AP2DX

Awesomizing the P2DX

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

UvA

June 30, 2011

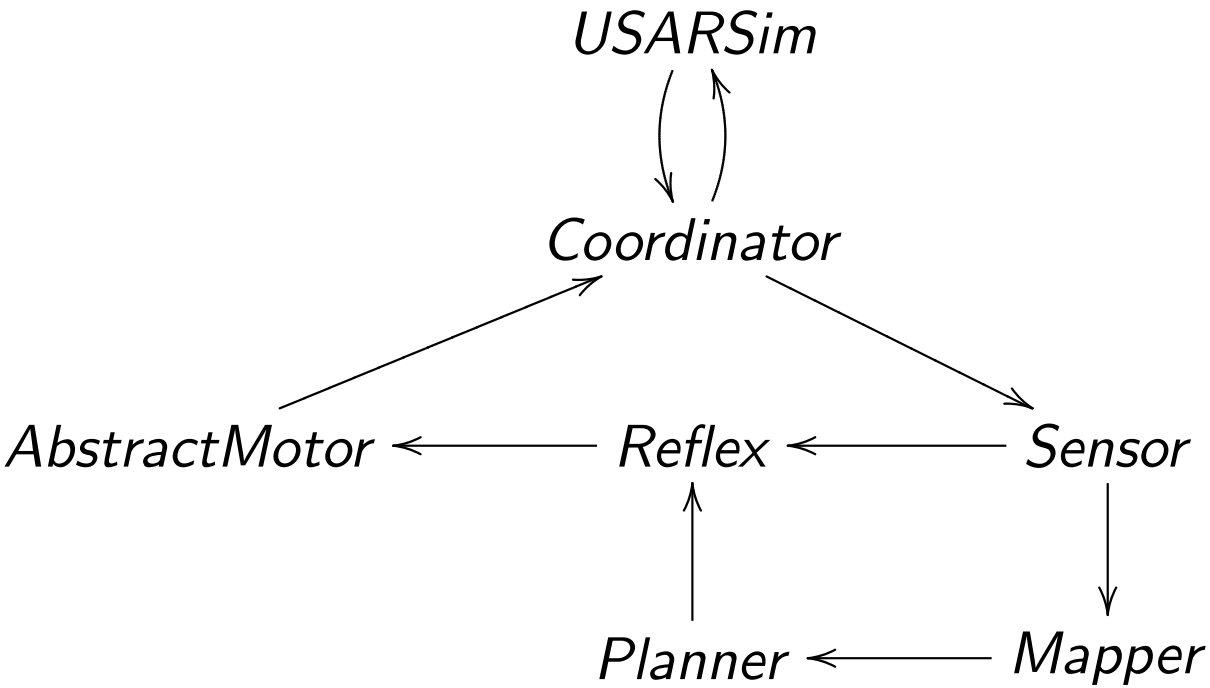
Introduction

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

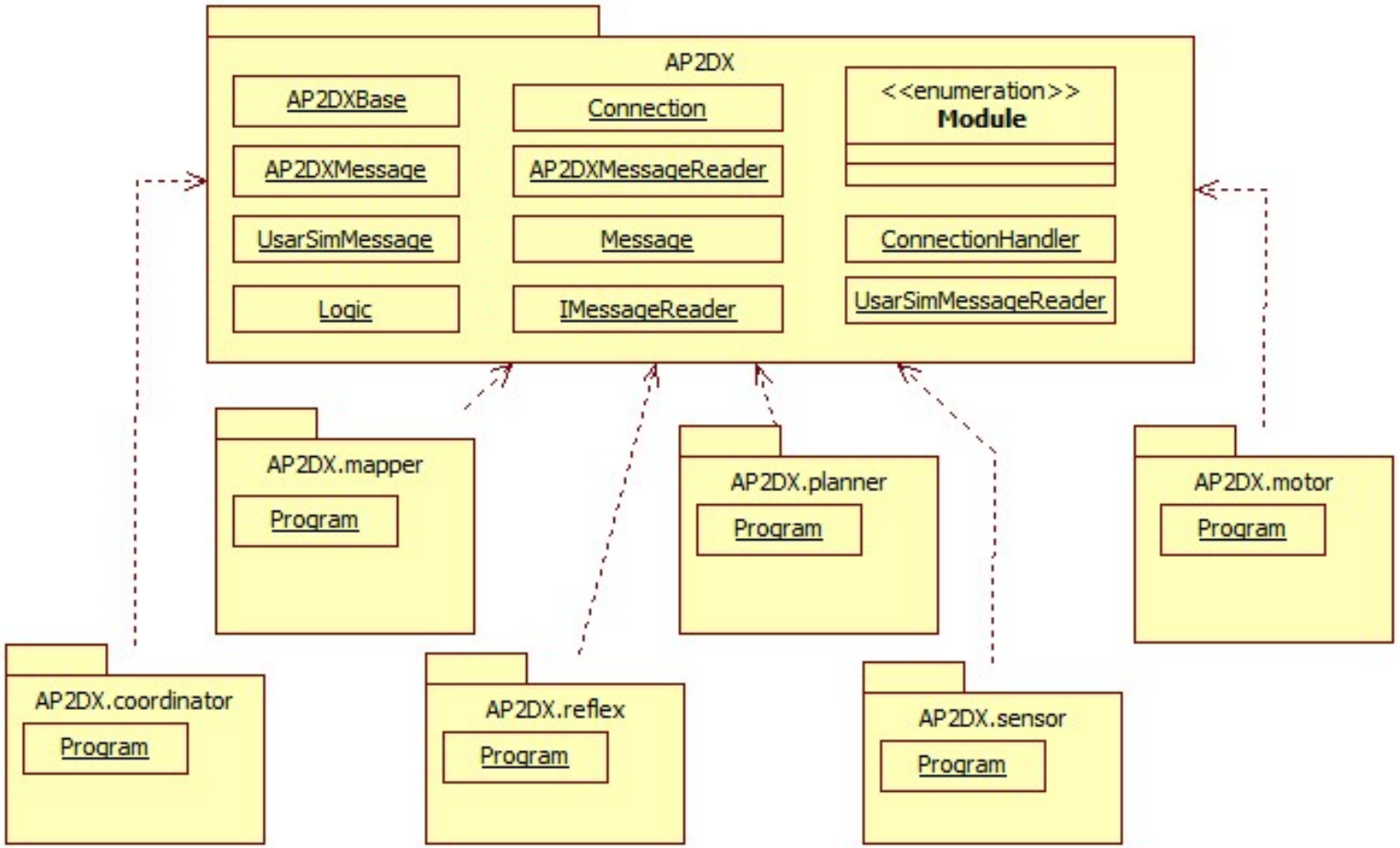
Goals

- 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

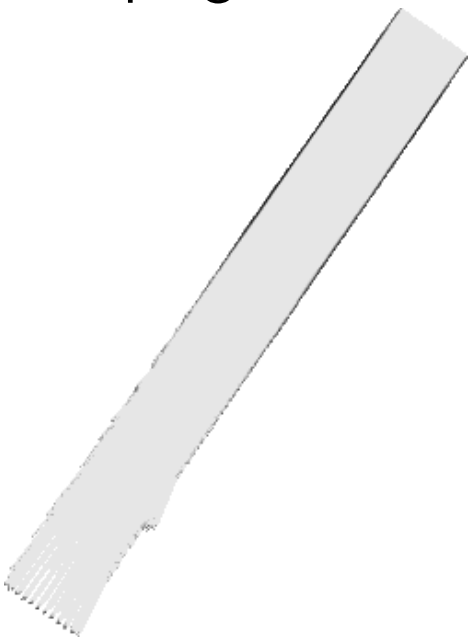
Disadvantages

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

boobs

Mapper

We did not make our own mapper. We used DP Slam[1][2].
The program works in C, and creates a map like this one:



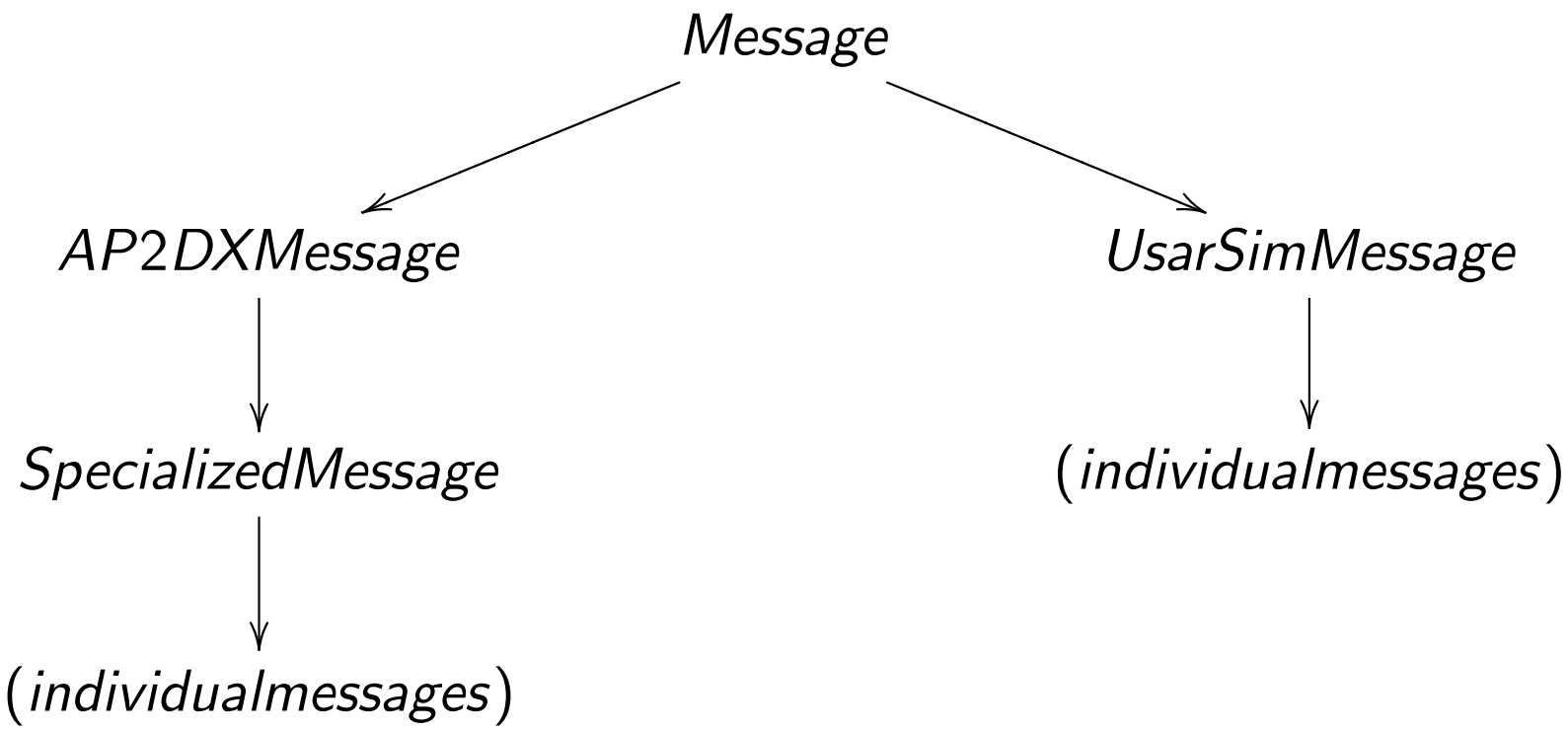
[1] http://www.cs.duke.edu/ parr/dpslam/

[2] Algorithm from: Austin Eliazar, Ronald Parr: DP-SLAM: Fast, Robust Simultainous 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


Message
AP2DXMessage
UsarSimMessage
SpecializedMessage
(individualmessages)
(individualmessages)

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.

Gmail - Alle berichten - jjwtimmer@gmai... | 12.1 Robots - Usarsim | AP2DX [Jenkins]

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!!!!

edit description

Disable Project

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

Coverage Report

Workspace

Recent Changes

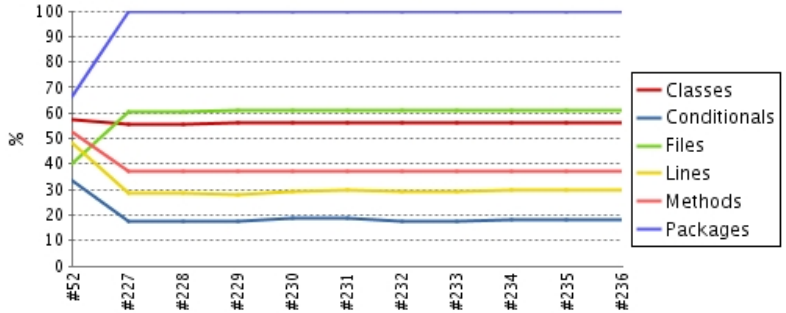
Latest Test Result (10 failures / ±0)

Permalinks

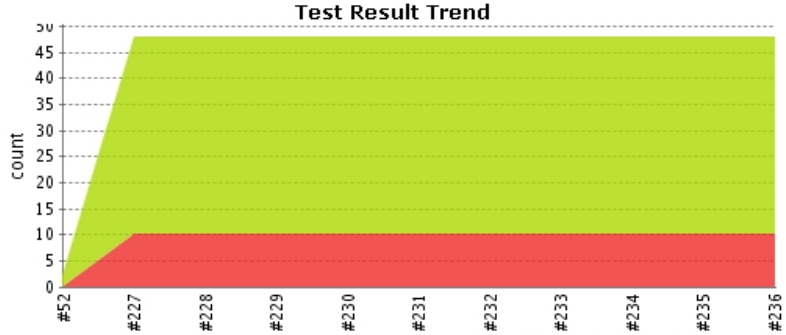
- Last build (#236), 4 hr 25 min ago
- Last stable build (#52), 20 days ago
- Last successful build (#236), 4 hr 25 min ago
- Last unstable build (#236), 4 hr 25 min ago
- Last unsuccessful build (#236), 4 hr 25 min ago

Code Coverage

Classes 56% **Conditionals** 18% **Files** 61% **Lines** 30% **Methods** 37%
Packages 100%



Test Result Trend



(just show failures) enlarge

Maarten zegt... | 12.1 Robots - Usarsim | Jenkins

Jenkins

search | Jasper | log out

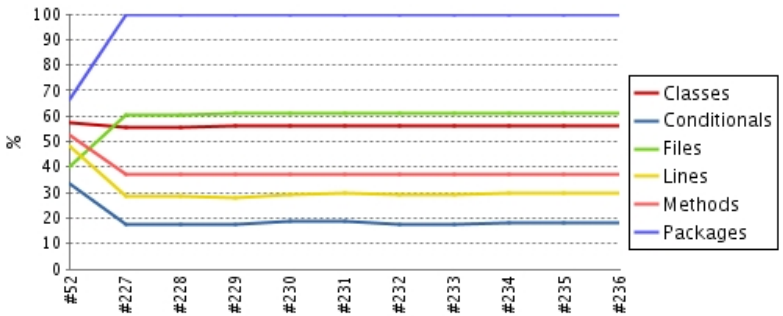
Jenkins » AP2DX » #236

- 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% 36/64	18% 54/295	61% 30/49	30% 535/1810	37% 145/389	100% 10/10

Coverage Breakdown by Package

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

Maarten zegt... | 12.1 Robots - Usarsim | Jenkins

```
24          * make a new UsarSimMessage
25          *
26          * @param in
27          */
28         public UsarSimMessage(String in) {
29  7              super(in, Module.UNDEFINED);
30  7      }
31
32         /**
33          * make a new UsarSimMessage
34          */
35         public UsarSimMessage(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 = UsarSimMessage.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, but the connections are hardcoded. This makes it harder to add an entirely new module to the set.
- bla