Case studies for model driven engineering in mobile robotics

Bruce MacDonald
Partha S Roop
Tanveer Abbas
Chandimal Jayawardena
Chandan Datta
Jamie Diprose*
John Hosking*
Zeeshan E Bhatti

Robotics Laboratory
Department of Electrical and Computer Engineering
*Department of Computer Science
University of Auckland

9 May 2011
Outline

• Model driven engineering
• Case studies:
  1. Customization tools for different human roles
  2. Defining interactions
  3. Programming by demonstration
  4. Visual programming
  5. Safety critical robotics
Model driven engineering

Designer provides high level specification

- Sound formal model
  - Ideal case: amenable to formal checking
- Visual design tools
- Cognitively appropriate for human designer roles
  - Developers (software development company)
  - Domain experts (clinicians)
  - Deployers (managers, IT staff)
  - Operators (caregivers)
  - Users (older people, staff, relatives)
- Automatic synthesis derives the implementation
Model driven engineering: Key engineering benefits

- Improved code quality
- Shorter development time
- Reusable modules
- Visual simulation
- Run-time visualization
Tool chain should include:

• Specifying robot behaviour, including:
  – The broad context of the robot's environment
    • eg physical and clinical context
  – Required interactions and contexts
    • eg interactions with a physician
  – Error and unusual conditions
    • eg prevent medication errors

• Deployment and configuration in the mission environment
  – eg retirement village business flow for medication management

• Reconfiguration during operation
  – eg modified medication administering

• Debugging robot behaviour, including:
  – Visualising sensor data, algorithm behaviour, interactions
  – Controlling visualisation and debugging modes
  – Augmented and mixed reality experiments
  – Workflow for debugging/editing at the model level
Our focus: *improve interaction*

Robotic Software Developer

Operator

User

Domain expert

Deployer

Robotic Software Developer
Case study #1
Customization tools for human roles
Chandan Datta

Objective: to develop an application development environment for end users to customize and define behaviour on a mobile robotic cognitive aid for older people
Tools for
Domain experts vs End users

- Interaction design
- Dialog creation
- Behaviour creation
- Eg medication reminding work flow
- Data customization
- Content personalization
- Interaction preferences
- Eg medication times
Eg Medication management

Post-It notes of a design session with domain experts (not programmers)

Key point: many issues!

Aim: enable domain experts to code requirements directly

Doctor/Pharmacist: enter meds
Manager:
Caregiver: customize meds
Older person/family: preferences
Interaction creation

Need an MDE Tool combining UI and robot behaviour
Data customization

- Caregivers, managers
- Robogen: forms based web tool
Preferences

• GUI on robot as part of the application/s
• Older person, family
Three different MDE tools

- Interaction design
- Operational customization of details
- User preferences
Case study #2
Defining interactions
Chandimal Jayawardena et al

• More about the interaction design tool/s
• Here about the representation and behaviour generation
HealthBots v.1 architecture

- Menu driven
- Simple user interactions
- No external data access

C++

TTS

RSF
Requirements of HealthBots v.2

• Semi-autonomous
• Reactive
• Interactive
• Context aware
• Scheduling
• More natural dialogs
• Biometric authentication
• Integration of distributed service modules
• Integration of web services
• Integration of third-party applications
• Customizable

“The robot should be able to navigate within a pre-defined work space without any user intervention. It should be able to identify people and visit places, and perform pre-defined or user-requested tasks.”
Overview of the architecture

- **Interaction Manager**
  - Task management
  - Dialog management

- **GUI**
- **TTS**
- **Speech Rec.**

- **Scheduler**
  - eg Medication reminding

- **Other event generators**
  - eg Falls detection

- **Connector to RSF**

- **Databases**
  - eg Medication, user details

- **Service modules**
  - eg VS measurement, navigation

- **External WS**
  - eg Medication info

- **3rd party apps**
  - eg Brain fitness

- **Robot Software Framework (RSF)**
Interaction manager

User Configurable (xml)

State Definition
- Current Session Data Required
- External Data Required
- Tasks Required

Interface Layout Definition
- Interface components (buttons, text inputs, images, videos etc.)
- Layout

User defined output templates

Dialog Manager

Current Session Data

Current Session Data Access

External Data

External Data Access

External Databases

External Services

Tasks

Layout Designer

Selection Algorithm

Selected template

Create voice response

Dialog Manager

“I am going to take your SPO2 reading, Dr Jekyll”
Customizability

- In HealthBots v.2, the following things are customizable
  - State transition
  - External data access
  - External service access
  - Dialog templates
  - Interface components
  - Interface layout
Interaction design as MDE

Future goals

• Redesign XML representation and interaction manager as an MDE system
• Create and integrate the visual front end MDE tool
• Eg: enable a psychologist to define the robot's behaviour for a formal user study
• Key requirement: include domain requirements, services, robot control and UI in MDE tools
Case study #3
Programming by demonstration
Tanveer Abbas
Programming by demonstration as MDE

Programming by demonstration process

Initial model is in the user's head

Transitory model is the physical environment

Stored model is the PbD representation (e.g., ETs and ECRs)

Generate behaviour from the stored model
PbD as MDE

- Requirement: match to cognitive models of human demonstrators
- Requirement: meet needs of task execution
- For MDE: formalize and manipulate PbD models using tools
Case study #4
Visual programming
Jamie Diprose

• Visual environment as a programming model
Ruru

- A visual language (VL) addressing the difficulties of engineering software for robots
  - Targeted at novice programmers
- Moody’s Physics of Notations used to:
  - Evaluate other related VL’s
  - Choose appropriate visual constructs
Ruru

• Semantic constructs derived from:
  – Collett’s robot program model
  – Novice robot programs, literature & robot API’s

• Visualisations
  – Real time
  – Set parameters intuitively

• Implemented with Microsoft's DSL Tools
Visual Programming as MDE

• Again: match human cognitive models to visual interaction models for defining behaviour

• Create MDE tools including debugging in the visual language domain (ie the model)
Case study #5
Safety critical robotics
Partha Roop and Zeeshan Bhatti

• New development in our robotics work
• Established tools in industrial control systems design
• IEC61499 function blocks model base
• TimeMe IDE tool:
  – MDE from function blocks
  – Includes formal verification
  – XML function blocks → C, C++, Esterel, System J
Key Advantages of TimeMe

Most efficient IEC 61499 code generator in the world

Generates deterministic and deadlock-free code

Only IEC 61499 implementation that doesn’t need a run-time environment

Provides Step-by-Step Simulation

Supports observer-based formal verification

Enables static timing analysis
Model Editors
Code Generation

```c
/* Function block initialization function */

void DistStnWithNetworkInit(DistStnWithNetwork* me)
{
    memset(me, 0, sizeof(DistStnWithNetwork));
    me->input.events = 0;
    SUBSCRIBE_5_1init(&me->Inputs);
    PUBLISH_5_2init(&me->Outputs);
    DistributionStationInit(&me->DistStn);
    SUBSCRIBE_Binit(&me->InitSub);
    PUBLISH_Binit(&me->InitConfirm);
    strcpy(me->Inputs._ID, "127.0.0.1:6777");
    me->Inputs.QI = true;
    me->Outputs.QI = true;
    strcpy(me->Outputs._ID, "127.0.0.1:6888");
    me->InitSub.QI = true;
    strcpy(me->InitSub._ID, "127.0.0.1:6666");
    me->InitConfirm.QI = true;
    strcpy(me->InitConfirm._ID, "127.0.0.1:9999");
    me->state = 1;
    me->entered = false;
    continue;
}
}
```

```c
if (me->_input.event.PosChar)
    me->state = 1;
    me->entered = false;
    continue;
}
break;

case 1:
    // State: GoingForward
    if (me->_entered)
        DistStnPusher_GoForward(me);
        me->output.event.PusherCtrl(1);
        me->entered = true;
    }
else {
    if (me->_input.event.PosChar)
        me->state = 2;
        me->entered = false;
        continue;
```

```c
```
Simulation
Industrial Impact

- Glidepath (airport baggage handling system)
- Powerplants (greenhouse controller)
- Integration with nxtControl Studio (commercial IDE)
- Auckland UniServices (IDE – editor, compiler, timing analyzer, and model checker for function blocks)
Possibilities Are Limitless: Robotics and ...

- Airport Baggage Handling
- Machine Tools
- Smart Building
- Mechatronics
- Smart Grid
- Food processing
- Monitoring
- Agriculture
Safety Critical MDE and Robotics

- Need to elaborate function block based models to the complex requirements identified for robotics models
- While maintaining advantages for verification, timing analysis
Summary

Model driven engineering for mobile robotics:
• Higher level of abstraction for behaviour description
• More appropriate cognitive model for humans in different roles
• Facilitation of multidisciplinary research and development
• Tools for verifying properties, such as safety
• Automatic generation of implementations
• Many different models result
• *MDE for cognitive match, not just engineering*
• Future: rationalization of appropriate models