Automated Exploration of Alternative System Architectures with VDM-RT

Kenneth Lausdahl[1] Augusto Ribeiro[1]

[1] Aarhus School of Engineering, Dalgas Avenue 2, DK-8000 Aarhus C, Denmark

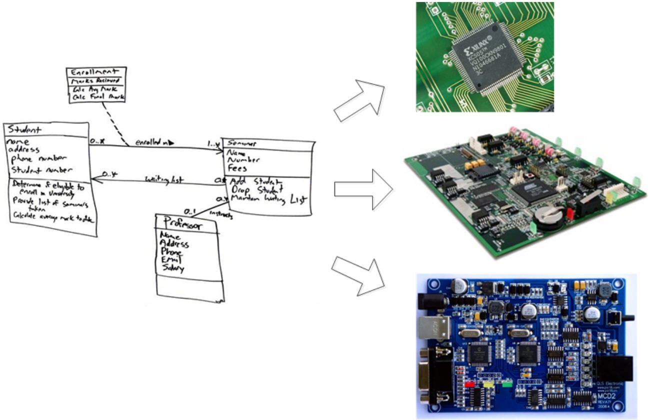
20 June 2011 / 9th Overture Workshop

Outline

1. Introduction
   - Motivation
   - The VDM Real-Time Dialect
2. Ensuring Separation Between Software And Hardware
   - Alternative Structure
   - Work-flow
3. Exploration of Alternative System Architectures
   - Exploring alternative artifact distribution on a fixed hardware configuration
   - Exploring alternative hardware configurations for an ASA
   - Exploring alternative deployment parameters for a fixed configuration
   - Case study

Choosing the best architecture for deployment

What is a good architecture?



Choosing the best architecture for deployment

Questions

Design questions:

1. Does the proposed architecture meet the performance requirements of all applications?
2. How robust is the chosen architecture with respect to changes in the application or architecture parameters?
3. Is it possible to replace components by cheaper, less powerful equivalents to save cost while maintaining the required performance targets?

Contribution by this work

This work addresses the design questions through:

- separation between the core model and deployment
- enabling automated exploration of system architectures
- architecture validation with system properties

VDM Real-Time System

A special **`system`** class defines hardware topology and deployment of a model.

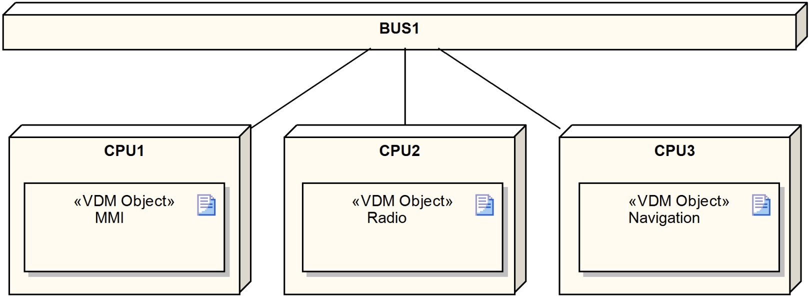
A system class consists of the following:

1. Artifacts to be deployed.
2. A number of `CPU`s.
3. A number of `BUS`es connecting `CPU`s together.
4. A constructor where artifacts are connected and deployed.

A single VDM-RT model is only allowed to define a single **`system`** class as part of the model.

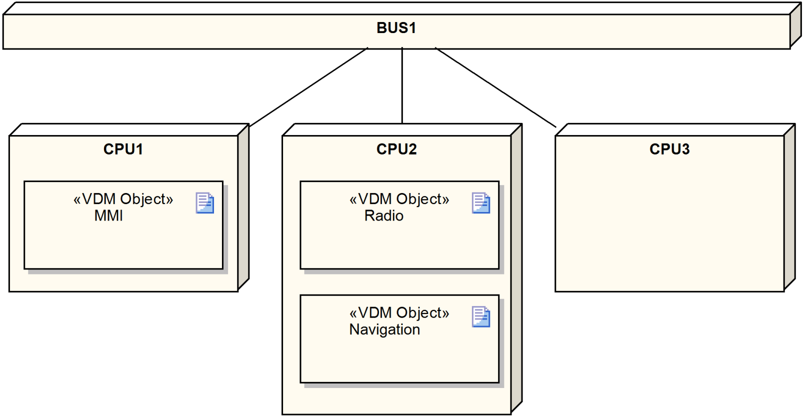
VDM Real-Time System

Example: In-car navigation system



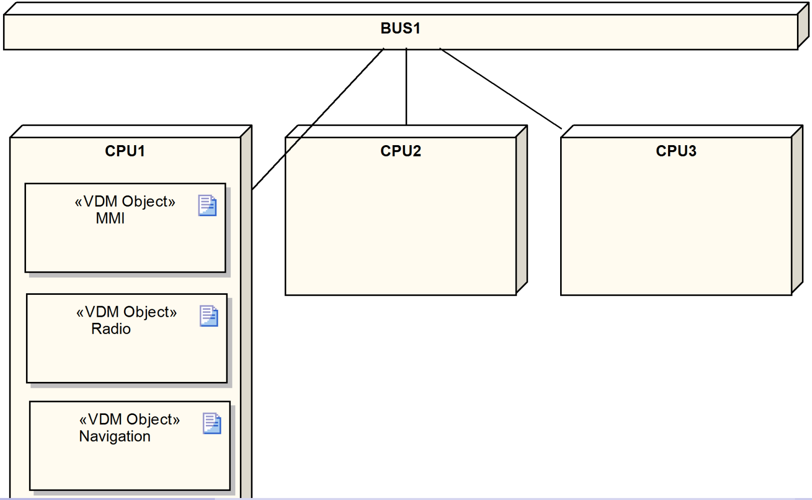
VDM Real-Time System

Example: In-car navigation system



VDM Real-Time System

Example: In-car navigation system



VDM Real-Time System

Example: In-car navigation system

```
system RadNavSys

instance variables
  -- artifacts to be deployed
  mmi : MMI := new MMI();
  radio : Radio := new Radio();
  navigation : Navigation := new Navigation();

  -- Hardware
  CPU1 : CPU := new CPU (...);
  CPU2 : CPU := new CPU (...);
  CPU3 : CPU := new CPU (...);

  BUS1 : BUS := new BUS (..., {CPU1, CPU2, CPU3})
```

VDM Real-Time System

Example: In-car navigation system

```
operations
  public RadNavSys: () ==> RadNavSys
  RadNavSys () ==
    ( navigation.setMmi(mmi);
      radio.setMmi(mmi);
      radio.setNavigation(navigation);
      mmi.setRadio(radio);

      CPU1.deploy(mmi,"MMI");
      CPU2.deploy(radio,"Radio");
      CPU3.deploy(navigation,"Nav");
      ...
    );

end RadNavSys
```

Outline

1. Introduction
   - Motivation
   - The VDM Real-Time Dialect
2. Ensuring Separation Between Software And Hardware
   - Alternative Structure
   - Work-flow
3. Exploration of Alternative System Architectures
   - Exploring alternative artifact distribution on a fixed hardware configuration
   - Exploring alternative hardware configurations for an ASA
   - Exploring alternative deployment parameters for a fixed configuration
   - Case study

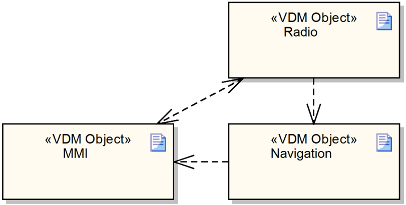
Enabling Exploration of Alternative Architectures

VDM-RT is currently limited to express a single deployment of a model.

Our alternative structure to the VDM-RT `system` class

- Abstract Software Architecture - ASA
- Abstract Hardware Architecture - AHA
- Configuration
- Deployment

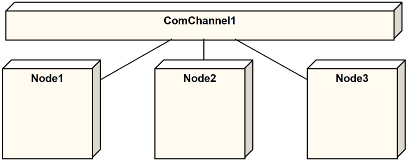
Abstract Software Architecture




```
types
Artifact : seq of char

ASA ::
  artifacts    : set of Artifact
  dependencies : map Artifact to set of Artifact
```

Abstract Hardware Architecture

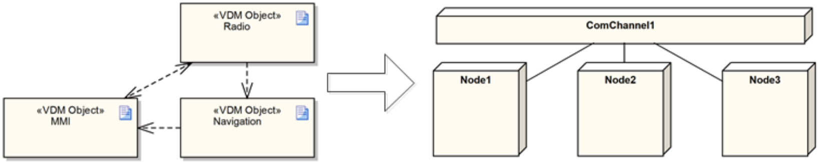



```
Node ::
  id : nat1;

ComChannel ::
  nodes : set of Node;

AHA ::
  nodes    : set of Node
  channels : set of ComChannels
```

Configuration

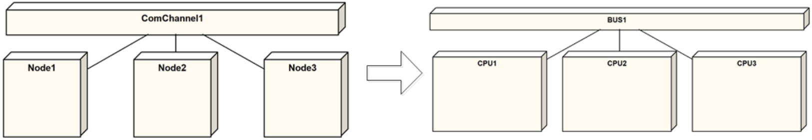



```
NodeArtifactRelation : map Node to set of Artifact;

Configuration ::
  asa : ASA
  aha : AHA
  relation : NodeArtifactRelation
```

Deployment

- Node and ComChannel are refined to CPUs and buses.




```
Deployment ::
  config : Configuration
  buses  : map ComChannel to BUS
  cpus   : map Node to CPU
```

New Work-flow

1. Specify the model.
2. Specify artifact configuration at system level.
3. Extract ASA.
4. Create or generate AHA from ASA dependencies.
5. Specify connection between ASA and AHA.
6. Specify deployment, refine Nodes and ComChannels to CPUs and buses.

$$\left(ASA + AHA\right) \rightarrow^{*} Configuration \rightarrow^{*} Deployment \equiv \texttt{system} \quad (1)$$

Outline

1. Introduction
   - Motivation
   - The VDM Real-Time Dialect
2. Ensuring Separation Between Software And Hardware
   - Alternative Structure
   - Work-flow
3. Exploration of Alternative System Architectures
   - Exploring alternative artifact distribution on a fixed hardware configuration
   - Exploring alternative hardware configurations for an ASA
   - Exploring alternative deployment parameters for a fixed configuration
   - Case study

Exploration Requirements

The general design questions lead to the following requirements:

It must be possible to:

- Explore alternative artifact distribution on a fixed hardware configuration.
- Explore alternative hardware configurations for an ASA.
- Explore alternative deployment parameters for a fixed configuration.

Additionally, automated validation of a deployment is required.

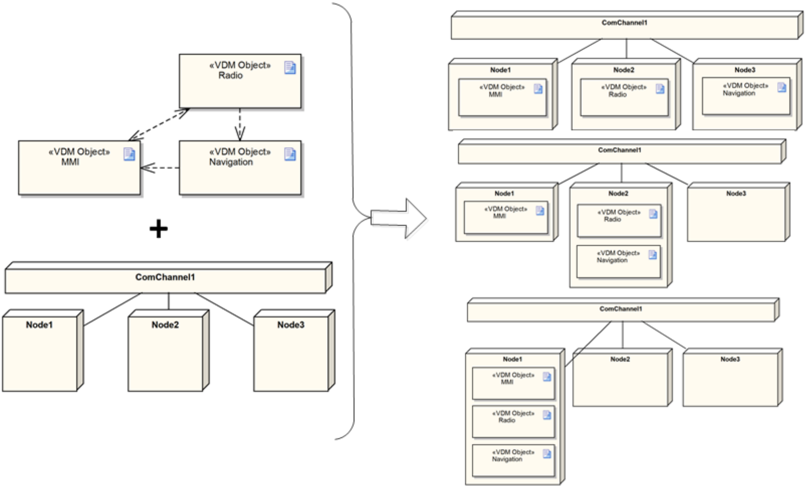
Exploring alternative artifact distribution on a fixed hardware configuration

The goal is to create all possible configurations which respects the fixed:

- ASA - with restrictions on possible distributions.
- AHA - with restrictions on communication paths.

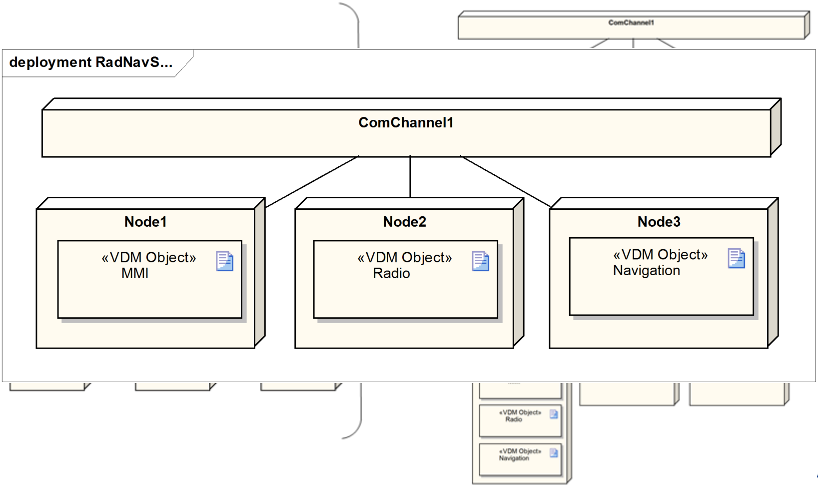
$$\left(ASA + AHA\right) \rightarrow^{*} \underline{Configuration} \rightarrow^{*} Deployment \equiv \mathtt{system} \tag{2}$$

Configurations
Example

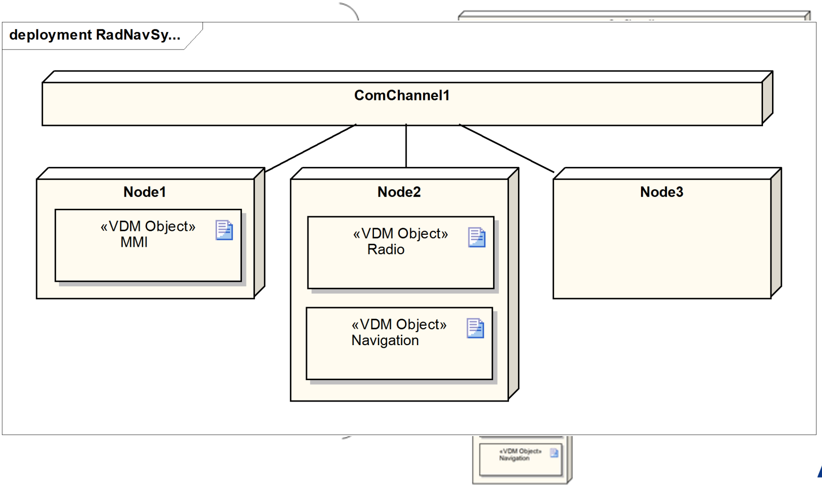


Configurations

Example

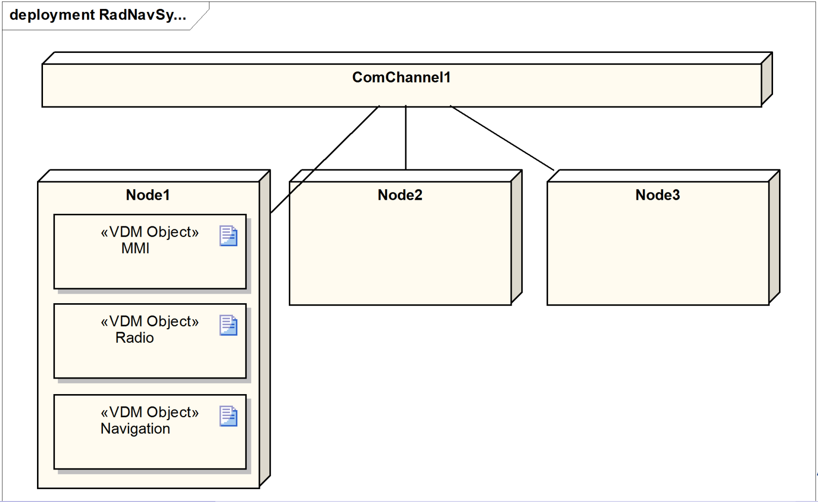


Configurations
Example



Configurations

Example



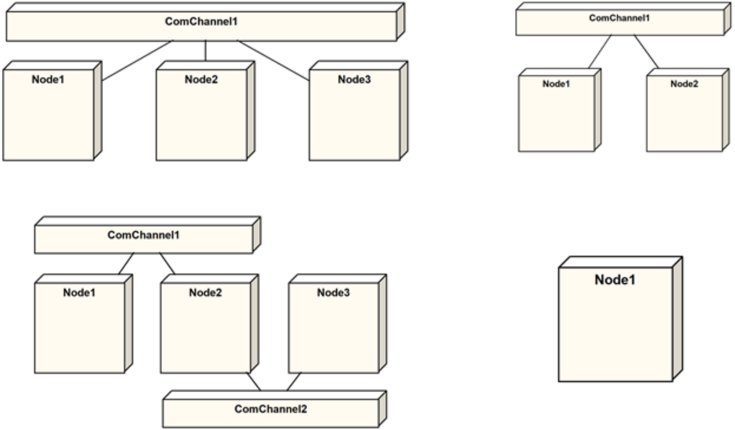
Exploring alternative hardware configurations for an ASA

The goal is to explore all possible hardware topologies which are compatible with the given ASA.

$$\left(ASA + \underline{AHA} \right) \rightarrow^{*} Configuration \rightarrow^{*} Deployment \equiv \texttt{system} \tag{3}$$

Exploring alternative hardware configurations for an ASA

Example



Exploring alternative deployment parameters for a fixed configuration

The goal is to try out different node refinements for an otherwise fixed system.

$$\left(ASA + AHA\right) \rightarrow^{*} Configuration \rightarrow^{*} \underline{Deployment} \equiv \texttt{system} \tag{4}$$

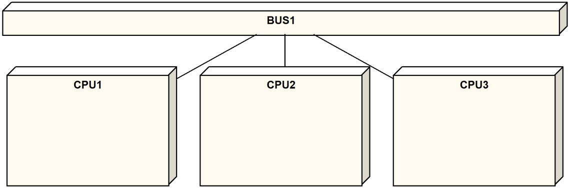
Exploring alternative deployment parameters for a fixed configuration

Example

Each Node, ComChannel is refined to a specific CPU or BUS with a specific calculation or transmission speed.

Exploration of a Node

- Try node1 with speeds { 200MHz, 100MHz, 50Hz} = CPU1
- Results in 3 different deployments.



Case Study with concrete syntax

Small case study with initial proposed syntax replacing the current **system** class.

```
system RadNavSys
instance variables
  -- artifacts to be deployed
  mmi : MMI := new MMI();
  radio : Radio := new Radio();
  navigation : Navigation := new Navigation();

operations
  public RadNavSys: () ==> RadNavSys
  RadNavSys () ==
    ( navigation.setMmi(mmi);
      radio.setMmi(mmi);
     ...
```

Case Study with concrete syntax

Syntax

```
aha
Channel1 := {node1, node2, node3}

configuration
node1 := {mmi};
node2 := {radio}
node3 := {navigation}

deployment
node1 := CPU(200MHz, <FP>)
node2 := CPU(100MHz, <FP>)
node3 := CPU(1000MHz, <FP>)
Channel1 := BUS(72E3, <CSMACD>)
```

Case Study with concrete syntax

Exploration of Deployment

```
deployment
node1 := {CPU(200MHz, <FP>),
          CPU(100MHz, <FP>),
          CPU(50MHz, <FP>)}
node2 := CPU(100MHz, <FP>)
node3 := CPU(1000MHz, <FP>)
Channel1 := BUS(72E3, <FCFS>)
```

Future Work

Planned work

- Development of exploration tool for:
  - AHA
  - Configuration
  - Deployment
- Future investigation of how priorities of functions/operations should be specified.
- Development of tool for run-time validation for time invariants.

Questions

Questions?