Meeting Purpose: Software Design Review for the Wandering Ambassador (Part 6) project (P11216).

Materials to be reviewed:

1. Sensor Information
2. Safety Concerns
3. Updated UML Design
4. Updated Sequence Diagrams

Meeting Date: January 28, 2011

Meeting Location: *70-2690*

Meeting Time: *1:30 PM*

Timeline:

Meeting Timeline		
Start Time	Topic of Review	Required Attendees
1:30	Project Overview (if needed)	P11215, P11216, Professors
1:32	Sensor Information	P11215, P11216, Professors
1:38	Safety Concerns	P11215, P11216, Professors
1:45	Updated UML	P11215, P11216, Professors
1:52	Updated Sequence	P11215, P11216, Professors

Project #	Project Name	Project Track	Project Family
P11216	Wandering Ambassador (Part 6)	Vehicle/Robotics	Land Vehicle
Start Term	**Team Guide**	**Project Sponsor**	**Doc. Revision**
20102	George Slack	KGCOE EE Dept	2

Project Description

Project Background:

RIT has always been interested in finding new and exciting ways to build interest and showcase student activities on campus. The primary customer conceptualized a robot which slowly moves around campus caring for an onboard plant and runs on sustainable energy. This would serve multiple purposes, which include showing RIT's commitment to sustainable energy, student innovation and technical ability, as well as providing a talking piece for visitors to campus.

Problem Statement (*Robot Track (2009/2010/2011)*:

The main goal of this project is to raise awareness of RIT innovation by designing a robot that acts as a guardian of a plant and who acts in a symbiotic relationship with the plant. The robot will support the needs of the plant, as well as its own, by managing sunlight and soil water content.

P11216 will join the P11215 team to evaluate previous development and then develop or mature needed functionality as well as test all robotic functions and continue to refine the design, as needed.

Objectives/Scope:

1. Work with P11215 efficiently to complete robot functions.
   - Improve the robot's navigation functions so that it may wander unattended.
   - Have the robot be able to care for the plant for a period of at least 1 week.
2. Define detailed test routines to uncover reliability issues.
3. Continue the implementation process of debugging hardware and software as the integration process begins. (i.e., characterize and evaluation the various robot sensors and output devices.)
4. Make maximum use of natural conditions by managing sun, shade, temperature, rain, and watering to allow the plant to grow and thrive and robot power to self-sustain.
5. Establishment of an environment which allows software development to proceed before hardware is available and integrates with hardware.
6. Full definition and implementation of software application programming interface to the robot navigation and plant support functions.
7. By the start of the spring quarter, evaluate the test results and issues from the P11215 team, and create a plan to eliminate critical known software issues.
8. Perform outdoor field testing, and debug software issues.

Deliverables:

1. Improved design with improved safety features.
2. Implemented plant care system.
3. Testing routine for drivetrain features including drive transmission and safety issue.
4. Testing routine for transport scheme needs to be evaluated.
5. Testing routine for plant portion of the robot including water reservoir and dispensing.
6. Robot navigates autonomously at the Innovation Festival in spring of 2011.

Expected Project Benefits:

1. Showcase the creativity and technical abilities of RIT's Multidisciplinary Senior Design teams.
2. Navigate autonomously and take care of the plant at RIT's Innovation Festival in Spring 2011.
3. Reinforcement of RIT's devotion to innovation, sustainability projects and energy resources.
4. Excellent demonstration of good testing procedures for integrated systems of its kind.
5. Define robust application programming interfaces for higher-level on-board processing.
6. Work in conjunction with the initial team during the winter term to expand the range of plant maintenance, environment interaction, and navigation functions that are available.

Core Team Members:

Nick Leathe (ME)
Anna Gilgur (ME)
Rui Zhou (EE)
Ken Hertzog (CE)
Joseph Stevens (SE)
Philip Gibson (SE)
Dave Ladner (SE)
Terra McAndrew (ID)
Project Team P11215

Strategy & Approach

Assumptions & Constraints:

1. The robot will not be a safety hazard to observers and campus visitors.
2. Existing locomotion system and frame will be used as a base for modification.
3. The robot needs to function for 1 week unattended.
4. The robot will take care of the plant.
5. The robot will be able to fit through doors.
6. On-board processing power: dual core single board computer, Linux operating system.
7. Sensors and actuators for robot navigation and status, and plant support and status.
8. Prototype design of software interfaces to the robot navigation and plant support functions.

Issues & Risks:

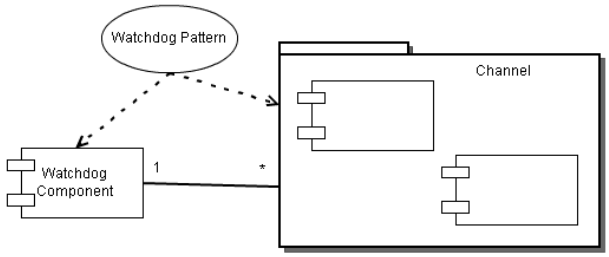
1. Difficulty in software-hardware interaction previous groups experienced continues to hinder progress.
2. Reliability issues
3. Legacy documentation not fully developed.
4. Sensor blind-spots need to be filled in order to prevent collisions.
5. Additional safety mechanisms are required to make the robot safe for curious children.
6. Conflicting customer needs lead to a failure in creating an interesting project.

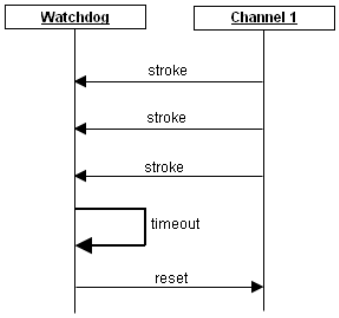
Sensor Information:

Sensor Name	Info goes to/from robot	Description	Comments
Moisture	To	Indicates moisture level of plant soil	
Water Level	To	Indicates how much water is in reservoir	May not be a sensor anymore
Solar / Light	To	Indicates how much light/sunlight is available to the plant	
Temperature	To	Indicates the temperature of the plant and/or surroundings	
Alarm	From	Loud noise generator to deter theft of plant	
Sprinkler	From	Sprays water to deter theft of plant and identify thief	
Water Pump	From	Used to water the plant	
Sonar	To	Indicates if something is in front of the sonar	Horizontally placed, side, front and back
GPS	To	Tells the robot where it is located	
Infrared Sensor	To	Indicates if there is a ledge/cliff	Vertically places facing the ground
Accelerometer	To	Indicates acceleration of robot	Deprecated?
Compass	To	Indicates direction of robot	Deprecated?
Stop Buttons	To	Immediately stops the robot wheels	May not be used by software at all
Bump Sensors	To	Indicates that the robot has run into something	
“Indiana Jones” Switch	To	Indicates whether or not the plant has been stolen	
Sensor Servos	From	Allows the software to turn the sonars	
Motors	From	Tells the wheels to move forward/backward. Used to determine speed of robot?	

Safety Concerns

Part of our new design is to try and incorporate some safety features, for in case some hardware and/or part of the software fails. One way that was proposed to do this was to have a Watchdog pattern in our design. We decided that this would be a good addition to the software, and that it helps to solve the safety concerns as well as add redundancy to the software.



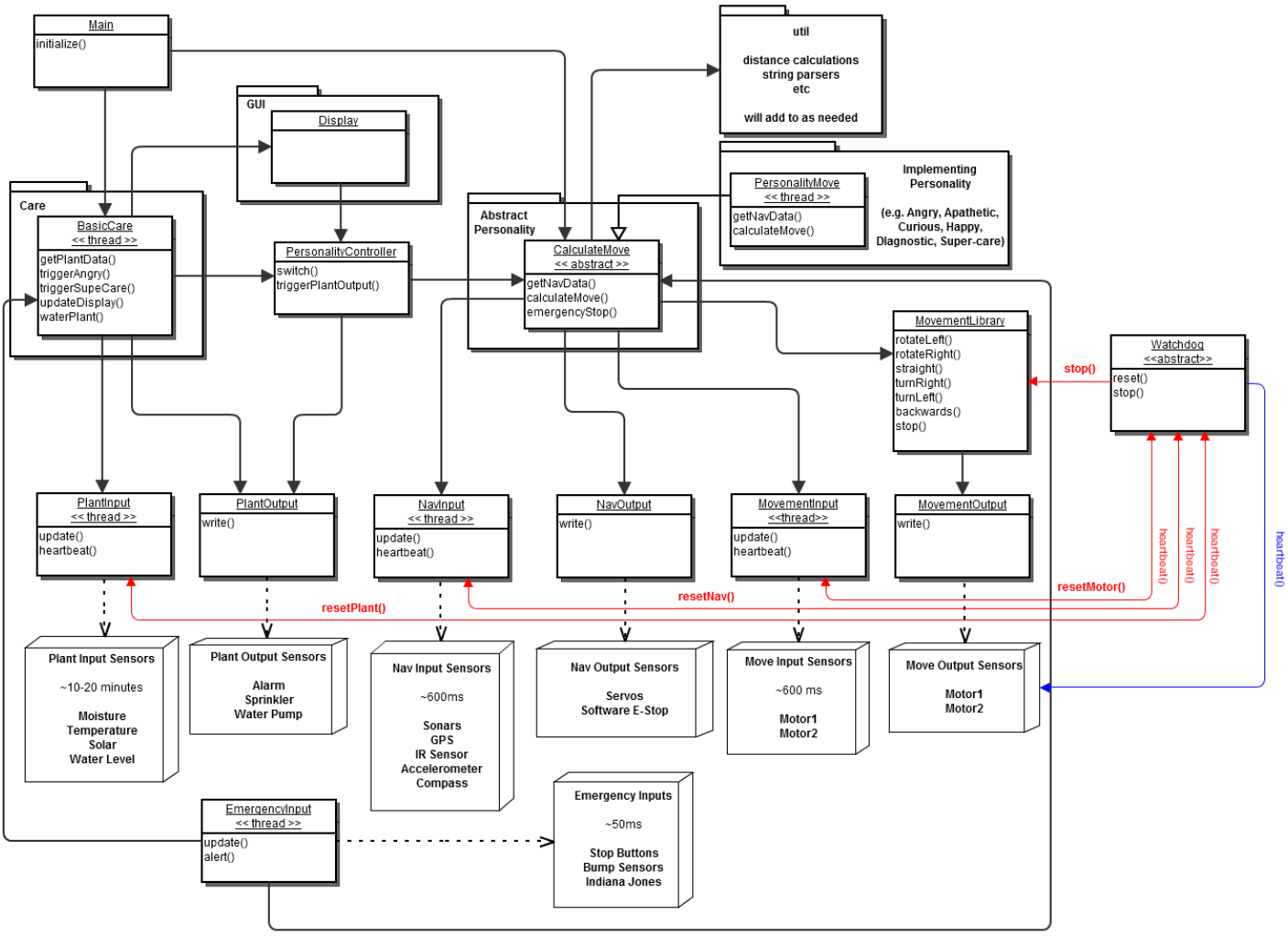


As part of implementing this pattern, we will have a watchdog monitoring the navigational and motor sensors. It will consist of the watchdog class receiving "heartbeats" from the sensors to indicate that the sensor is still "alive." Heartbeats can be any kind of message, and the specifics of the message will be discussed closer to implementation.

Most (if not all) of the input sensors would be monitored by a watchdog, to ensure that the program is responding to data that is current. Navigation, for example, would be unsafe if the sonars or IR sensors are stuck sending the same signal.

The watchdog will also introduce a new concept to the software design, that of a software "emergency" stop. By this we mean that the software will stop the motors, and attempt to somehow restart the sensors, or possibly initiate a full reboot of the robot. Either way, the software will not allow the robot to move until the erroneous sensor gets fixed. (This would most likely only happen with the navigation or motor sensors, as the moisture sensor not giving accurate readings should not cause the robot to stop everything it is doing.)

Updated UML Design:



Logic Classes

- Main
  - An initialize class that starts the BasicCare thread and the DiagnosticMove personality thread.
- PersonalityController
  - Will start and stop personality threads during runtime.
- Display
  - The GUI to be displayed. It will receive its data to display from BasicCare and change color schemes when PersonalityController switches the active personality.
- util classes
  - This represents any class that will be used to aid in calculations for the PersonalityMove. Exact functionality has not yet been determined and will be implemented as needed.
- MovementLibrary
  - Contains a set of movements the robot is able to take.
- BasicCare (thread)
  - This thread will check the data held in PlantInput every 10-20 minutes (TBD). Based on this data, it may water the plant, switch the personality, or just do nothing. It will also push the data it is checking to the Display class so it can be shown on the screen.
- CalculateMove (abstract)

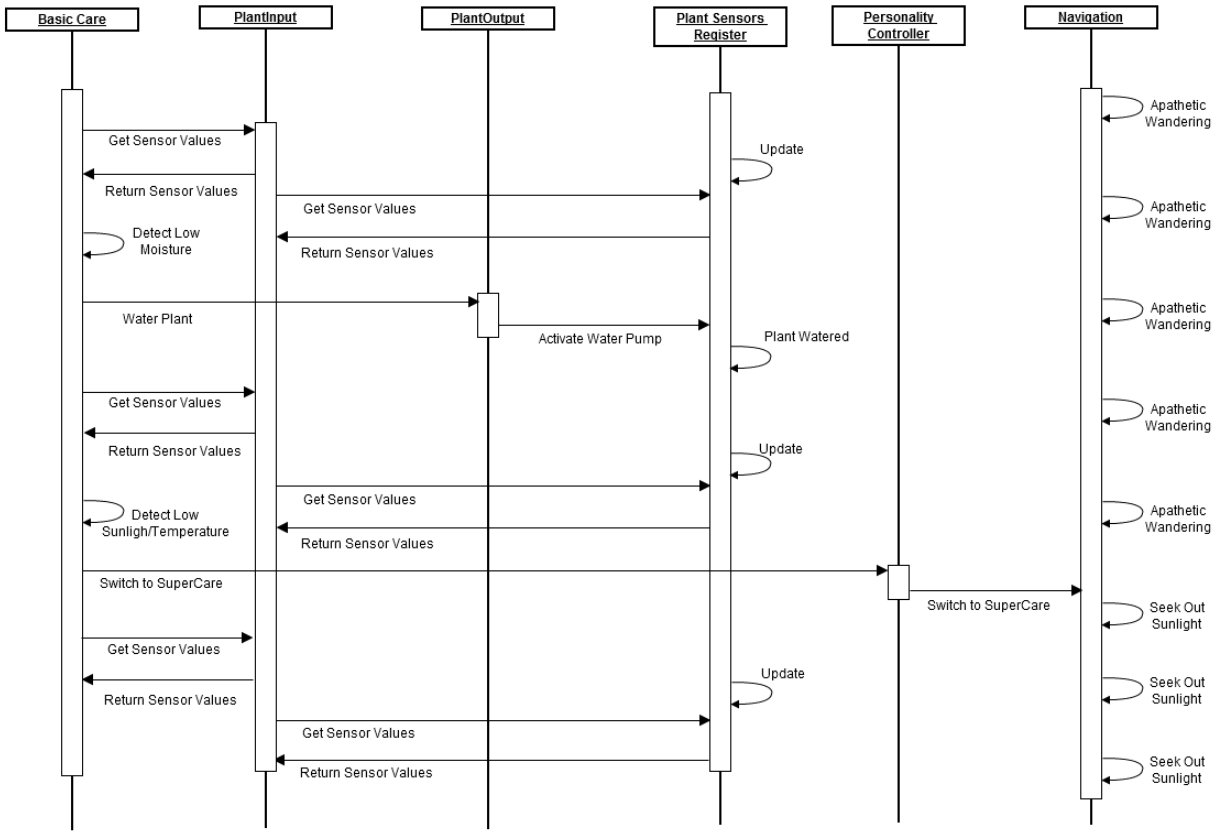
- An abstract class meant to be overwritten by PersonalityMove. It will contain the emergencyStop() method to be inherited by all personalities.

- PersonalityMove (thread)
  - A thread that represents the active personality of the robot. It will check the data stored in NavInput no faster than about every 600 ms (TBD), and based on this data it will determine what movement should be taken. It will then call the MovementLibrary to take that action.

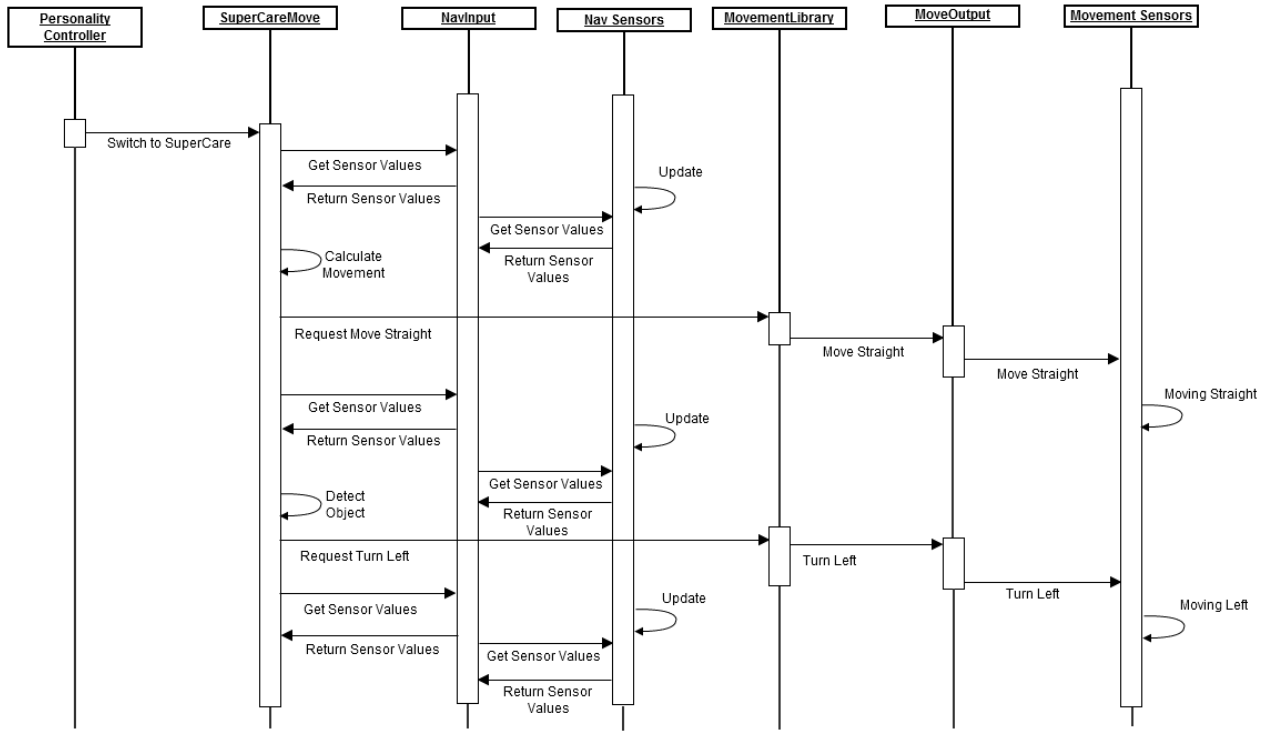
Sensors Input/Output

- PlantInput (thread)
  - Will continuously check register values every 10-20 minutes (TBD) and hold onto the data they contain
  - Monitors:
    - Moisture Sensor
    - Thermometer
    - Solar Panels
    - Water Level Sensor
- PlantOutput
  - Will write to the registers to indicate that an action should be taken.
  - Commands:
    - Alarm – can set the alarm off
    - Sprinkler – can spray water from the sprinkler
    - Water Pump – can water the plant
- NavInput (thread)
  - Will continuously check register values approximately every 600 ms (TBD) and hold onto the data they contain.
  - Monitors:
    - Sonars
    - GPS
    - IR Sensor
    - Accelerometer
    - Compass
- NavOutput
  - Will write to the registers to indicate that an action should be taken.
  - Commands:
    - Servos – rotates the sonars mounted on the servos
- MovementOutput
  - Will write to the registers to indicate that an action should be taken.
  - Commands:
    - Motor1 – rotates the wheel controlled by the motor
    - Motor2 – rotates the wheel controlled by the motor
- EmergencyInput (thread)
  - Will continuously check register values approximately every 50 ms (TBD). If the data indicates that an action needs to be taken, it will call the corresponding method to take that action in either BasicCare or CalculateMove.
  - Monitors:
    - Stop Buttons – calls emergencyStop() in CalculateMove
    - Bump Sensors – calls emergencyStop() in CalculateMove
    - Indiana Jones Switch – calls triggerAngry() in BasicCare
    - Motors – calls emergencyStop() in CalculateMove when jammed.

Sequence Diagram for Plant Care



Sequence Diagram for Navigation



Sequence Diagram for Triggering the Angry Personality