EXPERIENCES IN **SOFTWARE MANAGEMENT** AND **CONTINUOUS INTEGRATION** IN A **ROS**-BASED ROBOTICS PROJECT

Marc Hanheide
Lincoln Centre for Autonomous Systems

STRANDS
Spatio-Temporal Representation and Activities for Cognitive Control in Long-Term Scenarios

UNIVERSITY OF LINCOLN

STRANDS

STRANDS

Spatio-Temporal Representations and Activities for Cognitive Control in Long-Term Scenarios

UNIVERSITY OF BIRMINGHAM

KTH

RWTH AACHEN UNIVERSITY

TU WIEN

UNIVERSITY OF LEEDS

UNIVERSITY OF LINCOLN

SEVENTH FRAMEWORK PROGRAMME

Akademie für Altersforschung am Haus der Barmherzigkeit

G4S

http://strands-project.eu

Robust, intelligent, autonomous behaviour

Exploitation of structure for improved performance

Long run-times in everyday environments

Novel opportunities to learn structure environment

SEVENTH FRAMEWORK PROGRAMME

STRANDS

Objectives

Our overall objective is to enable a mobile robot to exploit a long-term understanding of space, and the activities that change it, for cognitive control in real-world environments.

O1: A unified understanding of space over time

O2: Semantic segmentation of space

O3: Understanding human activities

O4: Cognitive control of a robot's activities from spatio-temporal information

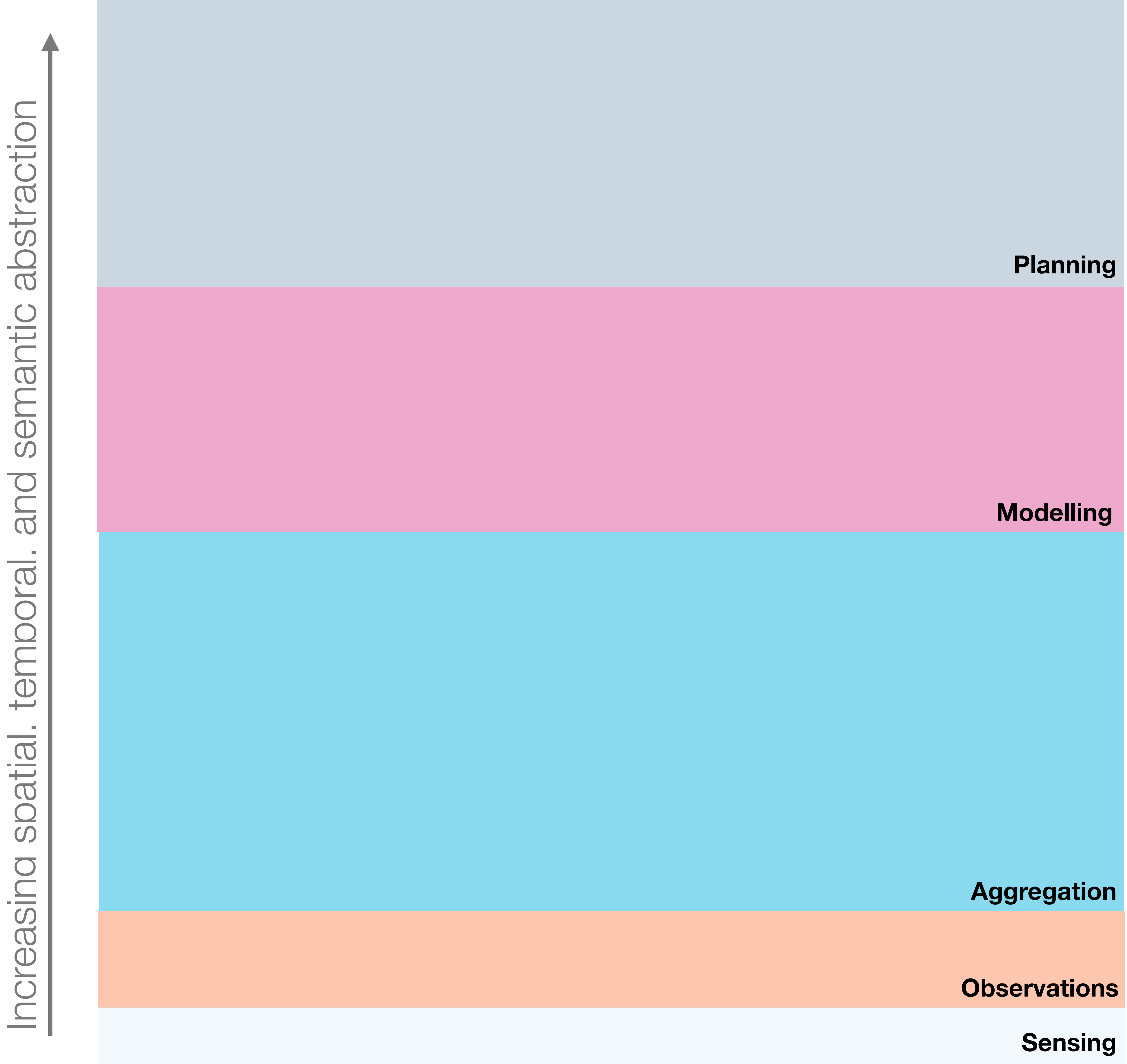
O5: Interpreting long-term experience from sparse observations

O6: Integration and validation of a long-lived cognitive robot for dynamic, real-world tasks

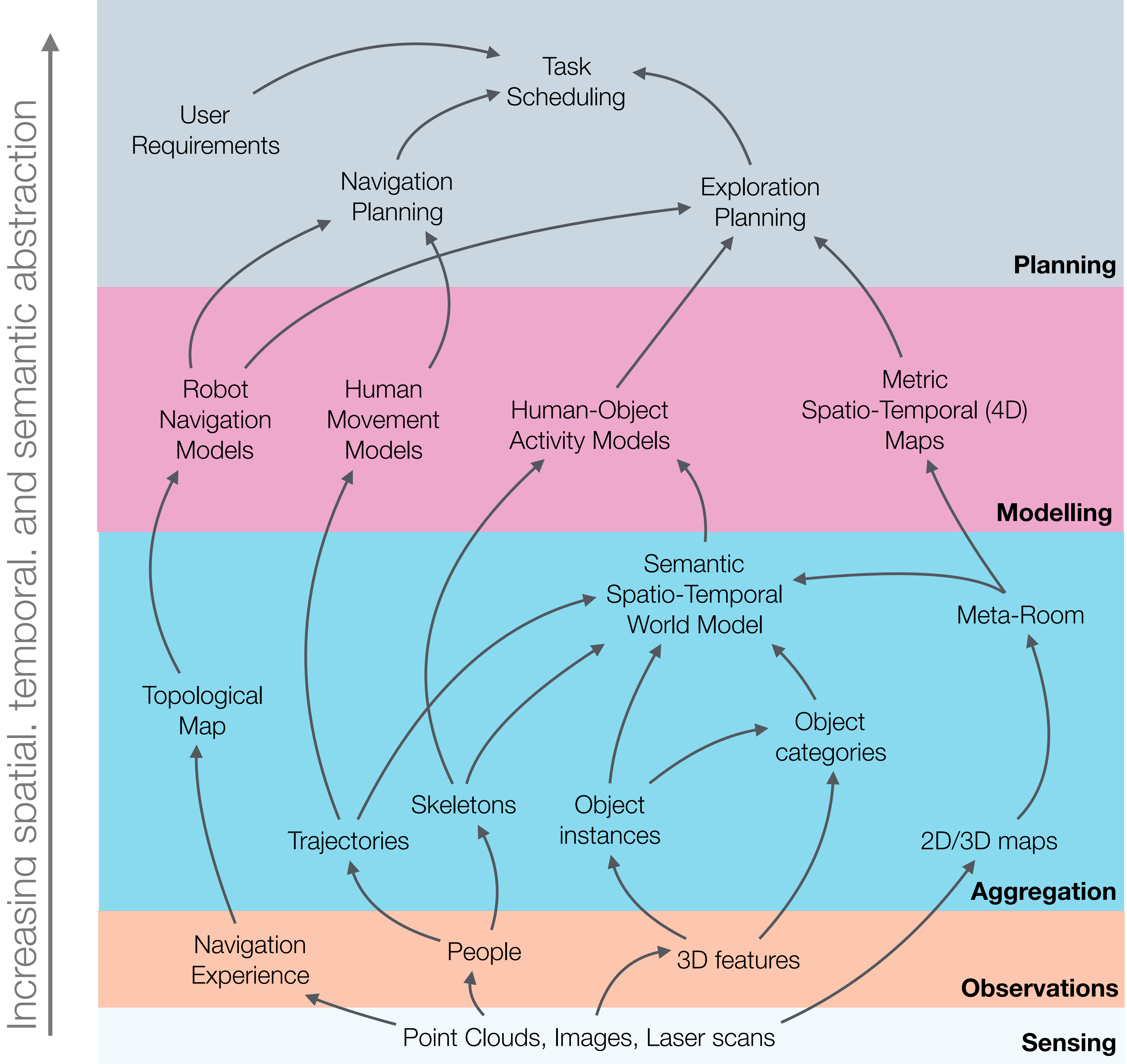
4D

SEVENTH FRAMEWORK PROGRAMME

STRANDS



Increasing spatial, temporal, and semantic abstraction
Planning
Modelling
Aggregation
Observations
Sensing



Increasing spatial, temporal, and semantic abstraction
Task Scheduling
User Requirements
Navigation Planning
Exploration Planning
Planning
Robot Navigation Models
Human Movement Models
Human-Object Activity Models
Metric Spatio-Temporal (4D) Maps
Modelling
Semantic Spatio-Temporal World Model
Meta-Room
Topological Map
Object categories
Skeletons
Object instances
Trajectories
2D/3D maps
Aggregation
Navigation Experience
People
3D features
Observations
Point Clouds, Images, Laser scans
Sensing

Year	MS	TSL	A%	Size	Tasks
3	MS8	60 days	30%	2000m^3	Security Task 2 (T4.2), Care Task 1 (T3.6), Care Task 2 (T6.2)

This milestone will see the addition of **attention and motivation mechanisms** for the robot **based on variations from predictable temporal and spatial structure** in the previously built representations, detecting such variations during patrols and autonomously examining them. In addition to this, **object and person tracking will be used to allow the system to track objects as they are manipulated by humans** and **learn the categories of objects that people regularly interact** with. Navigation will be influenced by the **predicted dynamics of the environment**, allowing the robot to **reduce travel times** by a significant amount of time and **guide humans appropriately** (Care Task 2). In Security Task 2. **arrangements of furniture will be detected through a comparison with existing spatial models**, and **basic activity models will be used to predict, and then verify, the movement of people in the robot's environment**.

SEVENTH FRAMEWORK PROGRAMME

STRANDS

Betty at
Transport Systems Catapult,
Milton Keynes, UK

Henry at
Haus der Barmherzigkeit,
Vienna, Austria

SEVENTH FRAMEWORK PROGRAMME

STRANDS

HENRY AT THE CARE HOME

Info-Terminal

Improve when and where to offer

STRANDS

Bellbot

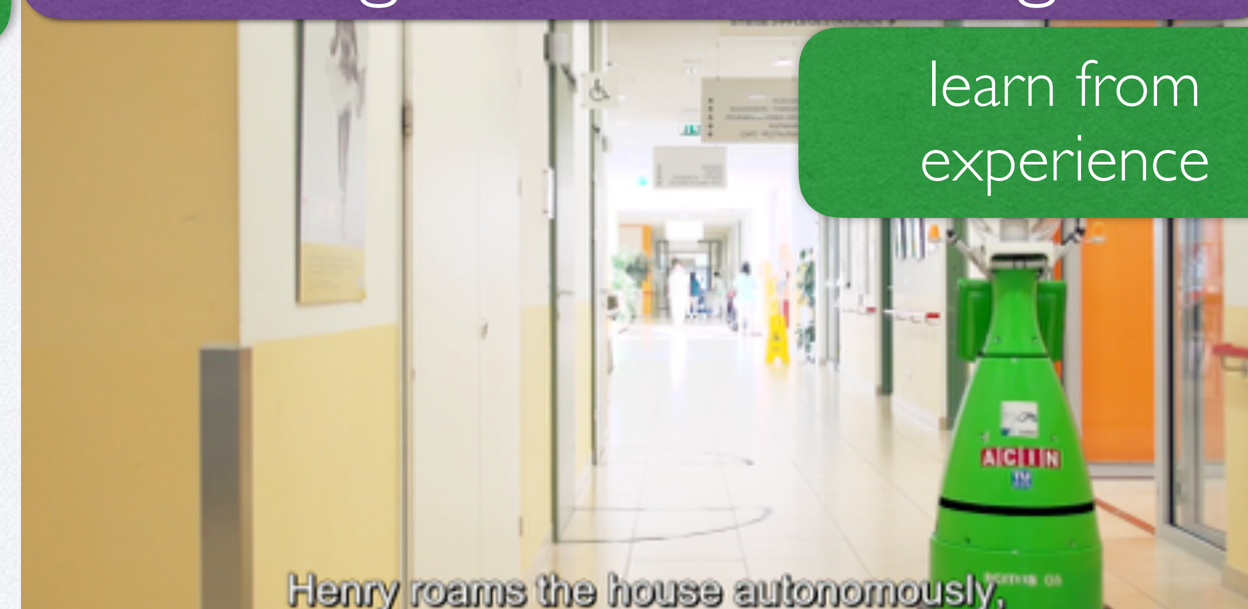
interaction with visitors

Walking Group

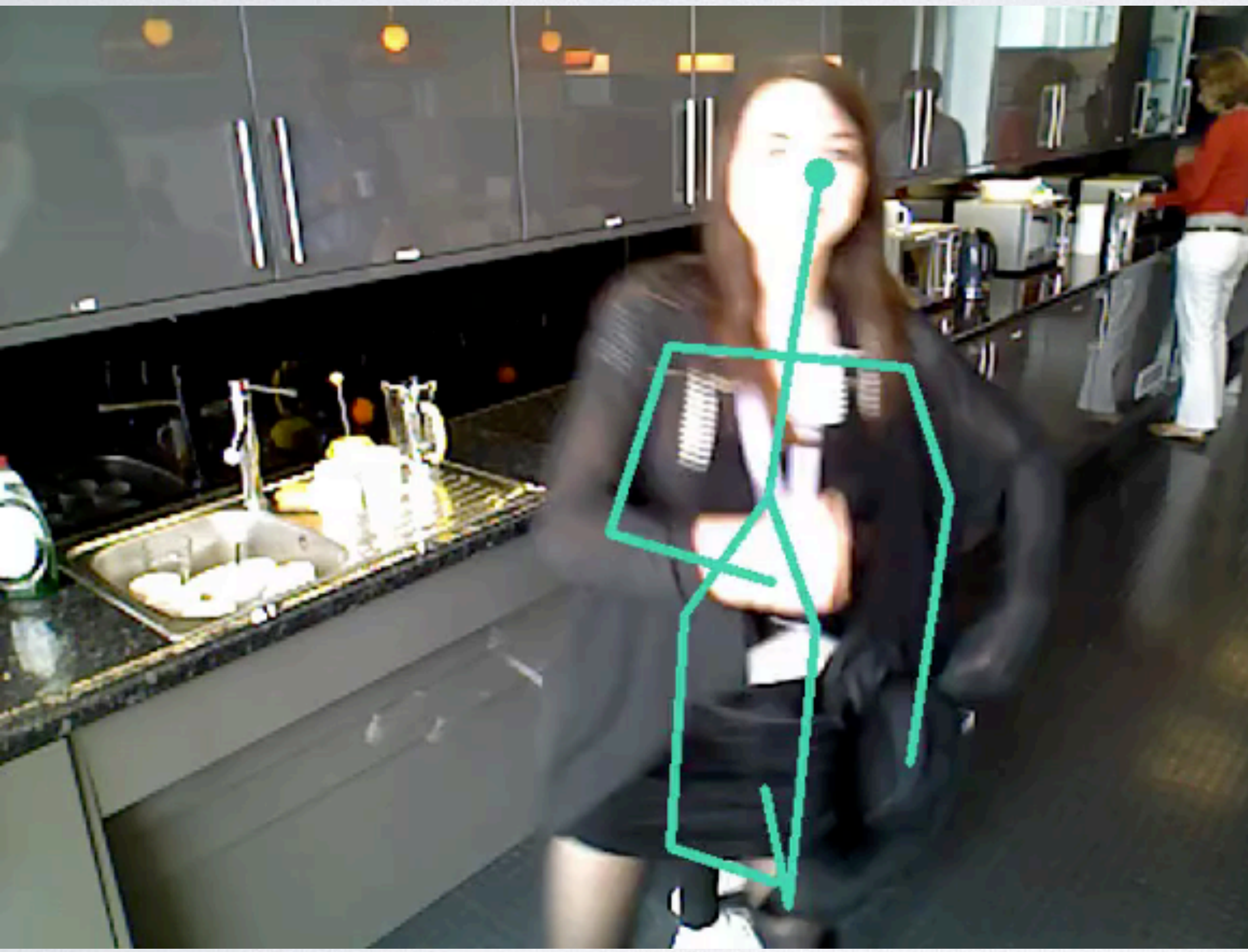
occupational therapy

Navigation is a challenge

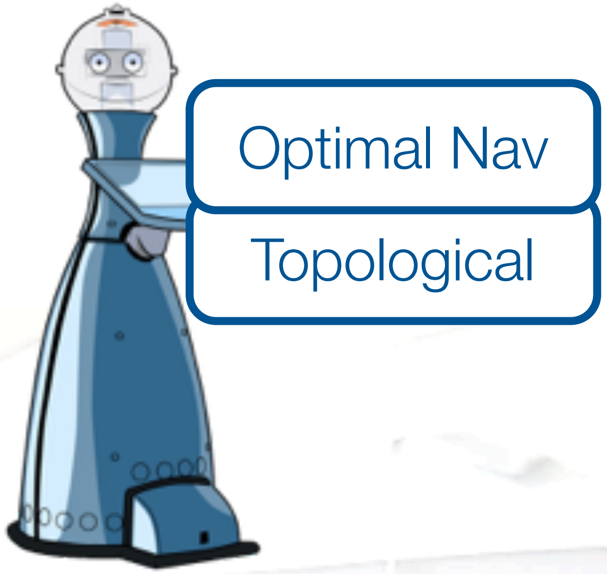
learn from experience



PEOPLE LOVE ROBOTS



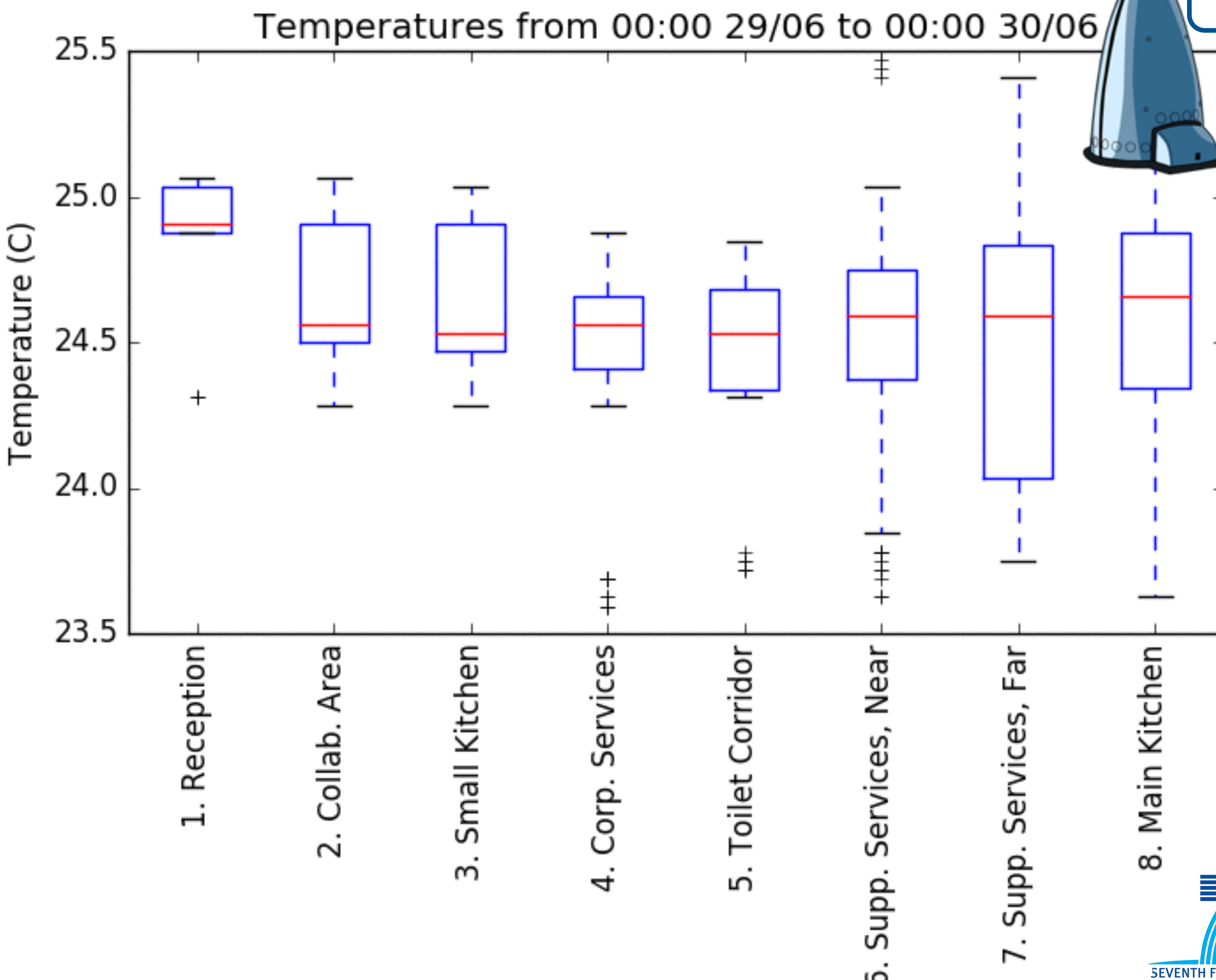
People are helpful to robots



Temperature Measurement

MongoDB

Topological



SEVENTH FRAMEWORK PROGRAMME

STRANDS

Surface/Worker Checking

SOMa

Meta-Rooms

Person Detect

Object Rec.

SEVENTH FRAMEWORK PROGRAMME

STRANDS

Autonomous Object Learning

SOMa

MongoDB

3D Model

Question

Please provide label(s) for the object you see. Below is a tic suggestions from the robot. Click the labels to add them to the list. Use th d more labels. Press the submit button in the end.

If the 3D model and images do not make any sense to you, just press the "It's junk!" button.

Suggested labels:

Pencil_sharpener Desk Eraser Notebook Hole_punch Paper_size Ring_binder Post-it_note Carbon_paper Liquid_Paper Filing_cabinet Pencil Staple_(fastener) Lazy_Susan Paper_clip

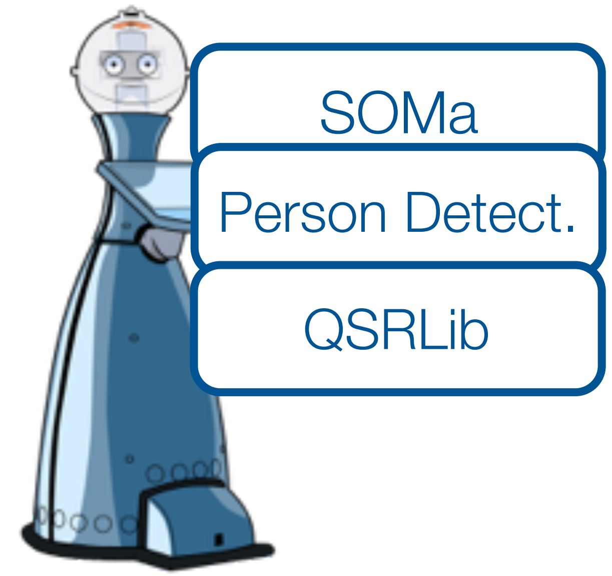
Images

Labels

Medkit ADD

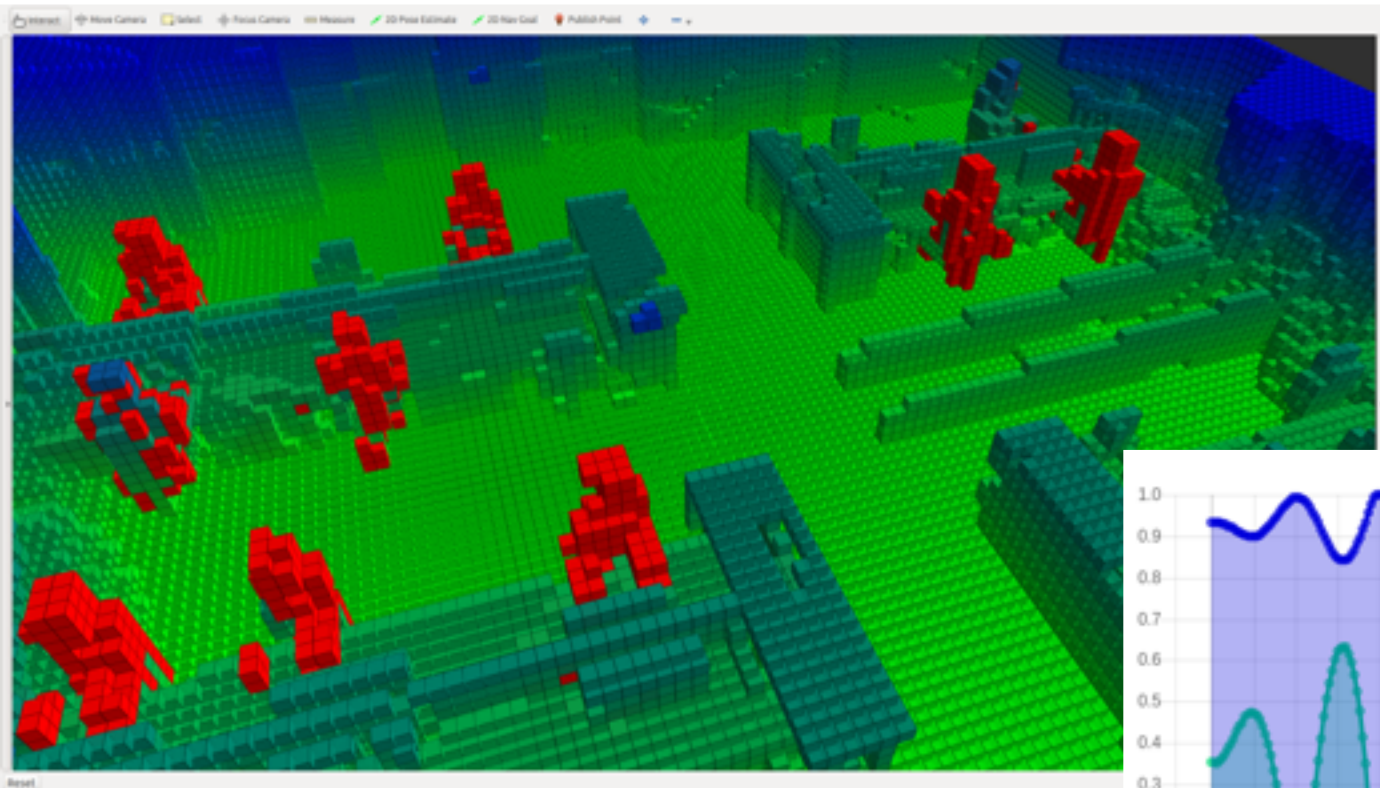
IT'S JUNK! SUBMIT

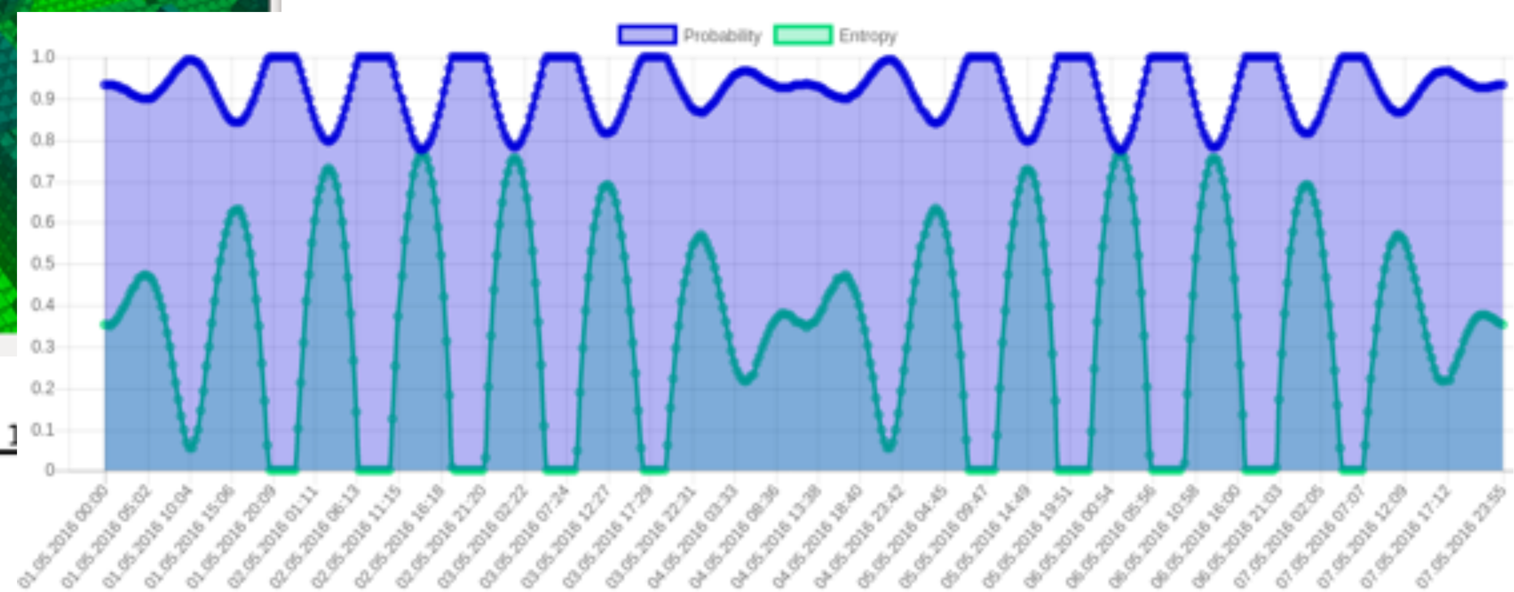
Activity Recording

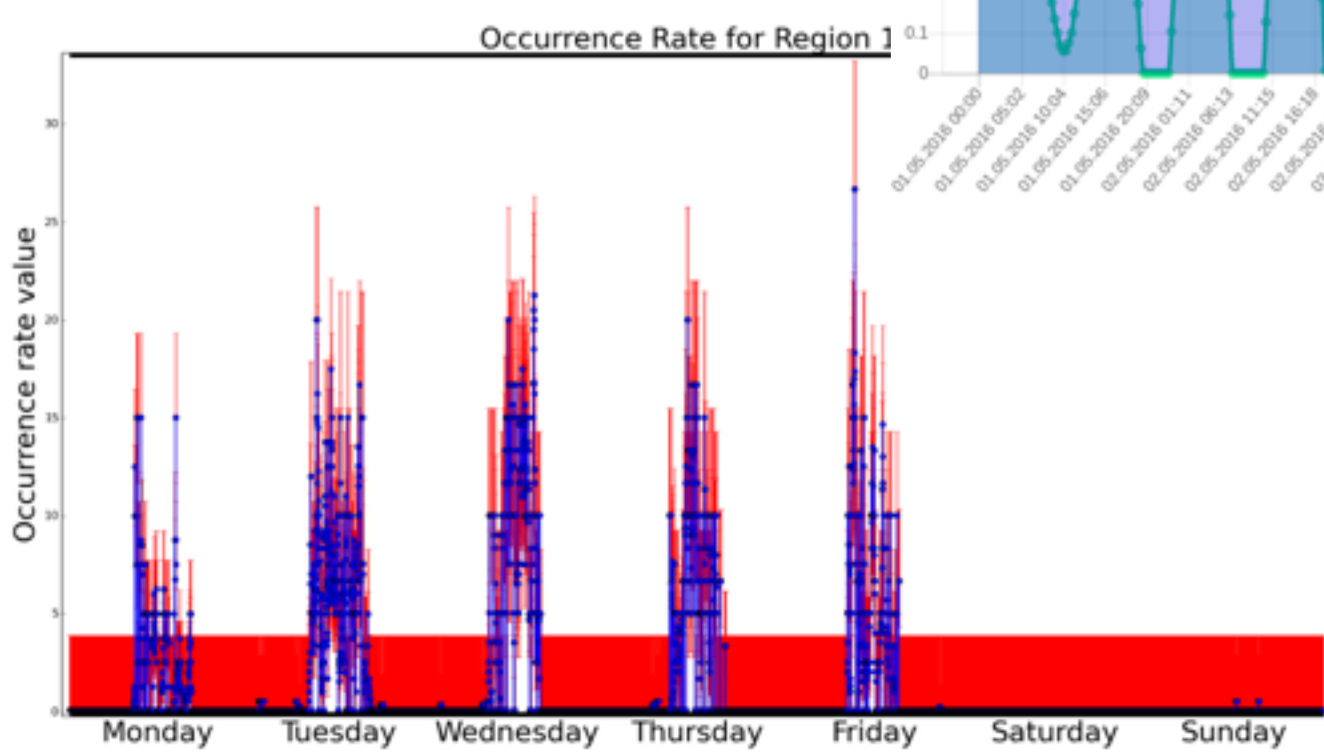


Exploration

SOMa
MongoDB
FreMEn
Exploration
Scheduler







	Security	Care Y3	Care Y2
Deployment	31/5/16 to ? 31/7/16	21/3/16 to 27/5/16	18/5/15 to 17/6/15
Working Hours	Weekdays, 6.00 to 17.45	Weekdays days 7.00 to 19.00	Most days 8.00 to 21.00
Distance		~50km	23.41km
Tasks		1890	865
Available Work Time		529 hours, 13 minutes	252 hours, 54 minutes
Autonomous Time		209 hours, 13 minutes	135 hours, 20 minutes
A%		39.53%	53.51%
	Total System Lifetime (TSL)		
Max		25 days, 11:29 hours (includes 8 days off)	15 days, 5:33 hrs (includes 5 days off)
2nd best		15 days, 9:30 hours (includes 4 days off)	
Cumulative		55 days, 9:57 hours (includes 16 days off)	29 days, 5:53 hrs (includes 10 days off)

no developers/ engineers on-site

Computer Science > Robotics

The STRANDS Project: Long-Term Autonomy in Everyday Environments

Nick Hawes, Chris Burbridge, Ferdian Jovan, Lars Kunze, Bruno Lacerda, Lenka Mudrová, Jay Young, Jeremy Wyatt, Denise Hebesberger, Tobias Körtner, Rares Ambrus, Nils Bore, John Folkesson, Patric Jensfelt, Lucas Beyer, Alexander Hermans, Bastian Leibe, Aitor Aldoma, Thomas Fäulhammer, Michael Zillich, Markus Vincze, Muhannad Al-Omari, Eris Chinellato, Paul Duckworth, Yiannis Gatsoulis, David C. Hogg, Anthony G. Cohn, Christian Dondrup, Jaime Pulido Fentanes, Tomas Krajník, João M. Santos, Tom Duckett, Marc Hanheide

(Submitted on 15 Apr 2016)

Thanks to the efforts of our community, autonomous robots are becoming capable of ever more complex and impressive feats. There is also an increasing demand for, perhaps even an expectation of, autonomous capabilities from end-users. However, much research into autonomous robots rarely makes it past the stage of a demonstration or experimental system in a controlled environment. If we don't confront the challenges presented by the complexity and dynamics of real end-user environments, we run the risk of our research becoming irrelevant or ignored by the industries who will ultimately drive its uptake. In the STRANDS project we are tackling this challenge head-on. We are creating novel autonomous systems, integrating state-of-the-art research in artificial intelligence and robotics into robust mobile service robots, and deploying these systems for long-term installations in security and care environments. To date, over four deployments, our robots have been operational for a combined duration of 2545 hours (or a little over 106 days), covering 116km while autonomously performing end-user defined tasks. In this article we present an overview of the motivation and approach of the STRANDS project, describe the technology we use to enable long, robust autonomous runs in challenging environments, and describe how our robots are able to use these long runs to improve their own performance through various forms of learning.

Subjects: Robotics (cs.RO)
Cite as: arXiv:1604.04384 [cs.RO]
(or arXiv:1604.04384v1 [cs.RO] for this version)

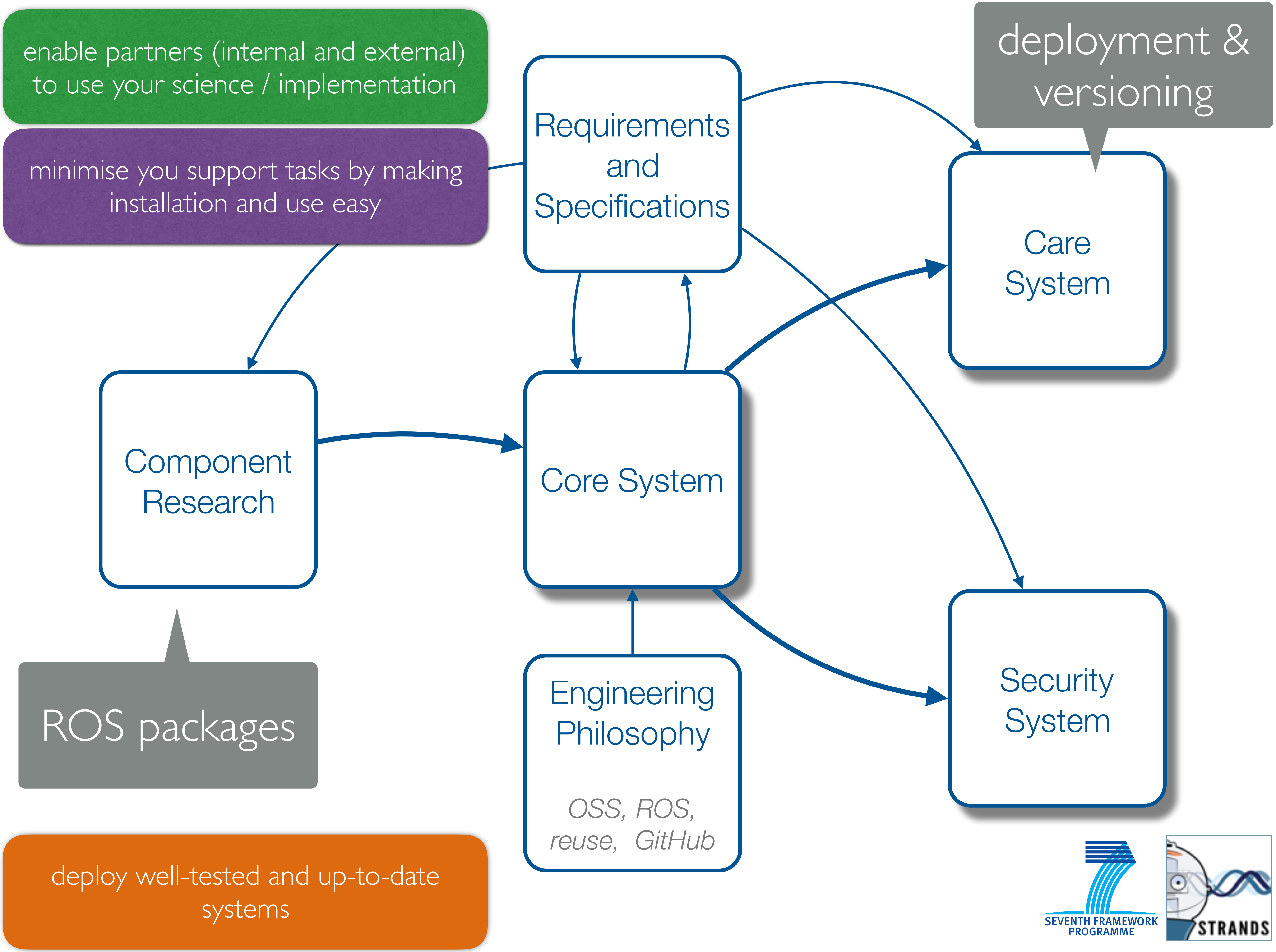
ENGINEERING SCIENCE
ENGINEERING

ENGINEERING
SCIENCE
ENGINEERING

enable partners (internal and external) to use your science / implementation

minimise you support tasks by making installation and use easy

deploy well-tested and up-to-date systems


enable partners (internal and external) to use your science / implementation
minimise you support tasks by making installation and use easy
Requirements and Specifications
deployment & versioning
Care System
Component Research
Core System
ROS packages
Engineering Philosophy
OSS, ROS, reuse, GitHub
Security System
deploy well-tested and up-to-date systems
SEVENTH FRAMEWORK PROGRAMME
STRANDS

>30 developers

>50 ROS packages

Information Terminal

Bellbot

alking Group

>150 ROS packages

Care Team **UOL**

Care System

Security Team **BHAM**

Core System

UNIVERSITY OF BIRMINGHAM

KTH

RWTH AACHEN UNIVERSITY

TU WIEN

UNIVERSITY OF LEEDS

UNIVERSITY OF LINCOLN

AKADEMIE FÜR ALTERSFORSCHUNG AM HAUS DER BARMHERZIGKEIT

G4S

Security System

Life-Long Object Learning

Activity Recording

Spatio-Temporal Exploration

Surface Checking

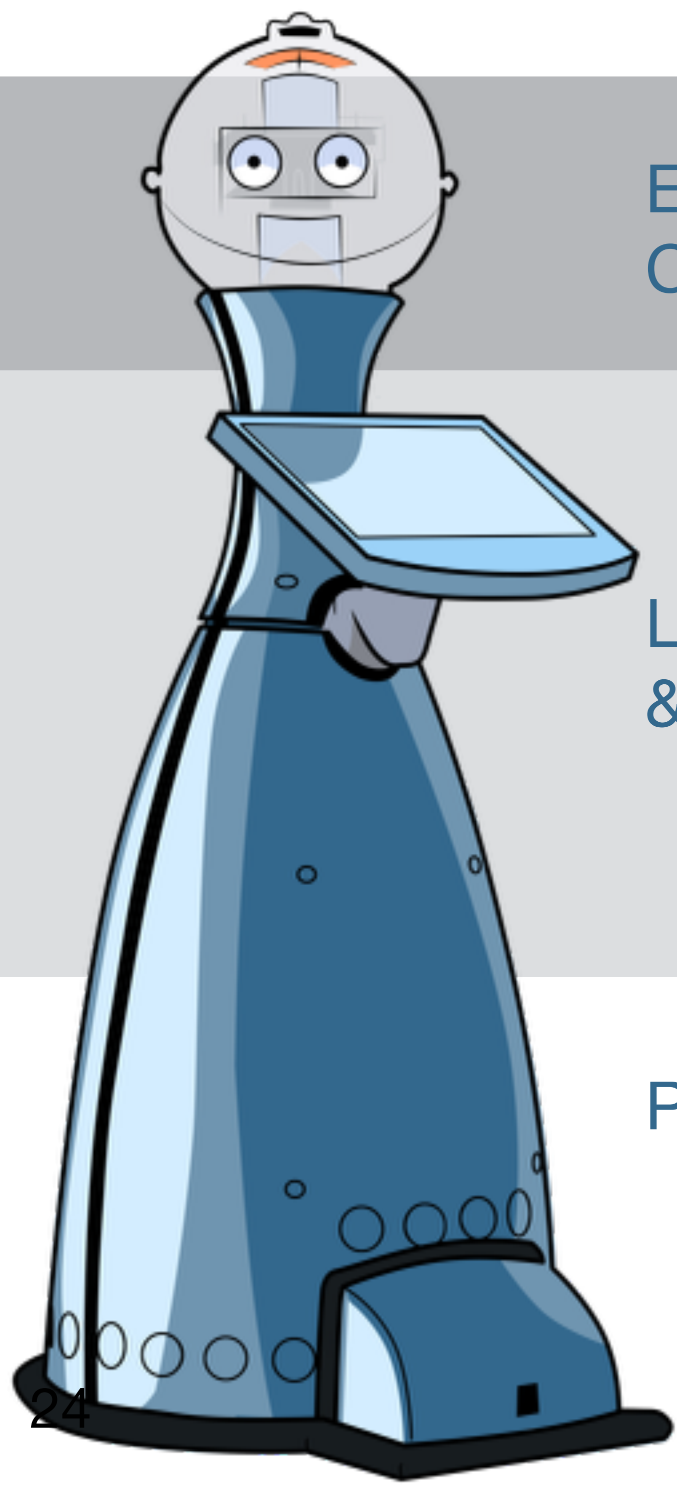
Person Checking

Temperature Measurements

Guest Greeting

SEVENTH FRAMEWORK PROGRAMME

STRANDS



Representation & Analysis

MongoDB

QSRLib

FreMEn

SOMa

View Plan.

Executive Control

Routine

Exploration

Task Executor

Scheduler

Optimal Nav

Localisation & Navigation

Monitoring

Topological

Continuous

Perception

Trajectories

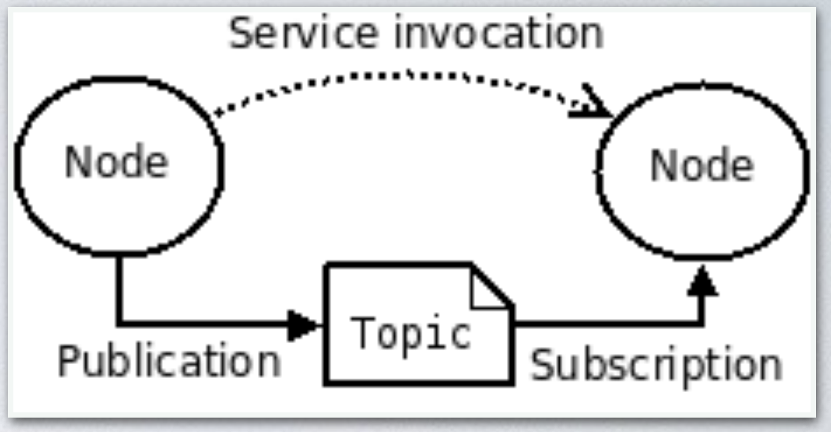
Meta-Rooms

Person Detect

Object Rec.

Object Disc.

WHAT IS ROS?



- forming a graph of peer-to-peer communicating components
- ROS is a middleware

- A component-oriented robotics framework,
- An Inter Process Communication middleware,
- A development suite,
- A (bad) package management system,
- An (active) community.

- **synchronous** RPC-stylecommunication over **services**
- **asynchronous** streaming of data over **topics**
- **storage** of data on a **Parameter Server**.

- build system (catkin) for C++ and Python based on CMake

ROS.org

- actually not that bad anymore… but now quite useful

UNIVERSITY OF LINCOLN

L-CAS

ROS lacks persistency, MongoDB employed as generic persistency layer

Repositories with ROS packages

MongoDB | QSRLib
FreMEn | SOMa | View Plan.

Routine | Exploration
Task Executor | Scheduler
Optimal Nav

Localisation & Navigation

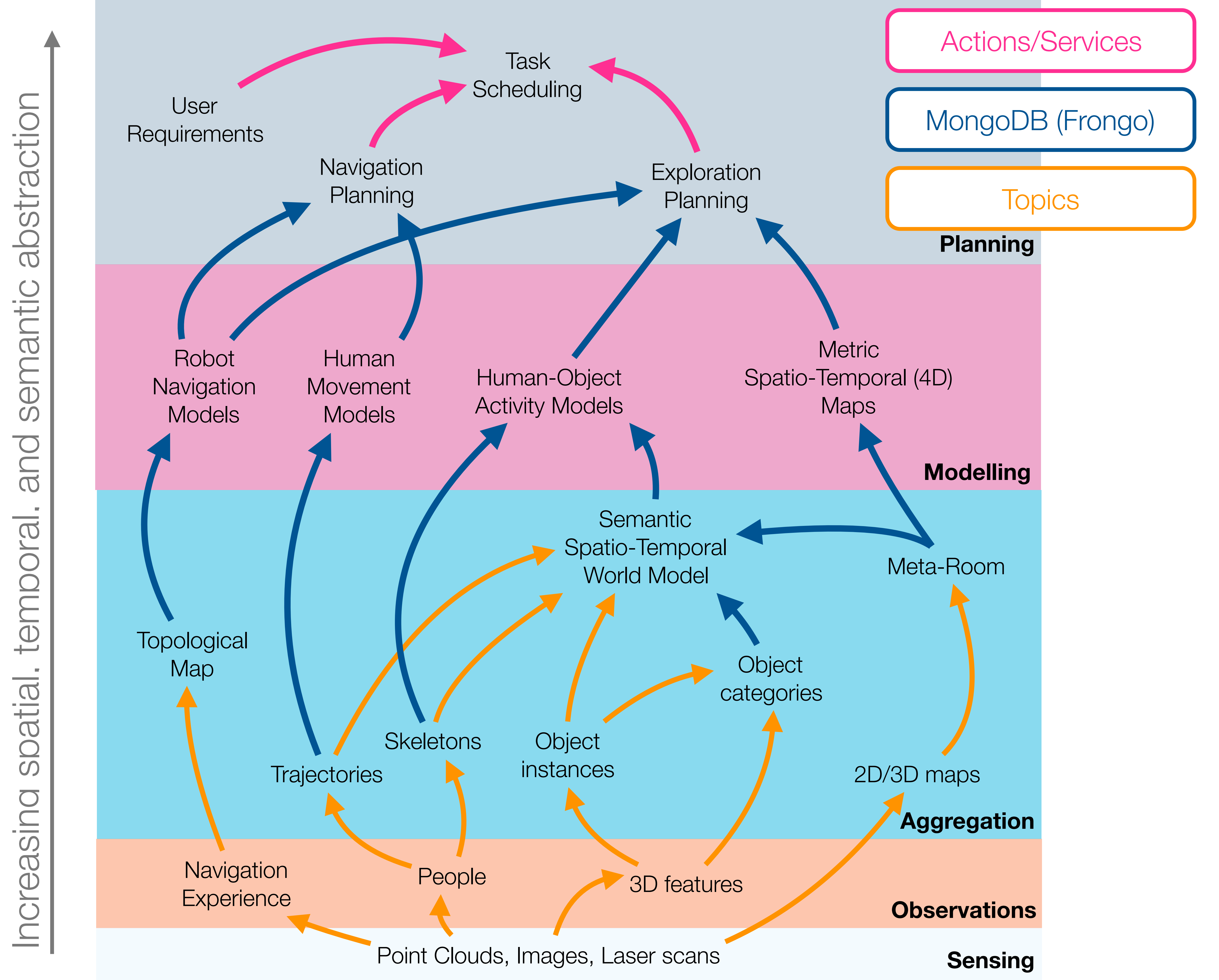
Monitoring
Topological
Continuous

Perception

Trajectories | Meta-Rooms
Person Detect | Object Rec.
Object Disc.

SEVENTH FRAMEWORK PROGRAMME

STRANDS



Actions/Services
MongoDB (Frongo)
Topics
Increasing spatial, temporal, and semantic abstraction
Task Scheduling
User Requirements
Navigation Planning
Exploration Planning
Planning
Robot Navigation Models
Human Movement Models
Human-Object Activity Models
Metric Spatio-Temporal (4D) Maps
Modelling
Semantic Spatio-Temporal World Model
Meta-Room
Topological Map
Object categories
Skeletons
Object instances
Trajectories
2D/3D maps
Aggregation
Navigation Experience
People
3D features
Observations
Point Clouds, Images, Laser scans
Sensing

GitHub

THE 10 COMMANDMENTS

(OF SUCCESSFUL INTEGRATION IN STRANDS)

THE 10 COMMANDMENTS

Code has to be packaged up as a package
(contain a package.xml)

Unit test (rostest) should be implemented

The sole officially supported OS is Ubuntu 14.04 64bit

A maintainer has to be named
(package.xml)

Code must only use other "released" code
(Debian/Ubuntu binaries)

If it uses ROS, it has to use ROS Indigo

Packages need to declare all their dependencies using rosdep keys
(in package.xml)

If it uses ROS, it needs to use catkin as a build scheme

Only code that has passed continuous integration tests is allowed to be merged
(enforced through github)

Code needs to be hosted on github.com
(normally in strands-project organisation)

THE 10 COMMANDMENTS

Code has to be packaged up as a package (contain a package.xml)

A maintainer has to be named (package.xml)

Packages need to declare all their dependencies using rosdep keys (in package.xml)

The sole officially supported OS is Ubuntu 14.04 64bit

If it uses ROS, it has to use ROS Indigo

If it uses ROS, it needs to use catkin as a build scheme

WHAT IS A ROS PACKAGE?

contains a package.xml definition

declare who's responsible!

Think about license

mostly in ROS, they use *catkin* to build

All dependencies need to be declared

```
<?xml version="1.0"?>
<package>
  <name>topological_navigation</name>
  <version>1.0.1</version>
  <description>The topological_navigation package</description

  <maintainer email="jpulidofentanes@lincoln.ac.uk">
    Jaime Pulido Fentanes
  </maintainer>

  <author>Jaime Pulido Fentanes</author>

  <license>MIT</license>

  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>rospy</build_depend>
  <build_depend>message_generation</build_depend>
  <!-- many more… -->

  <run_depend>rospy</run_depend>
  <run_depend>move_base</run_depend>
  <!-- many more… -->

  <test_depend>rosunit</test_depend>
  <test_depend>rostest</test_depend>
  <!-- many more… -->

</package>
```

THE 10 COMMANDMENTS

Code has to be packaged up as a package
(contain a package.xml)

A maintainer has to be named
(package.xml)

Packages need to declare all their dependencies using rosdep keys
(in package.xml)

The sole officially supported OS is Ubuntu 14.04 64bit

If it uses ROS, it has to use ROS Indigo

If it uses ROS, it needs to use catkin as a build scheme

THE 10 COMMANDMENTS

Code needs to be hosted on github.com (normally in strands-project organisation)

Unit test (rostest) should be implemented

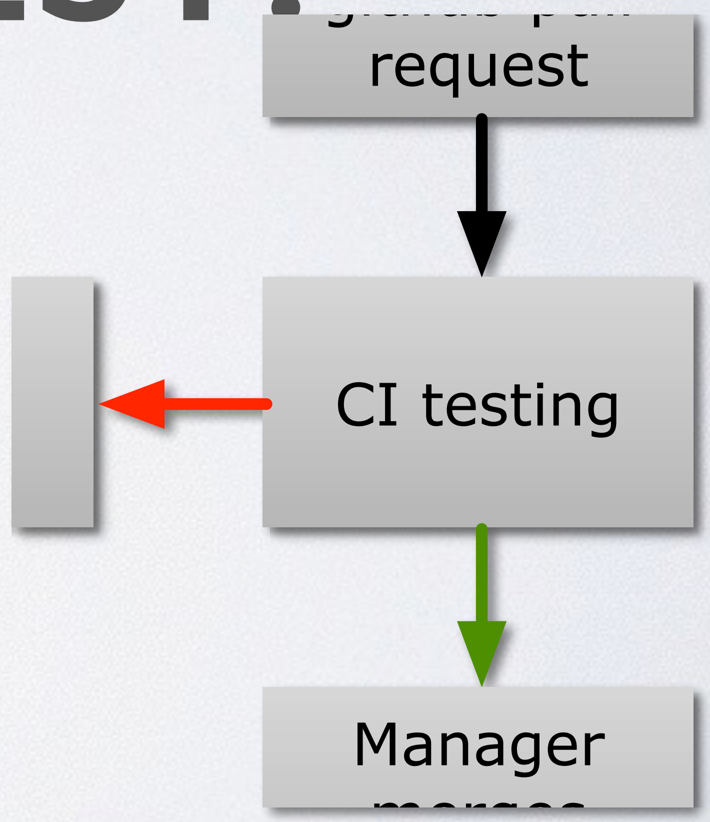
Only code that has passed continuous integration tests is allowed to be merged (enforced through github)

Code must only use other "released" code (Debian/Ubuntu binaries)

Unit test (rostest) should be implemented

TEST, TEST, TEST!

Only code that has passed continuous integration tests is allowed to be merged (enforced through github)



IT'S NOT AS EASY AS IT MAY SEEM

Fork me on GitHub

ROS

- Build on top of off-the-shelf ROS components
- long-term autonomy requires **robust software**

https://github.com/strands-project-releases/strands-releases/wiki

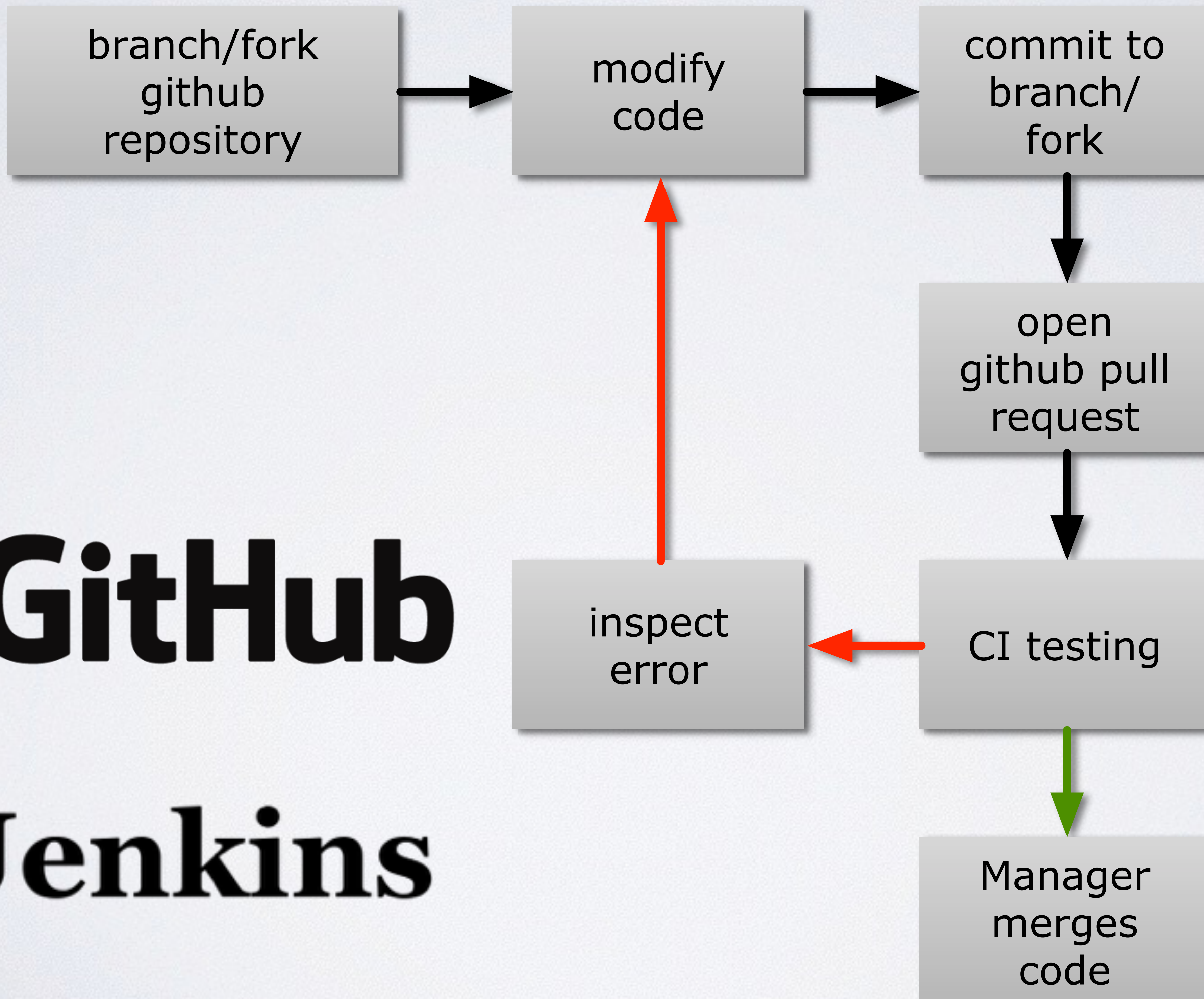
CONTINUOUS INTEGRATION

CI was intended to be used in combination with automated unit tests written through the practices of test-driven development.

Continuous integration involves integrating early and often, so as to avoid the pitfalls of "integration hell".

A complementary practice to CI is that before **submitting work**, each programmer must do a complete build and run (and pass) all **unit tests**. Integration tests are usually run automatically on a **CI server** when it detects a new commit.

THE STRANDS SOFTWARE WORKFLOW



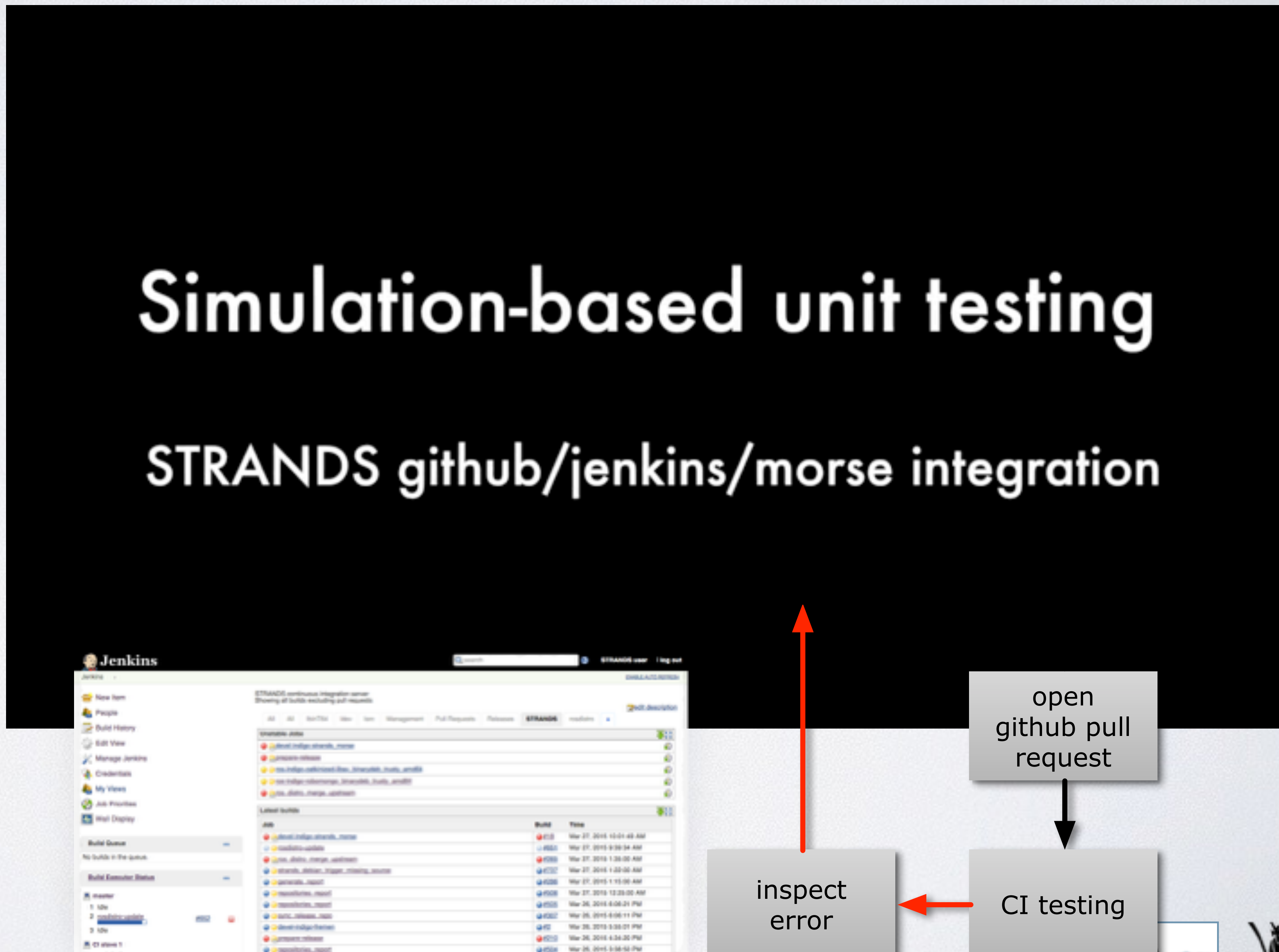
SIMULATION-BASED ROBOT TESTING

github pull request

jenkins pull request builder

launch MORSE simulation

run defined unit test & record result



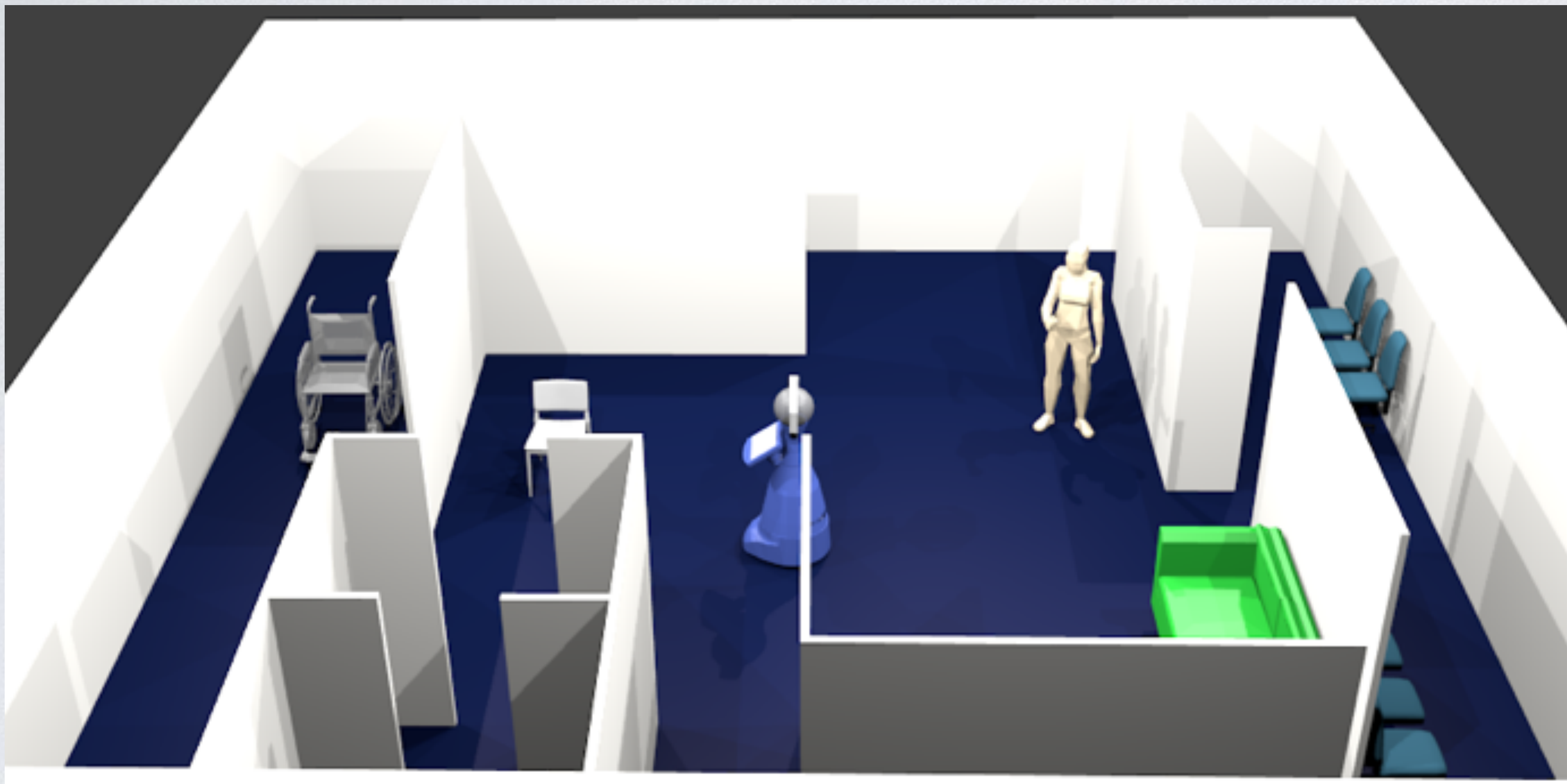
https://lcas.lincoln.ac.uk/jenkins/

UNIVERSITY OF LINCOLN

STRANDS

LINCOLN ROBOTICS

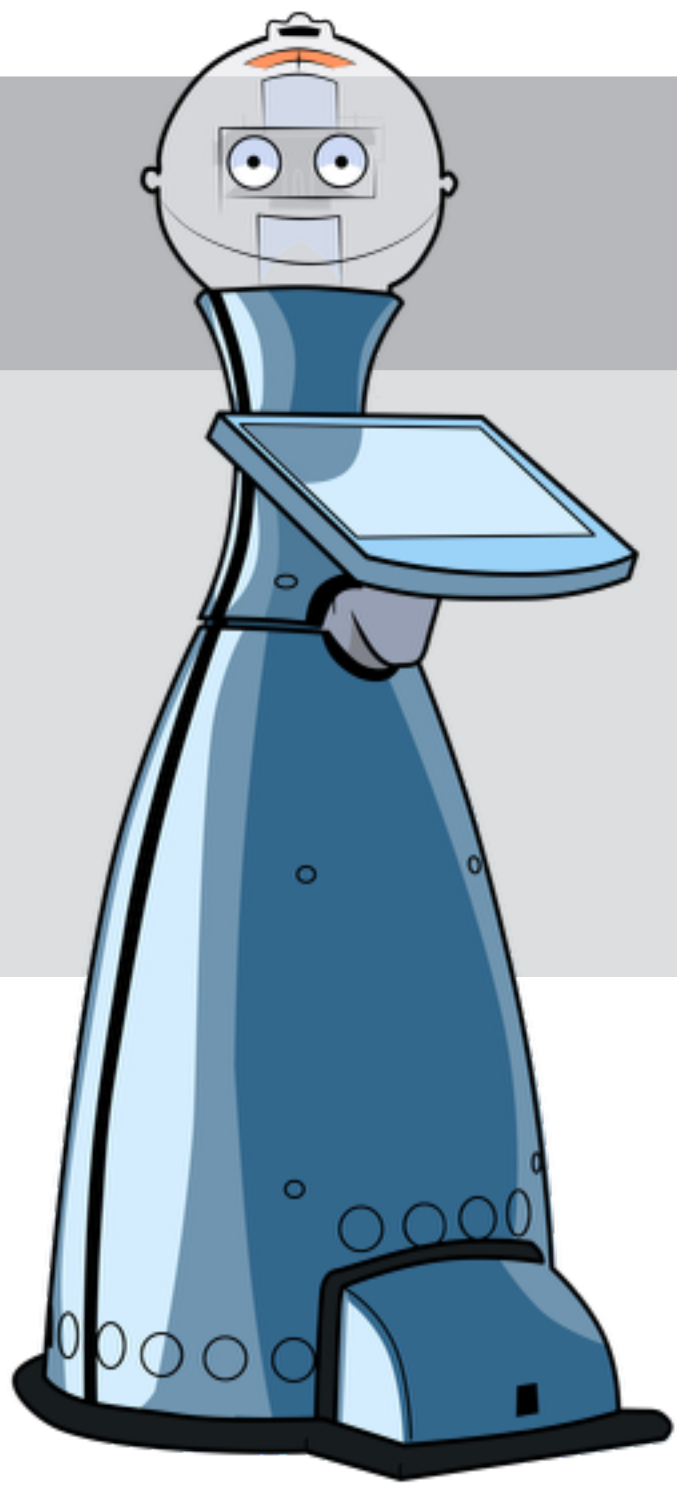
ROBOT TESTING IS ALSO ABOUT REALITY



NOT ONLY FOR STABILITY

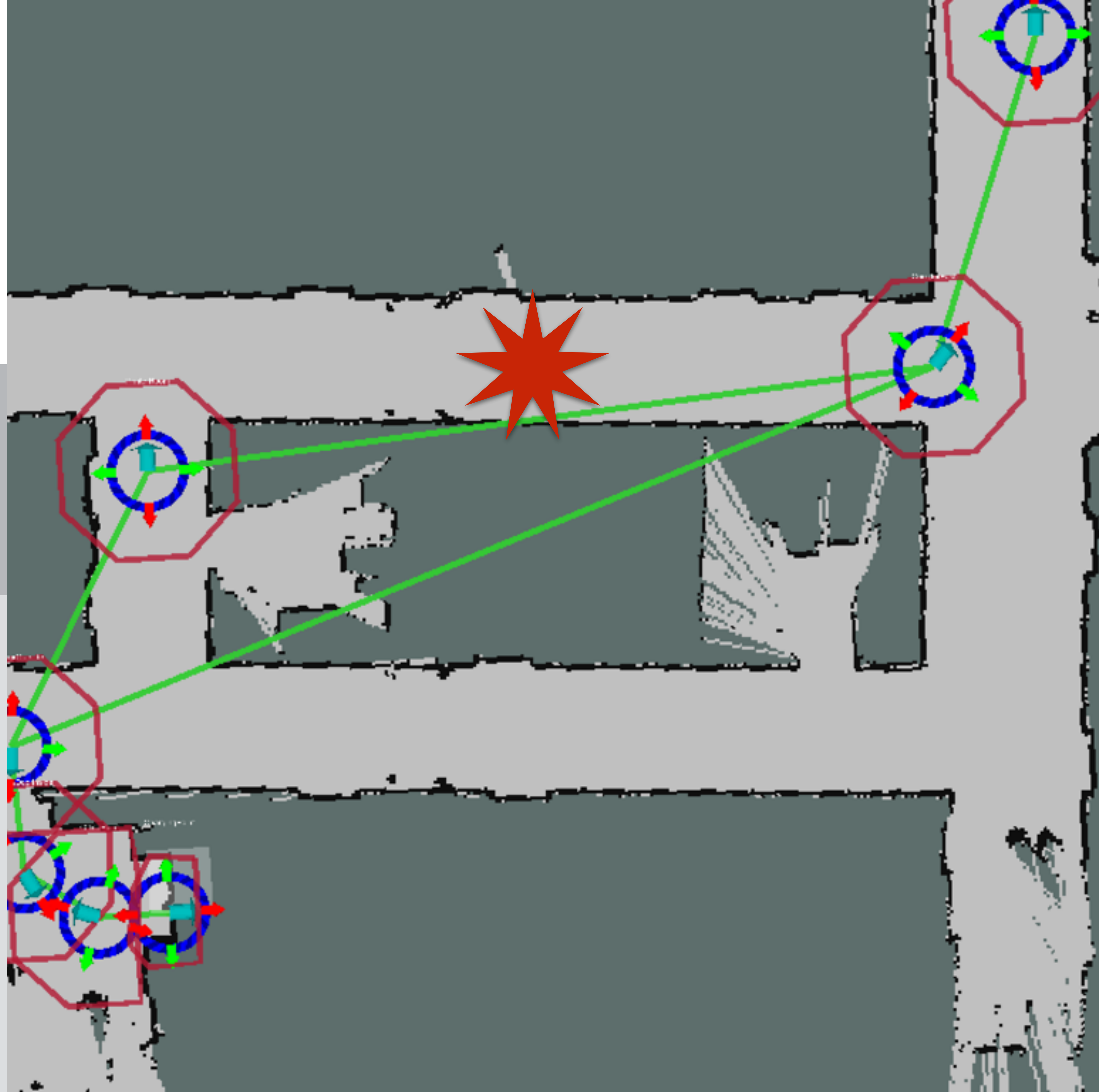
- github pull request and testing can also be used for automated benchmarking of systems/components
- live:
  - https://github.com/marc-hanheide/fremen_activity_benchmark
- adopt "proper" software development procedures for larger-scale collaborative projects

Reality and mature components still quite far from being perfect



Localisation
& Navigation

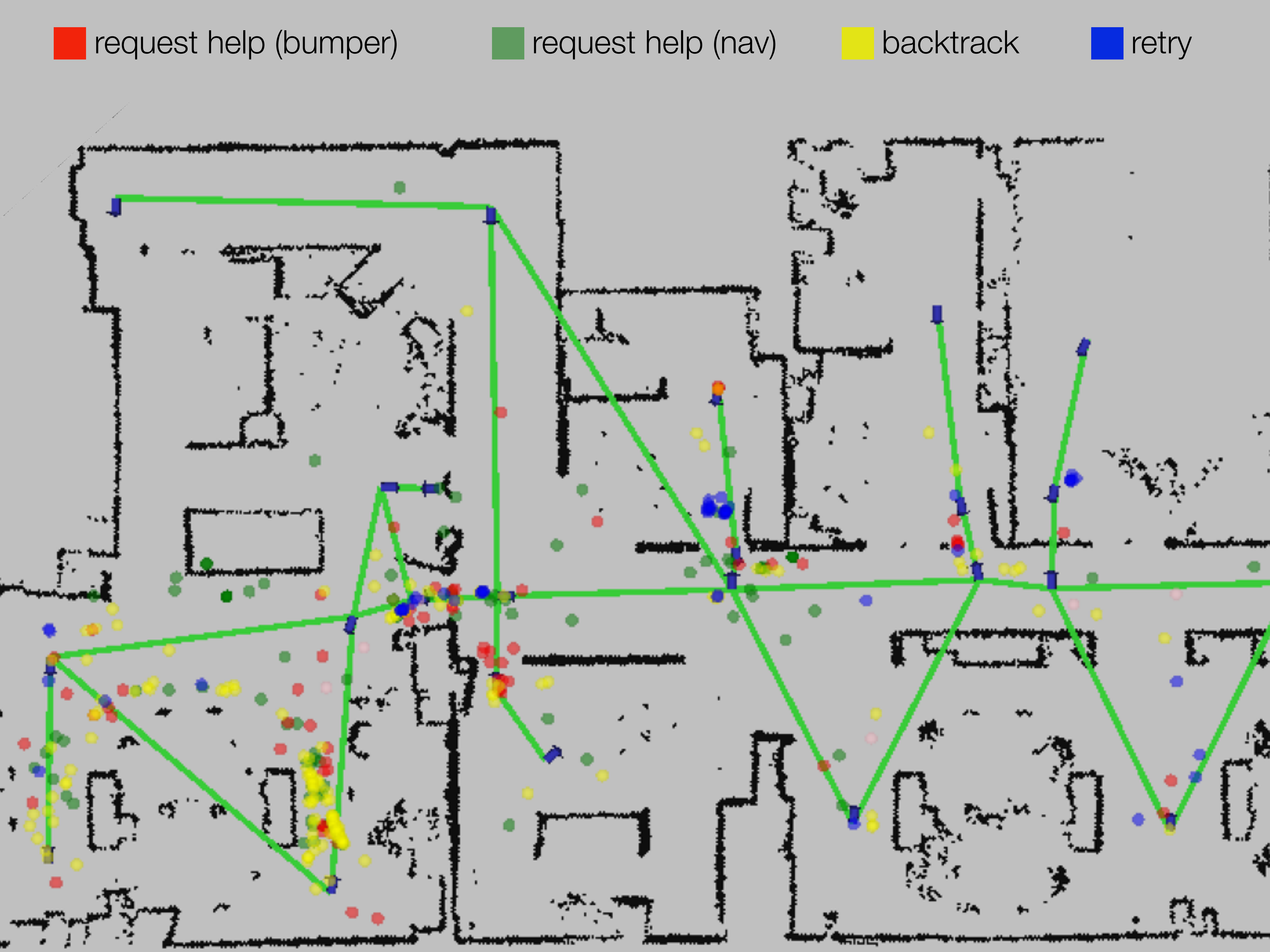
- Optimal Nav
- Monitored
- Topological
- Continuous

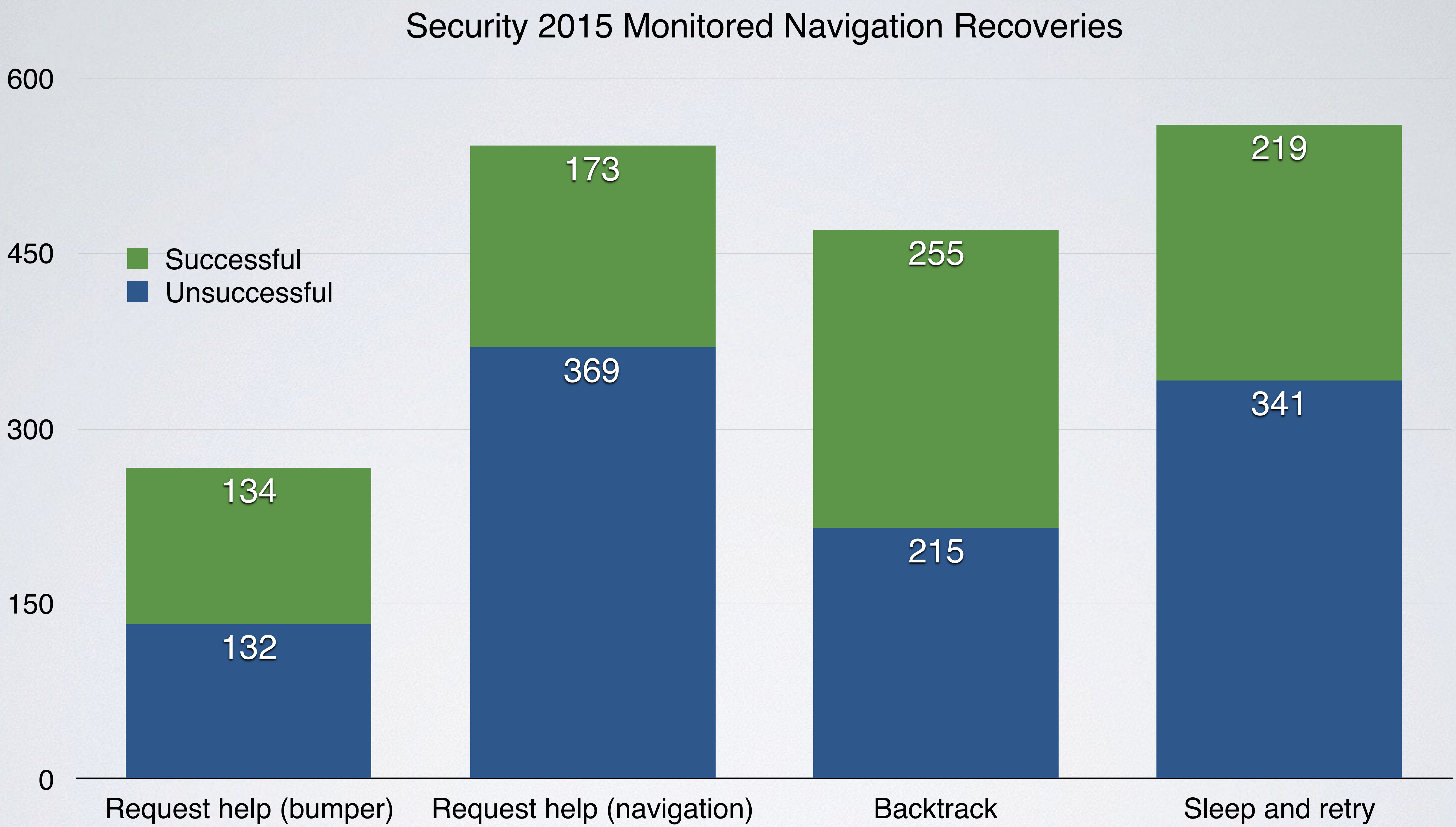


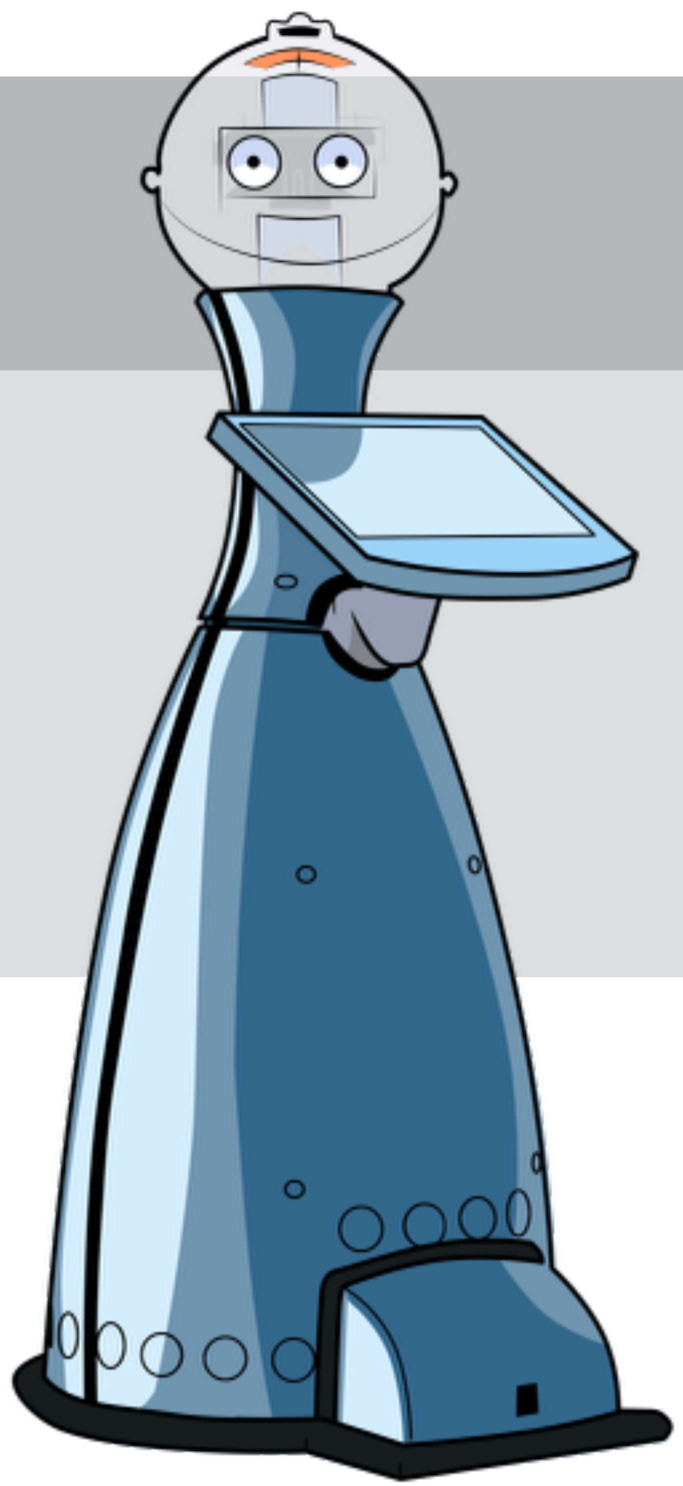
Monitored

Topological

Continuous


request help (bumper)
request help (nav)
backtrack
retry


Security 2015 Monitored Navigation Recoveries
600
450
300
150
0
Successful
Unsuccessful
134
132
173
369
255
215
219
341
Request help (bumper)
Request help (navigation)
Backtrack
Sleep and retry



Executive
Control

Localisation
& Navigation

Optimal Nav

Monitored

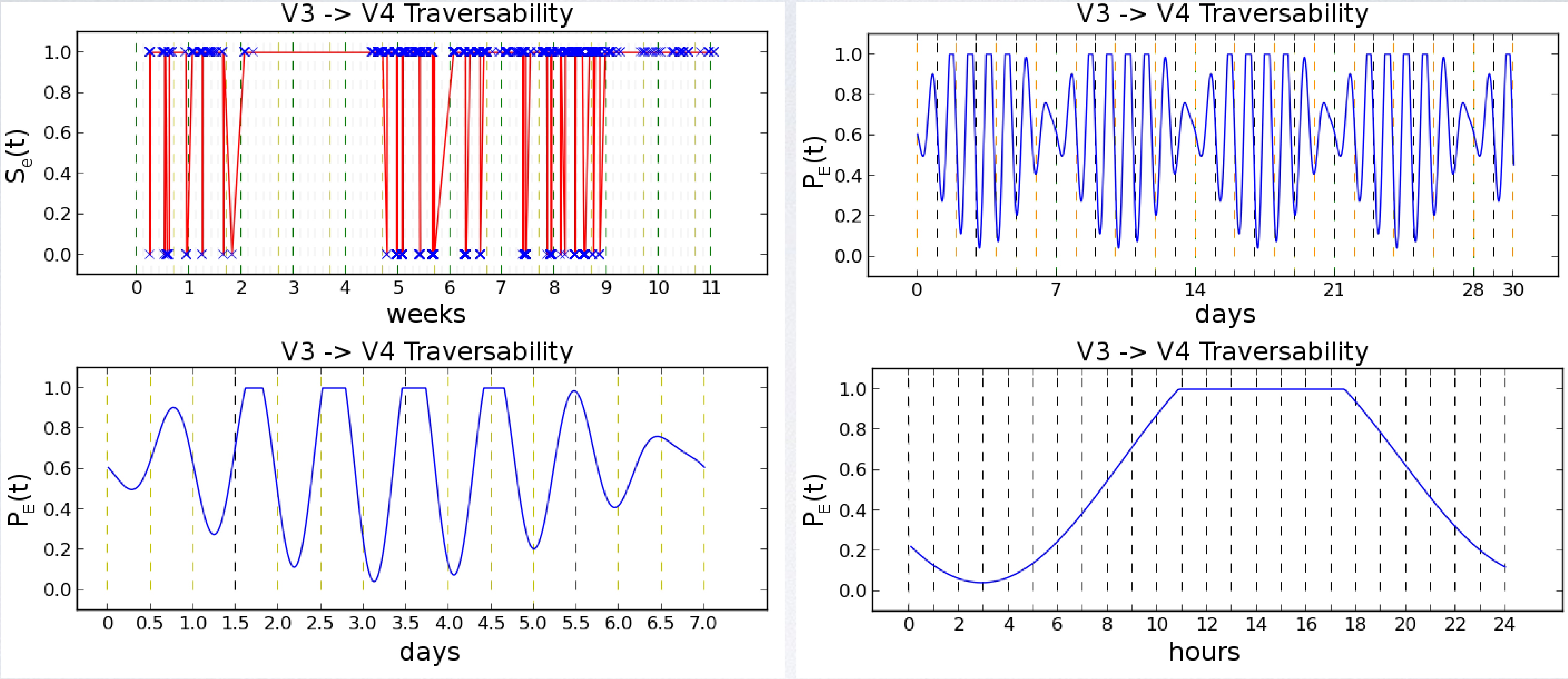
Topological

Continuous

SEVENTH FRAMEWORK PROGRAMME

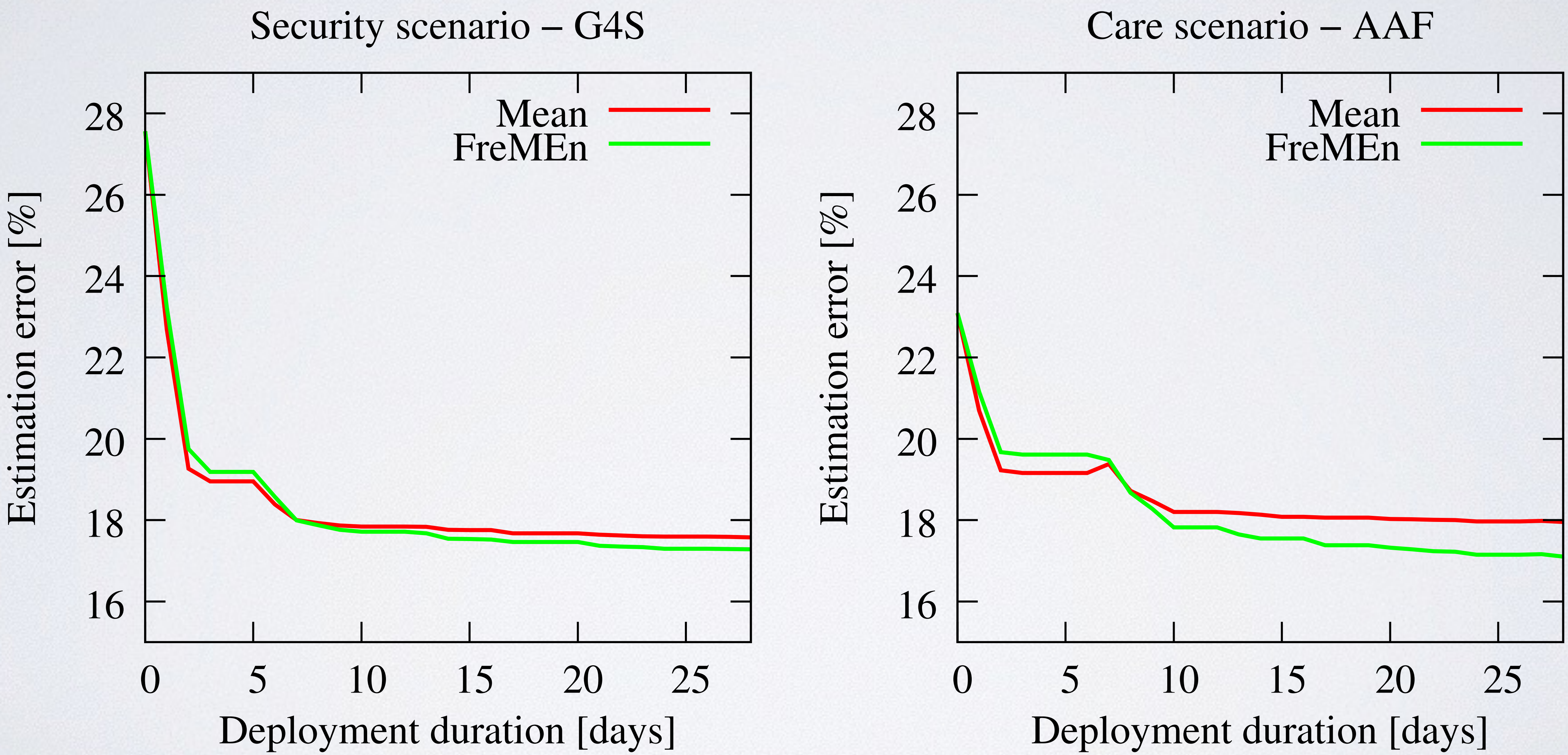
STRANDS

TOPOLOGICAL EDGE TRAVERSABILITY MODELLING USING FREMEN

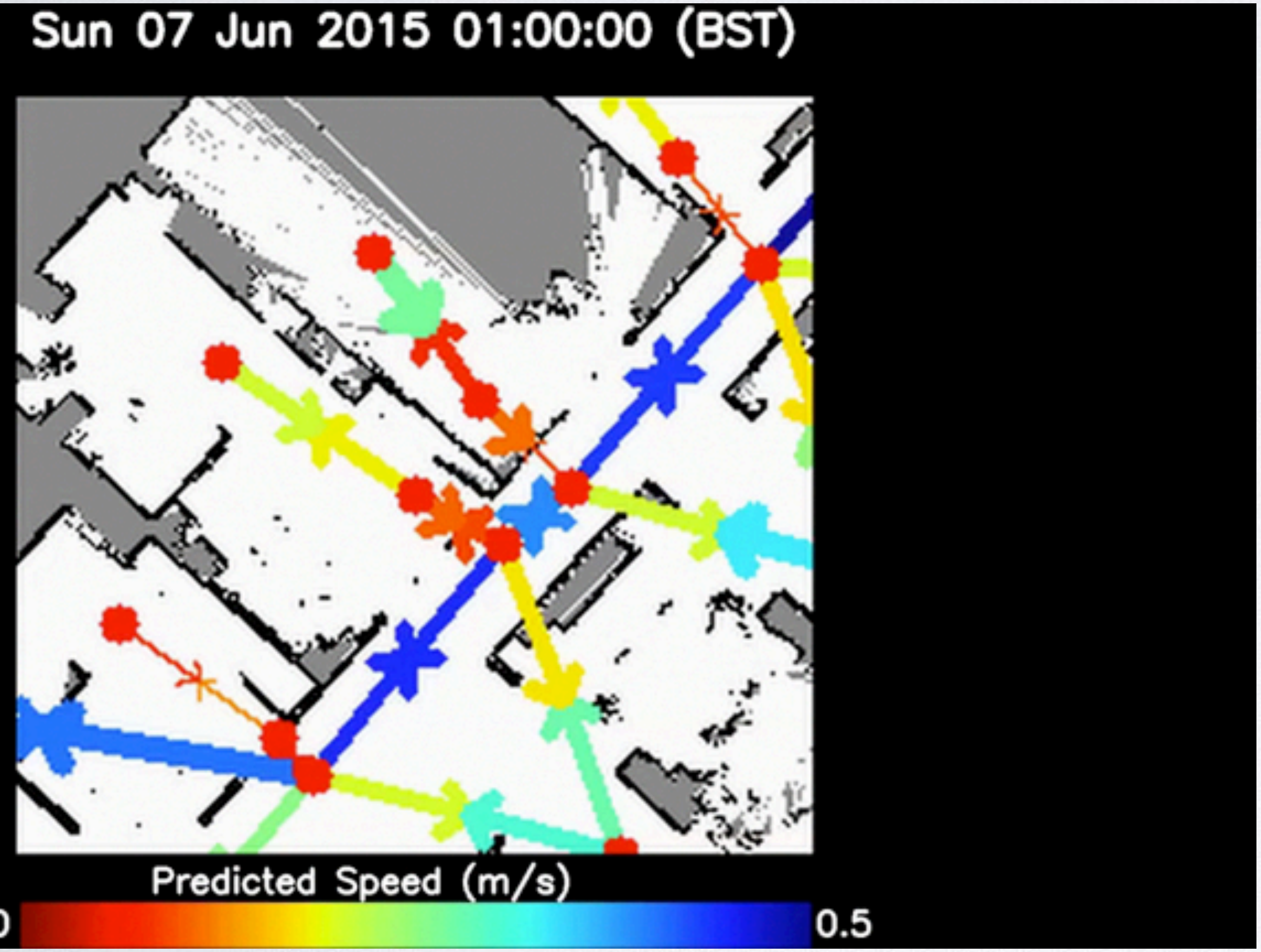


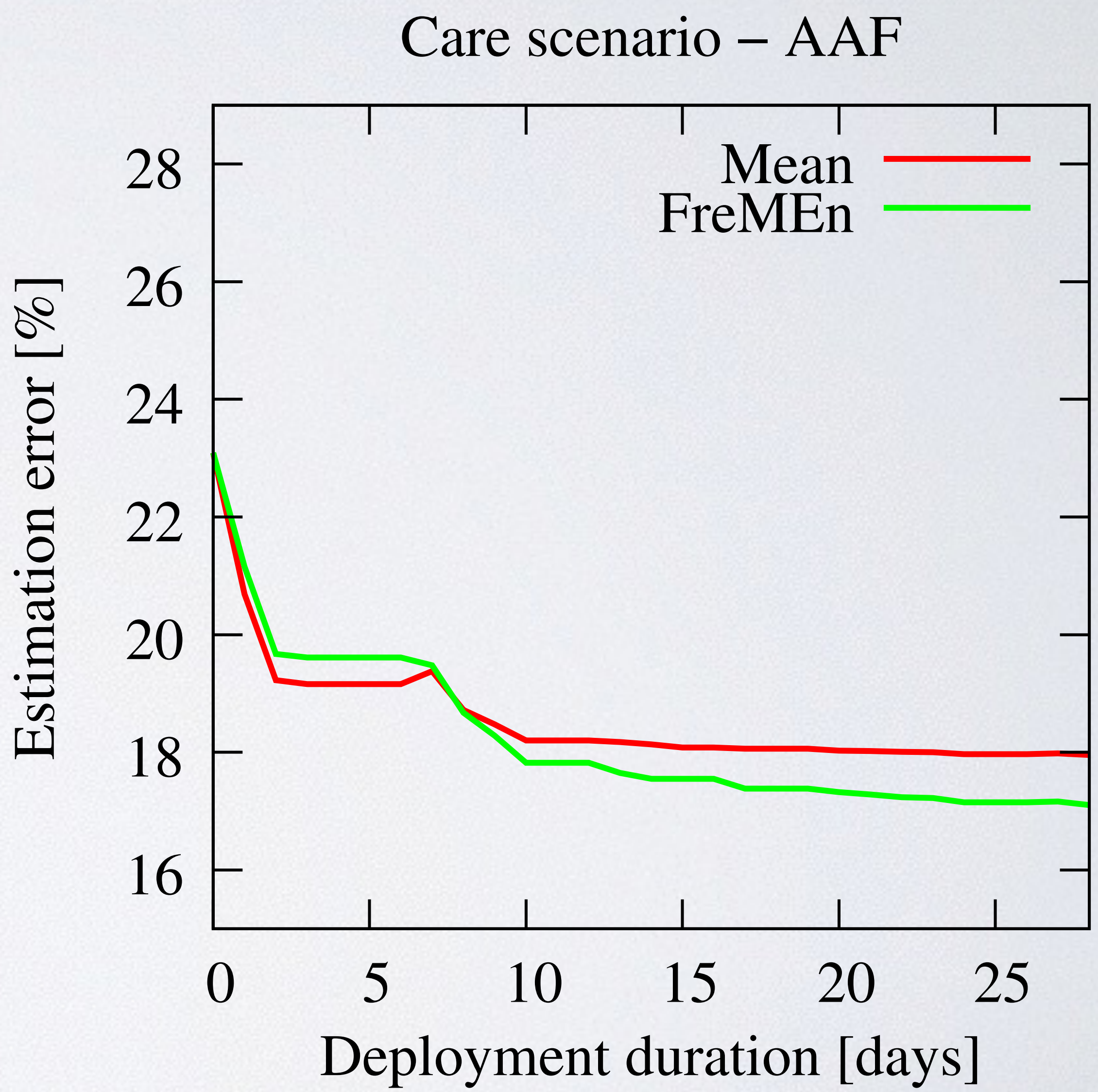
J. Pulido Fentanes, B. Lacerda, T. Krajník, N. Hawes, and M. Hanheide. Now or later? predicting and maximising success of navigation actions from long-term experience. In ICRA, 2015.

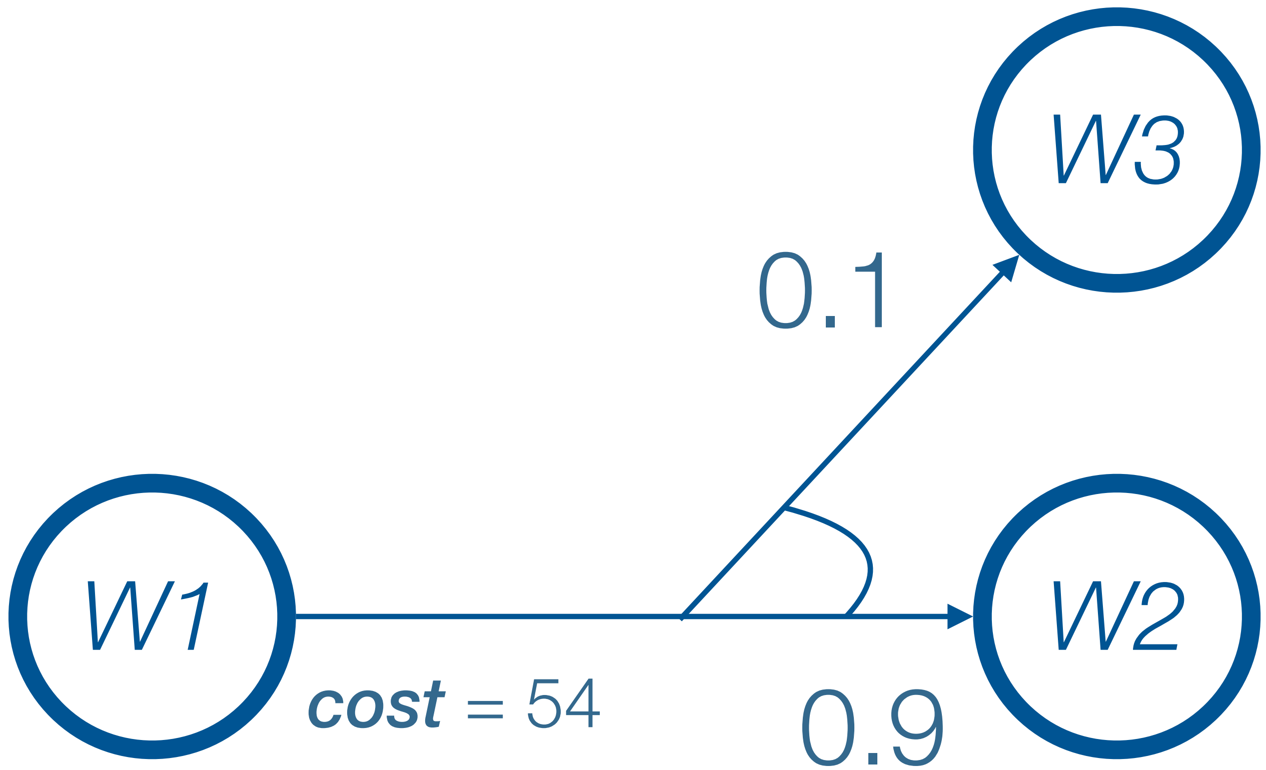
TOPOLOGICAL EDGE DURATION PREDICTION



TOPOLOGICAL EDGE DURATION PREDICTION







action goto *W2* from *W1*

B. Lacerda, D. Parker, and N. Hawes. Optimal and Dynamic Planning for Markov Decision Processes with Co-Safe LTL Specifications. In: IROS 2014.


PULL

GETTING IT OUT THERE

DEPLOYMENT ISSUES AND SOLUTIONS

INSTALLATION HELL

You need boost 1.35 to compile peekabot plus some additional libraries. Under Ubuntu 8.10 you would need to do

QUICK UPDATE DURING SESSION. DO NOT USE BZR! GET PEEKABOT FROM HERE: http://www.cs.bham.ac.uk/~nah/irlab/peekabot-0.6.0.tar.gz

```
sudo apt-get install libboost1.35-dev libfltk1.1-dev libpng12-dev libxerces-c2-dev bzr libfreetype6-dev libglut3-dev libtool automake autoconf

bzr branch lp:peekabot
cd peekabot

autoreconf -i
./configure --prefix=/usr/local
make
sudo make install
```

To install CURE on Ubuntu you need to do the following:

```
sudo apt-get install automake autoconf libtool

svn co https://codex.cs.bham.ac.uk/svn/nah/cosy/development/kth/cosycure
cd cosycure

./configure --prefix=/usr/local
make
sudo make install
```

If there is no ./configure file, then run:

```
autoreconf --install
./configure --prefix=/usr/local
make
sudo make install
```

Before compiling player please install the ltdl development files (stage needs player to be compiled with it)

```
sudo apt-get install libltdl7-dev libgtk2.0-dev
```

Also make sure that you have the file /etc/X11/rgb.txt. Get a copy of it under subarchitectures/nav.sa/config/rgb.txt and copy it into /etc/X11

Building and installing player follows the standard automake routine

```
tar -zxvf player-2.1.1.tar.gz
cd player-2.1.1

./configure --prefix=/usr/local
make
sudo make install
```

INSTALLATION HELL

You need boost 1.35 to compile peekabot plus some additional libraries. Under Ubuntu 8.10 you would need to do

QUICK UPDATE DURING SESSION, DO NOT USE BZR! GET PEEKABOT FROM HERE: http://www.cs.bham.ac.uk/~nah/irlab/peekabot-0.6.0.tar.gz

```
sudo                                      png12-dev libxerces-c2-dev bzr libfreetype6-dev libglut3-dev libtool automake autoconf
bzr
cd p
auto
./co
make
sudo
```

Dependencies are a mess

Shall each member in the project spend days to get a working system?

To install CURE on Ubuntu you need to do the following:

```
sudo apt-get install automake autoconf libtool

svn co https://codex.cs.bham.ac.uk/svn/nah/cosy/development/kth/cosycure
cd cosycure

./configure --prefix=/usr/local
make
sudo make install
```

If there is no ./configure file, then run:

```
autoreconf --install
./configure --prefix=/usr/local
make
sudo make install
```

Before compiling player ple

```
sudo apt-get install
```

Also make sure that you ha

Building and installing play

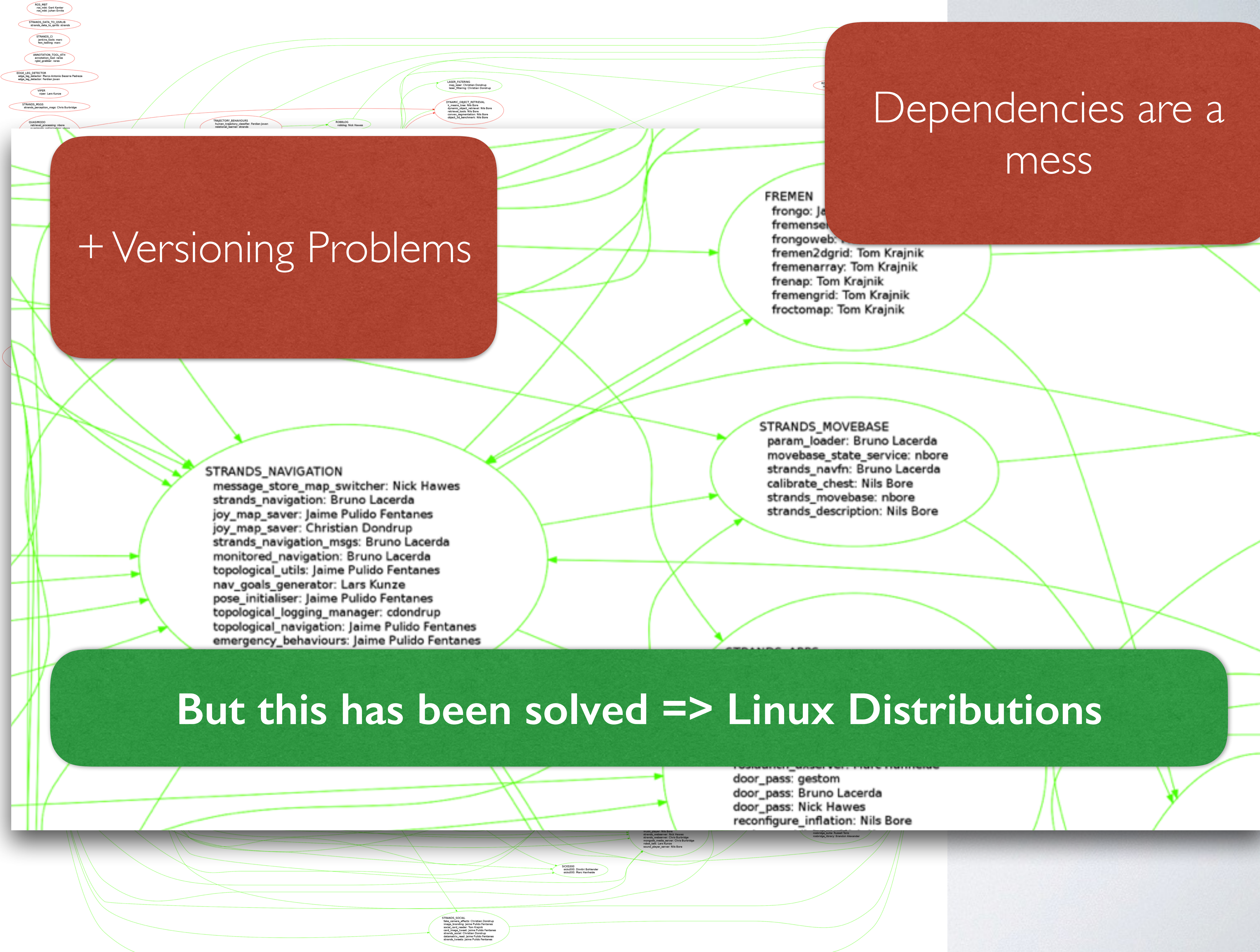
```
tar -zxvf player-2.
cd player-2.1.1

./configure --prefix
make
sudo make install
```

Why can you install an Ubuntu system in 30 minutes (>1000 packages) but not a simple robot system?

Dependencies are a mess

+ Versioning Problems



But this has been solved => Linux Distributions

All Open Source & Binary Released

Representation & Analysis

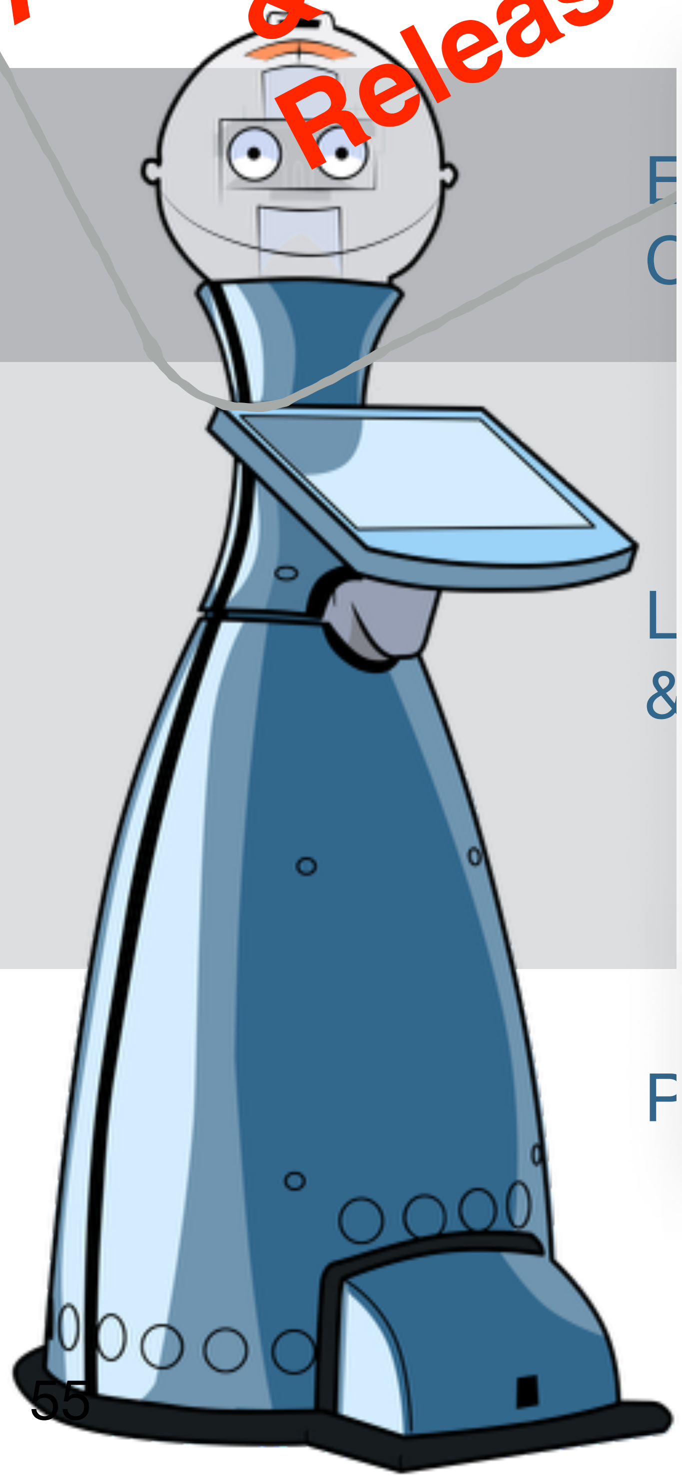
MongoDB

QSRLib

FreMEn

SOMa

View Plan.



Using the STRANDS repository

These steps are for a system administrator who wants to install STRANDS' release packages:

1. Enable the ROS repositories: Accomplish all the steps under `1. Installation` http://wiki.ros.org/indigo/Installation/Ubuntu#Installation.
2. Enable the STRANDS repositories:
   i. Add the STRANDS public key to verify packages:
   ```
   curl -s http://lcas.lincoln.ac.uk/repos/public.key | sudo apt-key add -
   ```
   ii. Add the STRANDS repository:
   ```
   sudo apt-add-repository http://lcas.lincoln.ac.uk/repos/release
   ```
3. update your index:
   ```
   sudo apt-get update
   ```
4. install any packages you want using `sudo apt-get install <pkg-name>`

Person Detect

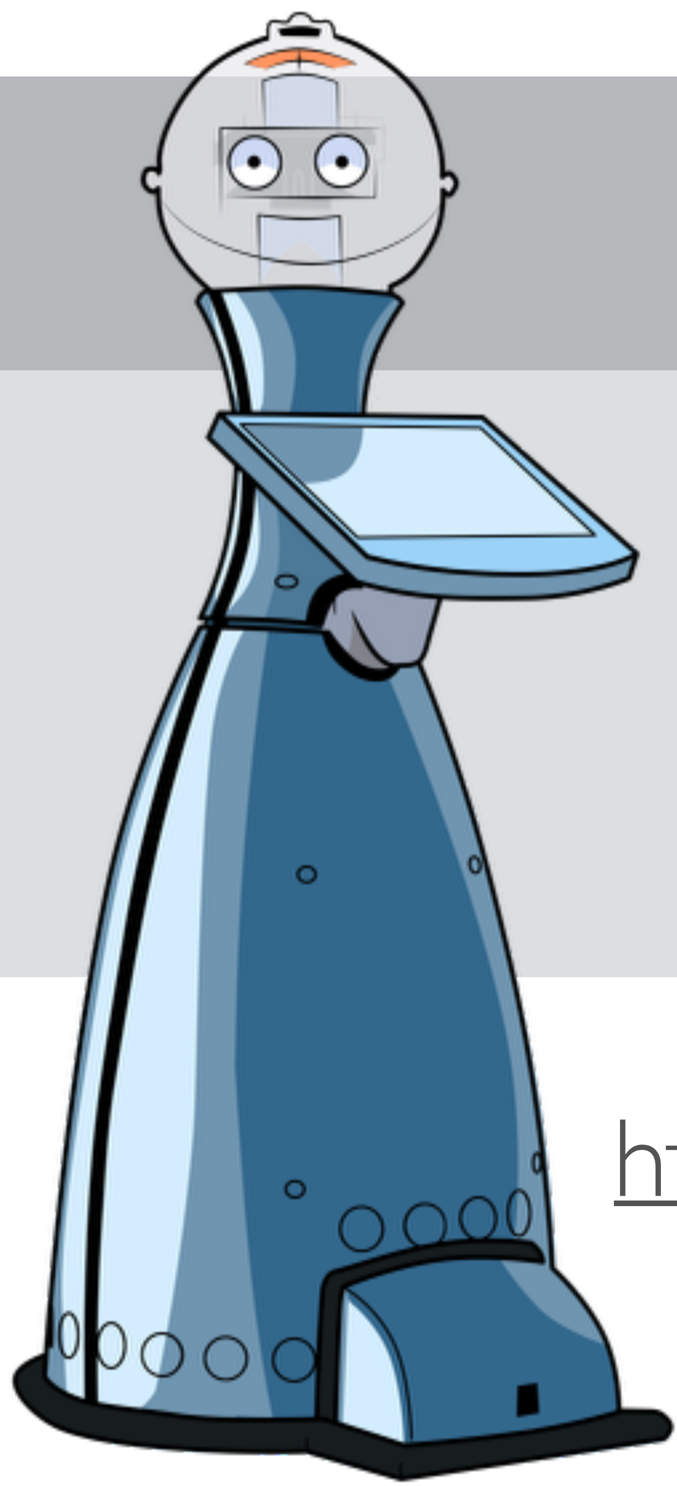
Object Rec.

Object Disc.

SEVENTH FRAMEWORK PROGRAMME

STRANDS

Jenkins



- ROS has a build farm (build on top of Debian deployment principles)
- STRANDS has implemented their own re-using OSRF's implementation
- everybody can submit their packages to ROS: https://github.com/ros/rosdistro/blob/master/CONTRIBUTING.md
- you get binary Ubuntu packages

https://github.com/strands-project/buildfarm

SEVENTH FRAMEWORK PROGRAMME

STRANDS

RECOMMENDATION

Write your own commandments (or adopt some of mine)!

Get people to commit to ONE OS/ROS/HW combination

Adopt established software engineering principles (pull requests, code reviews, CI)

Make use of the deployment toolchain (your own or OSRF ROS toolchain)

Use Python were possible

Always question researchers' software engineering decisions ;-)

RECOMMENDATION

Write your own commandments (or adopt some of mine)!

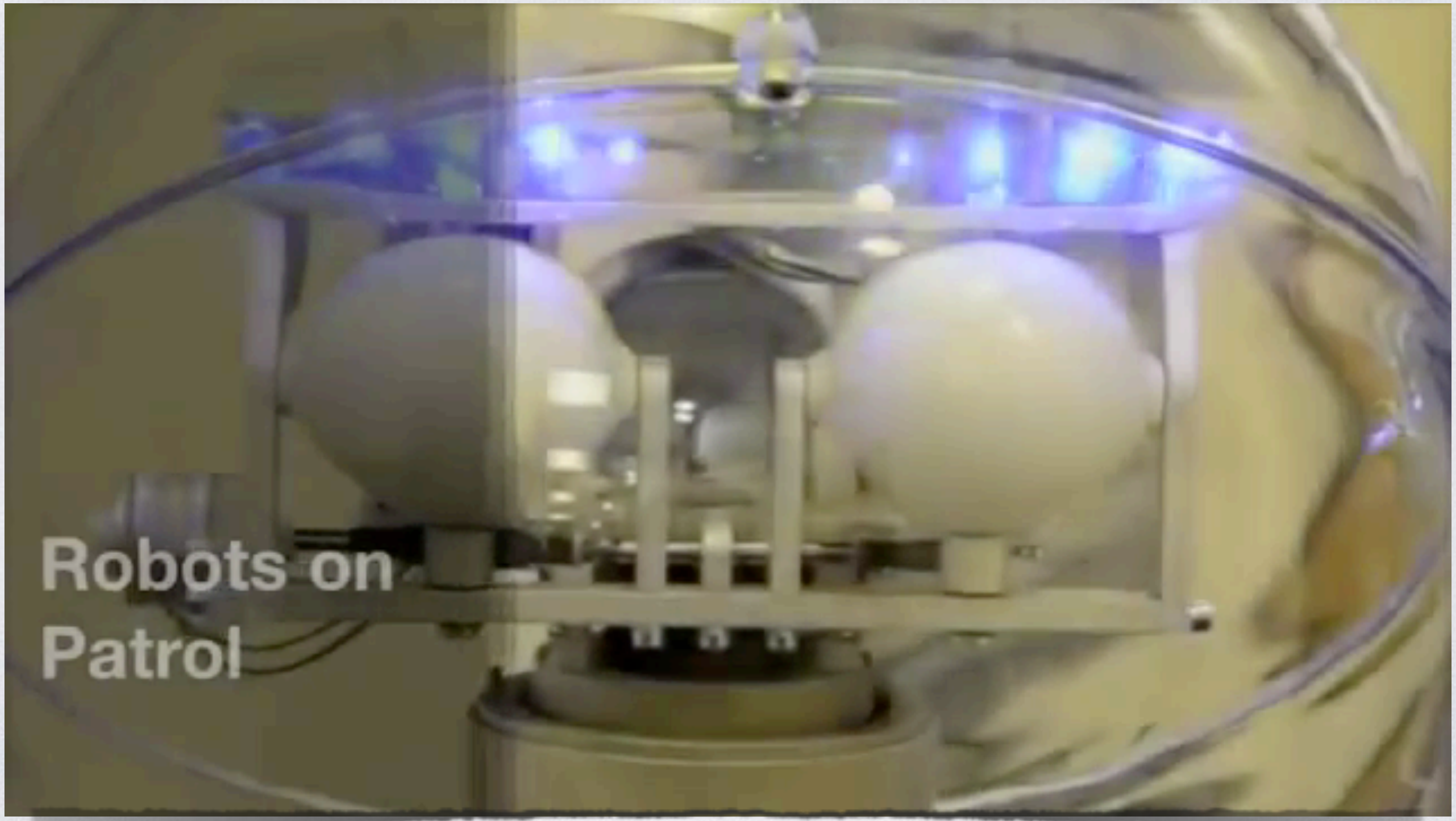
Get people to commit to ONE OS/ROS/HW combination

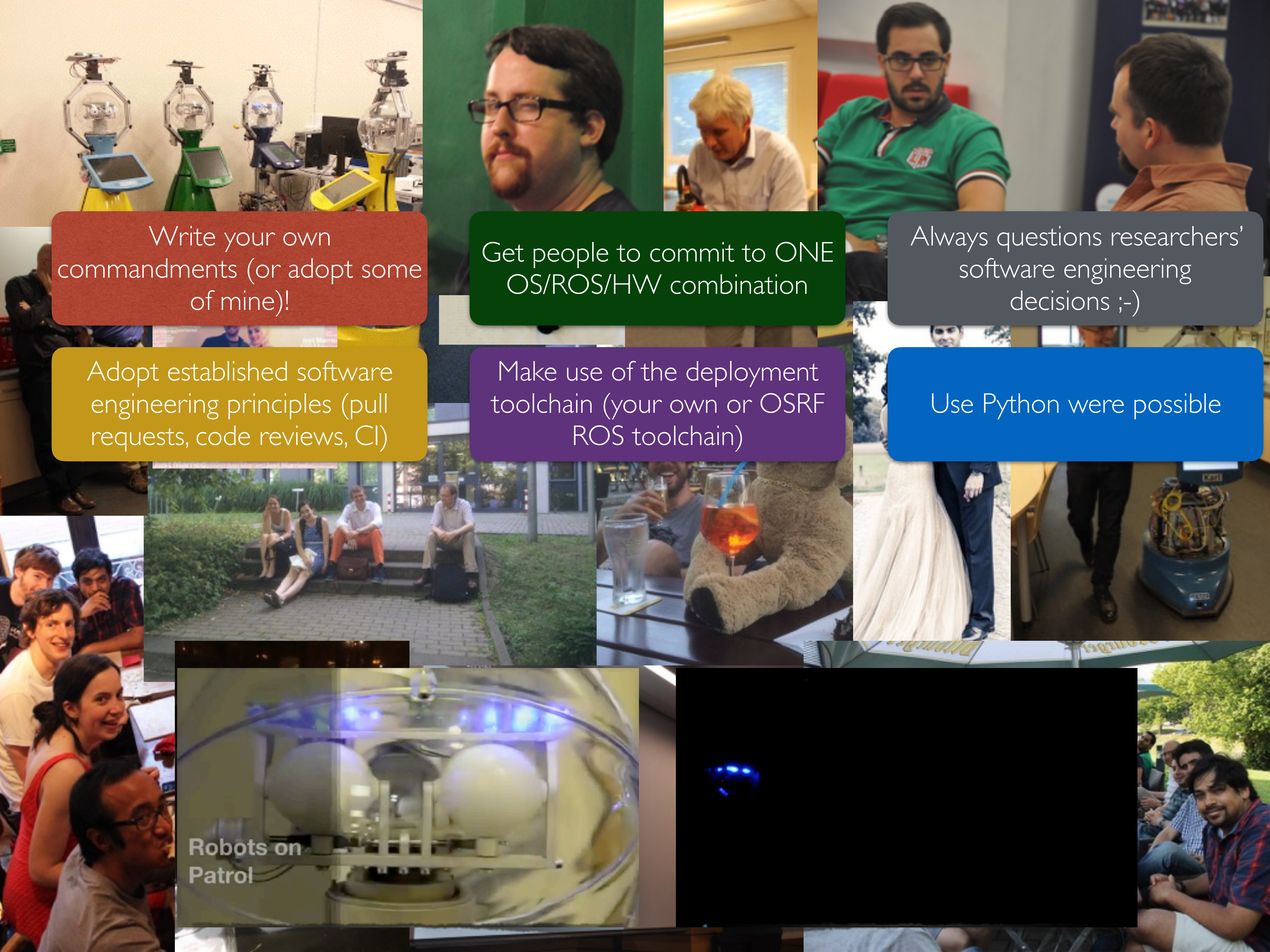
Always questions researchers' software engineering decisions ;-)

Adopt established software engineering principles (pull requests, code reviews, CI)

Make use of the deployment toolchain (your own or OSRF ROS toolchain)

Use Python were possible




Write your own commandments (or adopt some of mine)!
Get people to commit to ONE OS/ROS/HW combination
Always questions researchers' software engineering decisions ;-)
Adopt established software engineering principles (pull requests, code reviews, CI)
Make use of the deployment toolchain (your own or OSRF ROS toolchain)
Use Python were possible
Robots on Patrol