Automating the generation of platform specific models

W. El Hajj Chehade, A. Radermacher, A. Cuccuru, S. Gérard and F.Terrier.

CEA LIST, Laboratory of model driven engineering for embedded systems

{wassim.el-hajj-chehade, ansgar.radermacher, arnaud.cuccuru, sebastien.gerard, francois.terrier}@cea.fr

Fourth IEEE International workshop UML and AADL

Potsdam, June 2, 2009

Outline

Context

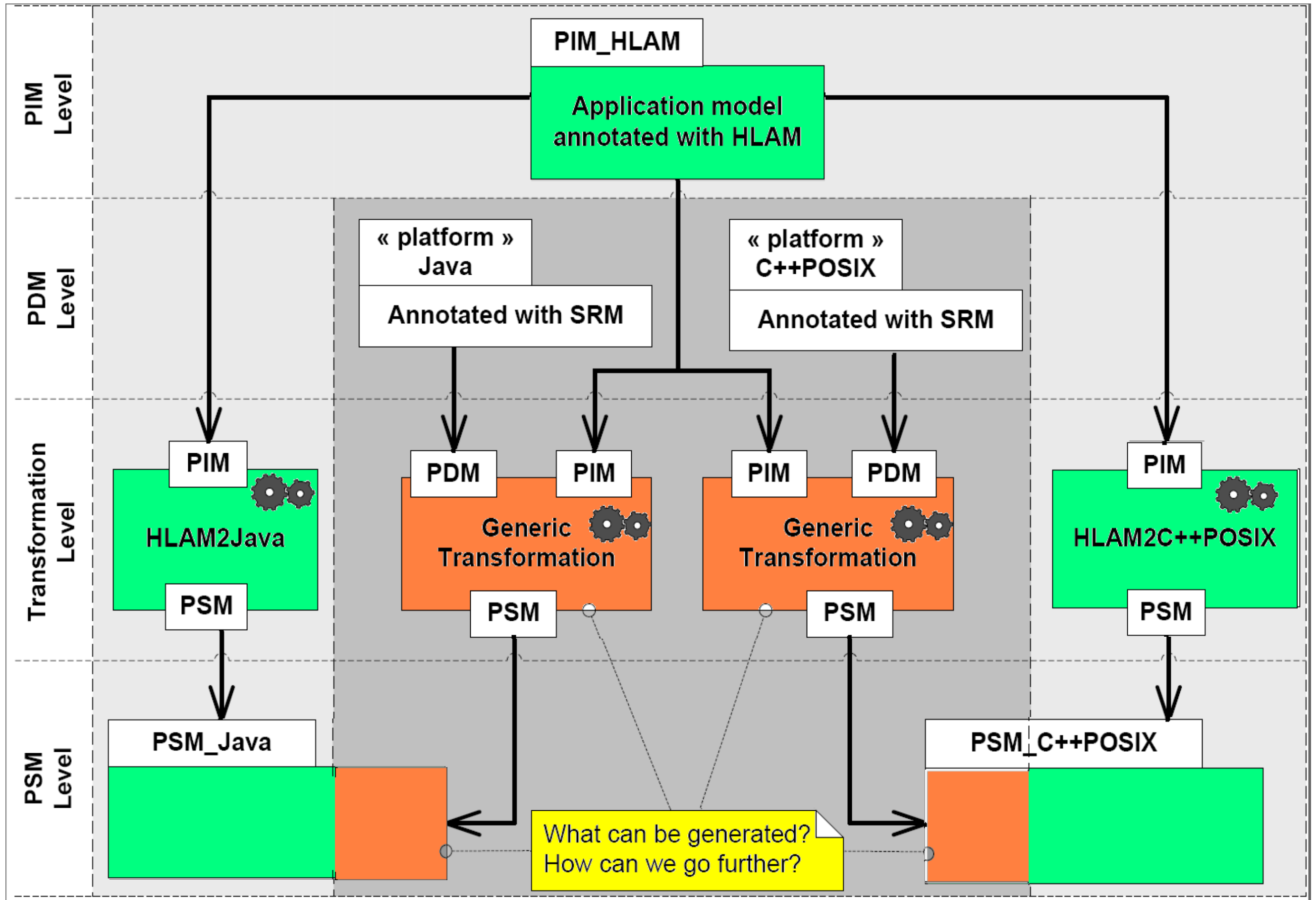
- **Goals**
  - ✓ **Studying the impact of programming languages and their libraries on Platform Specific Models**
- **Why:**
  - ✓ **Develop generic transformation for different implementation platforms**
  - ✓ **Automate the generation of Platform Specific Models**
- **Our approach:**
  - ✓ **Generating Platform Specific Models using:**
    - Dedicated model transformation
    - Generic model transformation with Software Resource Modeling (SRM) sub-profile of MARTE

AR2

Diapositive 3

AR2 Well, not fully automatic, also (I think):
  tool indicates violation of platform restrictions =>user interactively selects design patterns that repair the violation
Ansgar Radermacher; 27/05/2009

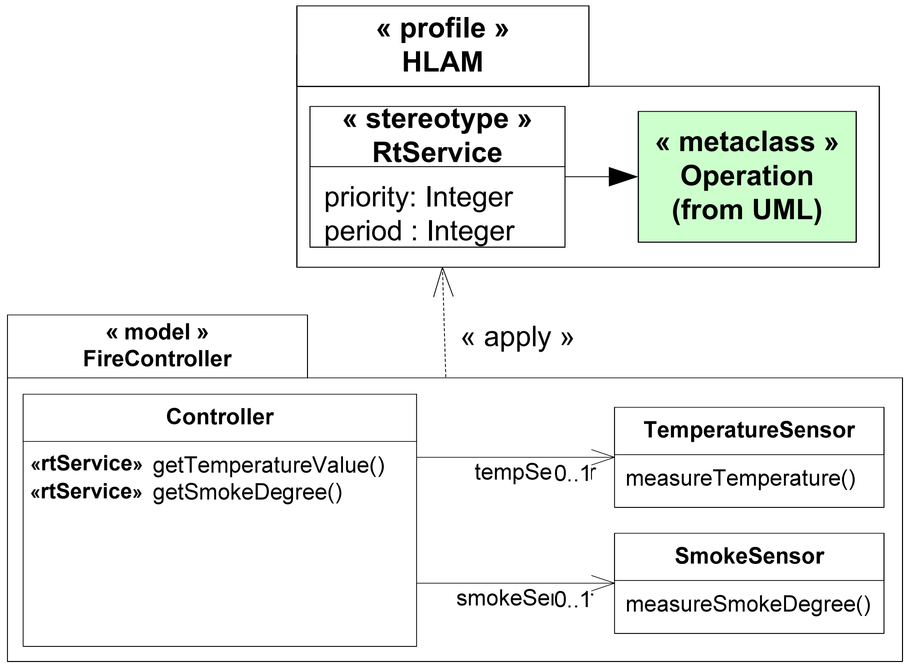
Global vision on the study



⬜	Usual MDE process (i.e., dedicated model transformations)
▨	SRM-Based MDE process (i.e., explict models of platform + generic model transformations)
■	Subset of the PSM that can be generated with the SRM-Based process

Example

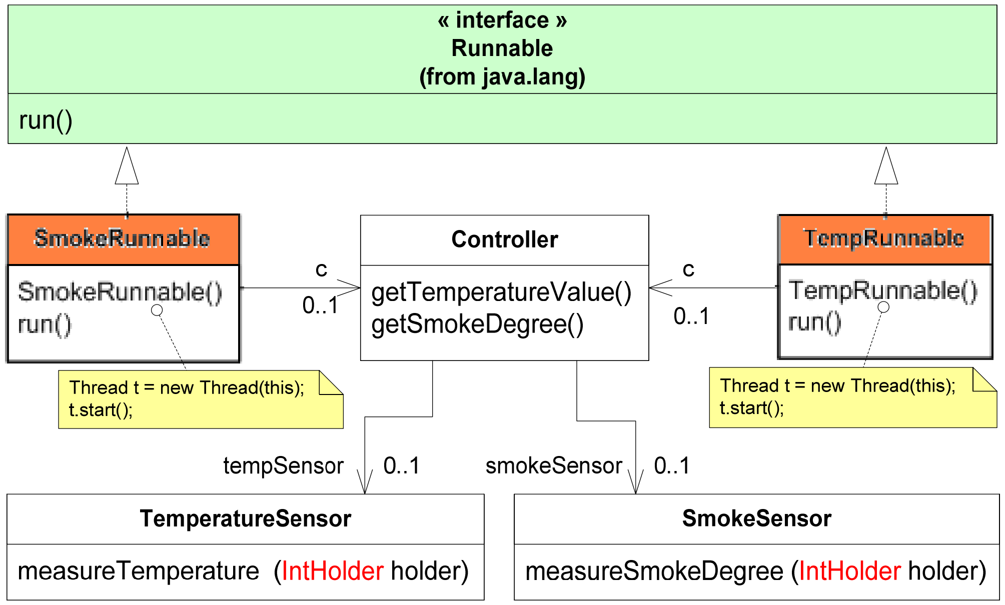
- Application model
  - ✓ fire controller system
- Controller class
  - ✓ 2 operations are executed by its own thread
- TemperatureSensor & SmokeSensor
  - ✓ Measure temperature and smoke degree
- Use HLAM of MARTE to capture multi task designs
  - ✓ “RtService” stereotype



Fire controller class diagram

Design for the Java platform

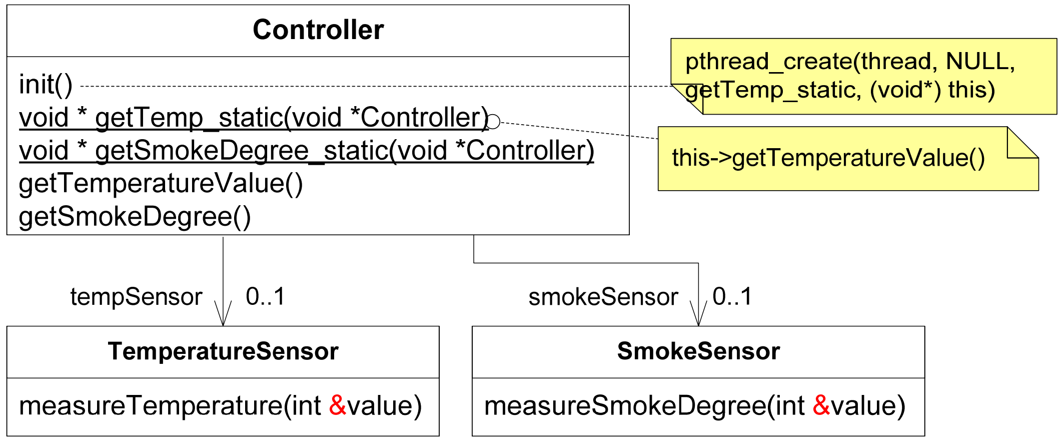
- **Impact of Java library**
  - ✓ **Thread implements the Runnable interface**
    - Each class owns the run() method
  - ✓ **Design is used to create and start the Thread**
    - Constructor of the class
- **Impact of Java language properties**
  - ✓ **Out parameters can't be typed by primitive types**
  - ✓ **Replace "int" by "IntHolder"**



Design for C++/POSIX platform

- ➢ **Impact of POSIX library**
  - ✓ **pthread_create() function create and start the thread**
    - Takes four parameters
  - ✓ **start_routine is a function or static method**
    - Class method in the C++ context
  - ✓ **Design is used to create and start the thread**
    - Init() method
    - Static method
- ➢ **Impact of C++ language properties**
  - ✓ **C++ supports passing primitive type by reference**

int **pthread_create** (pthread_t *thread, pthread_attr_t *attr, void *(***start_routine**) (void *), void ***arg**)

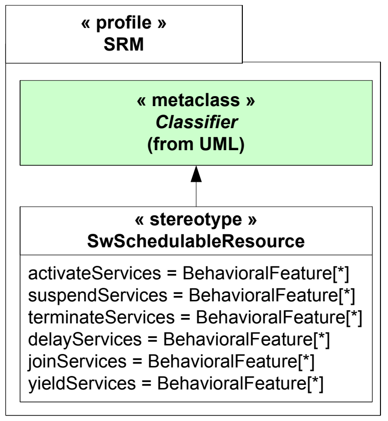

Controller
init()
void * getTemp_static(void *Controller)
void * getSmokeDegree_static(void *Controller)
getTemperatureValue()
getSmokeDegree()
pthread_create(thread, NULL, getTemp_static, (void*) this)
this->getTemperatureValue()
tempSensor
0..1
smokeSensor
0..1
TemperatureSensor
measureTemperature(int &value)
SmokeSensor
measureSmokeDegree(int &value)

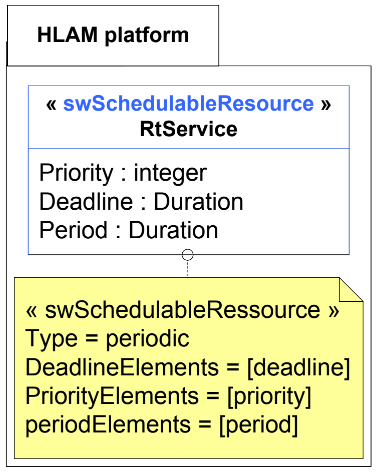
Overview on SRM

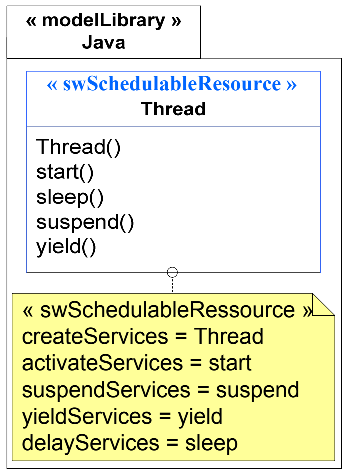
- What is the Software Resource Modeling framework ?
  - ✓ **SRM is:**
    - A UML profile to describe software execution APIs
      - → Real-time operating system (RTOS)
      - → Languages libraries (e.g., Java)
    - based on the Resource-Service pattern
      - → Resource: mechanisms to be used offered by the platform
      - → Service: set of operations to manipulate the resource
  - ✓ **What is supported by SRM profile?**
    - Concurrent computation contexts (e.g., interrupt, task)
    - Interactions between concurrent context
      - → Communication (e.g., mailbox)
      - → Synchronization (e.g., semaphore)
    - Hardware and software resources brokering (e.g., driver, scheduler)

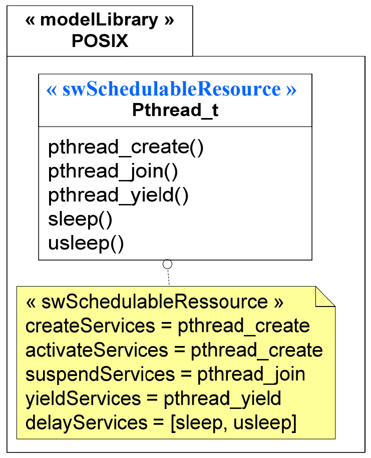
How should we use a SwSchedulableResource?

- Define a UML model for the Java and POSIX thread
- Apply “SwSchedulableRessource” Stereotype”
- Fulfill the properties of the applied stereotype to precise semantics
- Use SRM to annotate the domain model of the profile


« profile »
SRM
« metaclass »
Classifier
(from UML)
« stereotype »
SwSchedulableResource
activateServices = BehavioralFeature[*]
suspendServices = BehavioralFeature[*]
terminateServices = BehavioralFeature[*]
delayServices = BehavioralFeature[*]
joinServices = BehavioralFeature[*]
yieldServices = BehavioralFeature[*]

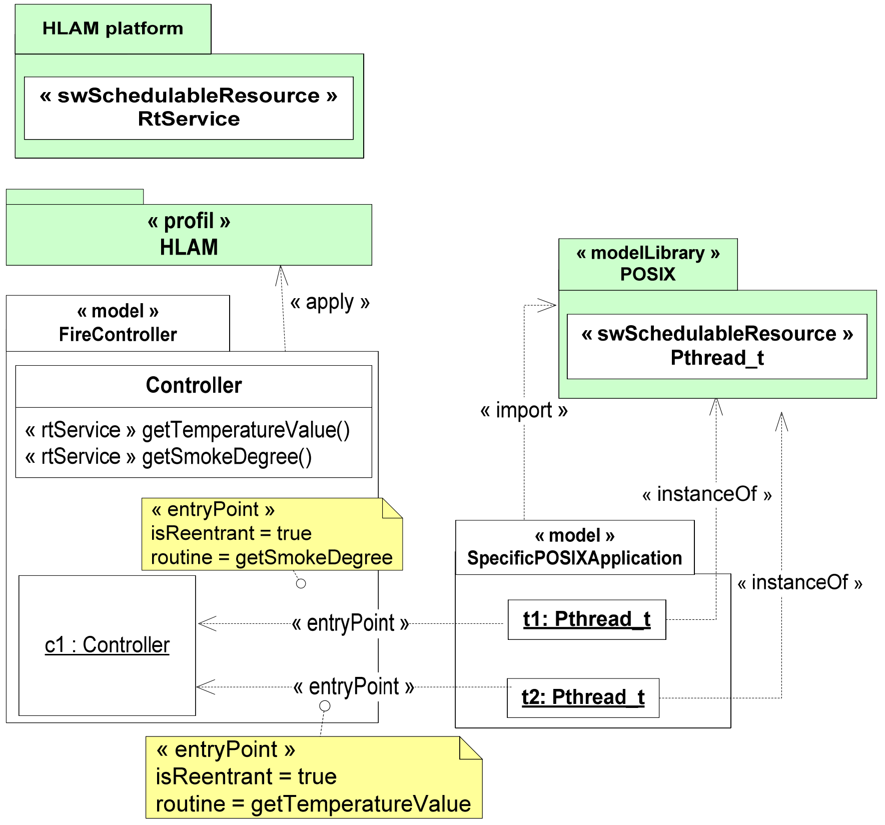

HLAM platform
« swSchedulableResource »
RtService
Priority : integer
Deadline : Duration
Period : Duration
« swSchedulableRessource »
Type = periodic
DeadlineElements = [deadline]
PriorityElements = [priority]
periodElements = [period]




« modelLibrary »
POSIX
« swSchedulableResource »
Pthread_t
pthread_create()
pthread_join()
pthread_yield()
sleep()
usleep()
« swSchedulableRessource »
createServices = pthread_create
activateServices = pthread_create
suspendServices = pthread_join
yieldServices = pthread_yield
delayServices = [sleep, usleep]

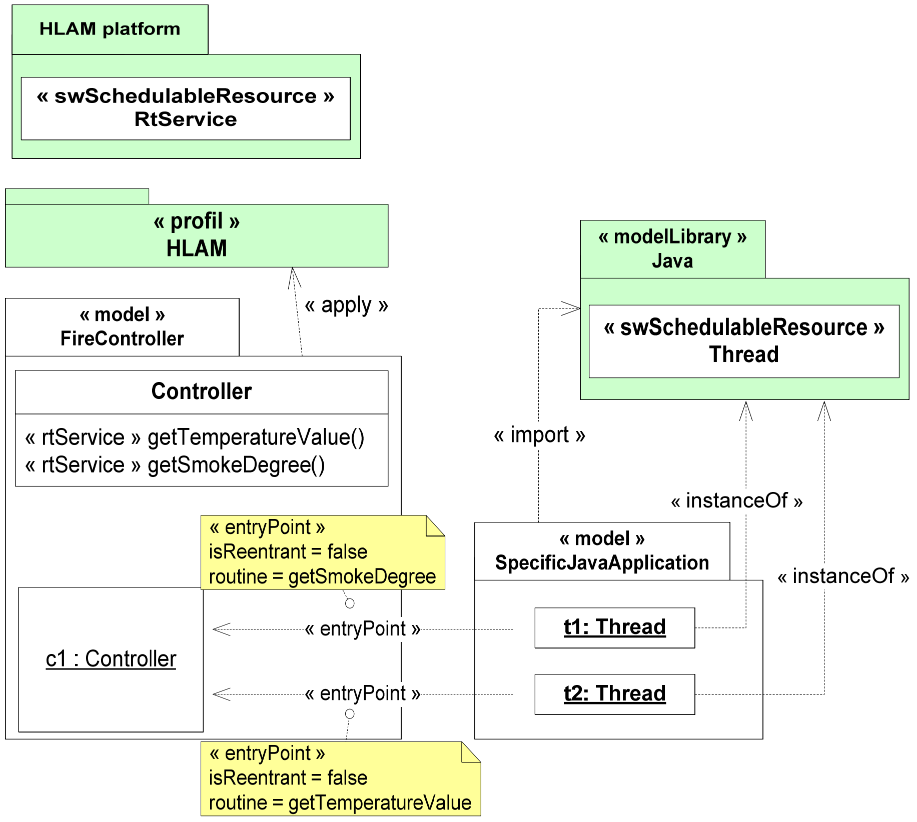
Design for C++ & POSIX platform with SRM

- Application model annotated with HLAM
- POSIX Platform model annotated with SRM
- Instantiation of two POSIX thread
- Usage of “entryPoint” stereotype to specify the code to be executed


HLAM platform
« swSchedulableResource » RtService
« profil » HLAM
« apply »
« model » FireController
Controller
« rtService » getTemperatureValue()
« rtService » getSmokeDegree()
« entryPoint » isReentrant = true routine = getSmokeDegree
c1 : Controller
« entryPoint »
« entryPoint »
« entryPoint » isReentrant = true routine = getTemperatureValue
« modelLibrary » POSIX
« swSchedulableResource » Pthread_t
« import »
« instanceOf »
« instanceOf »
« model » SpecificPOSIXApplication
t1: Pthread_t
t2: Pthread_t

Design for the Java platform with SRM

- Software designer creates the application model
- Domain model and Java model library are annotated with SRM
- Two method annotated with “RtService” = two instance of the Thread class
- “entryPoint” stereotype is used to specify the code to be executed


HLAM platform
« swSchedulableResource » RtService
« profil » HLAM
« apply »
« model » FireController
Controller
« rtService » getTemperatureValue()
« rtService » getSmokeDegree()
« modelLibrary » Java
« swSchedulableResource » Thread
« import »
« instanceOf »
« instanceOf »
« entryPoint » isReentrant = false routine = getSmokeDegree
« model » SpecificJavaApplication
c1 : Controller
« entryPoint »
« entryPoint »
t1: Thread
t2: Thread
« entryPoint » isReentrant = false routine = getTemperatureValue

Analysis

- **Actual use of SRM approach**
  - ✓ **Structural transition from application model to a platform specific model**

- **Limits of SRM**
  - ✓ **Differences between language capabilities are out of scope (e.g., passing of out parameter)**
  - ✓ **Behavior of real-time resources and their services**
    - “CreateService” for the “SwSchedulableRessource” has a behavior in Java and in C++/POSIX platform
    - Call operation action, parameter passing action
  - ✓ **Impact of library usage on the PSM structure**
    - In Java: each instance of the thread class must encapsulate run()
    - In C++/POSIX: start_routine function impose to encapsulate a static method in the class encapsulating a thread

Conclusion and future work

- Impact of programming languages on platform specific models
- Using SRM for the generation of platform specific models
- Limits of SRM
- Future work
  - ✓ Define the behavior of the resources and their services for Java and C++/POSIX platform
  - ✓ Study the structural impact that have the adoption of these behaviors

Thank you

Questions