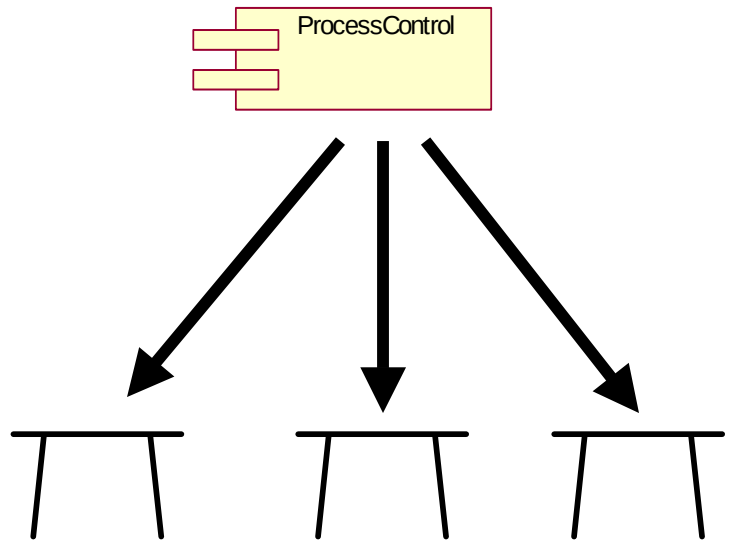
Applying MDA to Constrained Environments

Greg Eakman
Pathfinder Solutions
grege@pathfindermda.com

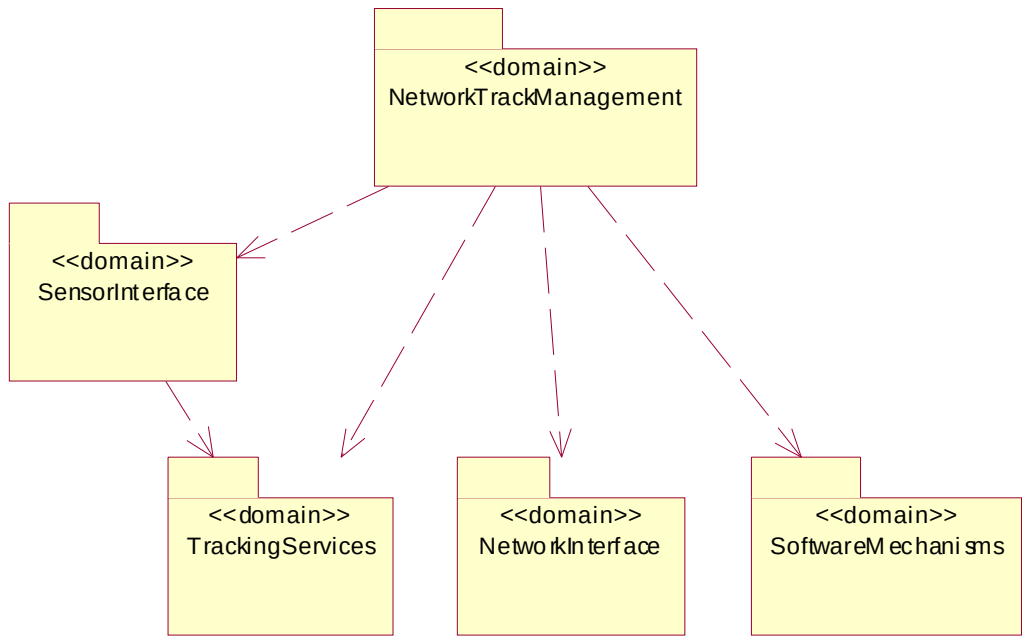
Embedded MDA

- Executable models
- Large number of embedded systems platforms
- Non-functional requirements
- One size does not fit all



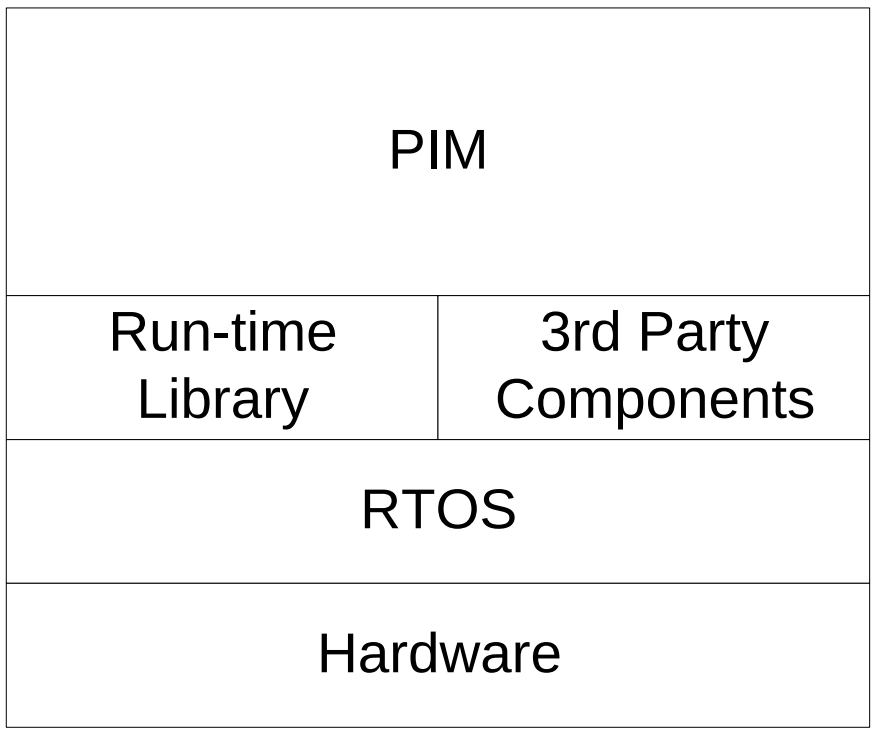
Executable Models

- UML profile
  - Not yet standardized
- Modeled domains consist of:
  - Subsystems
  - Classes
  - Statecharts
  - Operations
  - Action Language



Model Execution

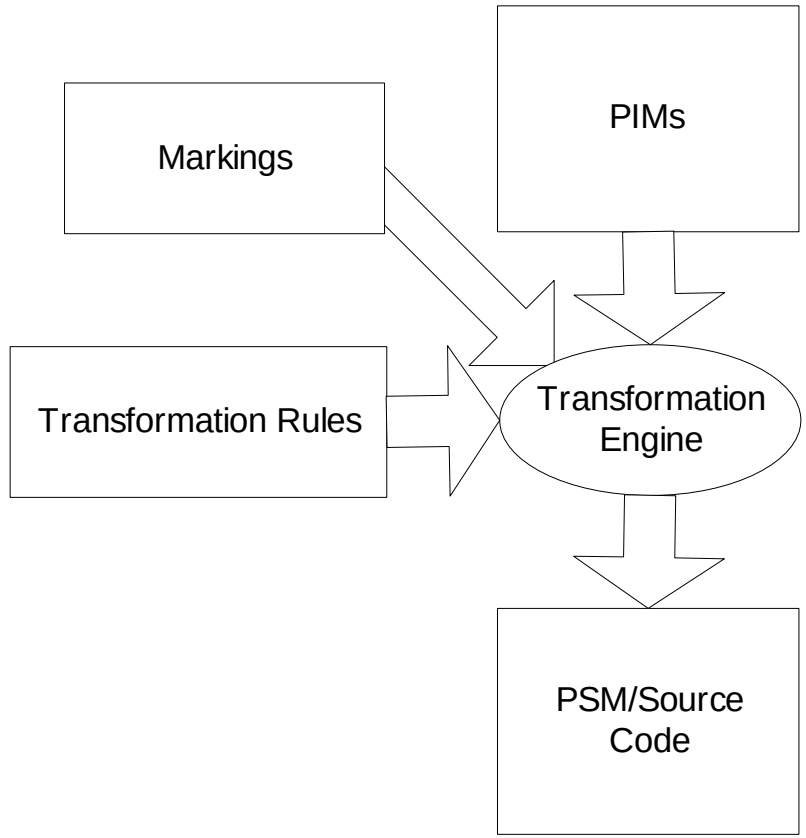
- Run-time library supports model execution



Template Based Transformation

Pathfinder Solutions

- Model components independent of implementation technologies
- Transformations map model to implementation
- Models and transformations can be developed, tested, and profiled independently



Transformation Example

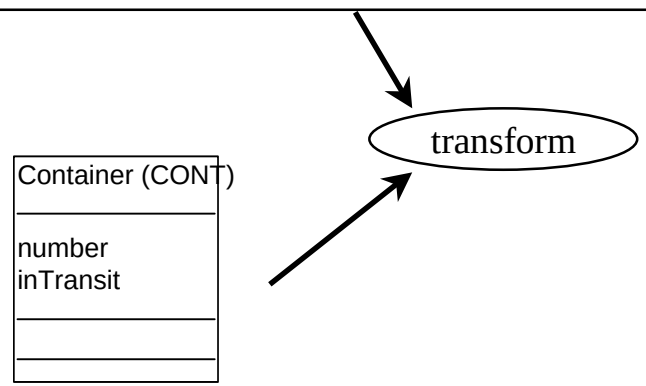
Pathfinder Solutions

- **Attributes -> Class Member Variables**
  - each Attribute maps to a public data member of it's C++ class

template

```
  . . .
public:
    // Attributes:
[FOREACH attribute in object.attributes]
    /* [attribute.description] */
    [attributedataType]  [attribute.name];

[ENDFOREACH /*attribute */ ]
  . . .
```



generated C++ code

```
public:
    // Attributes:
    /* External identification number. */
    Integer  number;
    /* Flag indicating if the container is moving towards
       its location, or if it has reached it. */
    Boolean  inTransit;
```

Implementation Patterns

Pathfinder Solutions

- Model “compiler”
- Expressed in templates
- Look for patterns in models
- Guided by markings

Constraint-Sensitive Areas

Pathfinder Solutions

- Models
- Transformation rules
- Run-time Library
- Platform

Platform and RT

Pathfinder Solutions

- Platform provides basis for RT support
- Platform model
  - Can be developed, tested, and evaluated for RT suitability separate from model development
- Models will drive some platform requirements
- Run-time mechanisms and basic data structures are foundation for model execution

Models and RT

Pathfinder Solutions

- Define the logic of the application
- All models are not necessarily good models (GIGO)
- PIM modeling very different from “traditional” OO modeling
  - Implementation focused models
  - Domain pollution
  - Inefficient models - the “Bit” class
- Resource contention

Transformation Rules and RT

Pathfinder Solutions

- Map models to code
  - Inefficient or unscalable mechanisms selected
  - Design and implementation patterns
- New mappings require thorough understanding
  - execution semantics
  - platform
  - transformations
  - metamodel
  - Real-time/Performance/QoS

Quality of Service

- Must be captured in the models
- Supporting standards
  - UML Quality of Service Profile
  - Schedulability, Performance and Time

Sample Patterns

Pathfinder Solutions

- Bitfields
- Associations
- Static initialization
- Memory Pools
- Lookup tables
- Distributed deployment

Bitfield Pattern

Pathfinder Solutions

- Template

```
[FOREACH attribute in class.attributes]
[  ASSIGN compressable = FALSE]
[  EXPAND attr_compress_lu(attribute; compressable)]
[  IF (compressable != TRUE)]
    /* [attribute.description] */
    [EXPAND gen_type (attribute.dataType, TRUE)]  [attribute.name];
[  ENDIF]
[ENDFOREACH /*attribute */ ]
```

- Code

```
PfdString name;
char bitfield;
bool_t getFlag() { return (bitfield & (0x01) >> 0); };
void setFlag(bool_t in_flag) { (bitfield & ~(0x01) | (char) (in_flag << 0)); };
```

Association Optimizations

Pathfinder Solutions

- List vs. Array
  - Upper bound from marking
  - Define exception handling
- Optimize based on use
  - Patterns examine action language to reduce code and RAM space
  - Remove unused associations in each direction

Static Initialization

Pathfinder Solutions

- Preconfigured, fixed set of objects
- Instance data stored as XML and fed into transformations

```
[/* Generate static initialization for C */]
[className] instances = {
[FOREACH instance IN instanceSet]
{\
[  FOREACH attribute IN class.attributes SEPARATOR “,”]
[   EXPAND instance_value(instance, attribute.name)]\
},
[  ENDFOREACH]
[ENDFOREACH]
};
. . .
```

```
/* attributes: id, volume, isEmpty */
Tank instances = {
{ 1, 100, TRUE },
{ 2, 100, TRUE },
};
```

Memory Pools

Pathfinder Solutions

- Dynamic object population
- Pre-allocated and recycled
- CREATE statement
  - Default new or malloc()
  - Memory pool overloads with memory pool run-time mechanism support

Distributed Deployment

Pathfinder Solutions

- Deploy components across threads, processes, and processors
- Communications channels
  - Shared memory
  - TCP/IP
  - Custom
- Lightweight message protocol

Lookup Tables

Pathfinder Solutions

- Action language lookup constructs
  - FIND this -> A1 WHERE (engaged == TRUE)
  - FOREACH track IN AirTrack WHERE (correlation > 0)
- Iteration implemnted by default
- Tradeoff space for time
- Create additional tracking tables to optimize lookups

Lookup Tables

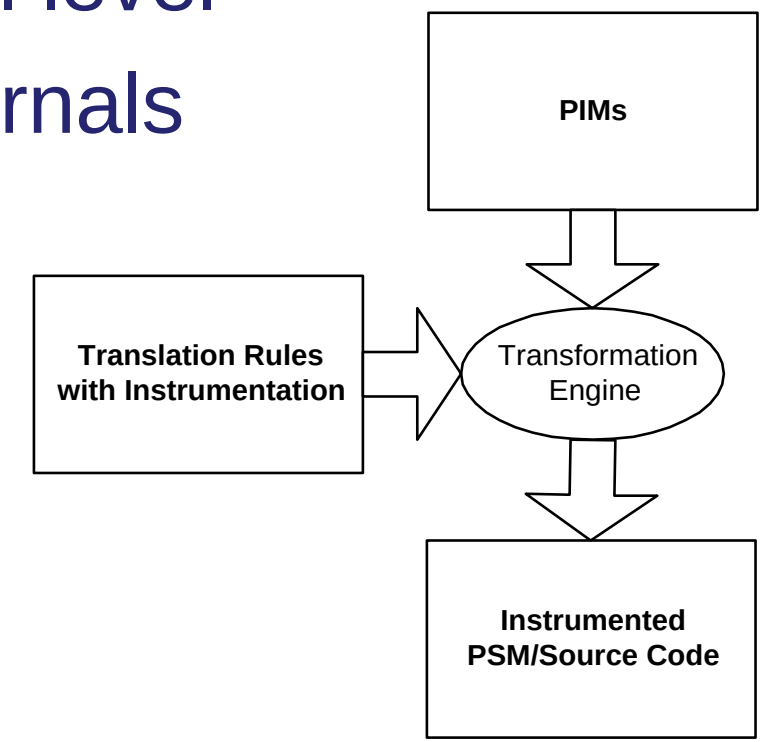
Pathfinder Solutions

```
[/* See if this attribute keys consideration for an equivalence table */]
[ASSIGN eq_tab_exp = PROPERTY (attribute, "EqTableExpression", "")]
[IF (eq_tab_exp != "")]
          if ([eq_tab_exp])
          {         if (!already_in_eq_tab)
                    {         // Add this instance to the eq table (addNoDupes to avoid
                              // duplication)
                              equivalenceTable_[attribute.name].addNoDupes(this);
                    }
          }
          else
          {         if (already_in_eq_tab)
                    {         // Remove this instance to the eq table
                              equivalenceTable_[attribute.name].removeThis(this);
                    }
          }

[ENDIF /* eq_tab_exp */]
. . .
```

Instrumentation

- Design-for-Test implementation patterns
- Source code debugger symbol table
- Test and debug at model level
- Visibility into system internals
  - Testability
  - Controllability
  - Observability
- Probe effect



Execution Profile

Pathfinder Solutions

- Run-time view of models
- Light weight, tailored instrumentation
- Stress tests
- Collect data on:
  - Class extents
  - Association extents
  - Maximum events pending
  - Time between points in execution
- Feed back results through markings

Conclusion

Pathfinder Solutions

- All aspects of the MDA process impact constraints
- Separation of models from platform
- Developers need to control the generated code
- Implementation patterns for QoS
  - Captured in transformation rules
  - Enforced globally
- Action language allows implementation patterns greater optimization capability

Thank You!

Pathfinder Solutions