Domain Specific V&V Strategies for Aircraft Applications

Aliki Ott
tsio@tzi.uni-bremen.de

Tobias Hartmann
hartmann@tzi.uni-bremen.de

Center for Computing Technologies, Safe Systems
University of Bremen

6th ICSTEST International Conference on Software Testing 2005
7th April 2005
In this presentation,...

- we describe a novel approach to improve development, verification and validation processes, providing means for
  - generating requirement documents for a specific aircraft based on generic development documents
  - deriving V&V documents for structured reviews, test, test validation and formal verification based on generic verification and validation documents
Motivation I: Improve Development Process

Development Process

A/C Functional Requirements
System Top-Level A/C Requirements
Top-Level System Requirements Document
Functional Requirement Document (FRD)
System Requirements Document (SRD)
System Description Document (SDD)
Purchaser Technical Specification (PTS)
Equipment PTS
System /Equipment Development

V & V Process

System Acceptance Test
Route Proving
Flight Tests
Ground Tests
System Integration Tests
Multi-System Tests
System Tests
Qualification Tests
Equipment Tests
Production
Motivation I: Improve Development Process

Specification Process

- FRD
- SRD
- SDD
- PTS

System Implementation

Code

Drawbacks

- State-of-the-art specification techniques are combinations of informal, semi-formal and formal languages
  - ambiguous, unprecise, over-flexible, incomplete, unknown semantics
- No tool support for refinement of specification and design documents
- Manual code generation
Motivation I: Improve Development Process

Aim

- Use widely accepted graphical specification techniques with formal semantics
- Tool support for
  - refinement of specification models
  - generation of executable code or code fragments
Motivation I: Improve Development Process

generic specification documents

Aim

- Use generic development documents which abstract from aircraft specific variations
- Apply instantiation rules to derive development documents for a specific aircraft
Domain Analysis

A/C Specifications, standards and directives

Domain Analysis: Generic abstraction from A/C specific variations

Domain-specific Generic Specifications

- Generic Development Documents
  - Architectural and behavioural specifications
  - Data Dictionary
  - HW specification and architectural mapping
### Specification Formalism: HybridUML

- **UML 2.0 Profile with formal semantics**
  - integrates extensions for time-discrete and time-continuous behaviour → feasible for real-time systems
  - concepts for hierarchy, parallel composition, separation of concerns → feasible for complex systems
  - graphical representation → "intuitive", widely acceptable
  - formal transformation into *Hybrid Low-Level Language Program* (HL³ Program) → executable for simulation and testing

- **Tool support possible**
  - development of specification docs
  - static analysis
  - code generation
Specification Formalism: HybridUML

Structural Specifications
- class diagrams
- composite structure diagrams

Behavioural Specifications
- statechart diagrams
Example: Fire and Smoke Detection System

Composite Structure Diagram
statemachine sm_FSDC_poll

/ t := maxT - ID * offset;
msgAlarm := 0; noReply := 0; msgFail := 0

[ condContinuePolling ]
/ sendRequest (RequestMsg::ID); t := 0

[ NOT(condInTime) ]
/ noReply++

[ condFailureStatus ]
/ msgFail++

[ condInTime ]
/ noReply := 0

[ condDetectorAlarm ]
/ alarm := true

[ condDetectorFailure ]
/ failed := true

[ condAlarmStatus ]
/ msgAlarm := 0; msgFail := 0

[ condStandbyStatus ]
/ msgAlarm := 0; msgFail := 0

[ flow: t := 1 ]
Hierarchy of Generic HybridUML Models

- Airbus
- A318
- A340
- A380

<<merge>>

<<realization>>

A380, customer 1
A380, customer 2
Detailing Generic HybridUML Models

- Refinement of datatypes
  Example:
  
<table>
<thead>
<tr>
<th>&lt;&lt;datatype&gt;&gt;</th>
<th>Telegram</th>
</tr>
</thead>
<tbody>
<tr>
<td>deck : Integer</td>
<td></td>
</tr>
<tr>
<td>comp : Integer</td>
<td></td>
</tr>
<tr>
<td>message : String</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

- Refinement of agents, statemachines, control modes
  - by adding more internals but without changing the public interface
  - by specialising the public interface

- Substitution or extension of labels in statemachines

→ HybridUML Profile defines what is possible and how
Instantiation of a Generic HybridUML Model

- Instantiation with specific configuration data
  Example:
  - *very small aircraft*: `numSD := 10`
  - *very large aircraft*: `numSD := 100`
Instantiation of a Generic HybridUML Model

```
<<agent>>
Cockpit_domain (conf : CockpitConfig)

<<agent>>
Cabin_domain (conf : CabinConfig)

<<agent>>
FSDS (fsds_conf : FSDSConfig)

<<agent>>
FSDC (fsds_conf : FSDSConfig)

<<agent>>
FSDC_poll (conf : FSDSConfig)

<<agent>>
FSDC_mon (conf : FSDSConfig)

class FSDS (fsds_conf : FSDSConfig)

: SmokeDetector [fsds_conf.numSD]
recvRequest (RequestMsg)
sendStatus (statusMsg)

: FSDC (fsds_conf)

sendRequest (RequestMsg) [fsds_conf.numSD]
recvStatus (statusMsg) [fsds_conf.numSD]

ptp
```

Aircraft
(ac : AircraftConfig)

fsds_conf.numSD

ptp

Universität Bremen

Aliki Ott
7th April 2005
Motivation II: Improve V&V Process

A/C Functional Requirements
System Top-Level A/C Requirements
Top-Level System Requirements Document
Functional Requirement Document (FRD)
System Requirements Document (SRD)
System Description Document (SDD)
Purchaser Technical Specification (PTS)
Equipment PTS
System/Equipment Development
Production

Development Process

V & V Process

System Acceptance Test
Route Proving
Flight Tests
Ground Tests
System Integration Tests
Multi-System Tests
System Tests
Qualification Tests
Equipment Tests
Production
Motivation II: Improve V&V Process

Drawbacks:
- validation and test planning usually start too late
- change of un-testable design is expensive
- clarification of requirement specifications needs re-validation
Motivation II: Improve V&V Process

Aim:
- consider V&V activities as early as possible to provide feedback for the requirement specification and design phase
- support automated generation of V&V templates or test cases
A/C Specific Instantiation Rules

Generic Domain Specific Req. Documents

Validation Patterns

Generic MC Proof Obligations and Models

Generic Test Case Generation Rules

Instantiation Process

A/C specific Req. Documents

Structured Reviews

Specific MC Proof Obligations and Models

Simulation Specifications

Test Specifications

Requirements Definition

Requirement Validation

System/Equipment Development

System Testing

Fault Diagnosis
A/C Specific Instantiation Rules

Generic Domain Specific Req. Documents

Validation Patterns

Generic MC Proof Obligations and Models

Generic Test Case Generation Rules

Instantiation Process

A/C specific Req. Documents

Structured Reviews

Specific MC Proof Obligations and Models

Simulation Specifications

Test Specifications

Requirements Definition

Requirement Validation

System/Equipment Development

System Testing

Fault Diagnosis

Aliki Ott
7th April 2005

Universität Bremen
Validation Pattern –
Generic Structured Review Templates

- Assembled during domain analysis
- Used for design and requirement validation

Different levels of abstraction

1. Domain level
   • Inter-system communication with AFDX or ARINC 429?

2. Generic level for dedicated systems
   • Correct protocols and communication medium for intra-system communication? (CAN, Ethernet, proprietary, ...)

3. Specific level for dedicated systems
   • Correct bits of ARINC labels used?
Validation Pattern –
Generic Structured Review Templates

Different areas under investigation

1. Architecture
   - Adequate consideration of safety requirements? Redundancy concepts?

2. Interface Verification

3. Configuration aspects
   - Compliance with standards and safety requirements
   - Consistency and completeness of descriptions

4. Behavioural aspects
   - Accuracy of flow conditions, data transformation, timing constraints?
   - Implementable solution? Compatible with target hardware?

5. Verifiability
   - Verifiable by HW/SW integration tests, SW integration tests, unit tests, or analysis?

6. Compliance with standards
Structured Review Templates – Instantiation

**Generic** Structured Review Templates

**Instantiation rules for a specific aircraft**

**Instantiation process**

**Specific** Structured Review Templates for manual analysis
### Structured Review Templates – Example

**Section 1: Architecture Verification**
...

**Section 2: Interface Verification**

#### 2.1 Domain level
- A429 interface between FSDS and FWS
  - [A340] [FSDS] [FWS]
- AFDX interface between FSDS and FWS
  - [A380] [FSDS] [FWS]
- A429 interface between FSDS and CMS
  - [A340] [FSDS] [CMS]
- AFDX interface between FSDS and CMS
  - [A380] [FSDS] [CMS]
- ...
- connected to normal and essential power
- ...

#### 2.2 Generic Level

##### 2.2.1 Generic Level for FSDS
- CAN interface to Smoke Detector
- ...

---

**Instantiation rules**
A380, FSDS

**Instantiation process**

---

**Section 2: Interface Verification**

#### 2.1 Domain level
- AFDX interface between FSDS and FWS
- AFDX interface between FSDS and CMS
- connected to normal and essential power
- ...

#### 2.2 Generic Level

##### 2.2.1 Generic Level for FSDS
- CAN interface to Smoke Detector
- ...

---

**Universität Bremen**

Ailiki Ott
7th April 2005
Concept for Simulation and Testing

HybridUML Model
- Graphical Representation
- Textual Representation (XMI,...)

Runtime Environment
- HL³ Scheduler
- Time Services
- Channel
- Cluster communication

Execution Strategy Selector for Hybrid UML
- Simulation
- Testing

Simulation / Test Execution

HybridUML – UML 2.0 Profile

Framework of Hybrid Low-Level Language (HL³)

transformation Φ
Conclusion: Current Research and Development Activities

- Simulation environment for HybridUML models
- Simulation of HybridUML models interacting with Matlab/Simulink models
- Test Strategies for HybridUML models
  - test data selection for discrete changes of observables (inspired by Chow, Binder,…)
  - test data selection for time-continuous observables (inspired by equivalence and model checking methods for analog circuits)
Conclusion: Advantages of the Approach

- Unified specification formalism HybridUML for generic and specific requirement and design documents
  - based on the widely accepted graphical specification language UML
  - tool support possible

- Domain analysis assembles domain-specific general information for
  - generic structural and behavioural descriptions of the systems
  - domain-specific data dictionaries and glossaries
  - structured review templates on different abstraction levels

- Transformation of (specific) HybridUML models into HL³ programs which are executable in hard real-time (for simulation and for testing)
Conclusion: Application Areas

- **KATO** with Airbus Deutschland GmbH
  - new methods, tools and technologies to improve the development and verification of future aircrafts including means for fault diagnosis

- **DFG research project HYBRIS**
  - Development and simulation of HybridUML models for train control systems
Slides and further information

- slides
  http://www.informatik.uni-bremen.de/~tsio/papers/info_TH05.html

- request for HybridUML related documents incl. HybridUML profile
tsio@tzi.uni-bremen.de
Appendix: Example
Example

Class Diagram

- Aircraft
  - Cockpit_domain
    - FWS
    - ACS
    - VCS
  - Cabin_domain
    - FSDS
    - SmokeDetector
  - Utilities_domain
    - FSDS
  - Energy_domain
  - OIS_domain
    - FSDC
    - SmokeDetector
  - PCES_domain
    - FSDC_poll
    - FSDC_mon

- OIS  Onboard Information System
- PCES  Passenger and Crew Member Communication and Entertainment Services
- FWS  Flight Warning System
- FSDS  Fire and Smoke Detection System
- ACS  Air Conditioning System
- VCS  Ventilation Control System
Example: Fire and Smoke Detection System
Example: Fire and Smoke Detection System

Class Diagram

```
public
    temperature : AnalogReal
    smoke : AnalogReal
    N-PWR : Boolean
    E-PWR : Boolean
signal
    sendMsg ( Telegram )
```
Example: Fire and Smoke Detection System

Composite Structure Diagram
Example: Fire and Smoke Detection System

Composite Structure Diagram

class FSDS ( fsds_conf : FSDSConfig )

: SmokeDetector [fsds_conf.numSD]
  recvRequest ( RequestMsg )
  sendStatus ( statusMsg )

: FSDC ( fsds_conf )
  recvRequest ( RequestMsg ) [fsds_conf.numSD]
  sendRequest ( RequestMsg ) [fsds_conf.numSD]
  recvStatus ( statusMsg ) [fsds_conf.numSD]
  sendMsg (...)

 sendMsg (...)

温度

smoke

E-PWR

N-PWR
Example: Fire and Smoke Detection System

```java
<<agent>>
FSDC (fsds_conf : FSDSConfig)

public
    N-PWR : Boolean
    E-PWR : Boolean

signal

recvStatus ( StatusMsg ) [fsds_conf.numSD]
sendMsg (...)

<<agent>>
SmokeDetector

class
    public
        temperature : AnalogReal
        smoke : AnalogReal
        N-PWR : Boolean
        E-PWR : Boolean

signal

recvRequest ( RequestMsg )
sendStatus ( StatusMsg )
```

sm_SmokeDetector
Example: Fire and Smoke Detection System

class FSDC (config: FSDS_Config)

: FSDC_poll (config) [config.numSD]
  sendRequest (...) [ ]
  recvStatus (...) [ ]
  N-PWR E-PWR

: FSDC_mon (config)
  failed alarm
  ptp

N-PWR E-PWR

sendRequest (...) [ ]
recvStatus (...) [ ]

numSD

<<agent>>
FSDC

<<agent>>
SmokeDetector

numSD

<<agent>>
FSDC_poll

<<agent>>
FSDC_mon

recvStatus (...) []
sendRequest (...) []

E-PWR sendMsg (...) sendMsg (...)

FWS

numSD

<<agent>>
FSDC_poll

<<agent>>
FSDC_mon

recvStatus (...) []
sendRequest (...) []
Example: Fire and Smoke Detection System

```
public
    failed : Boolean
    alarm: Boolean
    N-PWR : Boolean
    E-PWR : Boolean

private
    status : Status Message
    msgAlarm : Integer
    noReply : Integer
    msgFail : Integer
    t : Clock

signal
    sendRequest ( RequestMsg )
    recvStatus ( StatusMsg )
```

sm_FSDC_poll
Example: Fire and Smoke Detection System

Statechart Diagram

```plaintext
statemachine sm_FSDC_poll

wait_for_next_cycle
[ inv: 0 < t < maxT ]
/ t := maxT - ID * offset;
  msgAlarm := 0; noReply := 0; msgFail := 0
[ condContinuePolling ]
/ sendRequest (RequestMsg::ID); t := 0
[ NOT(condInTime) ]
/ noReply++
[ condFailureStatus ]
/ msgFail++
[ condInTime ]
/ noReply := 0
[ condStandbyStatus ]
/ msgAlarm := 0; msgFail := 0
[ condAlarmStatus ]
/ msgAlarm++; msgFail := 0
[ condDetectorAlarm ]
/ alarm := true
[ condDetectorFailure ]
/ failed := true
alarm -> failed

wait_for_reply
[ inv: condInTime ]
[ condInTime ]
recvStatus (status)
/ noReply := 0
[ condStandbyStatus ]
/ msgAlarm := 0; msgFail := 0
[ condAlarmStatus ]
/ msgAlarm++; msgFail := 0
[ condFailureStatus ]
/ msgFail++
failed
```

... ...
statemachine sm_FSDC_poll

\[ \text{inv: } 0 < t < \text{maxT} \]

\[ \begin{align*}
\text{wait_for_next_cycle} & \quad \text{[ flow: } t := 1 \text{ ]} \\
\text{wait_for_reply} & \quad \text{[ condInTime ]}
\end{align*} \]

- \[ \text{condContinuePolling} \]
- \[ \text{sendRequest (RequestMsg::ID); } t := 0 \]
- \[ \text{NOT(condInTime)} \]
- \[ \text{noReply}++ \]
- \[ \text{recvStatus (status)} \]
- \[ \text{noReply} := 0 \]

- \[ \text{condStandbyStatus} \]
- \[ \text{msgAlarm} := 0; \text{msgFail} := 0 \]

- \[ \text{condAlarmStatus} \]
- \[ \text{msgAlarm}++; \text{msgFail} := 0 \]

- \[ \text{condFailureStatus} \]
- \[ \text{msgFail}++ \]

- \[ \text{condDetectorAlarm} \]
- \[ \text{alarm} := \text{true} \]

- \[ \text{condDetectorFailure} \]
- \[ \text{failed} := \text{true} \]
Example: Fire and Smoke Detection System

Conditions for Statemachines \( sm_{FSDC\_poll} \)

- \( \text{condInTime} = (t < \max T) \)
- \( \text{condContinuePolling} = (\text{msgFail} < \max \text{MsgFail}) \land (\text{msgAlarm} < \max \text{Alarm}) \land (\text{noReply} < \max \text{NoReply}) \land (t == \max T) \)
- \( \text{condAlarmStatus} = (\text{status.alarm}) \land (\neg (\text{status.standby})) \)
- \( \text{condStandbyStatus} = (\neg (\text{status.alarm})) \land (\text{status.standby}) \)
- \( \text{condFailureStatus} = (\text{status.alarm} == \text{status.standby}) \)
- \( \text{condDetector Failure} = (\text{noReply} \geq \max \text{NoReply}) \lor (\text{msgFail} \geq \max \text{MsgFail}) \)
- \( \text{condDetectorAlarm} = (\text{msgAlarm} \geq \max \text{Alarm}) \)