Lab 5: Domain Modeling
Due: April 16, 2012 23:00 CDT

1. Key Lab Outcomes

- Perform object discovery to analyze a problem and determine candidate objects
- Construct a domain model showing collaborations and use cases for a system
- Use configuration management construct designs in a disciplined fashion
- Develop a proof of concept that shows the ability to program in the LeJOS environment
- Update and maintain a project work plan

2. Due Dates

Deliverable	Responsible Part	Due Date
Completed Domain Model and Candidate Object list - Domain model archived in SVN	Development Manager	April 16, 2012
Report submitted containing candidate object and domain model	Team Leader	April 16, 2012
Compass code archived into SVN	Support Manager	April 16, 2012
Compass demonstration	Planning Manager	Next lab session

3. Introduction

Thus far, for this lab, we have done very little with configuration management. That is about to change. In order to be successful in a team environment, it is vital that all of our work be conducted using the repository. This will ensure that all team members can work in an appropriate fashion and develop an effective project.

4. Lab Overview

In this lab, you will develop a domain model for your system. While the domain model is being developed, a subset of the team will work in the LeJOS environment to develop a simple application.

5. Developing a Domain Model

5.1. Update the work plan

To begin with, the Process manager should update the work plan tracking document in the Project Tracking folder of the archive. This document is an Excel worksheet, and it essentially

lists line by line the work that will be completed, and also serves to record the effort spent completing those tasks.

For last week, simply query the team members as to what they did to complete the lab and how long they thought it would take to complete the tasks. Then, record how long it actually took for them to complete the tasks in the second column.

For this week, enter the tasks that will be completed as well as the persons to whom they are assigned. For each task, estimate how long it will take to complete the item, as well as when it will be started and finished. Record these into the time tracking document before uploading the document to the repository. Be careful to include all tasks.

5.2. Determining the Objects in the System

Each team must develop and document a domain model for the LEGO MINDSTORM car system. This domain model includes all components necessary to implement your software application. The domain model should be developed based upon your use cases as well as what you know about your LEGO MINDSTORM car system.

While there are many approaches that can be taken for this, it is recommended that you start by using a technique similar to underlining the nouns. Essentially, with the development manager leading, brainstorm all of the nouns within the system, generating a list of nouns. When this is completed, think about the real world items that exist within the system and record these. Continue through the system discovering the objects that are part of the system domain. There are several starting points for this exercise. One starting point is the initial system description, but it is also perfectly valid to start with the use case scenarios as well.

When this step is finished, you should have a solid list of the pieces to the problem.

5.3. Developing the Relationships

Once the objects in the system have been created, create a series of diagrams which model the domain for the major use cases of the system. This model would be equivalent to the lower half of Figure 6-2 on page 335 of the textbook. While there may be some overlap, you will need to consider the creation of domain models for the “Drive Robot”, “Reset Odometer”, and “Play Radio” use cases. (Note: Your names may be slightly different.)

While the exact content of the domain model will be determined by the team, it is important that the domain model show all interactions between pieces in the system.

When all is finished, your domain model within enterprise architect shall; have several diagrams showing the relationship between items and all of the pieces of the problem will be defined.

In creating your diagrams, it is important to note that at this time, you do not provide any details about type, scope, etc. EA will automatically add this information to your diagram, but you will

want to suppress its display. This is accomplished by right clicking on the diagram, selecting "properties" "features" and suppressing parameter detail and only showing the name of attributes.

5.4. Updating your development Repository

When the development of your domain model is completed, make certain to update the package archive of your project.

First, from a learning standpoint, determine what has changed with your package. To do this, right click on the Domain Model, Select Package Control, and "Compare with XMI file". This will allow you to compare the state of the earlier model with the current state. Things which are listed as added are new. Things which have been deleted are marked as deleted.

Next, update your XMI file. To do this, right click on the package and click "Package Control" and "Save Packages to File". This will update the xml package for the given project for the domain model. Later on, we will learn how to control development with even finer granularity.

With this completed, re-run the difference tool, and you will see that both files should be the same.

6. Developing a LeJOS proof of concept program

6.1. Part 1

While the team is developing the domain model, the support manager (or designee) is responsible for developing a simple leJOS application. In essence, a simple digital compass should be developed. The compass will use the compass sensor of the Lego kit and display the angle and direction that the compass sensor is pointing on the LCD display of the Mindstorm computer. The direction should match the criteria shown in the table below

Angle	Display
$0 \leq \text{angle} < 22.5$ $337.5 \leq \text{angle} < 0$	Specific Degree Reading + "NORTH"
$22.5 \leq \text{angle} < 67.5$	Specific Degree Reading + "NORTH EAST"
$67.5 \leq \text{angle} < 112.5$	Specific Degree Reading + "EAST"
$112.5 \leq \text{angle} < 157.5$	Specific Degree Reading + "SOUTH EAST"
$157.5 \leq \text{angle} < 202.5$	Specific Degree Reading + "SOUTH"
$202.5 \leq \text{angle} < 247.5$	Specific Degree Reading + "SOUTH WEST"
$247.5 \leq \text{angle} < 292.5$	Specific Degree Reading + "WEST"
$292.5 \leq \text{angle} < 337.5$	Specific Degree Reading + "NORTH WEST"

The direction should be updated whenever the left or right button is pressed. If the escape or exit button is pressed, the program shall exit.

When completed this program should be archived in svn.

6.2. Part 2 Displaying the results on your PC

Now that the basic compass has been completed, you are responsible for developing a PC application. In essence, this application should use the Lego Mindstorm robot to update the display every second for 60 seconds. After 60 seconds, the program(s) shall exit.

To do this, you will need to use the USB cable and the code provided to you which allows one to send and receive an integer over the USB cable.

The PC software DOES NOT need to use a GUI. I can simply print the results to a console.

7. Lab Deliverables

7.1. Report

Each team shall submit a mini-report consisting of the following:

1. Title Page
    a. Name of team
    b. Team Members
    c. Submission Title
2. Introduction
    a. Explain, in your own words, what this lab was trying to accomplish
3. Candidate Object List
    a. List the candidate objects within the system (Note: This is a simple list, unless explanation is called for in which case a footnote would be appropriate)
    b. For any objects in which multiplicity may apply, state the multiplicity of the system
4. Domain Model Diagrams
    a. Submit copes of the domain model diagrams from the EA model. Simply copy and paste the model diagrams into a word document. Each diagram should be provided with its own separate section.
    b. For each diagram, provide a brief explanation of what the objects are and why they exist within the model.
5. Things Gone Right / Things Gone Wrong
    a. Describe briefly the things that went right with this activity.
    b. Describe briefly the problems you had completing this exercise.
6. Conclusions
    a. What has been learned from this exercise?
    b. Has this exercise helped to develop a more cohesive understanding of the problem and how to approach solving the problem?

The report shall be submitted in pdf format through the web submission page. The lab report also should be checked into your svn archive.

7.2. Work plan

The updated work plan showing estimates and effort should be checked into SVN

7.3. Demonstration

During the next lab session, the Support manager (or designee) shall demonstrate proper operation of the compass.

Appendix A: Preliminary Grading Rubric

	Weight Factor	Rubric Score	
	.25		Title Page - Name - Section - Instructor - Date - Assignment
	1		Introduction - Full explanation of the scope of the lab assignment - Written in one's own words - Written in multiple complete sentences
	1		Candidate Object List - Object list is a complete - Object list contains only nouns - Object list contains multiplicity where applicable - Object list generated from initial specification as well as analysis of causal objects, services, real world items, physical devices, key concepts, transactions, persistent information, visual elements, control elements, and scenarios.
	.5		Images - Images submitted in report are legible. - Images appropriately labeled and titled
	2		Domain Models - Domain model properly show associations between objects in the problem domain. - Associations between model objects labeled appropriately - Multiplicity provided for all relationships when appropriate - Appropriate usage of composition and aggregation as necessary. - Appropriate usage of generalization as necessary. - Attribute types not provided
	2		Domain Model Explanation - Brief explanation of the domain model diagrams provided - Multiple sentences - Provides details necessary to explain why the diagram and relationships exists as shown
	.5		TGR-TGW - Things gone right with lab discussed - Problems encountered in project discussed. - Multiple, complete sentences.
	1		Work plan - Work plan committed into archive at least twice, once during the initial planning and once at the end of the lab session with actual effort amounts recorded. - Work plan shows roughly equal contributions to project (at least during estimation) - Workplan shows effort from all parties involved in the project.

	Weight Factor	Rubric Score	
	.5		SVN Repository - Report checked into the SVN repository in native word doc or docx format. - Compass code checked into svn as appropriate - All submissions have meaningful submission comments.
	1		Conclusions filled out. - What was learned from this experience?
	1		Report generally grammatically correct -Written in 3rd person -No usage of I, we, etc. -No significant spelling or grammar errors. -Complete sentences with clear meaning.
Implementation	1		Compass - Com pass functions properly - Compass code cleanly written - Compass code properly documented with JavaDoc, etc. - Compass includes both desktop and Mindstorm implementations in separate files / projects.

8. Class LegoMindstormsProxy

```
java.lang.Object
  └ edu.msoe.se2890.supplied.LegoMindstormsProxy
```

All Implemented Interfaces:

java.lang.Runnable

```
public class LegoMindstormsProxy
extends java.lang.Object
implements java.lang.Runnable
```

Field Summary

static int	**BLUETOOTHPROXY**
static int	**USBPROXY**

Method Summary

void	**enqueue**(int valueToSend) This method will enqueue an integer that is to be transmitted to the pc.
static LegoMindstormsProxy	**getSingleton**(int type) This method will return the Singleton instance for the proxy which communicates with the PC proxy.
boolean	**isRunning**() This method will query the proxy to determine if it is still running or not.
int	**receiveInt**() This method will receive an integer from the pc.
void	**run**() This is the run method.
void	**shutdown**() This method will shutdown the running proxy.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

8.1.1. BLUETOOTHPROXY

`public static final int` **`BLUETOOTHPROXY`**

See Also:

Constant Field Values

8.1.2. USBPROXY

`public static final int` **`USBPROXY`**

See Also:

Constant Field Values

Method Detail

8.1.3. enqueue

`public void` **`enqueue`**`(int valueToSend)`

This method will enqueue an integer that is to be transmitted to the pc.

Parameters:

`valueToSend` - This is the value to send.

8.1.4. getSingleton

`public static` `LegoMindstormsProxy` **`getSingleton`**`(int type)`

This method will return the Singleton instance for the proxy which communicates with the PC proxy.

Parameters:

`type` - The type is either USBPROXY or BLUETOOTHPROXY.

Returns:

The singleton instance of the proxy will be returned.

8.1.5. isRunning

`public boolean` **`isRunning`**`()`

This method will query the proxy to determine if it is still running or not.

Returns:

true or false, depending on the state of the proxy.

8.1.6. receiveInt

`public int` **`receiveInt`**`()`

This method will receive an integer from the pc.

Returns:

The return value will be an integer from the pc. If there is nothing on the queue, the code will block.

8.1.7. run

`public void` **`run`**`()`

This is the run method. It simply runs the proxy.

Specified by:

`run` in interface `java.lang.Runnable`

8.1.8. shutdown

`public void` **`shutdown`**`()`

This method will shutdown the running proxy.

9. Class PCMindstormProxy

```
java.lang.Object
  └ edu.msoe.se2890.supplied.PCMindstormProxy
```

All Implemented Interfaces:

java.lang.Runnable

```
public class PCMindstormProxy
extends java.lang.Object
implements java.lang.Runnable
```

Field Summary

static int	**BLUETOOTHPROXY**
static int	**USBPROXY**

Method Summary

void	**enqueue**(int valueToSend) This method will enqueue an integer that is to be transmitted to the lego mindstorm.
static PCMindstormProxy	**getSingleton**(int type) This method will obtain a proxy which can be used to communicate with the LegoMindstorm.
int	**receiveInt**() This method will receive an integer from the lego mindstorm robot.
void	**run**() This is the run method.
void	**shutdown**() This method will shutdown the running proxy.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

9.1.1. BLUETOOTHPROXY

`public static final int` **`BLUETOOTHPROXY`**

See Also:

Constant Field Values

9.1.2. USBPROXY

`public static final int` **`USBPROXY`**

See Also:

Constant Field Values

Method Detail

9.1.3. enqueue

`public void` **`enqueue`**`(int valueToSend)`

This method will enqueue an integer that is to be transmitted to the lego mindstorm.

Parameters:

`valueToSend` -

9.1.4. getSingleton

`public static` `PCMindstormProxy` **`getSingleton`**`(int type)`

This method will obtain a proxy which can be used to communicate with the LegoMindstorm.

Parameters:

`type` - The type is either USBPROXY or BLUETOOTHPROXY.

Returns:

A proxy instance will be returned.

9.1.5. receiveInt

`public int` **`receiveInt`**`()`

This method will receive an integer from the lego mindstorm robot.

Returns:

The return value will be an integer from the mindstorm robot.

9.1.6. run

`public void` **`run`**`()`

This is the run method. It simply runs the proxy.

Specified by:

`run` in interface `java.lang.Runnable`

9.1.7. shutdown

`public void` **`shutdown`**`()`

This method will shutdown the running proxy.