AADL et la conception des logiciels
System Lifecycle

- System Engineering
- System Integration
- Hardware Engineering
- Software Engineering
Software Design

from System Engineering

Software Requirements Analysis

- SW requirements

Software Design

- coverage

Software Coding

- generation

Software Unit Testing

- unit tests scenarii

Software Functional Testing

- functional tests scenarii

to System Integration

System Requirements
- Hardware
- Operating System

Realization Constraints
- Team Organization
- Technical Choices
- Quality Standards

Coverage

Reverse Coverage
Les logiciels critiques
“faster, better, cheaper”

• General characteristics:
  – standardized development process
  – long term projects and many partners
  – collaborative ground & on-board software
  – dependable software

• Additional characteristics for on-board software:
  – for avionics: safety
  – for space vehicles: environment constraints

• Need for software development methods and tools:
  – ensure requested high level of quality
  – paradox: need for maturity and innovation
  – really operational concepts to increase productivity
  – enforced by standards: DO-178B, ECSS-E40, EN-50128, ISO-12207, etc...
“Good” Software Engineering practices:
- well defined architectural & detailed design process
- requirements traceability
- documentation framework, etc...
- model-based engineering

Managing the complexity:
- designing in the large:
  - hierarchical decomposition
- distributed development:
  - modularity
  - interfaces
- mixed paradigms:
  - function oriented
  - object oriented
  - task oriented for real-time SW

Modeling real-time:
- reaching deadlines:
  -> mastering control flows
- no loss of data:
  -> mastering data flows
- predefined RT components
- supporting RT executives
**Les techniques “basées modèle”**

- **Description du Système**
  - exemple: Mathlab/Simulink
  - utilisation intensive de bibliothèques de composants spécialisés
  - études paramétriques, dimensionnement, prototypage, simulation
  - si modèle formel: possibilité de générer du code, mais rarement

- **Description du Logiciel**
  - spécifier avant de concevoir
  - concevoir avant de coder
  - masquer la complexité: boîtes noires, composants Temps-Réal
  - automatiser la production du code et de la documentation

Langages de modélisation

AADL
UML2.0
HOOD
Avionics Architecture Description Language (AADL)

- Avionics Architecture Description Language: 
  - Emerging SAE Standard, AS-2C AADL working group 
  - Specification language for avionics systems 
  - Software & Hardware components, features & properties 

- An AADL specification describes: 
  - how components are assembled or integrated 
  - the functional interfaces (control and data flows) 
  - the non-functional properties (i.e. timing) 

- Field of application: embedded systems with: 
  - challenging resource (size, weight, power) 
  - strict real-time response requirements 
  - specialized input/output hardware 
  - need to be certified to high levels of assurance
**AADL components**

- **Components categories**
  - Software: data, threads and processes
  - Hardware: memories, processors, devices and buses
  - Compound: systems and modes

- **Components definition**
  - Component type: functional interface
  - Component implementation
  - Component instance: a subcomponent in a component implementation

- **Features**
  - Ports: data, event or event data
  - Subprograms
  - Data

- **Properties**: ex: real time properties
Emerging OMG standard

Structure Diagrams:
- new in UML2.0
- modeling Components, Interfaces, Connections
- black box and white box views

connectors in a UML 2.0 component
**HOOD**
Hierarchical Object Oriented Design

- Design Method:
  - Standardized by ESA
  - A set of **concepts**: Modules, Operations, Types, ...
  - A set of **notations**: Graphical and Textual
  - A set of **rules**: design process and verification
Additional design constraints to comply with the Ravenscar profile

HRT objects

HRT operation constraints

Real-Time attributes
Modélisation Temps-Réal

- Composants spécialisés
  - Objets cycliques, sporadiques ou protégés en HOOD
  - Process, threads et data en AADL

- Flots de contrôle et de données
  - Relations Use et contraintes d’exécution en HOOD
  - Ports et connections en AADL

- Propriétés spécifiques au temps-réel
  - Real-time attributes (deadline, wcet, ...) en HOOD
  - Scheduling properties en AADL

- Prise en compte de l’environnement d’exécution
  - Modélisation explicite: on décrit le RTOS comme une application
  - Modélisation implicite: règles de conception et de génération de code
History
All together

<table>
<thead>
<tr>
<th>AADL</th>
<th>UML 2.0</th>
<th>HOOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>component</td>
<td>component</td>
<td>(parent) module</td>
</tr>
<tr>
<td>subcomponent</td>
<td>part</td>
<td>(child) module</td>
</tr>
<tr>
<td>provides features</td>
<td>provided interface</td>
<td>provided interface</td>
</tr>
<tr>
<td>required subcomponents</td>
<td>required interface</td>
<td>required interface</td>
</tr>
<tr>
<td>(server) containment connection</td>
<td>delegate (provided)</td>
<td>implemented_by</td>
</tr>
<tr>
<td>(client) containment connection</td>
<td>delegate (required)</td>
<td>use (uncle)</td>
</tr>
<tr>
<td>components connection</td>
<td>assembly</td>
<td>use (sibling)</td>
</tr>
</tbody>
</table>

semantics can be formally specified with a UML meta model
Small design example

OBJECT SYSTEM_CONFIGURATION IS PASSIVE
   INTERNALS
   OBJECTS
      ex1AADL;
      Display_driver;
      OS;
   END SYSTEM_CONFIGURATION

OBJECT ex1AADL IS ACTIVE
   PRAGMA main(operation_name => start)
   PROVIDED_INTERFACE
      OPERATIONS
         start
   END PROVIDED_INTERFACE
   OBJECT_CONTROL_STRUCTURE
      CONSTRANDED_OPERATIONS
         start CONSTRANDED_BY ASER ;
      REQUIRED_INTERFACE
      OBJECT Display_driver;
      OPERATIONS
         refresh;
   END REQUIRED_INTERFACE
   INTERNALS
   OBJECTS
      client;
      server;
      sensor;
      refresh;
      monitor;
   END OBJECTS
   OPERATIONS
      start IMPLEMENTED_BY client.exec;
END ex1AADL
SYSTEM SYSTEM_CONFIGURATION IS
    start : IN EVENT;
END SYSTEM_CONFIGURATION;

SYSTEM IMPLEMENTATION SYSTEM_CONFIGURATION.default IS
    ARINC : SYSTEM system.ARINC;
    Exlaadl : PROCESS Exlaadl.default;
    Display_driver : PROCESS Display_driver.default;
    standard : PACKAGE standard.default;
CONNECTIONS
    EVENT Exlaadl.start <- start;
    EVENT Exlaadl.sensor_interrupt <- ARINC.sensor_interrupt;
    EVENT DATA Display_driver.value <- Exlaadl.value;
END SYSTEM_CONFIGURATION.default;
Generated AADL code (2)

PROCESS Ex1aadl IS
    start : IN EVENT;
    sensor_interrupt : IN EVENT;
    value : OUT EVENT DATA NIL.integer;
END Ex1aadl;

PROCESS IMPLEMENTATION Ex1aadl.default IS
    client : THREAD client( monitor => monitor ).default;
    server : THREAD server.default;
    sensor : THREAD sensor( monitor => monitor ).default;
    refresh : THREAD refresh( monitor => monitor ).default;
    monitor : MONITOR monitor.default;
CONNECTIONS
    EVENT client.exec <- start;
    SUBPROGRAM server.request <- client.request;
    EVENT sensor.start <- sensor_interrupt;
END Ex1aadl.default;
THREAD client IS
    exec : IN EVENT;
    request : CLIENT SUBPROGRAM
        (value : IN monitor.T_value, result : OUT monitor.T_value);
REQUIRES
    monitor : MONITOR monitor;
END client;

THREAD IMPLEMENTATION client.default IS
PROPERTIES
    Scheduling_protocol => aperiodic;
END client.default;

THREAD server IS
    request : SERVER SUBPROGRAM
        (value : IN monitor.T_value, result : OUT monitor.T_value);
END server;

THREAD IMPLEMENTATION server.default IS
PROPERTIES
    Scheduling_protocol => server;
END server.default;
THREAD sensor IS
  start : IN EVENT;
REQUIRES
  monitor : MONITOR monitor;
END sensor;

THREAD IMPLEMENTATION sensor.default IS
PROPERTIES
  Scheduling_protocol => sporadic;
  Deadline => 10;
  Compute_Time => 5;
END sensor.default;

MONITOR monitor IS
  T_value : TYPE;
END monitor;

THREAD refresh IS
REQUIRES
  monitor : MONITOR monitor;
END refresh;

THREAD IMPLEMENTATION refresh.default IS
PROPERTIES
  Scheduling_protocol => periodic;
  Period => 10;
  Deadline => 5;
  Compute_Time => 5;
END refresh.default;
OBJECT monitor;
  TYPES NONE
  CONSTANTS NONE
  OPERATION_SETS NONE
  OPERATIONS read, write;
  EXCEPTIONS

OBJECT server;

OBJECT client;

OBJECT display_driver;

OBJECT sensor;

OBJECT monitor;

OBJECT refresh;

INTERFACE Exload;

OPERATIONS start, stop, SENSOR_INTERRUPT;

TYPES

CONSTANTS

EXCEPTIONS

DATA

DEFINITIONS

PACKAGE body

BEGIN

END body;
Stood v5

The AADL semantics
(+ textual notation)

The HOOD design process,
(+ semantics, graphical & textual notations)

The UML 2.0 standard graphical notation
System Lifecycle improvement

System Engineering

Hardware Engineering

Software Engineering

System Integration