Modelling and Analysis of Distributed Embedded Real-Time Systems using VDM++ and VDMTools

ir. Marcel Verhoef
CHESS - The Netherlands

http://www.chess.nl
http://www.marcelverhoef.nl
Agenda - “Back to the Future”

• Introduction
• Using VDM++ and VDMTools: theory and practice
• In the past: successful projects in industry
• In the present: architecture and deployment
• In the future: continuous time co-simulation
• Conclusions
Who Is Marcel Verhoef?
The Past

Using VDMTools In The Commercial Enterprise
Dutch Government - Department of Defense (1)

- information management system
- mission-critical system component
- coupling to very large database
- 2 man-year development effort
- 98% VDM-SL (400 pages ± 15 kloc)
- ± 90 kloc C++ (code generation)
- delivered on-time and within budget
- no errors found after release
- still in use today
- return on investment: within project
Dutch Government - Department of Defense (2)

Traditional:

- Analysis & Design: 900
- Coding: 2000
- Testing: 700

VDMTools®:

- Analysis & Design: 1200
- Coding: 500
- Testing: 600

Cost:

0% - 64% - 100%
Flower Auction at Aalsmeer (1)

- largest covered market place in the world (1,000,000 m²)
- spot market for fresh cut flowers and plants
- 13 auction clocks in 4 halls
- world market share 45%
- Kenia, Israel, Ethiopia → Aalsmeer → Germany, Japan
- 44000 transactions per day
- 6.9 M Euro turnover per day
- obviously mission critical

1.0 e⁶ m² ≈ 200 football fields
6.9 e⁶ Euro ≈ 1.15 e⁹ JPY
Flower Auction at Aalsmeer (2)

- “Dutch auctioning” process
- up: 60 Hz  down: 30 Hz
- transaction: 5 seconds max
- new: on-line participation
- many challenges
  - unknown application area
  - use novel technology
  - unclear requirements
  - inexperienced team
  - no development process
- used VDM++ and UML
- completed successfully
The Present

Enhancing VDMTools For Embedded Systems
joint work with Peter Gorm Larsen and Jozef Hooman
[ LNCS 4085, FM 2006, pp 147-162 ]
Enhancing VDMTools For Embedded Systems

- Motivation: early life-cycle system architecting
- VDM++ for distributed embedded real-time
- Case Study: In-Car Radio Navigation System
early life-cycle system architecting (1)

• design paradox: “shooting at a moving target”
  – volatile requirements & many unknowns (not just technical)
  – nevertheless key architecture decisions must be made

• additional complications
  – business case always evolves over time
  – out-of-phase development usually occurs, for example
    mechanics → electronics → software
  – technology evolves much faster than project elapse time
early life-cycle system architecting (2)

• key problems
  – system-level overview is usually lacking (dominating view)
  – methods do not sufficiently support design iteration
    (building models takes too much time and effort)

• the proposed solution
  – notational extensions for context aware software models
    (explicit notion of architecture and deployment)
  – improve tool support for early model validation
    (enhanced visualisation)
VDM++ for distributed embedded real-time

- old VDMTools VICE version 6.6
  - all software implicitly deployed on a single CPU
  - only synchronous operation calls are allowed
  - only absolute notion of time (duration)
  - only strict periodic behaviour can be specified

- new VDMTools VICE version 8.0
  - hardware architecture can be described using BUS and CPU
  - explicit deployment of software (class instances) on CPUs
  - support for synchronous and asynchronous operation calls
  - absolute and relative notion of time (duration and cycles)
  - elaborate periodic behaviour (period, jitter or burst, offset)
Example: In-Car Radio Navigation System

- car radio with built-in navigation system
- several applications may execute concurrently
- user-interface needs to be responsive at all times
- traffic messages must be processed on time
- what is a suitable architecture for this product?
“Change Volume” application
“Handle TMC” application
Proposed Architecture Alternatives
Absolute and relative elapse time

class Radio
operations

async public AdjustVolume: nat ==> ()
AdjustVolume (pno) ==
  ( duration (150) skip;
   RadNavSys`mmi.UpdateVolume(pno) );

async public HandleTMC: nat ==> ()
HandleTMC (pno) ==
  ( cycles (10000) skip;
   RadNavSys`navigation.DecodeTMC(pno) )

end Radio
Absolute and relative elapse time

```plaintext
class Radio
operations

  async public AdjustVolume: nat ==> ()
  AdjustVolume (pno) ==
    ( duration (150) skip;
      RadNavSys`mmi.UpdateVolume(pno) );

  async public HandleTMC: nat ==> ()
  HandleTMC (pno) ==
    ( cycles (10000) skip;
      RadNavSys`navigation.DecodeTMC(pno) )

end Radio
```
Specifying the hardware architecture (1)

```java
system RadNavSys

instance variables

    -- create the class instances
    static public mmi := new MMI();
    static public radio := new Radio();
    static public navigation := new Navigation();
```
Specifying the hardware architecture (2)

... 

-- create the computation resources
CPU1 : CPU := new CPU(<FP>, 22E6, 0);
CPU2 : CPU := new CPU(<FP>, 11E6, 0);
CPU3 : CPU := new CPU(<FP>, 113E6, 0);

-- create the communication resource
BUS1 : BUS := new BUS(<FCFS>, 72E3, 0,
                           {CPU1, CPU2, CPU3})
Specifying the hardware architecture (3)

... 

operations

public RadNavSys: () ==> RadNavSys
RadNavSys () ==
  ( CPU1.deploy(mmi);
    CPU2.deploy(radio);
    CPU3.deploy(navigation) )

end RadNavSys
Modeling the environment of the system

class TransmitTMC
...
operations
  async createSignal: () ==> ()
  createSignal () ==
    ( dcl num : nat := getNum();
      e2s := e2s munion {num |-> time};
      RadNavSys`radio.HandleTMC(num) );

  async public handleEvent: nat ==> ()
  handleEvent (pev) == s2e := s2e munion {pev |-> time}
    post forall idx in set dom s2e &
      s2e(idx) - e2s(idx) <= 1000

thread
  periodic (3000, 4500, 1000, 0) (createSignal)
end TransmitTMC
In-Car Radio Navigation System Overview
Visualisation - ShowTrace "system view"
Visualisation - ShowTrace "resource view"
Enhanced Modelling Support - Summary

- Simple and intuitive language extensions
- Significant improvement expressiveness
- Much better domain applicability
- Significant decrease in model size
- "backward compatible" semantics
- Exploration of deployment and performance feasible
- Early detection of design bottlenecks by visualization
The Future

Embedding VDMTools Into System Engineering
joint work with Peter Visser, Jozef Hooman and Jan Broenink
[ LNCS 4591, IFM 2007, pp 639-658 ]
Embedding VDM++ Into System Engineering

• Motivation: multi-disciplinary system-level design

• Showdown: Continuous Time meets Discrete Event

• Case Study I: water tank level controller

• Case Study II: printer paper path controller
Beyond the Ordinary: Design of Embedded Real-time Control

- BODERC project @ ESI
- Sept 2002 - Apr 2007
- Multi-disciplinary design
  - mechanics
  - electronics
  - software
- High-tech systems focus
- Early life cycle trade-off analysis
- Industry as a laboratory
- http://www.esi.nl/boderc
Design of High-Tech Systems - State of Practice

- design is typically mono-disciplinary organised
- domain specific methods and custom tools are used
- out-of-phase development and system-level focus lacking
- cross-cutting concerns postponed to the integration phase
- late validation & feedback

"INTEGRATION HELL"
Multi-disciplinary Systems Design - The Vision

- system level approach
- model-driven design
- integrated models & tools
- rapid evaluation
- early feedback
- support design dialogue
- continuous integration
- continuous validation
- less effort overall
- higher quality

requirements

integrated models

mechanics
electronics
software

test & integration

PROJECT ELAPSE TIME
WEAK TENSION
The Challenge - Integrated Design Models (1)

• Notations and analysis techniques used by the disciplines are fundamentally different
  - mechanics : finite element methods
  - electronics : differential or difference equations
  - software : labelled transition systems

• Is a common notation feasible* at all?
  * [Henzinger & Sifakis, FM 2006 key note, LNCS 4085, pp 1-15]
The Challenge - Integrated Design Models (2)

- scope of discipline specific tools is widening
  - Matlab Simulink → Stateflow, Real-Time Workshop, TrueTime
  - Rhapsody → Simulink
  - UML → SysML

- bigger piece of the pie ≠ satisfy all stakeholders (disciplines)

- **problems**: poor abstraction, restrictive model of computation

- novel actor-based techniques*: Ptolemy-II

- **problems**: disruptive approach, weak semantics

* [http://ptolemy.eecs.berkeley.edu]
Our Approach - Integrated Design Models (3)

- Cross the continuous time - discrete event divide
- Select a well-defined (formal) notation on either side
- Explore semantic integration of those notations
- Implement tool support for this reconciled semantics
- Analyse combined models by (reliable) co-simulation
Expected *benefits* - Integrated Design Models (4)

- good abstraction facilities on both sides of the divide
- supports light-weight modelling required in early stages
- few a-priori MoC specific restrictions → avoid design bias
- fits in design flow → low threshold for industrial uptake

- inspired by previous experience (INFORMA EU project)
  weak coupling between VDMTools and Simulink
Continuous Time Realm - Bond Graphs

- dynamic systems modelling, physics domain independent
  - mechanics
  - electronics
  - pneumatics

- graphical notation: Bond graphs*

- formal analysis for algebraic loops and differential causalities

- model validation through simulation and visualisation

- industry grade tool support
  http://www.20sim.com

* [Gawthrop, Bevan, IEEE Control Systems Magazine, April 2007, pp 24 - 45]
Continuous Time Realm - Informal Semantics

- sets of differential equations
- approximate solution(s) numerically by
  - discrete integration over some time interval
- many “solver” algorithms available e.g. Euler
- CT shares state variables with DE model
- capture state events: zero-crossing detection
- capture time events: proceed to time $t > \text{now}$
Our Approach by Example - Water Tank Level Controller

\[ \frac{dV}{dt} = f_1 - f_0 \]

\[ f_0 = \begin{cases} 
\frac{\rho \cdot g}{A \cdot R} \cdot V & \text{if valve = open} \\
0 & \text{if valve = closed}
\end{cases} \]
Co-Simulation - Sharing State Variables

- **SENSOR**
- **ACTUATOR**
- **INTERRUPTS**

**LEVEL**

**VALVE**

DE: lwm hwm

CT

**LEVEL**

“ACTUATOR”

“INTERRUPTS”

“SENSOR”
Co-Simulation - State and Time Events

\[ \text{var} = p(t) \]

\[ \text{lwm} = \text{FEE} \text{(level, 2.0)} \quad \text{hwm} = \text{REE} \text{(level, 3.0)} \]
Our Approach by Example - water tank case (2)

01 variables
02 real volume, level;
03 parameters
04 real area = 1.0;
05 real gravity = 9.81;
06 real density = 1.0;
07 equations
08 // p.e = pressure, p.f = flow rate
09 // integrate flow to obtain volume
10 volume = int(p.f);
11 level = volume / area;
12 p.e = gravity * level * density;
Our Approach by Example - water tank case (3)

```java
01  class Controller
02
03  instance variables
04    static public level : real;
05    static public value : bool := false  -- default is closed
06
07  operations
08    static public async open: () ==> ()
09      open () == duration(0.05) value := true;
10
11    static public async close: () ==> ()
12      close () == cycles(1000) value := false;
13
14      loop: () ==> ()
15      loop () ==
16      if level >= 3 then value := true  -- check high water mark
17      else if level <= 2 then value := false;  -- check low water mark
18
19  threads
20      periodic(1.0,0,0,1.0)(loop)
21
22  sync
23      mutex(open, close, loop)
24
25  end Controller
```
Our Approach by Example - water tank case (4)
Printer paper path - case study (1)
Printer paper path - case study (2)

 continuous validation

co-sim results

SIL sim results

measurements
Printer paper path - case study (3)
public CtrlLoop: () => ()
CtrlLoop () ==
-- first retrieve the current encoder value
   ( dcl measured_value : real := ENCODER_GAIN * getEnc();
-- output the previous value if we are time synchronous
   if hold then setPwm(corr_profile, curr_setpoint, curr_error, next_output);
-- calculate the new pwm control value
   if feedback
     then next_output := limit(calcPID(measured_value))
   else next_output := limit(getSetpoint());
-- output the new value directly if we are not time synchronous
-- otherwise wait until the next period is due
   if not hold then setPwm(corr_profile, curr_setpoint, curr_error, next_output) )

thread
-- run the controller at 1 kHz and assume no jitter
periodic (0.001, 0, 0, 0.001)(CtrlLoop)
Printer paper path - case study (5)
Printer paper path - case study (6)
Printer paper path - case study (7)
Printer paper path - case study (8)
VDM++ In System Engineering - Summary

• Promising academic research results
  – coupling does not restrict tools or add complexity
  – co-simulation enables cross discipline design dialogue
  – small models due to powerful CT and DE abstraction
  – low effort design evaluation
  – discipline specific analysis on models is still possible
  – light-weight modelling can provide accurate answers

• generic integrated operational semantics
• (vendor) independent of continuous time simulator
• caveat: not yet available in VDMTools
Conclusions (1)

- VDM++ very suitable for managing complexity
- Applicable to embedded systems domain
- Not disruptive to current design practices
- Level of rigour can be chosen depending on task
- Improves quality of design dialogue dramatically
- Consistent documentation reduces maintenance cost
- Last but not least: excellent support from CSK
Conclusions (2)

- VDM++ and VDMTools help the system architect to
  - increase confidence in the design
  - to reduce project and product risks
  - while dealing with uncertainty
  - while working under high time pressure

- the system architect is empowered to
  - bridge the gap between the engineering disciplines
  - deal with design complexity in a cost-effective way
Back To The Future? The Future is Now!

Thank You for Your Attention

http://www.marcelverhoef.nl
Questions?

This week I mapped and gapped the requirements to consolidate everything into a program of work...

...to maximize synergy capture and optimize our resource utilization.

If any of that sounded like work, I'll do some more of it next week.