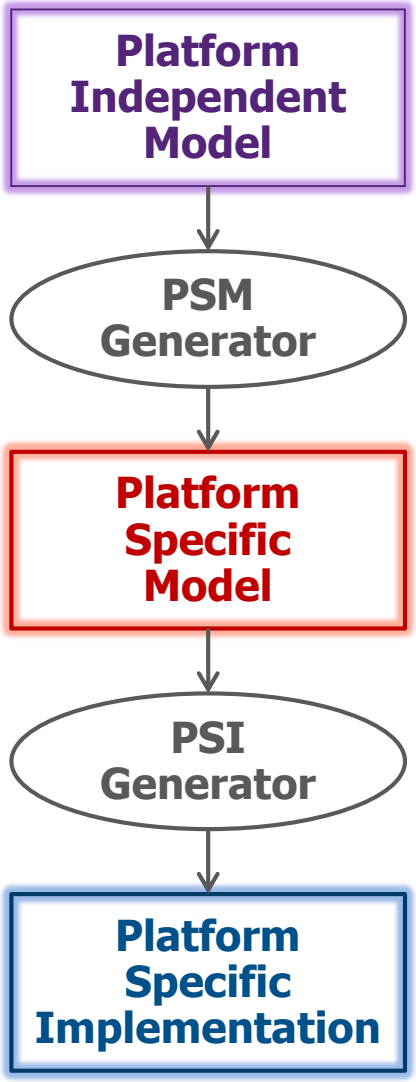
After 1800

- Automated production lines with no waste
- Consistent design strategies
- Premium is on the process

PIMs for Capabilities, PSMs and PSIs for Installations



Translation for Architecture Component Reuse and Maintainability



- ✓ Smaller, simpler and cheaper to maintain than PSM(s)
- ✓ Common service domains eliminate duplication of data items
- ✓ Independent of middleware and message definition languages
- ✓ Can be deployed on any execution platform in any installation (base, vehicle, soldier)
- ✓ Each PIM focusses on a single subject matter (domain)

- ✓ Easy to change system-wide design patterns
- ✓ Easy to adapt for different distribution and communication strategies

- ✓ Consistent use of design patterns across the system
- ✓ Coherent use of (one or more) software design strategies (SOA, Pub/Sub, Client/Server)
- ✓ Consistent inclusion of system-wide data items

- ✓ Easy to adapt for different middleware and message definition languages
- ✓ DDS and IDL specific translation rules ensure consistency

- ✓ Optimised for specific middleware technology (DDS, Web Services, Lean Services)
- ✓ Oriented towards specific message definition language (IDL, JSON, XML)
- ✓ Provides "on-the-wire" message interface definition

The smaller, simpler PIM becomes the maintained artefact.
The PSM and PSI are automatically generated and consistent

GVA Translator Properties

Project : GVA_Complete

General | Description | Relations | Tags | Properties

View Filtered By "gva"

AdditionalKeyName2	
AdditionalKeyName3	
CommandType	ArgumentBased
PSMName	
StateAttributeDescription	
StateAttributeName	
Translate	☑
Package	
PSMName	
Translate	☑
Project	
ClassTopicSourceAttributeDescription	
ClassTopicSourceAttributeName	sourceID
CommandTopicRecipientAttributeDescription	
CommandTopicRecipientAttributeName	recipientID
CommandTopicSourceAttributeDescription	
CommandTopicSourceAttributeName	sourceID
CommonPackageName	LDM_Common_New
IdentifierType	IdentifierType
ReferenceNumberAttributeDescription	
ReferenceNumberAttributeName	referenceNum
ReferenceNumberAttributeType	GVA_Long_Integer
TimeOfDataGenerationAttributeDescription	
TimeOfDataGenerationAttributeName	timeOfDataGeneration
TimeOfDataGenerationAttributeType	DateTimeType
State	
PSMName	
Translate	☑
Statechart	
PSMName	
StateAttributeDescription	
StateAttributeName	
Translate	☑
GVA_PSM_Translator	
Package	
Translate	☑
Project	
Compiler	RTI
Debug	☐
Mount	C:\Users\Chris

Class : Actual_Mount in Mount_PIM

General | Description | Attributes | Operations | Ports | Flow Ports | Relations | Tags | Properties

View Filtered By "gva"

GVA_PIM_Translator	
Class	
AdditionalKeyDescription1	
AdditionalKeyDescription2	
AdditionalKeyDescription3	
AdditionalKeyName1	
AdditionalKeyName2	
AdditionalKeyName3	
PSMName	
Translate	☑
Attribute	
PSMName	
Translate	☑
Operation	
AdditionalKeyDescription1	
AdditionalKeyDescription2	
AdditionalKeyDescription3	
AdditionalKeyName1	
AdditionalKeyName2	
AdditionalKeyName3	
CommandType	StateBased
PSMName	
StateAttributeDescription	
StateAttributeName	
Translate	☑
Statechart	
PSMName	
StateAttributeDescription	
StateAttributeName	
Translate	☑

Locate

Attribute : scan in Actual_Mount

General | Description | Relations | Tags | Properties

View Filtered By "gva"

GVA_PIM_Translator	
Attribute	
PSMName	
Translate	☑

Operation : setMaintenanceSubMode in Actual_Mount

General | Description | Implementation | Arguments | Relations | Tags | Properties

View Filtered By "gva"

GVA_PIM_Translator	
Operation	
AdditionalKeyDescription1	
AdditionalKeyDescription2	
AdditionalKeyDescription3	
AdditionalKeyName1	
AdditionalKeyName2	
AdditionalKeyName3	
CommandType	StateBased
PSMName	
StateAttributeDescription	
StateAttributeName	setMode
Translate	☑
Class	
AdditionalKeyDescription1	
AdditionalKeyDescription2	
AdditionalKeyDescription3	
AdditionalKeyName1	
AdditionalKeyName2	
AdditionalKeyName3	
PSMName	
Translate	☑

Statechart : statechart_1 in Actual_Mount

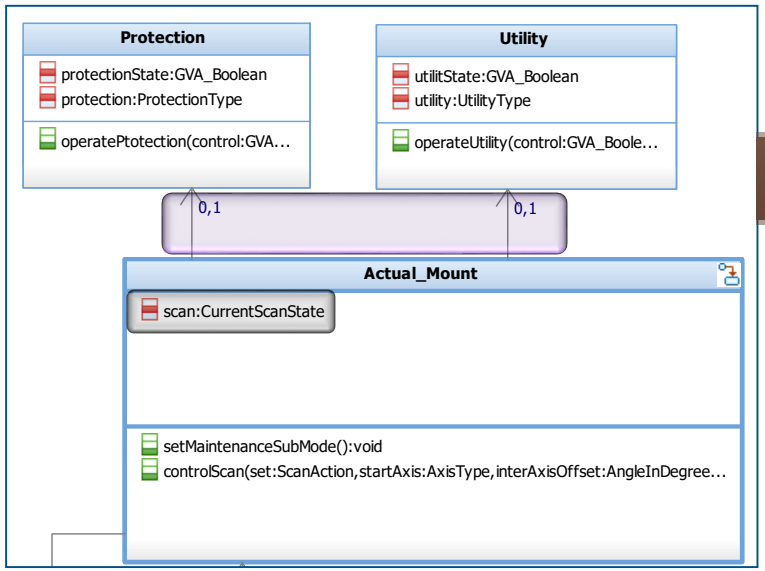
General | Description | Relations | Tags | Properties

View Filtered By "gva"

GVA_PIM_Translator	
Statechart	
PSMName	MaintenanceStateType
StateAttributeDescription	
StateAttributeName	maintenanceMode
Translate	☑

PIM to PSM Examples – Classes, Attributes and Associations

«pim» Mount_PIM (REF)



PIM + translator properties = PSM

«psm» Mount_PIM_PSM

Class : Actual_Mount in Mount_PIM_PSM

General | Description | Attributes | Operations | Ports | Flow Ports | Relations | Tags

Show Inherited

Name	Type
sourceID	IdentifierType in LDM_Common_New
timeOfDataGeneration	DateTimeType in LDM_Common_New
scan	CurrentScanState in Mount_PIM
maintenanceMode	MaintenanceStateType in Mount_PIM_PSM
itsProtection_sourceID	IdentifierType in LDM_Common_New
itsUtility_sourceID	IdentifierType in LDM_Common_New

Project : GVA_Complete

General | Description | Relations | Tags

View Filtered By "gva"

GVA_PIM_Translator	
AssociationEnd	
Attribute	
Class	
Generalization	
Operation	
Package	
Project	
ClassTopicSourceAttributeDescription	
ClassTopicSourceAttributeName	sourceID
CommandTopicRecipientAttributeDescription	
CommandTopicRecipientAttributeName	recipientID
CommandTopicSourceAttributeDescription	
CommandTopicSourceAttributeName	sourceID
CommonPackageName	LDM_Common_New
IdentifierType	IdentifierType
ReferenceNumberAttributeDescription	
ReferenceNumberAttributeName	referenceNum
ReferenceNumberAttributeType	GVA_Long_Integer
TimeOfDataGenerationAttributeDescription	
TimeOfDataGenerationAttributeName	timeOfDataGeneration
TimeOfDataGenerationAttributeType	DateTimeType

PIM to PSM Examples – Operations and States

«pim» Mount_PIM (REF)

Operation : controlScan in Actual_Mount

General | Description | Implementation | Arguments | Relations

void controlScan(const ScanAction& set,const AxisType& startAxi

Name	Type
set	ScanAction
startAxis	AxisType
interAxisOffset	AngleInDegreesType

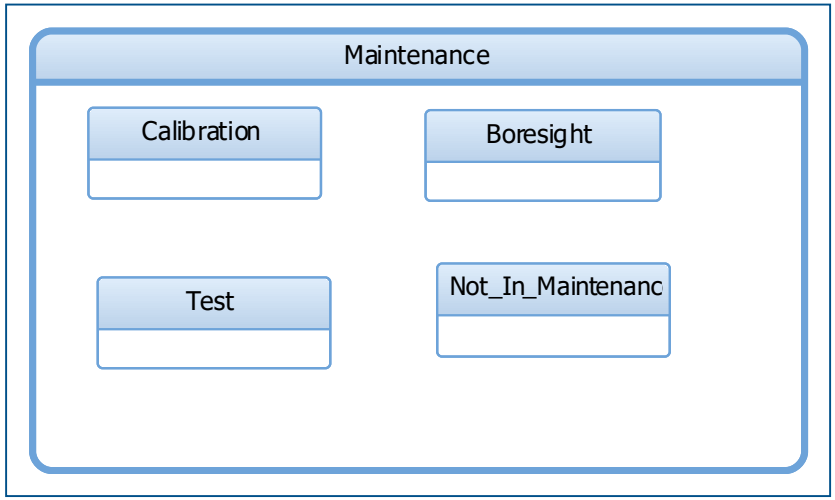
«psm» Mount_PIM_PSM

Class : Actual_Mount_controlScan in Mount_PIM_PSM

General | Description | Attributes | Operations | Ports | Flow Ports | Relations | Tags

Show Inherited

Name	Type
recipientID	IdentifierType in LDM_Common_New
sourceID	IdentifierType in LDM_Common_New
referenceNum	GVA_Long_Integer in LDM_Common_New
timeOfDataGeneration	DateTimeType in LDM_Common_New
set	ScanAction in Mount_PIM
startAxis	AxisType in Mount_PIM
interAxisOffset	AngleInDegreesType in LDM_Common_New



Type : MaintenanceStateType in Mount_PIM_PSM

General | Description | Literals | Relations | Tags | Properties

Name
MaintenanceStateType_Maintenance_Calibration
MaintenanceStateType_Maintenance_Boresight
MaintenanceStateType_Maintenance_Test
MaintenanceStateType_Maintenance_Not_In_Maintenance
<New>

PSM to IDL Examples

Project : GVA_Complete

General | Description | Relations | Tags | Properties

View Filtered By "gva"

GVA_PSM_Translator	
Package	
Translate	☑
Project	
Compiler	RTI
Debug	☐
Mount	C:\Users\Chris

« Local Disk (C:) ▸ Users ▸ Chris ▸ GVA_Complete

Search GVA_Compl...

File Edit View Tools Help

Organize ▾ Include in library ▾ Share with ▾ Burn New folder

Name	Date modified	Type	Size
LRF_PSM.idl	17/09/2015 16:54	IDL File	6 KB
Met_PSM.idl	17/09/2015 16:54	IDL File	5 KB
Mount_PIM_PSM.idl	17/09/2015 16:54	IDL File	18 KB
Navigation_Reference_System_PSM.idl	17/09/2015 16:54	IDL File	37 KB
New_Acoustic_PSM.idl	17/09/2015 16:54	IDL File	8 KB
Power_PSM.idl	17/09/2015 16:54	IDL File	21 KB

Class : Actual_Mount in Mount_PIM_PSM

General | Description | Attributes | Operations | Ports | Flow Ports | Relations | Tags

☐ Show Inherited

Name	Type
sourceID	IdentifierType in LDM_Common_New
timeOfDataGeneration	DateTimeType in LDM_Common_New
scan	CurrentScanState in Mount_PIM
maintenanceMode	MaintenanceStateType in Mount_PIM_PSM
itsProtection_sourceID	IdentifierType in LDM_Common_New
itsUtility_sourceID	IdentifierType in LDM_Common_New

```
// This reports the state of the mount.
struct C_Actual_Mount
{
  T_IdentifierType A_sourceID; //@key
  T_DateTimeType A_timeOfDataGeneration;
  T_CurrentScanState A_scan;
  T_MaintenanceStateType A_maintenanceMode;
  T_IdentifierType A_itsProtection_sourceID;
  T_IdentifierType A_itsUtility_sourceID;


};
```

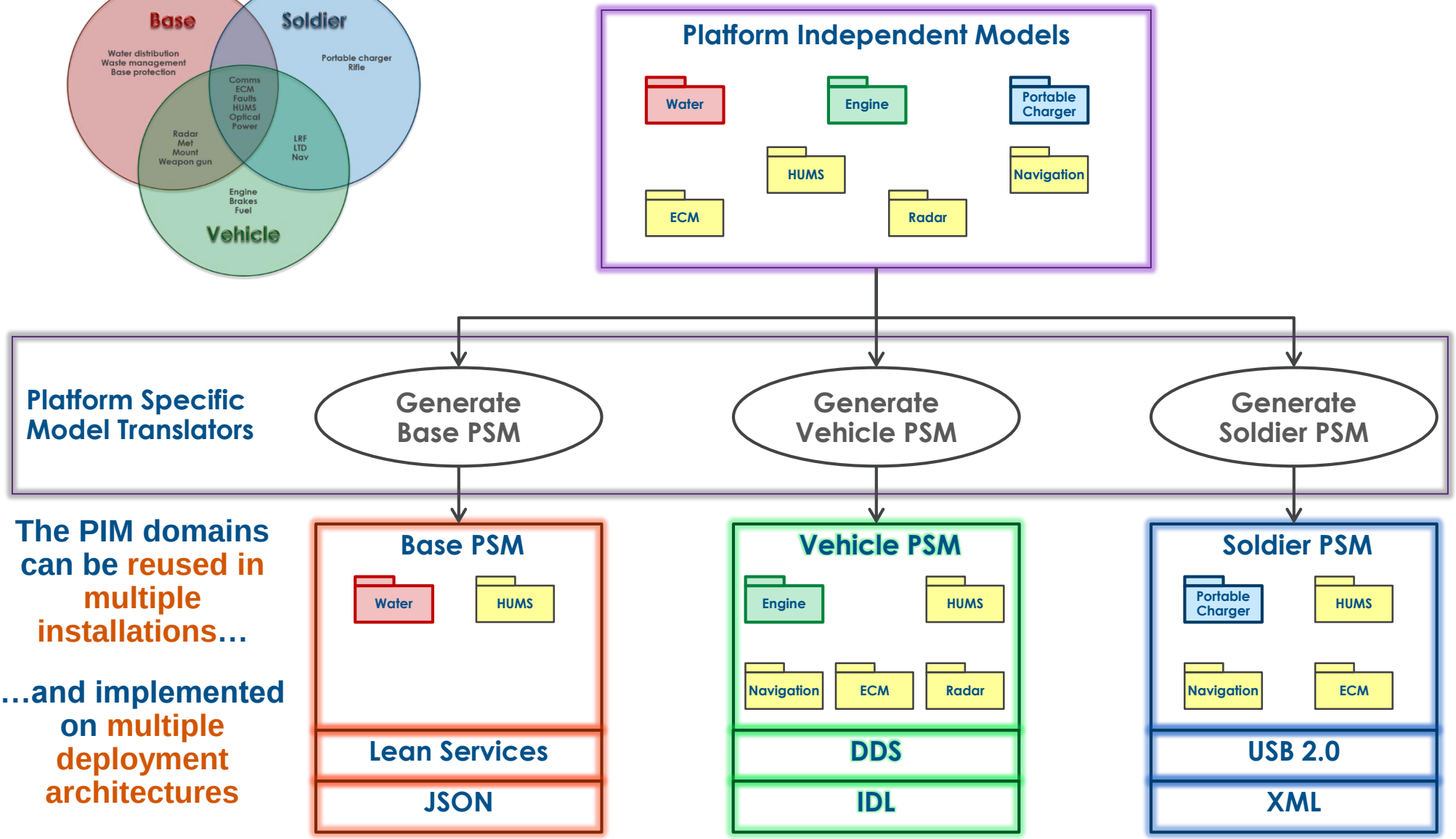
Type : MaintenanceStateType in Mount_PIM_PSM

General | Description | Literals | Relations | Tags | Properties

Name
MaintenanceStateType_Maintenance_Calibration
MaintenanceStateType_Maintenance_Boresight
MaintenanceStateType_Maintenance_Test
MaintenanceStateType_Maintenance_Not_In_Maintenance
<New>

```
// An enumeration of the Maintenance States that are possible.
enum T_MaintenanceStateType
{
  L_MaintenanceStateType_MaintenanceStateType_Maintenance_Calibration,
  L_MaintenanceStateType_MaintenanceStateType_Maintenance_Boresight,
  L_MaintenanceStateType_MaintenanceStateType_Maintenance_Test,
  L_MaintenanceStateType_MaintenanceStateType_Maintenance_Not_In_Maintenance
};
```

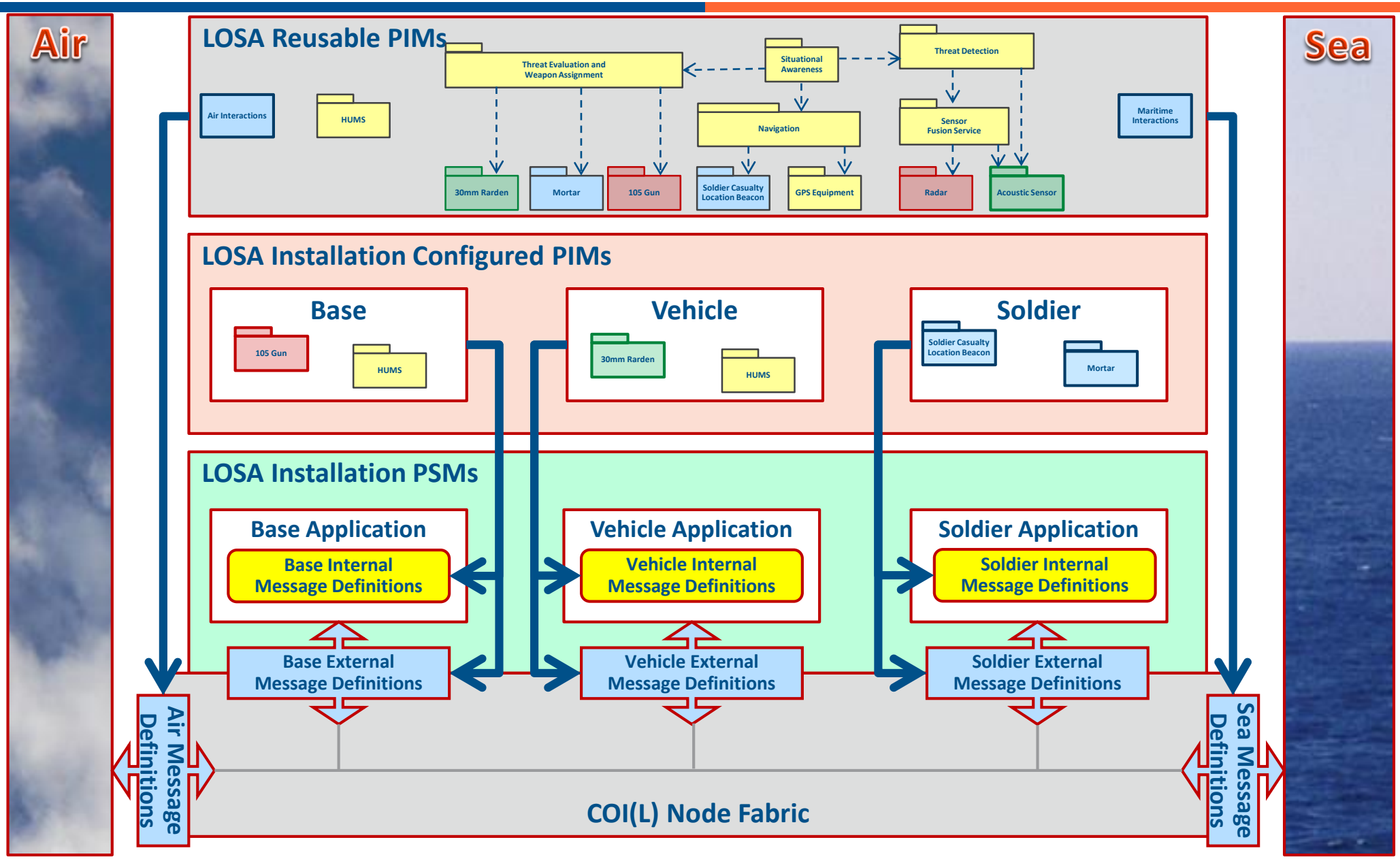
Architecture Component Reuse



The PIM domains can be reused in multiple installations…

…and implemented on multiple deployment architectures

Inter-Platform Communication: LOSA MDA Artefacts



MDA at the MoD

Summary

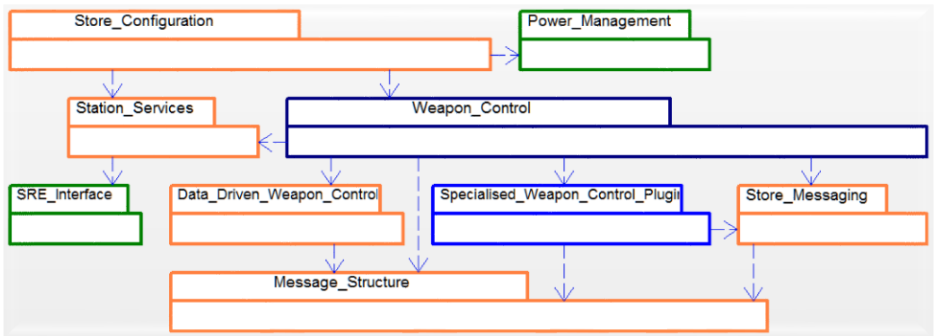
MINISTRY OF DEFENCE

Chris Raistrick
chris.raistrick@abstractsolutions.co.uk

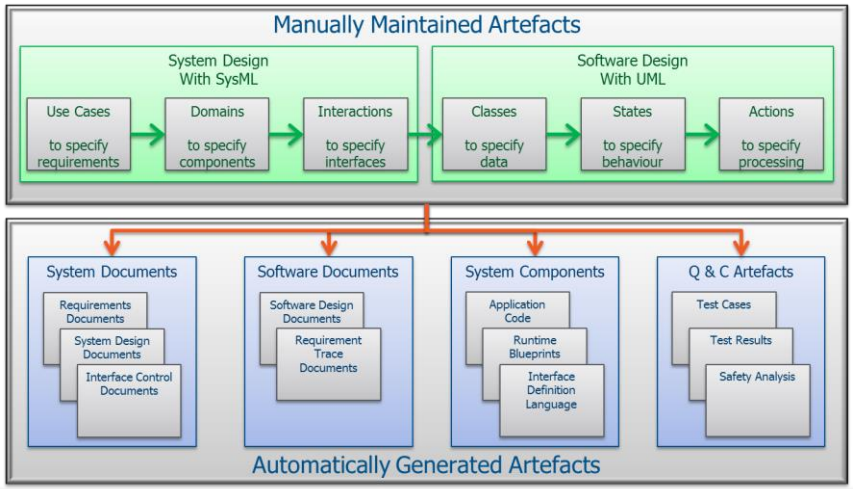
Simplify Complexity

- **Complexity and change are our enemies**
  - **Abstraction is how we attack complexity…**
    …and defend against change…
    …to reduce time and cost of system delivery
- **Abstraction is used for:**
  - **Separation of concerns into domains to manage complexity and support extensibility**
  - **Platform Independent Domain Models to facilitate technology change**
  - **Simple modelling formalism to enable universal communication**

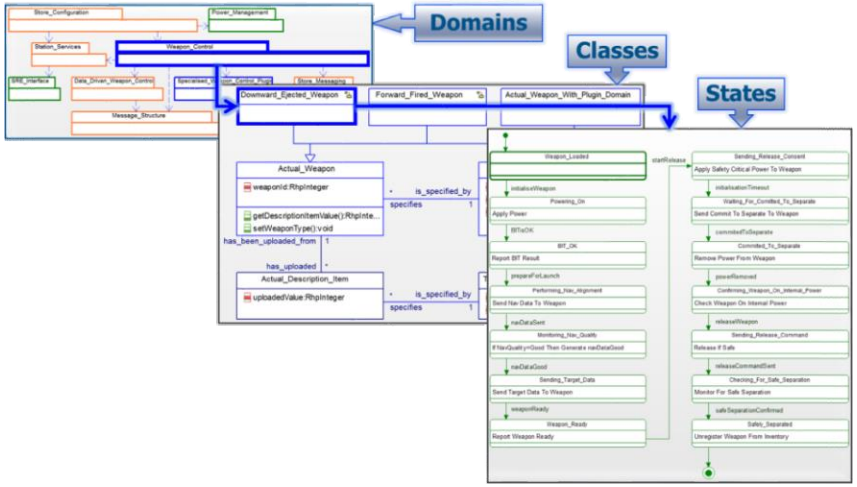
Simplify the Model
through domain based separation of concerns



Simplify the Process
through executable model translation



Simplify the Formalism
through platform independent xUML



Benefits of Model Driven Architecture Strategy

Model Driven Architecture for LOSA promotes:

- **Effective management of complexity**
- **Cost-effective maintenance of a single common model**
- **Large-scale reuse across base, vehicle and soldier architectures**
- **Compatibility with any present or future middleware**
- **Easier technology and capability insertion (including UORs)**
- **Automatic generation of consistent open interface definitions based on multiple languages and multiple messaging strategies**

MDA at the MoD

The End - Thank You

MINISTRY OF DEFENCE

Chris Raistrick
chris.raistrick@abstractsolutions.co.uk