

# AP2DX

## Awesomizing the P2DX

Jasper Timmer, Maarten Inja, Maarten de Waard, Wadie Assal

UvA

June 30, 2011

## 1 Introduction

## 2 Architecture

- Framework
- Movement and collision detection
- Creating a map
- Communication

## 3 Developing process

## 4 Discussion

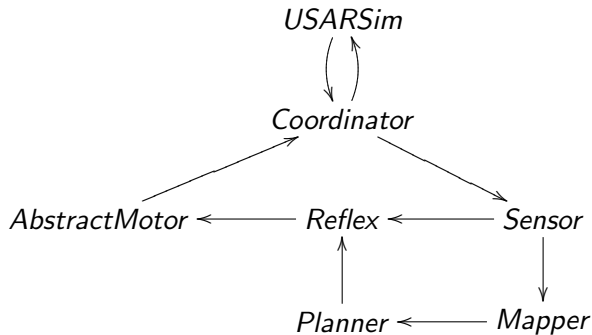
# Introduction

- The same assignment
- Who we are
- What is special in our case

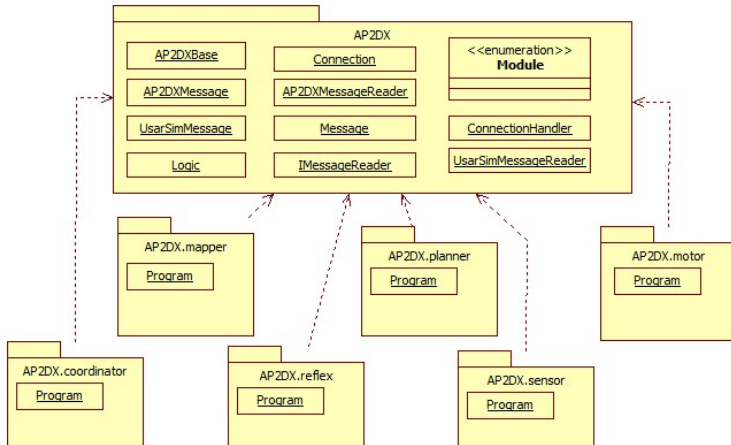
# Goals

- Loosely coupled modules based on network communication
- Robot should be safe, i.e. stop for obstacles
- Robot should be able to drive autonomously through the environment
- Robot should be able to create a map of the environment
- No user input will be required

# Architecture - the basics



# Architecture - into the depths



# Abstract base class

We decided to use an abstract class, to base all our classes on.

## Advantages

- Very easy to work with
- Only have to make the connection protocol once
- Strict contracts with team mates

## Disadvantages

- Stuck with one language
- Hard debugging
- Hard to make big changes, or add things we had not thought of

# Movement - how we do it

- Based on Sonar sensor data
- Not using the outer two sonars ( $-90^\circ$  and  $90^\circ$ )
  - Only use them for detecting “gaps in the wall”, places we may want to go
  - “hey, this time the wall is  $> 1$  meter further away in one tick, maybe there is a hole!”
  - Turn to the hole, scan the width, and decide to drive through or not.

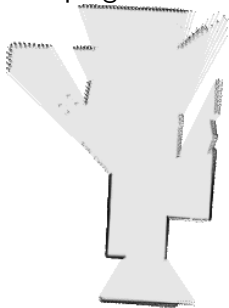


# Movement - reflexes

- If reflex stops the robot, because of an object in the way, turn to the direction with the furthest view angle
- If all four middle sonar sensors detect  $> 0.5\text{m}$ , we can drive forward.
  - The bot's width is 38cm, angle between outer-middle 2 sonars (sensor 3 and 6) = 60 degrees, lets say the hole needs to be 45cm wide to drive through it, those two sensors need both to see at least 45cm. (cosine rule)

# Mapper

We did not make our own mapper. We used DP Slam<sup>12</sup>.  
The program works in C, and creates a map like this one:



---

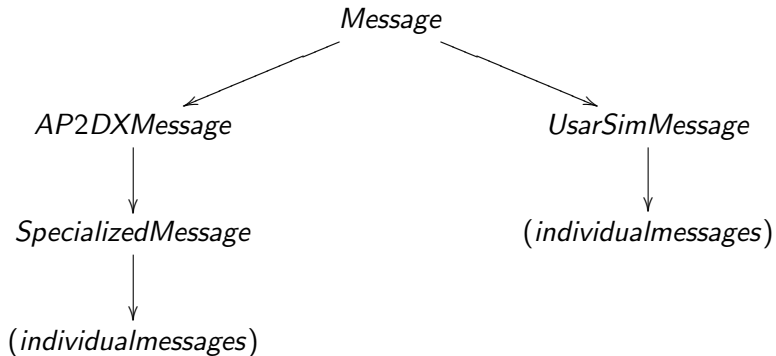
<sup>1</sup><http://www.cs.duke.edu/~parr/dpslam/>

<sup>2</sup>Algorithm from: Austin Eliazar, Ronald Parr: DP-SLAM: Fast, Robust Simultaneous Localization and Mapping Without Predetermined Landmarks

# Mapper - What makes it special

- Two ways to use a mapper:
  - While driving
  - After driving (with saved sensordata)
- We make a map, while driving
- Mapper uses Odometry and Laser range scanner data
- Currently only works on linux

# Messages



# Messages - explained

There are a couple of advantages and disadvantages:

## Advantages

- Very easy to work with
- Easy to add a new kind of message
- Strict restrictions to how a message should look (and thus uniformity)

## Disadvantages

- Very hard to debug
- Hard to add a type of message that doesn't fit in

# Developing process - 1

## Test driven programming

- Unit test in front
  - Smallest testable part of code
  - Every dependency is mocked
- A lot of overhead for small project

## File management

We used Git, to easily synchronize our code. There are some advantages:

- Distributed, so it is not like subversion
- It has change tracking
- It automatically merges
- It is installed on the UvA computers.

## Developing process - 2

### Automatic building

Ant was used to build everything. This includes compiling, testing, publishing test reports, javaDoc, creating jar files.

### Continuous integration server

Jenkins was used as integration server. It uses:

- Git checkout
- Ant build
- Clarifying coverage reports

Together this makes a nice way to discover every detail about our project. Screenshots will follow.

# Jenkins

search Jasper | log out

Jenkins > AP2DX

DISABLE AUTO REFRESH

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Workspace](#)

[Build Now](#)

[Delete Project](#)

[Configure](#)

[GitHub](#)

[Coverage Report](#)

[Build History](#) (trend)

- #236 Jun 29, 2011 7:01:17 PM
- #235 Jun 29, 2011 6:01:17 PM
- #234 Jun 29, 2011 5:01:17 PM
- #233 Jun 29, 2011 4:01:17 PM
- #232 Jun 29, 2011 3:01:17 PM
- #231 Jun 28, 2011 7:00:30 PM
- #230 Jun 28, 2011 6:00:30 PM
- #229 Jun 28, 2011 5:00:30 PM
- #228 Jun 28, 2011 4:00:30 PM
- #227 Jun 28, 2011 3:00:30 PM
- #52 Jun 9, 2011 10:00:33 AM

[RSS for all](#) [RSS for failures](#)

## Project AP2DX

Super Awesome! It's P2DX! :D Nice!!!!



[Coverage Report](#)



[Workspace](#)



[Recent Changes](#)



[Latest Test Result](#) (10 failures / #0)

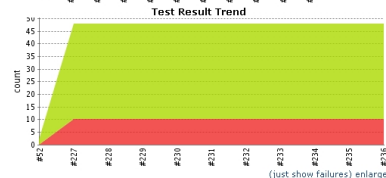
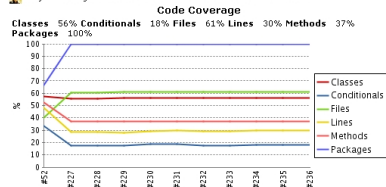
### Permalinks

- Last build (#236), 4 hr 25 min ago
- Last stable build (#52), 20 days ago
- Last successful build (#236), 4 hr 25 min ago
- Last unstable build (#236), 4 hr 25 min ago
- Last unsuccessful build (#236), 4 hr 25 min ago

[edit description](#)

[Disable Project](#)

Project managers never ask Chuck Norris for estimations... ever.





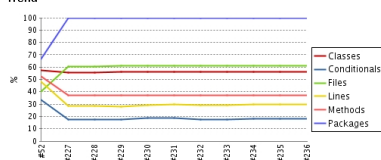
Jenkins > AP2DX > #236

- [Back to Project](#)
- [Status](#)
- [Changes](#)
- [Console Output](#)
- [Edit Build Information](#)
- [Git Build Data](#)
- [Coverage Report](#)
- [Test Result](#)
- [Previous Build](#)

## Code Coverage

### Cobertura Coverage Report

#### Trend



#### Project Coverage summary

Name	Classes	Conditionals	Files	Lines	Methods	Packages
Cobertura Coverage Report	56% <span>36/64</span>	18% <span>54/295</span>	61% <span>30/49</span>	30% <span>535/1810</span>	37% <span>145/389</span>	100% <span>10/10</span>

#### Coverage Breakdown by Package

Name	Classes	Conditionals	Files	Lines	Methods
AP2DX.specializedMessages	31% <span>5/16</span>	5% <span>1/20</span>	27% <span>4/15</span>	23% <span>66/287</span>	24% <span>20/85</span>
AP2DX.usarsim	100% <span>2/2</span>	69% <span>11/16</span>	100% <span>2/2</span>	69% <span>34/49</span>	78% <span>7/9</span>
AP2DX.coordinator	40% <span>2/5</span>	14% <span>2/14</span>	33% <span>1/3</span>	27% <span>29/106</span>	50% <span>5/10</span>
AP2DX.reflex	100% <span>2/2</span>	7% <span>2/27</span>	100% <span>1/1</span>	23% <span>14/62</span>	38% <span>5/13</span>
AP2DX.sensor	67% <span>2/3</span>	4% <span>1/23</span>	50% <span>1/2</span>	13% <span>19/149</span>	27% <span>6/22</span>
AP2DX.mapper	60% <span>3/5</span>	0% <span>0/15</span>	100% <span>3/3</span>	28% <span>34/121</span>	26% <span>7/27</span>
AP2DX	64% <span>9/14</span>	14% <span>10/69</span>	100% <span>8/8</span>	35% <span>168/478</span>	51% <span>41/81</span>
AP2DX.planner	33% <span>1/3</span>	0% <span>0/27</span>	50% <span>1/2</span>	8% <span>9/112</span>	14% <span>3/21</span>
AP2DX.usarsim.specialized	67% <span>8/12</span>	37% <span>26/70</span>	67% <span>8/12</span>	38% <span>147/387</span>	40% <span>46/115</span>
AP2DX.motor	100% <span>2/2</span>	7% <span>1/14</span>	100% <span>1/1</span>	25% <span>15/59</span>	83% <span>5/6</span>



```

Maarten zegt... 12.1 Robots - Usersim Jenkins
23  /**
24   * make a new UserSinMessage
25   *
26   * @param in
27   */
28  public UserSinMessage(String in) {
29  7      super(in, Module.UNDEFINED);
30  7  }
31
32  /**
33   * make a new UserSinMessage
34   *
35   public UserSinMessage(MessageType type) {
36  4      super(type);
37  4  }
38
39  @Override
40  public Message.MessageType getMsgType() {
41  6      if (this.type == MessageType.UNKNOWN || this.type == null) {
42  2          String startPatternStr = "[A-Z]+";
43  2          Pattern startPattern = Pattern.compile(startPatternStr);
44  2          Matcher startMatcher = startPattern
45              .matcher(this.getMessageString());
46
47  2          if (startMatcher.find()) {
48  2              this.type = UserSinMessage.MessageType
49                  .getEnumByString(startMatcher.group(0));
50              } else {
51  0              this.type = null;
52              }
53          }
54
55  6      return this.type;
56  }
57
58  @Override
59  protected void parseMessage() throws Exception {
60  0      throw new Exception(
61          "Not possible on this class, try casting to a specialized message type.");
62  }
63
64  /**
65   * This method uses annotated fields to build the output of the message.
66   * field.toString() is the value of the field like so: {name value}.
67   * @throws IllegalAccessException

```

# Discussion

- Framework is handy and dynamic, due to the JSON config files. Nevertheless, if you would add an extra module, some code would have to change.
- Framework took a lot of time to set up and debug
- Messages took a lot of time to debug, because we were using JSON