An Intuitive and Flexible Architecture for Intelligent Mobile Robots

By Xiao-Wen Terry Liu
Dept. of Computer Science
The University of Manitoba
Winnipeg, Manitoba
November 17 2005

Outline

- Introduction & Motivation
- Background – Robot Architectures
- Design
- Evaluation
- Conclusion
Goal: Develop an intuitive, adaptive, and flexible architecture for controlling intelligent mobile robots

An architecture is a unifying, coherent form or method of construction, which provides the foundation for creating powerful intelligent systems.

Intelligent:
- Pragmatic definition of intelligence
- Must act autonomously
- Must perform appropriate action in controlled and uncertain situations

Intelligent behaviour

Performing appropriate actions demonstrating behaviours that are working towards completing the system’s objectives

Difficulties:
- Selecting the correct action among a very large set of possibilities
- Real-time constraints
  - Reaction to danger and other events
  - Noisy sensors and imprecise actuators
Motivation

- Developing, maintaining, and modifying systems to control intelligent mobile robots in the real world can be a daunting task.

Motivation (II)

- Systems often limited by initial design
- New architecture’s focus:
  - Adaptable: deals with noise and uncertainty
  - Flexible: add new tasks, change & remove existing tasks
  - Extensible: add new sensors/actuators
  - Intuitive: make the things above easy
Test Domain

- Robotic soccer
  - Requires common fundamental skillset for mobile robots
    - Real-time control
    - Perception
    - Awareness
    - Planning
    - Coordination and Communication

Robotic soccer research

- Excellent testbed for research
  - Dynamic and complex domain
  - Large community of researchers
Architecture classifications
- Domain Relevance vs. Domain Independence
- Analysis vs. Synthesis
- Top-down vs. Bottom-up
- Deliberative vs. Reactive

Domain Relevance vs. Domain Independence
- Domain Relevance: specialized for a domain/task
- Domain Independence: functions for multiple domains
- Matter of Utility vs. Efficiency
Analysis vs. Synthesis

- **Analysis**: starts with an intelligence model
- **Synthesis**: starts with a basic unit/component of intelligence (unified field theory)

Top-down architecture

- The first type of architecture
- **Knowledge-driven** – takes a problem and decomposes it to further sub-problems
- Good for simple/routine tasks, however cannot cope with errors well.
**Bottom-up architecture**

- More reactive by design
- Good for dealing with unexpected situations
- Example: Brooks’ Subsumption Architecture

**Hybrid Architecture**

- Incorporates the advantages of both Top-down and Bottom-up design
  - But also the disadvantages
  - Importance is on finding a balance
- Examples: 3T, Atlantis, Aura, RCS, SSS
- Belief-Desire-Intention (BDI) Architecture
- Blackboard Architecture
- Ubiquitous Robot Architecture
Behaviour Specification

- There are some specialized languages for behaviour language
- eXtensible Agent Behavior Specification Language (XABSL)

```
<option xsi:schemaLocation="http://www.ki.informatik.hu-berlin.de/XABSL2.1 ...
./../xabsl2/xabsl-2.1/xabsl-2.1.option.xsd" name="striker" initial-state="initial">
  <state name="initial">
    <subsequent-basic-behavior ref="go-to">
      <set-parameter ref="go-to.x"> 
        <decimal-input-symbol ref="ball.x"/>
        <decimal-value value="8"/>
      </set-parameter>
      <subsequent-basic-behavior>
        <transition-to-state ref="initial"/>
      </decision-tree>
    </state>
  </option>
```

Design

- Requirements: adaptable, flexible, extensible, and intuitive
- Archangel architecture
  - Sensor and Actuator modules
  - World Model
  - Sequencing
  - Timing Constraints
  - MR Clients
    - Flexible Behaviour Selection Mechanism
  - Explicit Representation of behaviour specifications
Design Overview

Sensors and actuators use a loosely-coupled methodology to allow for extensibility

- i.e. perceptual processing routines and command generator abstracts hardware from logical

- Use perceptual processing routines to link sensors to World Model
  - Filtering of object coordinates – e.g. obstacles, ball
  - Useful to deal with errors
World Model

- Very useful for purpose-driven (proactive) behaviours
- Can be used when sensors fail
  - For example, when the ball gets hidden behind another robot from the camera sensor

Sequencing

- Different levels of sequencing
  - Task Sequencing – ordering subtasks to complete the goal
    - E.g. Steal the ball, go behind the ball, then kick
  - Action Sequencing – e.g. a set of waypoints to move to destination
  - Actuator Command Sequencing – more performance related
    - E.g. help smooth out turns
Sequencing (II)

- Trend
  - Units more abstract up the pyramid
  - Fewer units queued up the pyramid
- More queuing usually means better efficiency, however less reactivity

Timing Constraints

- Necessary to deal with real-time constraints
- Minimum time allowed for behaviours
  - Useful to avoid behaviour oscillation
- Maximum time allowed for behaviours
  - Useful to avoid local minima/maxima situations
Mobile Robot Client

- Controls one (physical) robot

Behaviour System

- Uses competition as behaviour selection
- Behaviour with the largest activation (applicability + reward) value:
Command Generator & Sender

- Relates to the actuator command sequencing
- Abstracts the physical actuators

Explicit Representation

- Using eXtensible Markup Language (XML) to describe behaviours
- Different types of behaviours
- High-level complex behaviour representation
  - Behaviour-tree Composition
  - Finite State Machine
- Low-level action behaviours
Sample Behaviour Representation

- `<behaviour name="sampleBehaviour">
  - `<init>
    `<target_list ofType="obstacle" src="World::videoObjects/>
  </init>
- `<draw_env>
  `<!-- Home location -->
  `<pen colour="red">
  `<rect x="360" y="40" width="20" height="20"/>
  </draw_env>
- `<behaviour_list>
  `<behaviour_ref name="chaseTarget"/>
  `<behaviour_ref name="goHome"/>
  `<behaviour_ref name="dance"/>
  </behaviour_list>
- `<reward value="0.5"/>
- `<applicability>
  - `<robot fartherThan="50">
    `<reference_position reference="closestTarget"/>
    `<add value="0.8"/>
  </robot>
  </applicability>
- `<execute useBehaviourList="true"/>
</behaviour>

Implementation

- Robots

- System: Linux, QT3
Evaluation

- Difficult to evaluate robot architecture.
  - (No universally accepted standard)
- User study – costly (time, money, and other resources)
- Evaluation between architectures are more of a question of efficiency rather than computability – R.C. Arkin
- Anecdotal evidence on several challenge tasks
  - Used soccer domain
  - RoboCup challenge tasks
- Bias

Challenges

- Simple tasks required of mobile robots applicable to many domains
  - Path Tracking (Racetrack)
  - Obstacle Avoidance (Obstacle Run)
  - Path Planning (Treasure Hunt)
  - Goal-Scoring (Shooting)
  - Robotic Interaction (Ball-Passing)
Path Tracking

Sample state

```xml
- <state name="state1">
  - <draw_env>
    <pen colour="red"/>
    <rect x="360" y="40" width="20" height="20"/>
  </draw_env>
  - <goto>
    <absolute_position x="370" y="50"/>
  </goto>
  - <next_state name="state2">
    - <condition>
      - <robot within="60">
        <absolute_position x="370" y="50"/>
      </robot>
    </condition>
  </next_state>
</state>
```

VS. 370 50
Treasure Hunt

- <behaviour name="treasureHunt">
  - <init>
    <target_list ofType="obstacle" src="World::videoObjects"/>
  </init>
  - <behaviour_list>
    <behaviour_ref name="treasureHuntChase"/>
    <behaviour_ref name="treasureHuntTurn"/>
  </behaviour_list>
  <applicability value="1"/>
  <execute unlessBehaviourList="true"/>
</behaviour>

- <behaviour name="treasureHuntChase">
  <init/>
  <reward value="1"/>
  <applicability>
    <condition>
      <robot fartherThan="50">
        <reference_position reference="closestTarget"/>
        <add value="0.8"/>
      </robot>
    </condition>
    <applicability>
      <execute>
        <goto within="40">
          <reference_position reference="closestTarget"/>
        </goto>
      </execute>
    </applicability>
  </applicability>
</behaviour>
Obstacle Run

- Impossible scenario using single destination point

Obstacle Run

- Explicit trigger mechanism for zone
Obstacle Run

- <behaviour name="obstacleRun">
  - <init>
    - <reward value="1!" />
    - <applicability value="1!" />
  - <execute initial_state="RunToZone1" />
    - <state name="RunToZone1">
      - <goto>
        - <control_command avoidWall="true" />
        - <absolute_position x="2610" y="760" />
      - <robot>
      - <condition>
        - <robot within="50" />
        - <absolute_position x="2610" y="760" />
      - <robot>
      - <next_state name="RunToZone2"></next_state>
    </state>
  - <trigger_set>
    - <trigger_next_state target_name="RunToZone2">
    - <condition>
      - <robot>
      - <within_rect x="2520" y="260" width="220" height="1000" />
      - <robot>
    </condition>
    - <trigger_next_state>
  </trigger_set>
</state>

Obstacle Run

- <state name="AfterZone1Pause" minExecMicroSecs="5000000">
  - <goto>
    - <reference_position reference="World::self" />
  - <goto>
  - <next_state name="RunToZone2">
    - <condition met="true" />
  </next_state>
</state>
Robot's Path

Goal Scoring

Nov. 17, 2005

Goal Scoring

Shooting

- <behaviour name="shooting">
  - <behaviour_list>
    <behaviour_ref name="GoBackHome"/>
    <behaviour_ref name="MoveBehindBall"/>
    <behaviour_ref name="LineUpToKick"/>
    <behaviour_ref name="KickForward"/>
  </behaviour_list>
  <reward value="1"/>
  <applicability/>
  <execute useBehaviourList="true"/>
</behaviour>
Goal Scoring

Passing challenge
Passing

- Passing
- Sub-behaviours:
  - GoBackHome
  - PassingDrill

PassingDrill

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<robot>
  <start name="start">
    <condition>
      <sensor name="ballFound">
        <robot name="robot">
          <sensor name="ball_found">
            <variable name="ballFound">
              <value>true</value>
            </variable>
          </sensor>
        </robot>
      </sensor>
    </condition>
    <state name="dribble">
      <control_command avoidBall="true"/>
      <relative_x y="0" z="0" w="0"/>
      <absolute_position x="0" y="0" z="0" w="0"/>
      <motion_width x="0" y="0" z="0" w="0"/>
      <motion_height x="0" y="0" z="0" w="0"/>
      <motion_depth x="0" y="0" z="0" w="0"/>
      <motion_updown x="0" y="0" z="0" w="0"/>
      <motion_left_right x="0" y="0" z="0" w="0"/>
      <motion_up_down x="0" y="0" z="0" w="0"/>
      <motion_right_left x="0" y="0" z="0" w="0"/>
      <motion_forward x="0" y="0" z="0" w="0"/>
      <motion_backward x="0" y="0" z="0" w="0"/>
      <motion_stop x="0" y="0" z="0" w="0"/>
      <motion_slide x="0" y="0" z="0" w="0"/>
      <motion_tilt x="0" y="0" z="0" w="0"/>
      <motion_turn x="0" y="0" z="0" w="0"/>
      <motion_rotate x="0" y="0" z="0" w="0"/>
      <motion_twist x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_scale x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
      <motion_pose x="0" y="0" z="0" w="0"/>
Passing

- <state name="goBehindBallForPass">
  - <gate>
    <relative_abs_focus_position offset="behind"
      reference="World:ball" focusPoint.x="2740" focusPoint.y="200"/>
  </gate>
  - <next_state name="pass">
    - <condition>
      - <robot within="50">
        <relative_abs_focus_position offset="behind"
          reference="World:ball" focusPoint.x="2740" focusPoint.y="200"/>
      </robot>
    </condition>
  </next_state>
  - <trigger_set>
    - <trigger if ball crosses the line ->>
    - <trigger_next_state targetName="headHome">
      - <condition>
        - <ball isFound="true">
          <within_rect width="1370" x="1410" y="0" height="1520" />
        </ball>
      </condition>
    </trigger_next_state>
  </trigger_set>
</state>

Nov. 17, 2005

Evaluation

How easy is it to add/change/remove behaviours?

- Variations in tasks
- Metrics
  - Lines of Code (LoC) for changes
  - Time required for changes

Nov. 17, 2005
# Results

<table>
<thead>
<tr>
<th>Code size</th>
<th>Original C++ Client</th>
<th>Archangel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Racetrack original design</td>
<td>50 lines</td>
<td>88 lines of XML</td>
</tr>
<tr>
<td>Racetrack change No. 1</td>
<td>added 1 line</td>
<td>edited 19 lines</td>
</tr>
<tr>
<td>Racetrack change No. 2</td>
<td>modified 7 lines</td>
<td>modified 21 lines</td>
</tr>
<tr>
<td>Treasure Hunt original design</td>
<td>50 lines</td>
<td>40 lines of XML</td>
</tr>
<tr>
<td>Treasure Hunt change No. 1</td>
<td>edited 10 lines</td>
<td>edited 14 lines</td>
</tr>
<tr>
<td>Treasure Hunt change No. 2</td>
<td>edited 1 line</td>
<td>remove 6 lines</td>
</tr>
<tr>
<td>Obstacle Run original design</td>
<td>40 lines</td>
<td>30 lines of XML</td>
</tr>
<tr>
<td>Obstacle Run change No. 1</td>
<td>added 4 lines</td>
<td>Add 18 lines of XML</td>
</tr>
<tr>
<td>Obstacle Run change No. 2</td>
<td>edited 30 lines</td>
<td>edited 18 lines</td>
</tr>
<tr>
<td>Goal Scoring original design</td>
<td>172 lines</td>
<td>84 lines of XML</td>
</tr>
<tr>
<td>Goal Scoring change No. 1</td>
<td>add 8 lines</td>
<td>add 8 lines</td>
</tr>
<tr>
<td>Goal Scoring change No. 2</td>
<td>edited 10 lines</td>
<td>add 11 lines</td>
</tr>
<tr>
<td>Goal Scoring change No. 3</td>
<td>edited 20 lines</td>
<td>edited 19 lines</td>
</tr>
<tr>
<td>Goal Scoring change No. 4</td>
<td>modify 2 lines</td>
<td>add 4 lines</td>
</tr>
<tr>
<td>Goal Scoring change No. 5</td>
<td>modify 14 lines</td>
<td>add 20 lines</td>
</tr>
<tr>
<td>Passing original design</td>
<td>320 lines</td>
<td>322 lines of XML</td>
</tr>
<tr>
<td>Passing change No. 1</td>
<td>modify 4 lines (x2)</td>
<td>modify 1 line (x2)</td>
</tr>
<tr>
<td>Passing change No. 2</td>
<td>modify 8 lines</td>
<td>added 12 lines (x2)</td>
</tr>
<tr>
<td>Passing change No. 3</td>
<td>modify 1 line (x2)</td>
<td>modify 1 line (x2)</td>
</tr>
<tr>
<td>Passing change No. 4</td>
<td>modify 1 line (x2)</td>
<td>add 1 line (x2)</td>
</tr>
<tr>
<td>Passing change No. 5</td>
<td>modify 3 line (x2)</td>
<td>modify 2 line (x2)</td>
</tr>
</tbody>
</table>

# Results (II)

<table>
<thead>
<tr>
<th>Development Time</th>
<th>Original C++ Client</th>
<th>Archangel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Racetrack original design</td>
<td>7 days</td>
<td>4 days</td>
</tr>
<tr>
<td>Racetrack change No. 1</td>
<td>8 minutes</td>
<td>8 minutes</td>
</tr>
<tr>
<td>Racetrack change No. 2</td>
<td>15 minutes</td>
<td>18 minutes</td>
</tr>
<tr>
<td>Treasure Hunt original design</td>
<td>3 days</td>
<td>3 days</td>
</tr>
<tr>
<td>Treasure Hunt change No. 1</td>
<td>20 minutes</td>
<td>8 minutes</td>
</tr>
<tr>
<td>Treasure Hunt change No. 2</td>
<td>6 minutes</td>
<td>4 minutes</td>
</tr>
<tr>
<td>Obstacle Run original design</td>
<td>2 days</td>
<td>1 hour</td>
</tr>
<tr>
<td>Obstacle Run change No. 1</td>
<td>40 minutes</td>
<td>2 hour</td>
</tr>
<tr>
<td>Obstacle Run change No. 2</td>
<td>30 minutes</td>
<td>10 minutes</td>
</tr>
<tr>
<td>Goal Scoring original design</td>
<td>7 days</td>
<td>5 days</td>
</tr>
<tr>
<td>Goal Scoring change No. 1</td>
<td>10 minutes</td>
<td>7 minutes</td>
</tr>
<tr>
<td>Goal Scoring change No. 2</td>
<td>9 minutes</td>
<td>7 minutes</td>
</tr>
<tr>
<td>Goal Scoring change No. 3</td>
<td>50 minutes</td>
<td>40 minutes</td>
</tr>
<tr>
<td>Goal Scoring change No. 4</td>
<td>9 minutes</td>
<td>6 minutes</td>
</tr>
<tr>
<td>Goal Scoring change No. 5</td>
<td>30 minutes</td>
<td>13 minutes</td>
</tr>
<tr>
<td>Passing original design</td>
<td>14 days</td>
<td>3 days</td>
</tr>
<tr>
<td>Passing change No. 1</td>
<td>23 minutes</td>
<td>15 minutes</td>
</tr>
<tr>
<td>Passing change No. 2</td>
<td>1 hour</td>
<td>1 hour</td>
</tr>
<tr>
<td>Passing change No. 3</td>
<td>10 minutes</td>
<td>6 minutes</td>
</tr>
<tr>
<td>Passing change No. 4</td>
<td>15 minutes</td>
<td>10 minutes</td>
</tr>
<tr>
<td>Passing change No. 5</td>
<td>25 minutes</td>
<td>15 minutes</td>
</tr>
</tbody>
</table>
Conclusion

- Lines of code is imperfect measurement
  - How much of the code is functional?
- Time required difficult to remove implication of bias
- Future Work
  - Additional domains and tasks

The End

(Questions?)