Model Based Systems Engineering at DARP

Alek Radjenovic
(Malcolm Wallace, Philippa Conmy, John McDermid, Richard Paige)
Outline

• Background to HIRTS DARP
• Architectural Descriptions and Modelling
• Contracts and Incremental Certification
• Model Testing and Analysis
• Work in Progress & Beyond
DARP HIRTS

• Several DARPs in the UK
  – Defence & Aerospace Research Partnership
    • Aero-engine design, helicopters …
• High Integrity Real Time Systems (HIRTS)
  – Based at The University of York
  – Partners: BAE SYSTEMS, Rolls Royce,
    QinetiQ, MBDA, MoD,…
  – 3 Strands (topics, themes)
    • Strand 1: model-based systems engineering
HIRTS Operation

- **Research programme**
  - Undertaken at York
  - Direction and input, e.g. case studies, from industrial partners

- **Broader engagement**
  - Annual workshops
  - Review DARP work, and obtain wider view
    - E.g. talk from Steve Vestal on AADL
Strand 1 Approach

• Objective
  – Improve the development process for HIRTS by improving architectural definition/analysis
    • Address the Incremental Certification problem
  – Three lines of attack
    • Architectural Descriptions and Modelling
    • Incremental Change and Certification (Contracts)
    • Static Analyses and Automated Testing of Models (including failure behaviour)
Connections

- **Soc. Automotive Eng.**
  - AADL modelling language

- **IEEE 1471**
  - Architectural description, esp. views

- **Rolls-Royce**
  - TAY case study, SWAN modelling

- **QinetiQ**
  - IFPCS case study

- **BAE Systems**
  - IFPCS case study, IMA development, contracts

- **DARP Strand 1**
  - Architectural modelling
  - Incremental certification
  - Safety analysis

- **EuroCAE WG 60**
  - Civil standards for IMA

- **MBDA**
  - LRAAM case study, RTN modelling

---

**Defence & Aerospace Research Partnership**

**The University of York**
Outline

• Background to HIRTS DARP
• Architectural Descriptions and Modelling
• Contracts and Incremental Certification
• Model Testing and Analysis
• Work in Progress & Beyond
Modelling

- different modelling languages used by our industrial partners
- all different in terms of how they express HIRTS and what they can express

To support our partners, we:
- define an information superset for the models above, and
- organise it in such structured way…

…so that interchange is possible

Interchange platform called “AIM”
Contracts and static analysers (FPTC) plug-in to AIM
AIM Key Features

- AIM = Architectural Information Modelling
  - Abstraction layers
    - from high level specification to detail-rich lower levels to source code
  - View driven design
    - AIM is a relational database; views are queries
    - enabling specialist engineers (control, comms, safety) to work under same umbrella
  - Change management
    - monitor-detect-propagate-notify mechanism
    - supported by contracts technology
  - Extensible
    - usable for other domains (e.g. distributed systems, SoS)
AIM Views

Requirements View
- more detail
  (abstraction layering)
- requirements
  engineers

Deployment View
- change control
- view inter-dependency
- different structure
  (but same underlying model)
- system architects

Temporal View
- schedulability
  analysis
AIM – Top Level Elements

- **Model**
  - single instance
  - points to the ‘system’ element
- **Environment**
  - automatic/manual creation
  - in support of simulation
- **Classes, Implementations**
  - structural elements (extension through classes)
  - components, connections, topologies
- **Properties**
  - non-functional behaviour (extension)
- **Constants, Types, Collections**
  - supporting elements (configuration, extension & grouping)
- **Contracts**
  - assertions, constraints
AIM and AADL

- Features/capabilities of AIM which might enrich AADL
  - support for trusted component development embedded in component description
    - signatures – unique ID, version, dates, namespace (e.g. *RollsRoyce.Engines.Controls.TAY.Oil*), owner (company, dept., team, project, manager, etc.)
  - classes
    - root class *Undefined*, abstract (implementation-less) class → reuse, rapid model prototyping
    - composable interfaces (ports, port groups, inverted interface)
    - optional and multiple choice services
      - example: MissileCommsLink service = wired or wireless
AIM and AADL (2)

- **classes** - multiple layers of abstraction
  - AL0 (boxes and lines - unqualified) AL1 (system, subsystem, hardware, software, binding, data link...), AL2 (process, data, task, processor, memory, device,...), AL3 (application, variable, package, subprogram,..)

- **implementations** - *conceptual* (class-less) → rapid (semi-formal) modelling

- **properties** - as in AADL, defined in sets, but also within a namespace

- **collections** - support for various important arch. concepts:
  - layers, mode enabled component groups, real-time transactions, schedulability analysis, packages, ...
  - grouping of components in non-structural (conceptual, logical) way

- **types** - predefined types (integer, boolean, string, float, component,..), but also a template mechanism to allow extension
  - templates: enumeration, array, set, range, custom, class
Outline

• Background to HIRTS DARP
• Architectural Descriptions and Modelling
• Contracts and Incremental Certification
• Model Testing and Analysis
• Work in Progress & Beyond

THE UNIVERSITY of York

Defence & Aerospace
Research Partnership
Contracts

• Contract binds two or more entities
  – Describes functionality offered and required by a component
  – Describes required relationship between components

• Pre and post conditions on component/function operation
  – Typically used for data, used to compose functions

• Contracts extended to capture constraints for:
  – Timing - e.g. required refresh times
  – Other non-functional properties - e.g. scientific units
  – Possible additions to AADL:
    • Extension of property definitions to support more complete descriptions and ensure compatibility (e.g. Mars Pathfinder mission)
Incremental Certification

• Incremental certification (definition):
  – Ability to alter component(s) in a completed system without the need to completely recertify the system
  – Also has a role in development

• Role of contracts:
  – Check when system model is changed
  – Highlight unintentional “breakage”
Contracts example

- Interlock Handler
  - Executes periodically to check value in Mode pool
  - If detects ABANDON mode then initialises the firing interlock
  - “Last gasp” defence against firing

```cpp
contract ILHBehaviour {
    safety_case_link = AbortMissileGoal;
    attached = objInterlockHandler;
    assertion ilhpre pre {
        time@now >= LastExecutionTime + period
    }
    assertion ilhpost post {
        if pMode == ABA then
            pStopFiring = TRUE
        else
            pStopFiring = FALSE
        endif
    }
}
```
Managing change

1. Infrastructure
2. Properties
3. Contracts/Constraints

support for incremental certification
Model Driven Development

• Research based around Object Management Group MDA initiative
  – Automatic model transformations
  – Convert platform independent models to platform specific models

• Our research
  – Developing rules for automatic transformations of AIM models and contracts, whilst preserving essential constraints
  – Adding components e.g. an OS
  – Reconfiguring system models e.g. moving processes
MDA Approach

• Supports incremental certification by:
  – Supporting impact assessment
  – Allowing assessment of multiple configurations of a system e.g. to determine tolerance to failures or optimise configuration
  – Ability to check new models with contract checking and model animation tools
Outline

• Background to HIRTS DARP
• Architectural Descriptions and Modelling
• Contracts and Incremental Certification
• Model Testing and Analysis
• Work in Progress & Beyond
FPTC

• Failure propagation and transformation calculus
  – Unique amongst architectural modelling analyses because it directly addresses safety
• With FPTC, given an architectural model
  – Decide what failures are of interest
  – Describe the response to failure of each component and communications mechanism
  – Thence calculate the whole-system response
• Failure types
  – Early, late, omission, commission, stale value, detectable value, undetectable value
FPTC: Example

CLK 1

SENSOR

motion

FLOW RATE CALCULATION

rate

DECISION CONTROLLER

action

ACTUATOR

CLK 2

MODE CONTROLLER

mode

events
FPTC: Component Behaviour

SENSOR

CLK 1

* → late

FLOW RATE CALCULATION

(late, *) → stale value

(rate, _) → v

DECISION CONTROLLER

v → (v, v)

EVENTS

MODE CONTROLLER

mode

ACTION

CLK 2

ACTUATOR

.motion

late

stale value

early

late

stale value

* → late

(v, _) → v

(omission) → late

commission → stale value

v → v
FPTC: System Response

SENSOR

FLOW RATE CALCULATION

DECISION CONTROLLER

ACTUATOR

CLK 1

CLK 2

* → late

(late, *) → stale value

(v, _) → v

v → (v, v)

mode

motion

rate

events

omission → late
commission → stale value
early → *

v → v

(v, v)
FPTC Benefits

• FPTC can directly answer these questions:
  – What faults get expressed into the environment?
  – How did they arise?
  – Are there common causes of failure?
  – How likely are different failure types?
  – FPTC could be a likely candidate for one of the AADL Error Model implementations
FPTC: Probability

Fault type = LOSS

Wiring A

SENSOR

P(loss) = 0.1

P = 0.1

loss → loss

P(loss) = 0.2

Wiring A

v → v

CPU

P(loss) = 0.28

P(loss) = 0.1

SENSOR

P = 0.1

loss → loss

*(*) → *

(*(loss, loss)) → loss

P(loss) = 0.136

SPLIT

P(loss) = 0.1

P(loss) = 0.1

Wiring B

loss → loss

*(loss, loss) → *

P = 0.2

CONSOLID.
Benefits Continued

• Comparing different designs, FPTC can:
  – Show the impact of a change
  – Altered component
  – Altered communication
  – Predict whether likelihood of failure is smaller/larger
  – Reveal effects far distant from the site of change
Outline

• Background to HIRTS DARP
• Architectural Descriptions and Modelling
• Contracts and Incremental Certification
• Model Testing and Analysis
• Work in Progress & Beyond
Work in Progress & Beyond

• DARP ends September 2006
• Integrating activities
  – Stronger links AIM, contracts and FPTC
• Case studies
  – TAY, work with MBDA
• Future exploitation
  • Via tools
  • Via standards such as AADL