AADL and MDA

Early Experience Applied to Aircraft-Weapon Integration

Yves LaCerte
Yves.LaCerte@gd-ais.com

27 January 2005
Agenda

- Introduction
- A Weapons Management System
- Embedded Systems: Solutions / Trends
- Plug and Play Modeling Concepts
- Architecture Analysis and Description Language
- Model Driven Architecture
Introduction

• The aircraft-weapon integration challenge is part of a larger integration problem, i.e.
  › Independent system-specific models often create unsolvable interoperability problems
Weapons Management Systems

Primary electrical, functional and logical interface between the Mission Management Computer(s), weapons, launchers and other equipment used to release and deliver stores
Motivation

- Today, $100M (typically)* to field a new weapon
- 40% to 60% associated with software updates (typically)*

* Source: AFRL/MN, 1999
Test and Integration Challenge

- **Platform Perspective**
  - Provide relevant functions and data
  - Observe resource, performance, and timing constraints
  - Do not change platform software

- **Weapon Perspective** *
  - Identify/define relevant functions and data
  - Identify/define resource, performance, and timing constraints

* Weapon software does change over time, which may impact the platform
Cost / Schedule Perspective (Notional)

WMS Development Costs

- Traditional
- New
- MDA
- AADL
- New WMS
- Weapon 3
- Weapon 2
- Weapon 1

Number of Months to Integrate a New Weapon

- Traditional
- New
- Weapon 3
- Weapon 2
- Weapon 1

Do not change platform software

No need to take down the whole squadron
State of the Art

- Increasing complexity / decreasing productivity
  - six (or fewer!) lines per day *

- The inefficiency of the embedded software development process will prevent novel technologies from entering the marketplace in time

* Typical of embedded software industry
Embedded Systems: Solutions / Trends

- Components
  - Less dedication to specific functions
  - Design - Improved abstraction
  - Synthesis - Auto code generation
  - Models - Assess before final implementation
  - Specification languages
    - Unambiguous representation of behavior and constraints - Rigorous semantics
    - Widely accepted
Embedded Systems: Solutions / Trends

- Semantic Interface Specification
  - Syntax can be performed by any type of Interface Definition Language (IDL, XML)
  - Assess semantic properties of an interface by an executable interface model
  - Assess interoperability by analyzing provided and required interfaces, and contracts
  - Adapt interface to improve interoperability

**AADL Semantic Interface Specification**
Dynamic Reflective Systems
- Change internal behavior depending upon attached devices
- Capable of integrating devices which provide new functions
- Capable of providing unforeseen functionality
- Foundation of aspect-oriented programming
Embedded Systems: Solutions / Trends

- Applied to Aircraft-Weapon Integration
  - Integrate new capabilities within a given design space (domain)
  - Prevent waiting for an aircraft upgrade cycle to integrate new weapons
- But
  - Likely difficult to implement a dynamic system that meets performance constraints
  - Aircraft-Weapon interface standards is a recent development, change is slow
The Plug and Play Concept
The Plug and Play Concept

- Demonstrate interoperability at design time
  - In terms of Functionality and Data
    - Open system approach via standards
  - In terms of Non-Functional Quality Attributes
    - Safety, real-time, reliability, fault tolerance, security,….

- A system that can exchange information and services with multiple systems is more interoperable than one that can't
  - Assess the quality of interoperability
  - Formulate strategies to improve interoperability
Domain Model – Weapon Interface View

The WMS accepts generic missile commands. These are subsequently passed on to the missile.

**Weapon Agnostic**
- Select Missile
- Configure
- Select State
- Get Status

**Weapon Specific**
- Apply power
- Initialize
- Set mission parameters
- Set missile modes etc
- Release
- Jettison
- IBIT
- Get current configuration

Platform → WMS → Missile

**Platform**
The WMS accepts generic missile commands. These are subsequently passed on to the missile.
The WMS offers mission services that are non-missile specific. Configuration data is subsequently passed on to the missile.
Domain Model - Data View

**Mission Plan**
- **MNF**
- `setQvoid`
- `getQvoid`

**Target Acquisition**
- **Acquisition Technique**: `void`
- **Target ID**: `void`
- `setQvoid`
- `getQvoid`

**Environment**
- **Tether Attitude**: `void`
- **Pressure Attitude**: `void`
- **Static Temperature**: `void`
- `setQvoid`
- `getQvoid`

**Target**
- **Position**: `void`
- **Velocity**: `void`
- **Type**: `void`

**Handover**
- **Time**: `void`
- **Status**: `void`
- **Missile_ID**: `void`

**Utility**
- **Validity**: `void`
- **Correlation**: `void`

**Platform Characteristics**
- **Type**: `void`
- **ID**: `void`

**Platform Quality**
- **Heading Error**: `void`
- **True_Heading_Viability**: `void`

**Attitude**
- **Pitch_Angle**: `void`
- **Bank_Angle**: `void`
- **Elevation_Angle**: `void`
- **Vertical_Acceleration**: `void`
- **Poth_Angle_Rate**: `void`
- **Roll_Angle_Rate**: `void`
- **Yaw_Rate**: `void`
- **XYZ_Velocity**: `void`

**Position**
- **Latitude_MSNW_INDEX**: `void`
- **On_Ground**: `void`
- **Pylon_Position**: `void`

**Weapon**
- **Launch Acceptability Region**

**General Dynamics**
Advanced Information Systems
State Machines

- Preferred for the specification of controllers
- Useful for
  - Verification against requirements
  - Test-case generation
  - Automatic code generation

API and events used to cause state changes
Ports – Connectors – Contracts

Weapon requires power.... Power provided by WMS

**Weapon**

<table>
<thead>
<tr>
<th>Pre and Post conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter types</td>
</tr>
<tr>
<td>Synchronization constraints</td>
</tr>
<tr>
<td>QoS features</td>
</tr>
</tbody>
</table>

**Power**

- 115 volts(), AC
- 270 v, DC
- 28 v, DC 1 (std)
- 28v, DC 2 (safety)

**WMS**

Component

Port

Provided Interface

Connector

Required Interface

Contract

«interface»

Power

- 115 volts()
- AC
- 3 phase

<<interface>>

Power
Platform Challenge

- Component models and supporting frameworks often rely on the specifics of the underlying platform.

- There is a need for techniques to handle functional and non-functional properties of components and systems.
Non-Functional Properties

Architecture Analysis and Design Language (AADL)
System Example
System Construction Example

system MILSystem
end MILSystem;

system implementation MILSystem.MissilePlatform
subcomponents

-- buses
Power_A : bus MILPower;

-- components
WMS1 : system WMS.WMS;
FCS : system FireControlSystem.FCS;

connections
-- buses
bus access Power_A -> Power1.IFPower_A;
bus access Power_A -> WMS1.IFPower_A;

-- ports
port group WMS1.Out1553_A -> Launcher1.In1553_A;

end MILSystem.MissilePlatform;
system FireControlSystem features
   IFGeneric: requires bus access MILGeneric;
   InMission: port group Rx_Port;
   OutMission: port group Tx_Port;
end FireControlSystem;

system implementation FireControlSystem.FCS
end FireControlSystem.FCS;
System Construction Example

Black Box

system WMS
features
-- Buses
IF1553_A: requires bus access MIL1553;
IFGeneric: requires bus access MILGeneric;
IFPower_A: requires bus access MILPower;
IFDiscrete_A: requires bus access MILDiscrete;
-- Ports
In1553_A: port group Rx_Port;
Out1553_A: port group Tx_Port;
InDiscrete_A: port group Rx_Port;
OutDiscrete_A: port group Tx_Port;
InMission: port group Rx_Port;
OutMission: port group Tx_Port;
end WMS;

port group Rx_Port
features
Rx: in data port;
inverse of Tx_Port
end Rx_Port;

port group Tx_Port
features
Tx: out data port;
end Tx_Port;
WMS System Implementation Example

```
system implementation WMS.WMS
subcomponents
...connections
...modes
end WMS.WMS;
```

**Diagram:***

- **Subcomponents**: Processors, Processes, Bindings
- **Connections**: Port Groups
- **Modes**: MainMode: initial mode; BackupMode: mode;
Process

process PlugandPlayDispatcher
features
  InGeneric: port group Rx_Port;
  OutGeneric: port group Tx_Port;
  InLauncher: port group Rx_Port;
  OutLauncher: port group Tx_Port;
  InWeapon: port group Rx_Port;
  OutWeapon: port group Tx_Port;
end PlugandPlayDispatcher;
Process Implementation

process implementation PlugandPlayDispatcher.WMS

subcomponents

Bus_Listener: thread Listener.Bus_Listen;
PnP_Dispatcher: thread PnPDispatcher.WMS;

connections

port group InGeneric -> Bus_Listener.Bus_Listener;
port group PnP_Dispatcher.OutLauncher -> OutLauncher;
port group InLauncher -> PnP_Dispatcher.InLauncher;
port group PnP_Dispatcher.OutWeapon -> OutWeapon;
port group InWeapon -> PnP_Dispatcher.InWeapon;

end PlugandPlayDispatcher.WMS;
Thread – Producer/Consumer

features
Buffer_Listener: port group Rx_Port;
Bus_Dispatcher: port group Tx_Port;
Buffer_2: requires data access Buffer;

properties
Source_Text => "abc";
Source_Code_Size => 100 kb;
Source_Data_Size => 10 kb;
Source_Stack_Size => 10 kb;
Source_Heap_Size => 10 kb;
Dispatch_Protocol => periodic;
Period => 100 ms;
Deadline => 100 ms;
Compute_Execution_Time => 50 ms;
Trade – Offs

Producer – Consumer
Event-Driven with
Publish-Subscribe
Tools - OSATE

- Open source AADL tool environment
  - Software Engineering Institute
  - www.aadl.info

- Set of plug-ins on top of Eclipse
  - www.eclipse.org
Lessons Learned

- **Model Conventions**
  - Readable
  - Extensible
  - Maintainable

- **Ubiquitous Connectivity**
  - Port connections
  - Bus interfaces

- **Model Scale**
  - Complex coding
    - Visualization will help
  - Multiplicity
    - Arrays, loops
  - Model navigation
    - Tree editor
  - Compartments
    - Ease of integrating multiple models
Future Work

- Use / create OSATE analysis plug-ins
  - Schedulability
  - System scalability
  - Safety
  - End-to-end flow analysis

- AADL support from UML vendors
Model Driven Architecture
Approach to MDA

TE = Transformation Engine
CG = Code Generator
VE = Validation Engine
Approach to MDA / AADL

Derive system properties

UML Model

Executable System

Product Instance

Prove system properties

Platform Requirements

AADL Model

Executable System

Product Instance

Validated Product

Prove system properties

General Dynamics
Advanced Information Systems
MDA Work Products

Application

Requirements

Platform Independent Model

Context View

Requirements View

Analysis View

Design View

Product Instance

Platform Specific Model

Guidelines and Rules

Platform Model

Libraries

Patterns

Producer/Consumer Factory

AADL Model ready for analysis

UML Model ready for code generation
MDA / AADL Expectation

Development Costs for New Weapons

Traditional

New

New WMS
Weapon 3
Weapon 2
Weapon 1

With MDA

Without MDA

New MWS
Weapon 3
Weapon 2
Weapon 1

With MDA

Without MDA

Traditional

New

$