Model-Driven Software Development in Robotics:
Composability of Software Components, Robot Behaviors and Reuse of Action Plots

May 9th, 2011 / SDIR VI, Shanghai, China

Dennis Stampfer,
Andreas Steck
and Christian Schlegel

ZAFH Servicerobotik
Computer Science Department
University of Applied Sciences Ulm, Germany

http://smart-robotics.sourceforge.net/
http://www.zafh-servicerobotik.de/ULM/index.php
MDSD in Robotics

Outline

- Part 1: SmartSoft MDSD Toolchain
- Part 2: Deployment and reuse of components
- Part 3: Dynamic State Charts
- Part 4: Vision: The Robot App-Store
- Video
SmartSoft MDSD Toolchain implements the complete process from idea to runnable code.
Toolchain workflow: an example
- Idea: Simple wander mode using laser range data

Laser range data

LaserObstacleAvoid

Steering commands
MDSD in Robotics

MDSD Toolchain: Step 2: PIM

- Create Platform Independent Model of the component

**MDSD Toolchain**

1. Idea
2. PIM
3. PSM / PSI
4. Deployment
5. Runtime
MDSD in Robotics

MDSD Toolchain: Step 3: PSM/PSI

- Implement avoidance algorithm in generated hull
MDSD in Robotics

MDSD Toolchain: Step 4: Deployment

- Deployment

1. Idea
2. PIM
3. PSM / PSI
4. Deployment
5. Runtime

Laser range data

LaserObstacleAvoid

Steering commands
MDSD in Robotics

MDSD Toolchain: Step 4: Deployment

- Deployment

- Laser range data

- Steering commands
MDSD in Robotics

MDSD Toolchain: Step 5: Execute

MDSD Toolchain: 1. idea 2. PIM 3. PSM / PSI 4. deployment 5. runtime

Diagram showing a robot with components labeled as parts of the MDSD Toolchain.
MDSD in Robotics

SmartSoft Component

- Communication Patterns
  - downward interface / internal
    - invisible to user, is handled by MDSD toolchain
    - can be mapped onto different middleware systems, e.g. Ace, Corba
  - upward interface / user
    - no adjustments at user visible API

- Communication Objects for marshalling
- Tasks etc.

Details to be found in:
MDSD in Robotics

SmartSoft Component: Further Information

- **SIMPAR 2010 Tutorial**
  - Model-Driven Software Development in Robotics, 2nd International Conference on SIMULATION, MODELING, and PROGRAMMING for AUTONOMOUS ROBOTS (SIMPAR 2010)

- **Technical Reports**

- **Further publications and slides**
MDSD in Robotics
Extending the example

- Previous example very simple
- Expanding to navigation task by using readily available components for:
  - Motion control
  - Mapping
  - Path planning
  - ...

What is the benefit?
- Make use of models
- Reuse of work & quality
- Separate experts
- Make the step from programming to composing
MDSD in Robotics
Extending the example

The navigation task

(Further extensions: Localization, SLAM, Speech, Simulation, etc)
MDSD in Robotics
Extending the example

- How can individual components form a real scenario?
- Where is the business logic that defines the scenario plot?

(Further extensions: Localization, SLAM, Speech, Simulation, etc)
MDSD in Robotics
Extending the example

- How can individual components form a real scenario?
- Where is the business logic that defines the scenario plot?

The challenge is system integration

**Challenge 1: Integration of parts to a whole:**
Solved in the previous slides using SmartSoft and MDSD

**Challenge 2: Coordination of the parts next**

(Further extensions: Localization, SLAM, Speech, Simulation, etc)
MDSD in Robotics

Coordination of parts

- A scenario consists of several steps including substeps: the task plot
- Every step reflects a certain configuration of the components in the system
  - Situated selection of configurations to manage the complexity of the task
- The coordination of activities cannot be done in if/else constructs

Scenario:
Task plot

Different variants for a step: e.g. approaching depends on the object type (shelf, person, object on table, etc)
MDSD in Robotics

Task coordination

Two approaches for task coordination

- **Task Coordination Language: SmartTCL**
  - Flexible task tree structure
  - Implemented in LISP
  - Interfaces with symbolic planners
  - Work in progress but in productive use

- **Dynamic State Charts**
  - Uses the structure of UML State Machines
    - Structured, accepted, well known
    - Intuitive use
    - Existing tools (graphical modeling, code gen)
  - Prototype implementation
MDSD in Robotics

Configuring components

Step 1:
Wait for told positions

Step 2:
Drive to the told position
MDSD in Robotics

Configuring components

Step 1:
Wait for told positions

- wait for command
  - entry / configure speech for commands
  - exit / deactivate speech
- drive_to_place
  - entry / configure navigation components, set goal
  - exit / stop navigation
- arrived

Communication via SmartSoft patterns state, event, param, wiring

Step 2:
Drive to the told position

- active
  - Motion execution
  - state: neutral
  - Path planning
  - state: neutral
  - Map building
  - state: neutral

- inactive
  - Speech Input
  - state: active
  - parameter: grammar=commands
  - events: notify on new command

- active
  - Motion execution
  - state: active
  - parameter: goal=(1000,500)
  - Path planning
  - state: active
  - Map building
  - state: active
Dynamic State Charts

- Based on UML “State Machines”
- Adequate for behavior coordination in robotics
  - Offer hierarchy and parallelism
  - Base on events and states
  - Entry/Exit actions to reconfigure components according to progress of task execution as reported by events
- Highly reusable
  - Reuse as a goal, not just a by-product
  - Onetime representation of repeating state charts: reuse task descriptions
- Dynamic States
  - Modeling all execution variants is too complex
  - A hull of a state with the internals selected at runtime
MDSD in Robotics

Dynamic State Charts

- Hierarchy
  - Encapsulated state machines: scoped and labelled events
  - Interfacing between hierarchies: parameter passing, attributed event firing

- Parallelism
  - Separation of parallel control flow vs. parallel execution in skills

- Interfacing with components
  - Configure components in Entry/Exit actions
  - Decide on plot: Activate events the coordinator is interested in but being notified when it happens

- Reuse
  - Abstract and primitive state machines
  - Instantiation of state machines
  - Enable the composition of complex tasks out of simpler tasks in a building blocks manner
Dynamic States are a hull of a state that will select the internals at runtime.

The dynamic state refers to possible candidates.

Candidates include conditions whether they match.

Reuse the task knowledge independent from their refinements.

Robustness by context and situation dependent instantiation.

### Dynamic State Charts

![Dynamic State Chart Diagram]
Dynamic State Charts are a tool for robot behavior coordination. They enable the composition of complex tasks out of simpler tasks in a building blocks manner.

- Developed state chart
  - Overall task plot

- Reused state chart for navigation
  - Wraps the configuration of the components for navigation
MDSD in Robotics
Vision: Robot App Store

- Not only reuse components but also behavior
  - Distribute bundles of components+ behavior
  - e.g. for navigation
- Reuse task plots
  - Provide task plots that are independent from refinement
  - e.g. delivery task uses navigation
- Benefit:
  - Task knowledge can be separated from algorithmic knowledge
    - Separation of experts
  - Build new applications in a building blocks manner at behavior level.

Stop programming. Start composing.
Conclusion

- Impressive algorithms and abilities for robotics are available
  - The challenge is to bring them all together, to coordinate them: **compose applications**!
- Complex tasks shall be composable out of simpler ones by a task expert
  - Separation of experts is important for robotics: makes it accessible to domain experts
- Dynamic State Charts are a generic method for composition and coordination of complex robot behavior
- The use of models provides the necessary abstraction
- The SmartSoft framework, MDSD Toolchain and related methods guide through the presented process
What is available?

- Components
- Documentation
  - Examples
  - Technical Reports
  - Tutorials
  - Screencasts
  - ...
- SmartSoft MDSD Toolchain
- VmWare Image to try
- Two implementations
  - CORBA/SmartSoft
  - ACE/SmartSoft
- LGPL and GPL
• On Youtube: [http://www.youtube.com/roboticsAtHsUlm](http://www.youtube.com/roboticsAtHsUlm)
• The robot Kate cleans up the table, stacks objects (cups, red bull, potato sticks) and puts them into trashbin or kitchen sink.