Applying Model Driven Development to the DDS Domain

Bruce Trask
Hans van ‘t Hag
<table>
<thead>
<tr>
<th>Object Oriented</th>
<th>Component Based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multithreaded/MultiProcess</td>
<td>Real-time</td>
</tr>
<tr>
<td>Embedded</td>
<td>C++/Java/Ada/VHDL</td>
</tr>
<tr>
<td>Platform Independent</td>
<td>High Performance</td>
</tr>
<tr>
<td>Heterogeneous</td>
<td>Distributed</td>
</tr>
<tr>
<td>Vital</td>
<td>Secure</td>
</tr>
<tr>
<td>Fault Tolerant</td>
<td>Portable</td>
</tr>
<tr>
<td>Portable</td>
<td>Standardized</td>
</tr>
<tr>
<td>Standardized</td>
<td>Declarative</td>
</tr>
<tr>
<td>Dynamic</td>
<td>Imperative</td>
</tr>
<tr>
<td></td>
<td><strong>So what’s the big deal?</strong></td>
</tr>
<tr>
<td></td>
<td>Each one <em>by itself</em> is difficult,</td>
</tr>
<tr>
<td></td>
<td>let alone doing them all <em>at the</em></td>
</tr>
<tr>
<td></td>
<td><em>same time</em></td>
</tr>
</tbody>
</table>
Computer Science, Software Engineering

- “Computer Science is the discipline that believes all problems can be solved with one more layer of indirection”
- Formalize the level of indirection
- Provide ways to program at this new level
- Provide automatic ways to get back to executable lower levels of abstraction
- Standardize
- Optimize

1 Dennis DeBruler, Refactoring, Fowler et. al. Addison Wesley 1999
## Levels of Abstraction

<table>
<thead>
<tr>
<th>Programming</th>
<th>Platform</th>
<th>More Productive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less Flexible</td>
<td></td>
<td>More Productive</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Higher Order Languages</td>
<td>Frameworks</td>
</tr>
<tr>
<td></td>
<td>Assembly</td>
<td>Middleware</td>
</tr>
<tr>
<td></td>
<td>Opcodes</td>
<td>Operating Systems</td>
</tr>
<tr>
<td>More Flexible</td>
<td>Custom Hardware</td>
<td>Less Productive</td>
</tr>
</tbody>
</table>

### Custom Hardware
- start()
- myFunc1()
- myFunc2()
- name()

### DTUs e Port
- (from CF)
- Interface

### DTPort
- (from C++ Reverse Engineered)
- _virtual_ ~PO_DTPort()
- (from CF)

### Resource
- Provides Refs: std::map<std::string, DT_var>
- p Resource: ResourceNameIt*

### Resource identifier: string
Levels of Abstraction – more accurate

- Less Flexible
  - Higher Order Languages
  - Assembly
  - Opcodes
  - Custom Hardware

- More Flexible

- More Productive
  - Frameworks
  - Middleware
  - Libraries

- Less Productive
  - Operating Systems
  - Custom Hardware
The Problem

- General Purpose programming languages have not kept pace with the growth of platform complexity\(^1\)
- Families of systems are the order of the day
- Systems and requirements have become more complex

\(^1\)Model Driven Engineering, Douglas C Schmidt IEEE Computer Feb 2006
Families of Systems

http://www.army.mil/fcs/
More Complex Systems and Requirements
Insufficient tools to do the job

C++  UML
Traditional tools

- **UML**
  - Insufficiently domain specific
  - Some say not semantically precise enough to be machine processed
  - Not much different than programming with OO languages

- **3GL**
  - C++, Java, Ada etc.
  - Insufficiently domain specific
  - Not at a sufficient level of abstraction

- **Frameworks, libraries and components**
  - So many
  - Complicated
  - Manipulated via 3GL languages

- **DDS-tools**
  - IDL pre-processing: Insufficiently domain specific
Traditionally – a separation

modeling

Programming
Traditionally – a separation

Systems Engineering

Software Engineering
Critical Innovations

- Technologies that enable systematic vs. ad hoc reuse and enable the systematic development of families of systems
- Component technologies
- Domain Specific Languages\(^1\)

\(^1\)Software Factories, Greenfield et. al Wiley 2005
"Instead of forcing a separation of the high-level engineering/modeling work from the low-level implementation programming. It brings them together as two well-integrated parts of the same job" \(^1\)

1. Eclipse modeling Framework, Budinsky et.al Addison Wesley 2004
Some ramifications

- Modeling and programming blend into one activity
- Models become *development* artifacts as well not just *design* artifacts
- Domain Specific Models can be machine processed and thus integrate well with generators
- Developers can “program in terms of their design intent rather than the underlying, computing environment”

---

3 IEEE Computer Magazine, February 2006 Model Driven Engineering, Douglas C. Schmidt
“Instead of forcing a separation of the high-level engineering/modeling work from the low-level implementation programming. It brings them together as two well-integrated parts of the same job”\(^1\)

\(^1\) Eclipse modeling Framework, Budinsky et.al Addison Wesley 2004
Some ramifications

- Increased cross discipline collaboration
- Speaking the same language
- Flattens learning curves

3 IEEE Computer Magazine, February 2006 Model Driven Engineering, Douglas C. Schmidt
Providing sufficient tools to do the job

DSML Tool

C/C++/Java UML

DSML Tool

C/C++/Java UML

DSML Tool

C/C++/Java UML

Copyright © PrismTech 2006

Slide 18
How is this done?

- Domain Specific modeling Languages
- Domain Specific Editors
- Transformation engines and generators
- Combine the above three – Called Model Driven Development (a.k.a Model Driven Engineering, or Model Integrated Computing)

---

2 IEEE Computer Magazine, February 2006  Model Driven Engineering, Douglas C. Schmidt
Domain Specificity

Domain Independent

Domain Specific - Tools

The task at hand
The steps

<table>
<thead>
<tr>
<th>In general</th>
<th>In our domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isolate the abstractions and how they work together</td>
<td>The DDS specification</td>
</tr>
<tr>
<td>Create a formalized grammar for these - DSL</td>
<td>Create a formalize DDS meta-model</td>
</tr>
<tr>
<td>Create a graphical representation of the grammar – GDSL</td>
<td>Create a DDS specific graphical tool</td>
</tr>
<tr>
<td>Provide domain-specific constraints – GDSCL, DSCL</td>
<td>Program into the tool the constraints</td>
</tr>
<tr>
<td>Attach generators for necessary transformations</td>
<td>C++, C and Java generators</td>
</tr>
</tbody>
</table>
Applying MDD to the DDS Domain
DDS: An Information Centric Approach

Information MEANING

MODEL

Analysis

Server BEHAVIOR

Client -> Server

Data AVAILABILITY

App

OMG - DDS

Slide 24 Copyright © PrismTech 2006
MDD-aspects in ‘Coalition Based Development’

Design Authority

Common Information Model

Company ‘a’  
Company ‘b’  
Company ‘c’

DDS Information Backbone

Information modeling

Application modeling

Deployment modeling
Introducing The Example

- **SENSOR PROCESS**
  - Optical sensor
  - Scans the environment
  - Produces ‘Tracks’
  - Position of ‘objects’
  - Reports ‘pointTrack’

- **CLASSIFICATION PROCESS**
  - Classifies tracks
  - Determines their identity
  - Analyses the trajectories
  - Determines hostility
  - Reports ‘trackState’

- **DISPLAY PROCESS**
  - Displays track info
  - Both position & identity
  - Raises alerts
  - Requires ‘pointTrack’
  - Requires ‘trackState’
MDD and the Example

- Track
  - trackId
- PointTrack
  - pos
- TrackState
  - identity

- Display Process
- Identification Process
- Remote-Sensor Process

- Computing-Node Services
  - App-1
    - Library
  - App-2
    - Library
  - App-3
    - Library

- Shared Data Storage
  - Library
  - Library
  - Library

- Disk
  - Storage

- Network

- MDD Tool suite
  - Information modeling
  - Application modeling
  - Deployment modeling

- MDD and the Example

- Track Model

- Information-Model
  - Track
  - trackId
  - PointTrack
  - pos
  - TrackState
  - identity

- Network-Service
- Durability-Service
- Soap-Service
- Library

- Config (XML)
- Library

- MDD Tool suite
- Network

- PRISM Tech
(1) Information modeling
Information modeling: Scope & Purpose

- **Language**
  - Define the ‘spoken language’ in the system
    - Requirements phase: to talk with the customer
    - Development phase: to talk between components
  - DDS-Domain specific ‘Topics’ (Type + QoS)

- **Editor**
  - Semantics: UML (‘annotated’)
  - Syntax: IDL (optional MDD-input if predefined)
  - Behavior: QoS (reliability, urgency, durability)

- **Generator**
  - Types (IDL) from UML (or direct IDL-import)
  - Topics from Types
  - DDS-entities
    - Topic-creation, QoS-setup, typed readers/writers
Information modeling: The Example

**Task: Information modeling**
- Model the ‘PointTrack’ and ‘TrackState’ topics
- Model the system-wide behaviour of topics
- Model the relationships between the Topics
- Separate these concerns from applications

**MDD: Features / Advantages**
- Graphical modeling of structure and QoS (intuitive, fast)
- DDS-entity code generation (topics, interfaces)
- Allowing for direct utilization (by applications or tool)
- Documented packages of re-usable topic-sets

**Topic**
PointTrack
- long trackId;
- Position pos;
- **Key**: trackId
- **QoS**: best-effort, volatile

**SPLICE-DDS Real-time Information Backbone**

**Sensor**
- Sensor
- **PointTrack Publisher**
- **Display**
- **PointTrack Subscriber**
(2) Application modeling

- Display Process
- Identification Process
- Remote-Sensor Process

Application modeling
Application modeling: Scope & Purpose

- **Language**
  - Processing-language
  - DDS ‘keywords’
    - publishers/writers, subscribers/readers,…
    - Content-filters, queries, waitsets, listeners,…

- **Editor**
  - Templates & Patterns (loop, MVC, …)
  - Application-level QoS modeling (history, filters)
  - Process modeling (waitsets, listeners)

- **Generator**
  - Application-framework from templates
  - DDS entities from (graphical) model
  - Annotated application- and test-code

*Simple DDS-Loop Pattern*

- Register with DDS
- Declare publications
- Declare subscriptions
- Define queries (read_conditions)
- Set query parameters
- Select input data
- Wait for input data
- Process data
- Publish output data
Application modeling: The Example

Task: Display Process modeling
- Model ‘aggregate interest’
  - ‘multi-topic’ or ‘DLRL-object’ with local ‘QoS’
- Model display process (periodic loop)
  - Handle first-appearances of hostile tracks with prio

MDD: Features / Advantages
- Import information-model
- Provide Application-framework (from ‘loop’ template)
- Model DLRL-objects, Multi-Topics, Queries, Waitsets,…
- Developer can concentrate on ‘business-logic’ (GUI)
Content Filtered Multi-Topics

**dataWriter (sensor)**

- **PointTrack**
  - `long trackId;`  
  - `Position pos;`
  - **Key**: trackId  
  - **QoS**: best-effort, volatile

- **TrackState**
  - `long trackId;`  
  - `long identity;`
  - **Key**: trackId  
  - **QoS**: reliable, transient

**dataReader (MyTrack)**

- **read**
  - **query**
    - **Parameterized-Query-Conditions**
    - **Multi-topic**
      - `long Id;`  
      - `Position p;`  
      - `long env;`
      - **projection**
        - **Timed-waitset or Call-backs**

- **take**
  - **Multi-topic**
    - **aggregation**
      - **Content-Filters**
        - **Local data-cache**
          - **PointTrack**
          - **TrackState**

- **Multi-topic**
  - **Selection**
    - `Any`  
    - `Not_Read`  
    - `New`  
    - `Modified`  
    - `Read`  
    - `No_Writers`  
    - `Disposed`
Application modeling: MDD-Tool ‘Sneak-Peak’
(3) Deployment modeling
Deployment modeling: Scope & Purpose

- **Language**
  - Deployment-language
  - DDS ‘keywords’
    - Participants, Partitions, Resource limits,…
    - Latency-budget, Transport-priority,…

- **Editor**
  - Configure ‘physical’ DDS system
    - Networking, Durability, Resource-limits, …
  - Map DDS (QoS) properties to Deployment properties
    - DDS ‘partitions’ to Network-partitions (multicast groups)
    - ‘transport-priority’ to Network-channels (prio/diffserv)
    - ‘latency-budget’ to Network-dispatcher (efficiency)

- **‘Generator’ (Control & Monitoring)**
  - (remote) set-up, configure and monitor DDS-system
  - Deploy information-models (create readers/writers)
  - Deploy applications (monitor, QoS-tuning)
  - Deploy systems (logging/replay, QoS tuning,…)

---

**Computing-Node Services**

- **App-1**
  - Library

- **App-2**
  - Library

- **App-3**
  - Library

**Shared Data Storage**

- **Disk**

---

Slide 37 Copyright © PrismTech 2006
Deployment modeling: The Example

Task: Deployment modeling (‘Control & Monitoring’)
- Configure Durability-service (transient TrackState topic)
- Configure Network-channels (priority, reactivity)
- Configure Resources (resource-limits)
- Remote control & Monitoring of deployed system

MDD: Features / Advantages
- Round-trip engineering
  - Deploy & tune models (info & application)
- Control & Monitor deployed system
  - Using the domain specific ‘DDS-language’
Deployment modeling: MDD-tool ‘sneak-peak’
QUESTIONS ?