Software and systems design process: Introducing the whole lifecycle

Davide Brugali
University of Bergamo, Italy

Software variability

2024 ROS Metrics Report

Katherine Scott & Tully Foote

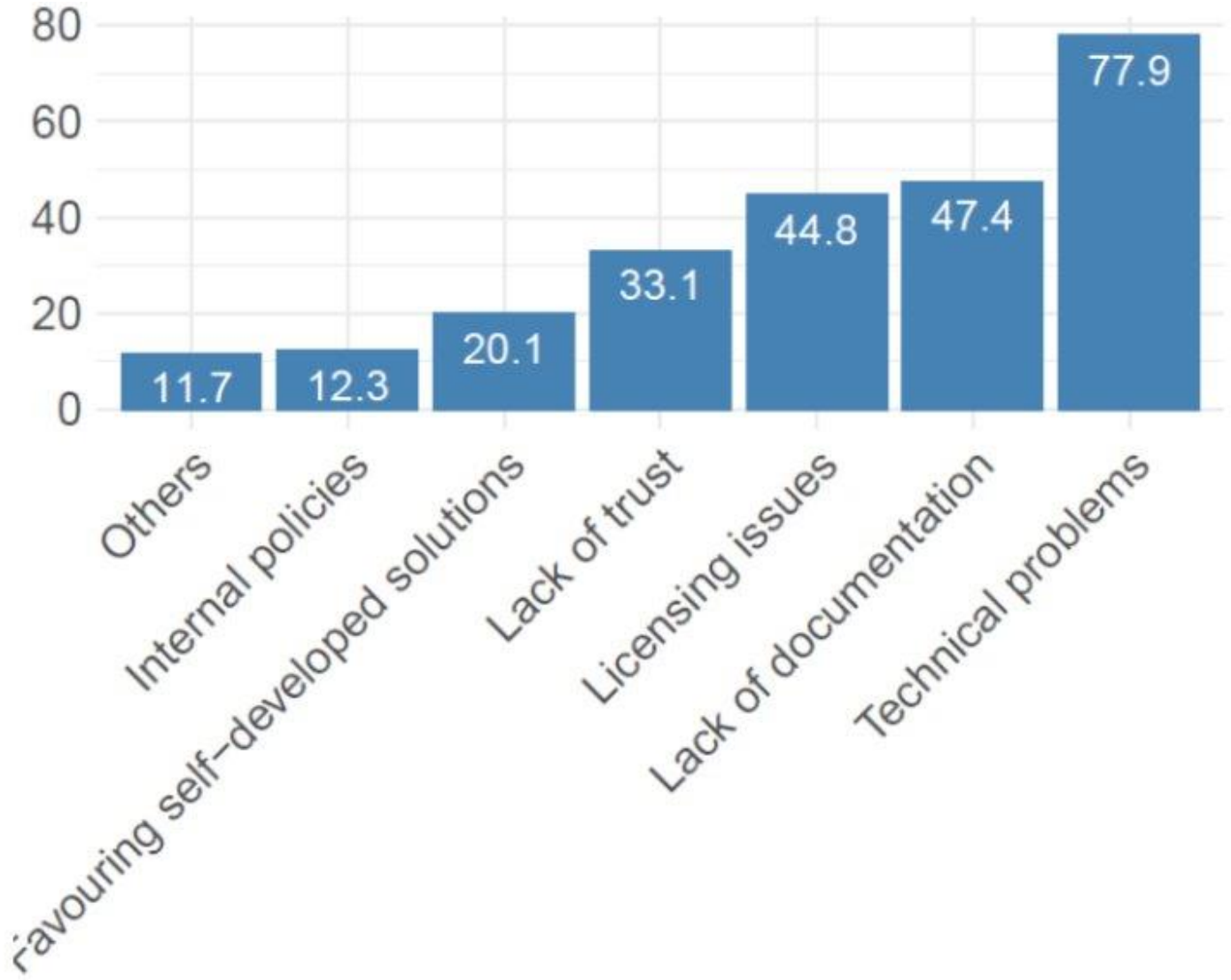
8,349

Number of Github repositories tagged with #ROS on 2025-01-22

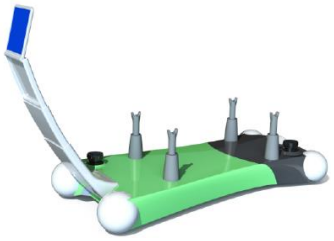
2,760

Number of Github repositories tagged with #ROS2 on 2025-01-22.

Reasons for not reusing software



The cost of software development for service robots



container-transporting robot

care utensil robot

floor-cleaning robot

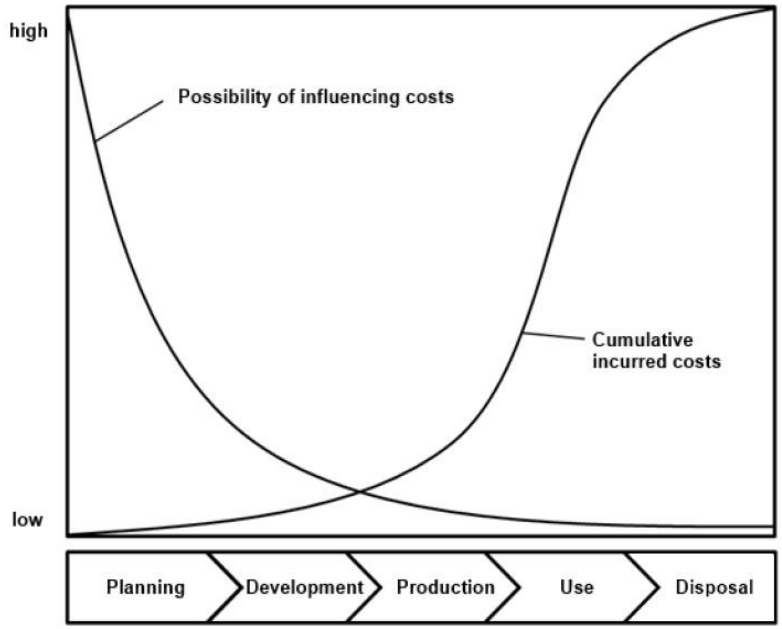
NIKOLAUS BLÜMLEIN

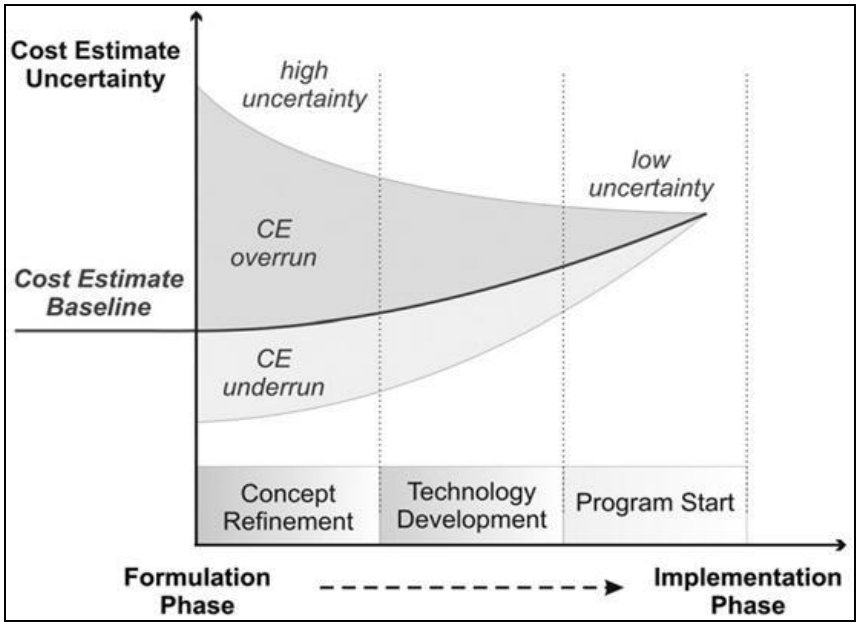
Function-based Cost Estimation for Service Robot Prototypes in Early Design Phases

Universität Stuttgart — Fraunhofer IPA

Cost type	Expected value	Standard deviation
Material (per unit)	121,389.58 €	11,183.26 €
Hardware installation (per unit)	21,900.00 €	13,950.00 €
Administration	36,575.00 €	1,741.67 €
Software development	5,814,611.29 €	972,242.91 €
Software installation (per unit, not for prototype)	568.18 €	190.91 €
System designing	1,456,657.64	378,552.25 €

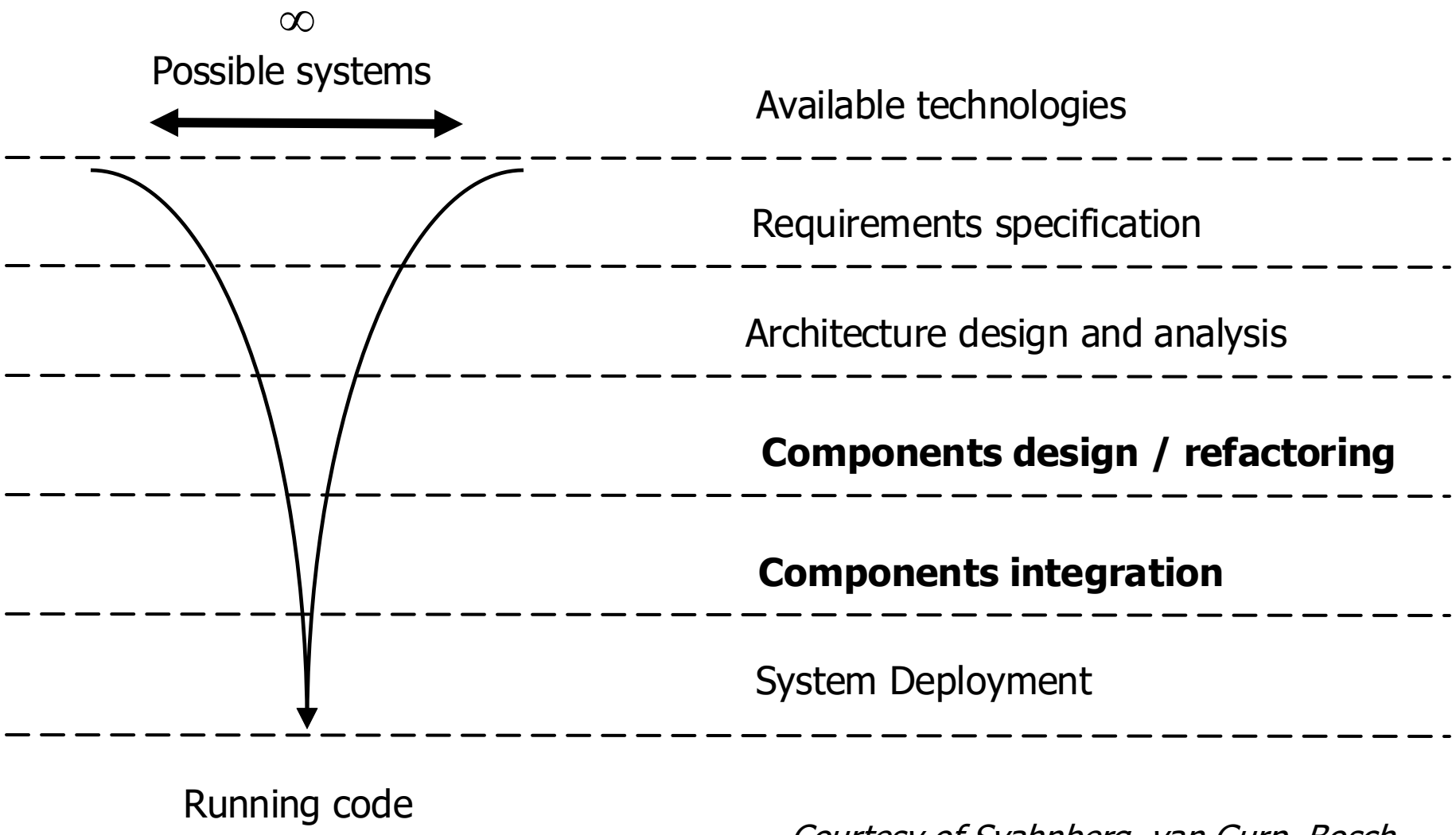
The cost of software development for service robots





- up to 90% of the life-cycle costs are influenced by decisions taken in early development phases
- the cost estimate uncertainty is higher in early phases
- the majority of the costs are incurred during production, and use (i.e. maintenance)

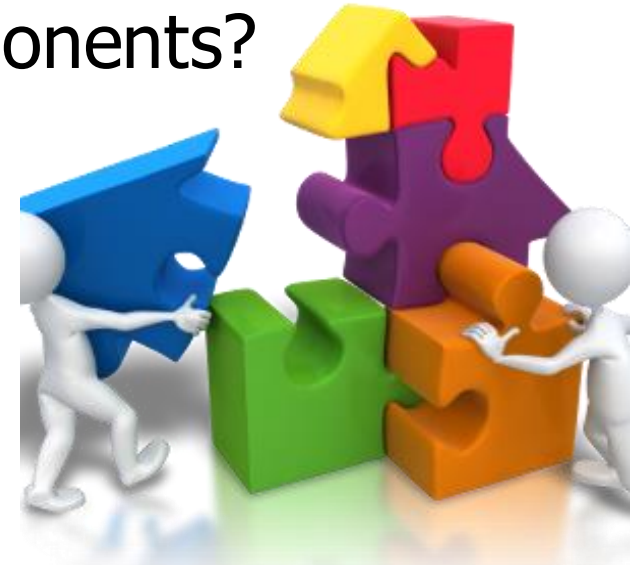
Current practice : single system development



Courtesy of Svahnberg, van Gurp, Bosch

Reusing Components

Is it really possible to build systems out of independently developed components?



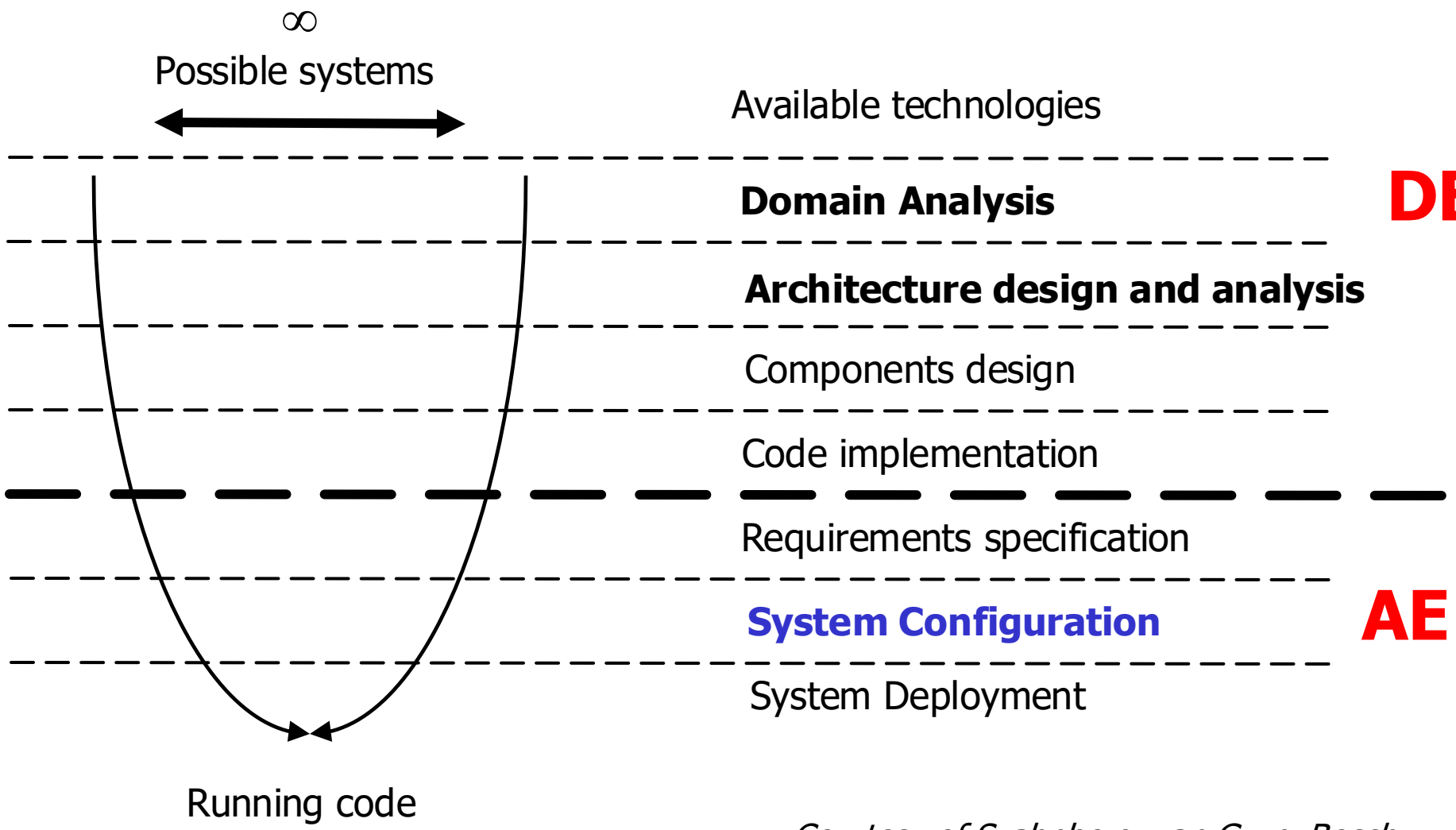
Courtesy of http://ramsplus.com/services/

Courtesy http://www.gobeyondthecube.com

ROS mantra:
"we do not wrap your main"

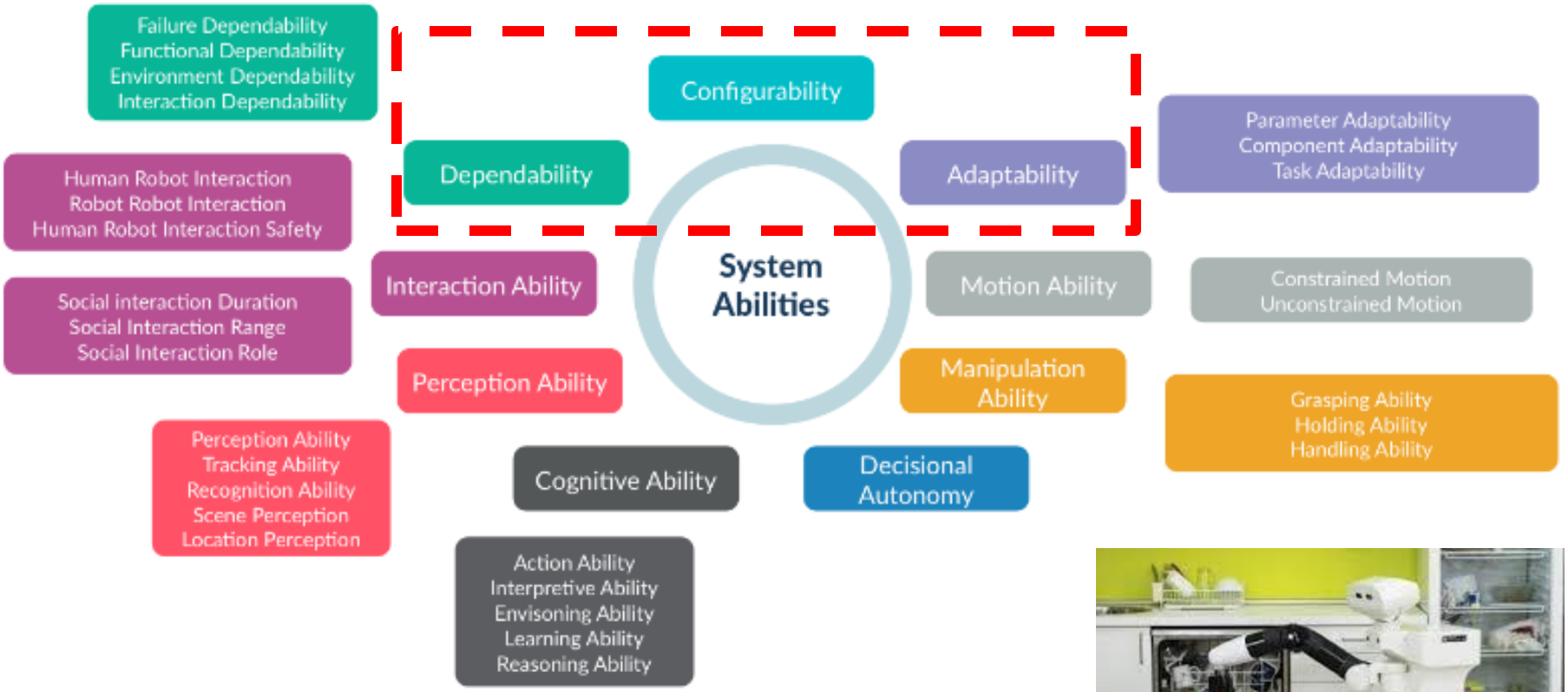
Or should someone imagine how they will fit together in future applications?

Software Product Lines : configurable systems



Courtesy of Svahnberg, van Gurp, Bosch

Service robotics



Robotics 2020
Multi-Annual Roadmap

For Robotics in Europe

Horizon 2020 Call ICT-2017 (ICT-25, ICT-27 & ICT-28)

Definitions : Configurability

- Robotics 2020 Multi-Annual Roadmap (MAR) defines configurability as

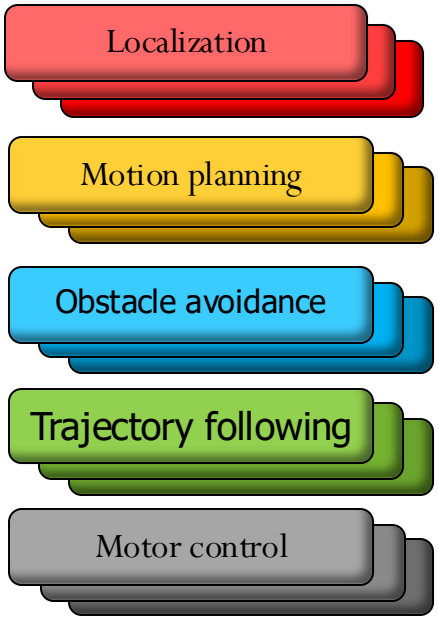
  "the ability of the robot to be configured to perform a task or reconfigured to perform different tasks."

- Configurability implies
  - an ease of modification and
  - an absence of irreversible or rigid commitments
- in some aspects of the robotic system.

Motion ability

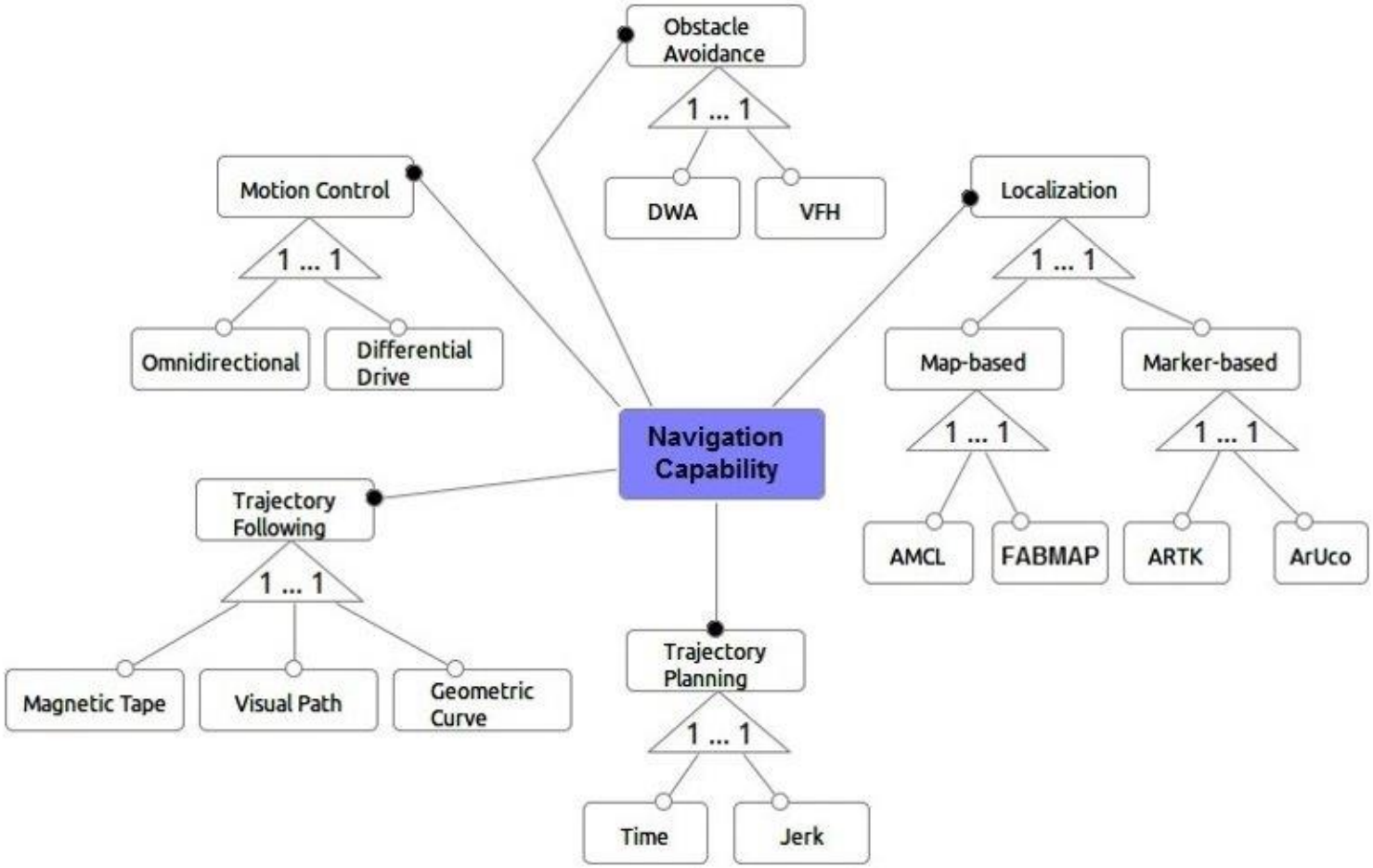
Requires a lot of functionality

. . .

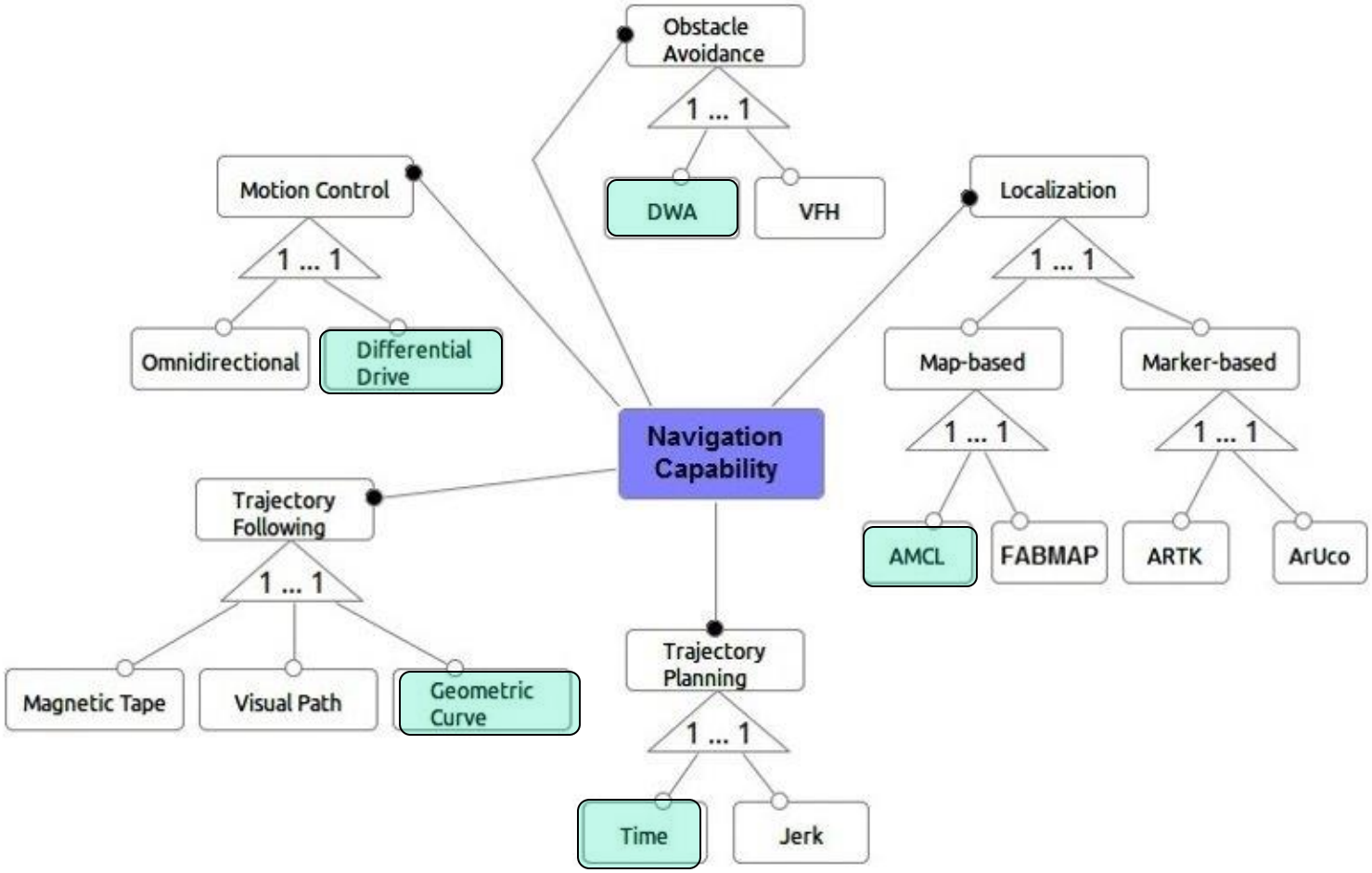


Software Variability

Software variability in robot navigation



Software variability in robot navigation



Two steps towards configurability

Domain Analysis: identifying the drivers of variability

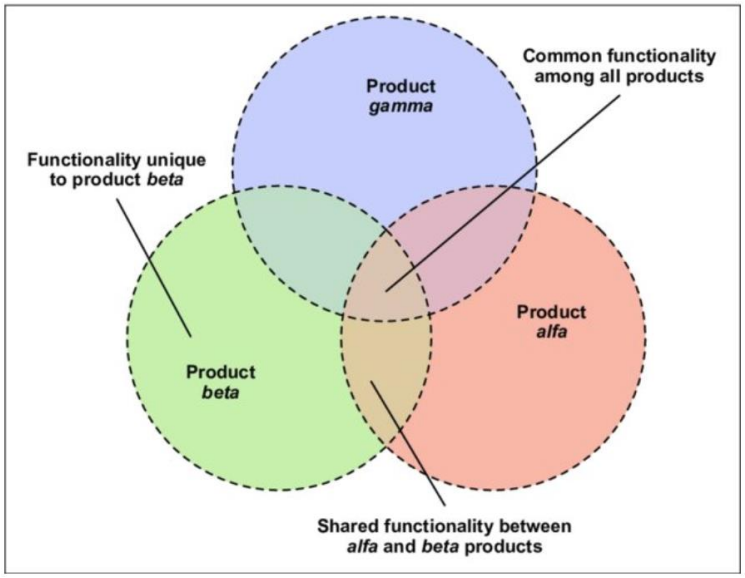
System infrastructures: supporting the development of configurable component systems

DOMAIN ANALYSIS

Bottom Up: Commonality and Variability Analysis

Identifying :

- **common abstractions and variations,**
- **relationships between them,**
- **assigning them responsibilities, and then**
- **linking them together** (Rebecca Wirfs-Brock)



8,349

Number of Github repositories tagged with #ROS on 2025-01-22

2,760

Number of Github repositories tagged with #ROS2 on 2025-01-22.

Commonality and variability in software engineering

J. Coplien; D. Hoffman; D. Weiss, IEEE Software 1998

Top Down: Drivers for Variability

Software variability in service robotics

*Sergio García, Daniel Strüber, **Davide Brugali**, Alessandro Di Fava, Patrizio Pelliccione, Thorsten Berger,*

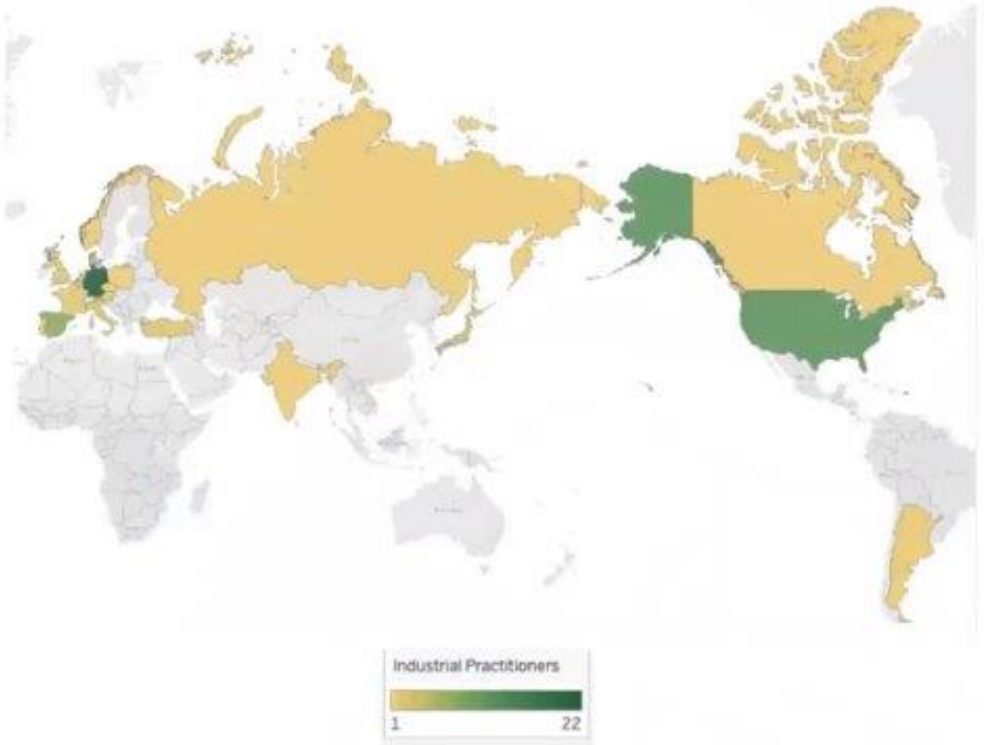
Empirical Software Engineering 2023

Interviews with industrial practitioners

- 18 interviews (16 hours)
- 15 different companies
- 9 different countries

Online questionnaire

- 156 answers
- 72 industrial practitioners
  - 58 different companies
  - 20 different countries
- 22 academic practitioners
  - 16 different universities
  - 10 different countries



Software variability in service robotics

*Sergio García, Daniel Strüber, **Davide Brugali**, Alessandro Di Fava, Patrizio Pelliccione, Thorsten Berger,*

Empirical Software Engineering 2023

RQ1: What are the **drivers of variability** in the service robotics domain?

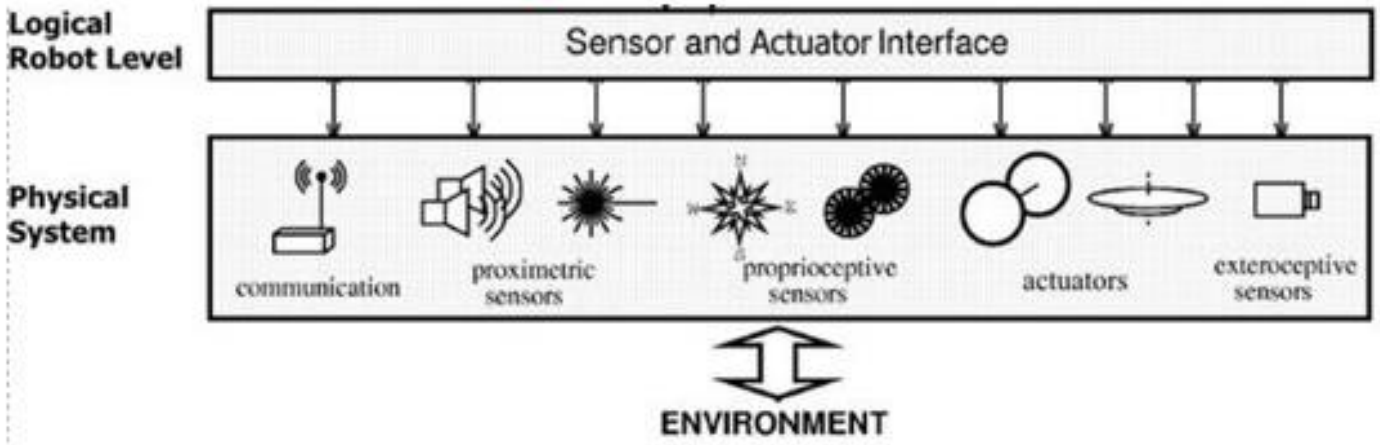
RQ2: What variability **management practices** are applied by the companies to address the drivers of variability?

RQ3: What **challenges** do service robotics companies face when managing variability?

Hardware variability

Hardware variability

Software variability



from LAAS Architecture

Challenges

- There is the need for design methodologies and tools promoting software/hardware decoupling.
- The main goal is that hardware can evolve without affecting the codebase.

Environment variability

Indoor

Artificial illumination

Highly dynamic

Mostly static

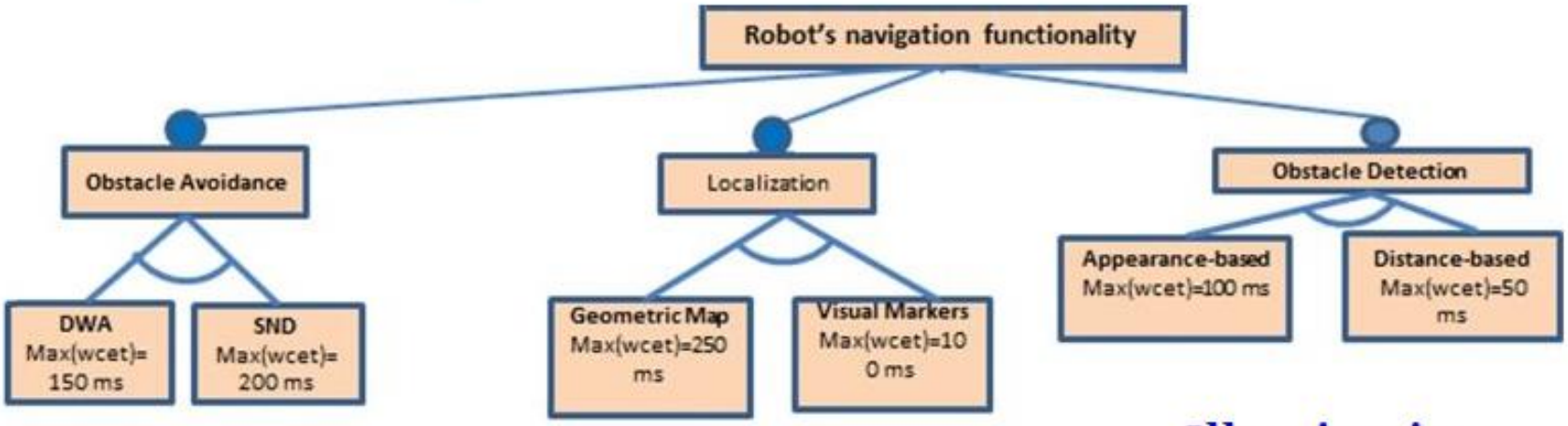
Surveillance

Outdoor

Natural illumination

Environment variability

Software variability



Illumination

Static / Dynamic

Structured / Unstructured

Challenges

- Companies need tools for
  - configuring the robot software control system
  - validating each possible configuration.

Task variability

BLUE OCEAN ROBOTICS
- for humans

PAL ROBOTICS

Task variability

Software variability

- Means of Human-Robot Interaction (textual/graphical, speech-based, gesture-based), mission type (single purpose, multi-purpose), level of autonomy.

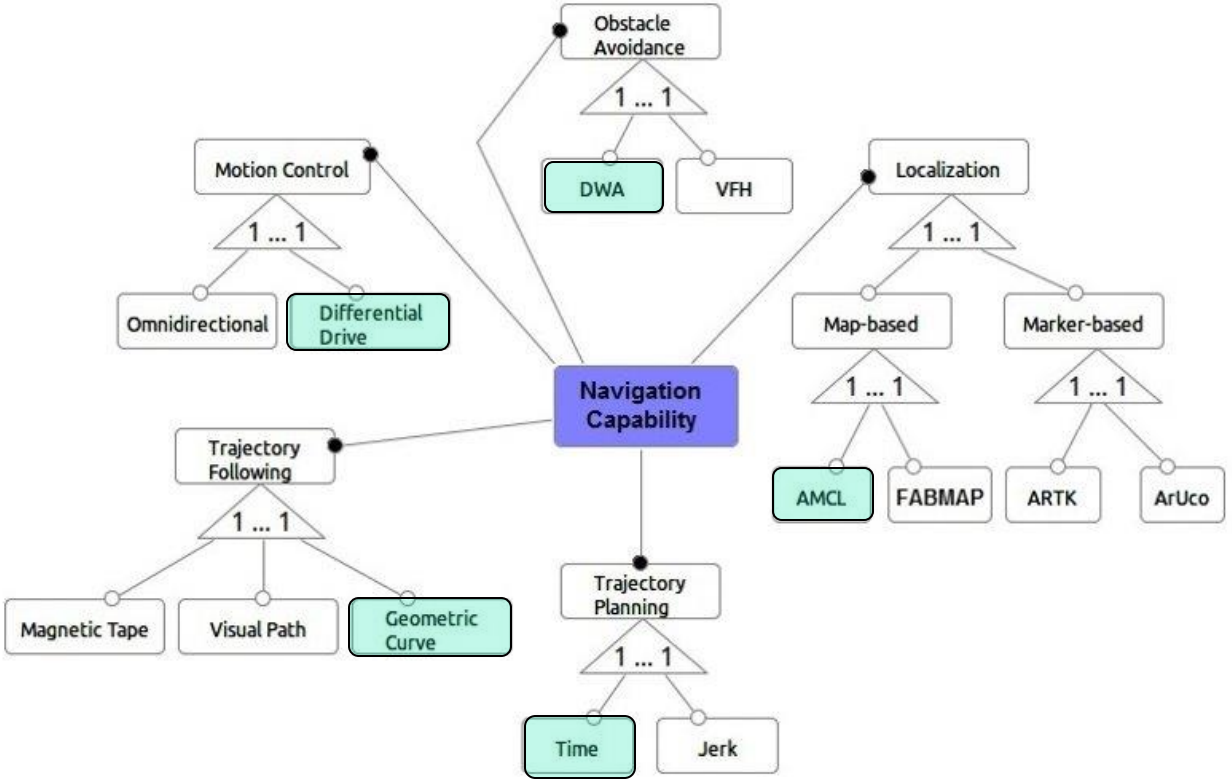
Challenges

- Companies need new approaches and tools to let non-technical customers specify complex missions for multi-purpose robots, while guaranteeing correct and safe operations.

Drivers of software variability

Variability in Hardware, Mission, Environment

Software Variability

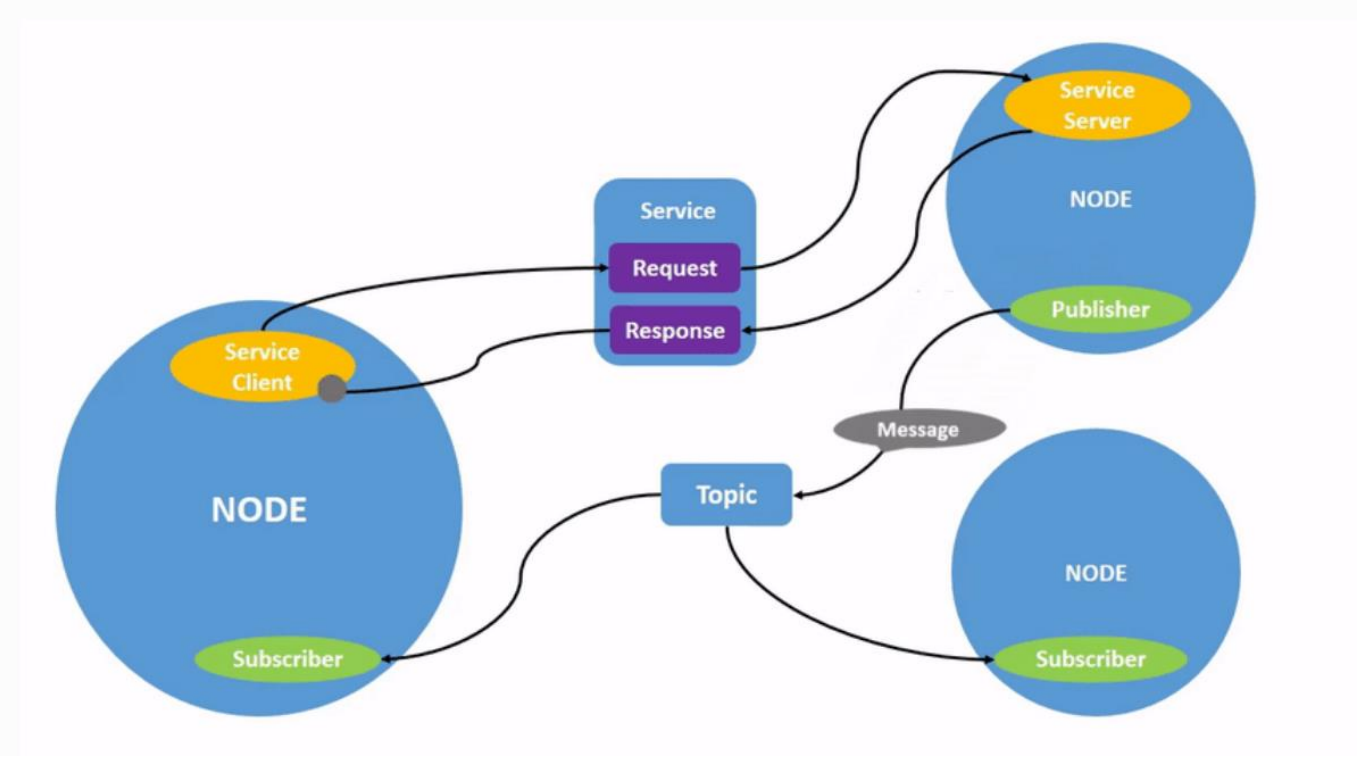


SYSTEM INFRASTRUCTURE FOR RECONFIGURATION

Definitions: Configuration

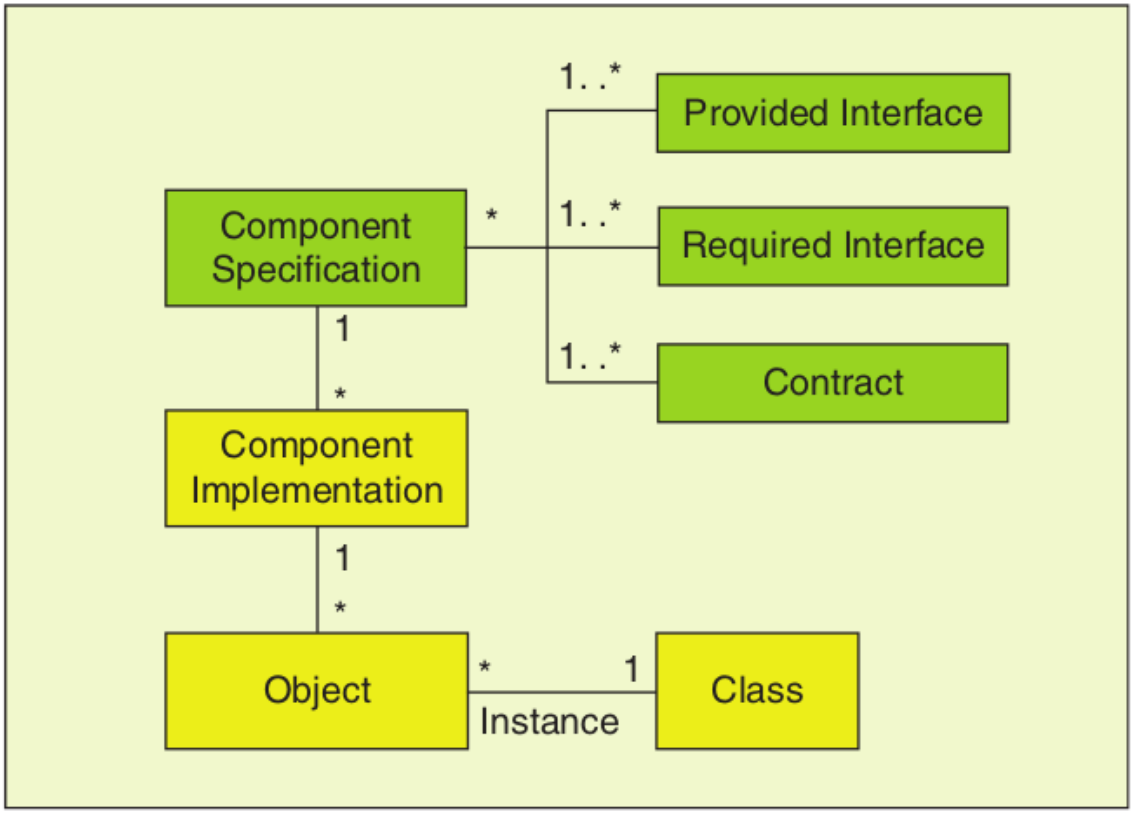
- A robotic control system is
- "an interconnection of components forming a system configuration that will provide a desired system response"

(Modern Control Systems, Dorf and Bishop, 2011)



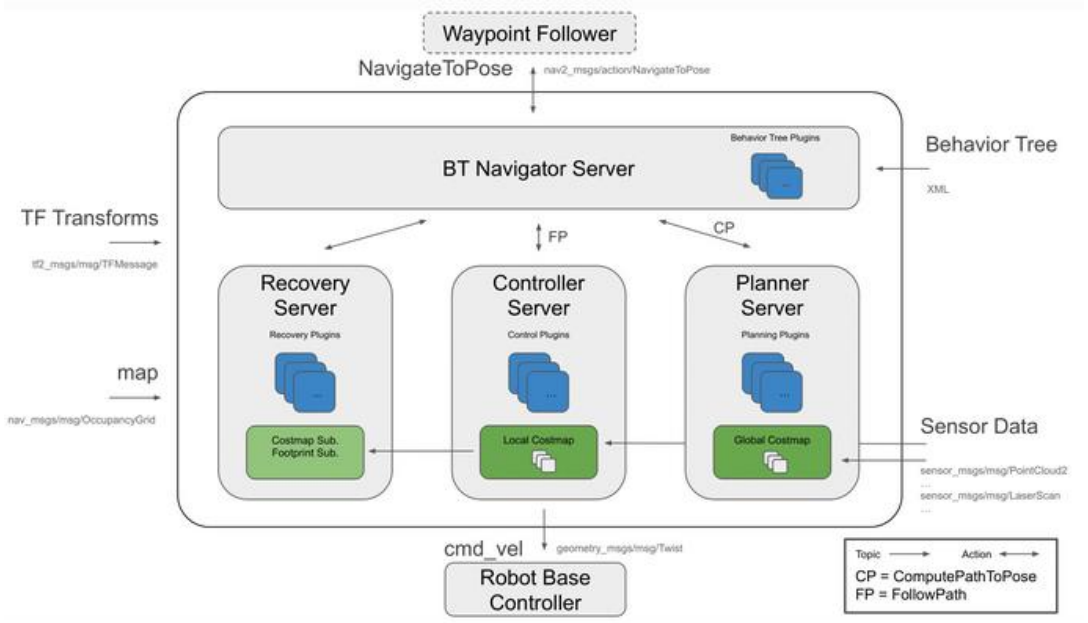
Definitions : Component

- Components are software modules designed with a well-defined interface, which is an abstraction from the details of their (possibly numerous) implementations (Brugali 2009)



Definitions: Re-configuration

- Re-configuration of the robot control system entails
  - activating or deactivating components,
  - altering their connections,
  - replacing their implementation,
  - setting new values for their parameters.



Definitions: Re-configuration

- Static configuration consists in selecting and integrating reusable software components during deployment

- Dynamic re-configuration is the robot ability to self-reconfigure at run-time according to context changes

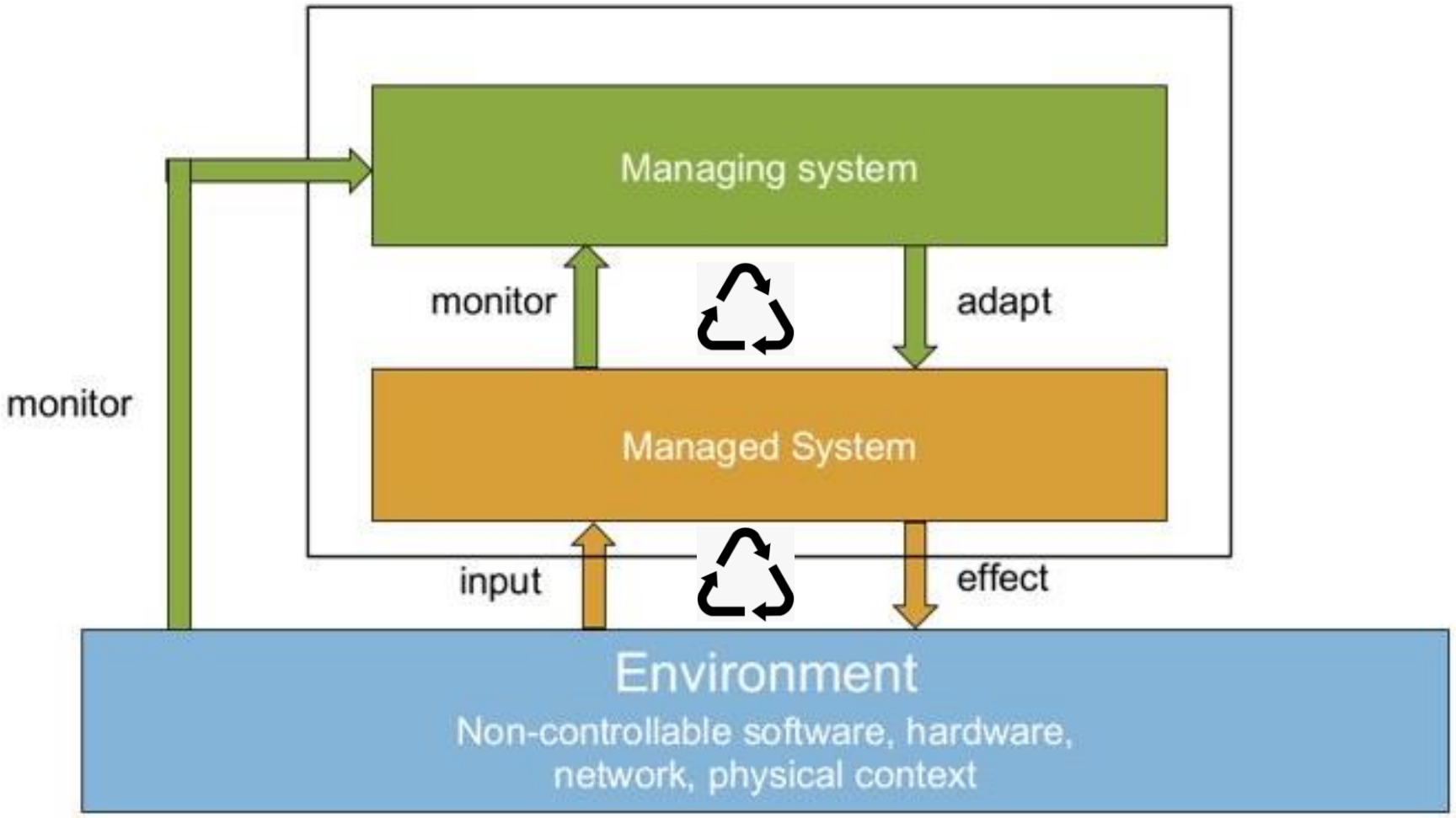
- Additional challenges
  - define constraints among the configuration options
  - define triggers of reconfiguration
  - assure that reconfiguration is timely, and puts the system into the desired and valid state of operation

Research questions

- RQ1: What software parts of a reconfigurable robot control system are reconfigured and at what granularity?

- RQ2: What mechanisms exist and how they are used for developing reconfigurable robotic software systems?

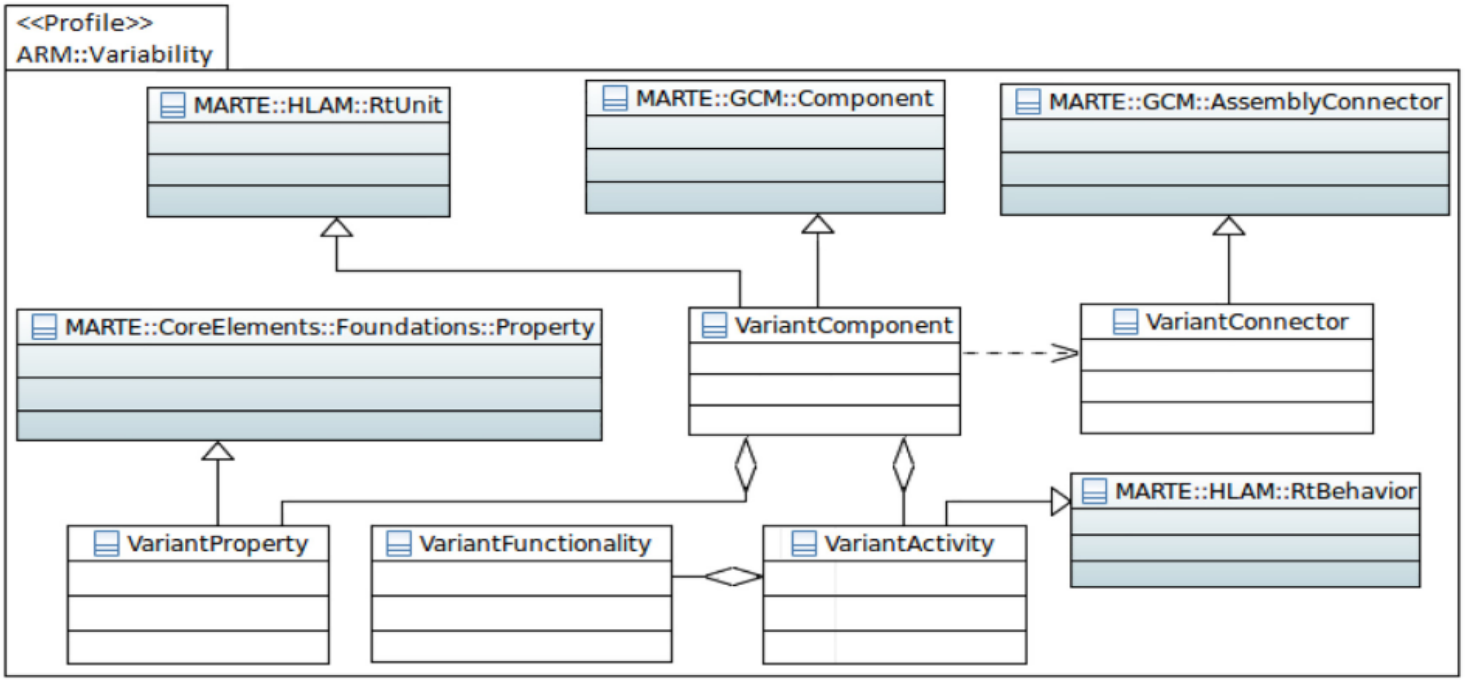
Reconfigurable Systems



Slide credit: Danny Weyns, "Tutorial: Engineering Self-Adaptive Systems - An Organized Tour"

Variability Metamodel

- What software parts of a robot control system can be reconfigured and at what granularity?



Modeling variability in self-adapting robotic systems
Davide Brugali,
Robotic and Autonomous Systems, 2023

ROS Mechanisms for Runtime Reconfiguration

C1 : activate / deactivate components

- A ROS Node is implemented as a c++ process. ROS Nodes are started using the ***roslaunch*** command-line tool that loads a textua configuration file (the launch file).

Roslaunch can be invoked programmatically by another Node (i.e. the Task Manager) :

Reconfiguration Manager

```
p = subprocess.Popen ( [ "roslaunch" ,"navigtor .launch" ] ,
                                        stdout = subprocess.PIPE)
```

Navigator

```
<launch>
        <node name="Navigator" pkg="rover_navigation" />
            <param name="max_speed" value="2" />
            <param name="controller_name" value="OmniDir" />
            <param name="gps_available" value="false" />
            <remap from="odometry" to="youbot_odometry"/>
        </node>
</launch>
```

Runtime reconfiguration of robot control systems: a ROS-based case study
Davide Brugali, IEEE Robotic Computing, 2020

ROS Mechanisms for Runtime Reconfiguration

C2 : set properties values

- The ROS framework offers the **Parameter Server**
- It is a shared dictionary that is accessible by every ROS Node.
- The dynamic_reconfigure package provides a means to update parameters at runtime without having to restart the node.

Reconfiguration Manager

```
nodehandle ->setParam("maximum speed " , max_speed);
```

Navigator

```
nodehandle ->getParam("maximum speed " , max_speed);
```

Runtime reconfiguration of robot control systems: a ROS-based case study
Davide Brugali, IEEE Robotic Computing, 2020

ROS Mechanisms for Runtime Reconfiguration

C3 : connect /disconnect components

- ROS Nodes exchange messages according to the publish/subscriber communication paradigm.
- A callback function is invoked by the ROS core when a message is received.
- A Node can activate/deactivate callback functions at runtime.

Reconfiguration Manager

```
nodehandle ->setParam("GPS" , gps_available);
```

Navigator

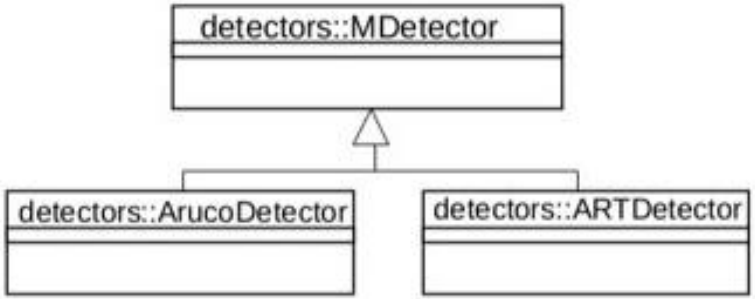
```
nodehandle ->getParam("GPS" , gps_available);

if(gps_available)
  gpssub=nodehandle->subscribe("globalPose", 10, gpsCbk ) ;
else
  gpssub.shutdown();
```

Runtime reconfiguration of robot control systems: a ROS-based case study
Davide Brugali, IEEE Robotic Computing, 2020

ROS Mechanisms for Runtime Reconfiguration

C4 : replace algorithms

- The ROS framework includes the **pluginlib package**



Reconfiguration Manager

```
nodehandle ->setParam("detector" , detectors::ArucoDetector);
```

Navigator

```
node.getParam("detector ", detector_class_name);

pluginlib::ClassLoader<detectors::MDetector>  plugin_loader("detectors", "detectors::MDetector");
try {
        boost::shared_ptr<detectors:: MDetector> marker_detector =
                                                plugin_loader.createInstance(detector_class_name);
}
catch(pluginlib::PluginlibException& ex) {ROS_ERROR("The plugin failed to load: %s", ex.what()); }

int marker_id = marker_detector ->getMarkerID();
```

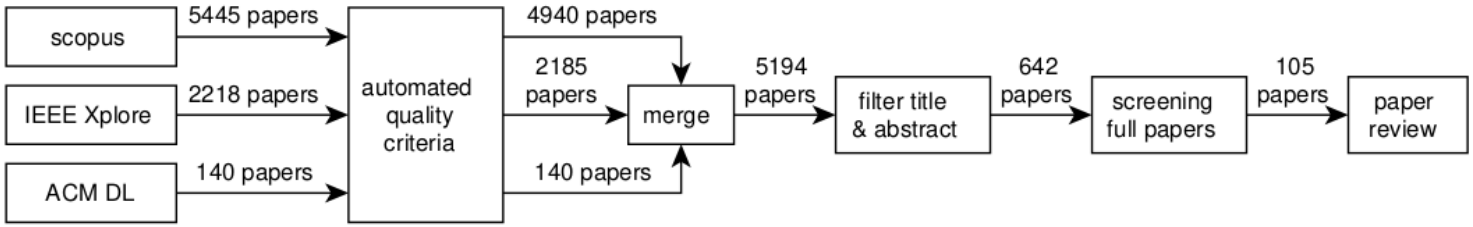
Research questions : RQ2

- What mechanisms exist and how they are used for developing reconfigurable robotic software systems?

Software Reconfiguration in Robotics

Sven Peldszus, **Davide Brugali**, Daniel Strüber, Patrizio Pelliccone, Thorsten Berger

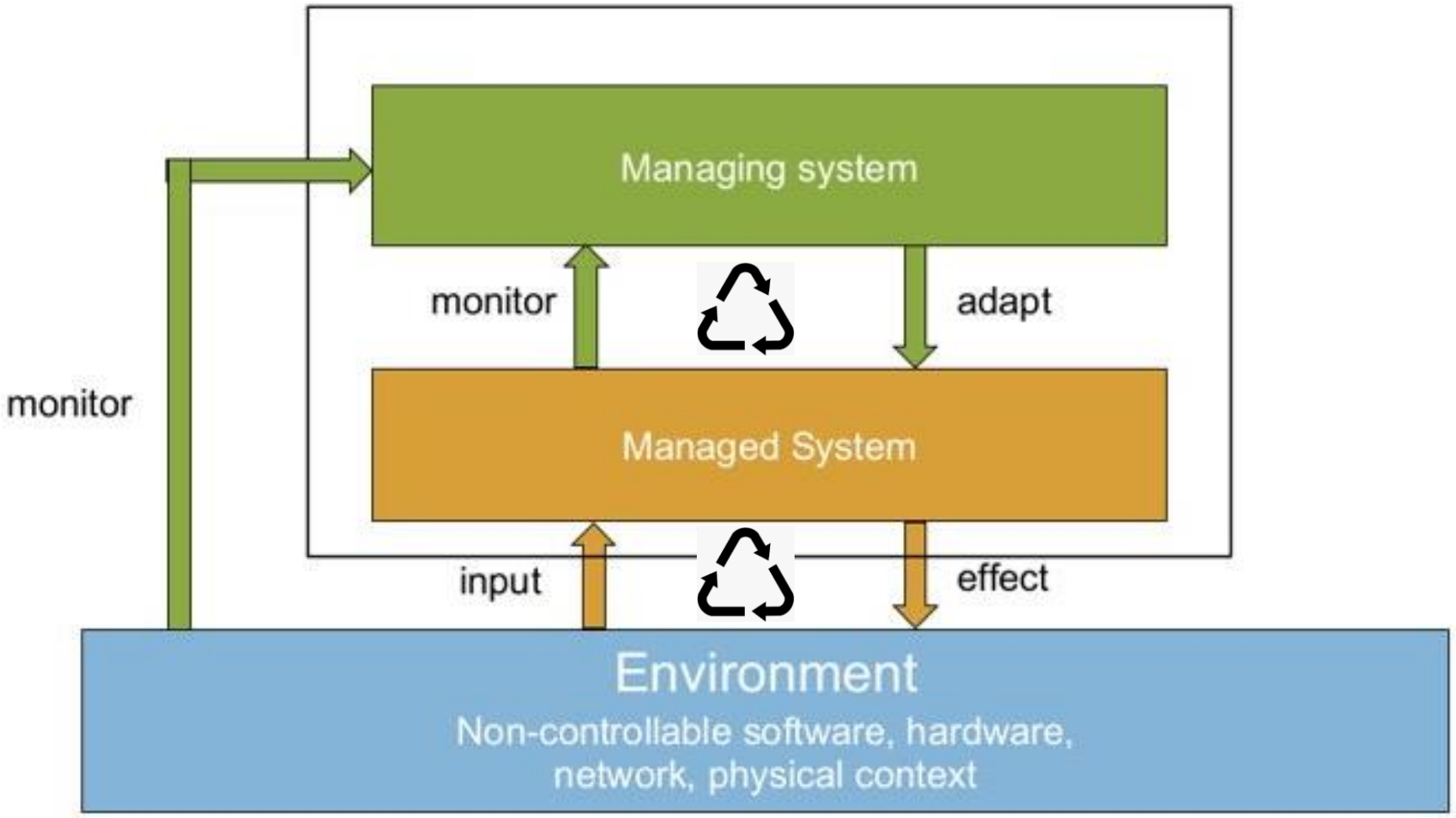
Empirical Software Engineering, 2025



	Wikis	Websites	Publications
ROS (ROS2)	wiki.ros.org docs.ros.org	www.ros.org	Quigley et al. (2009); Estefo et al. (2019); Cousins et al. (2010); Gerkey (2014)
OROCOS	docs.orocos.org	orocos.org	Bruyninckx (2001)
YARP	wiki.icub.eu/wiki/YARP	yarp.it	Metta et al. (2006) Paikan et al. (2015)
RobMoSys (SmartSoft)	robmosys.eu/wiki wiki.servicerobotik-ulm.de	robmosys.eu smart-robotics.sourceforge.net	Lotz et al. (2013a) Schlegel and Worz (1999) Schlegel et al. (2021)

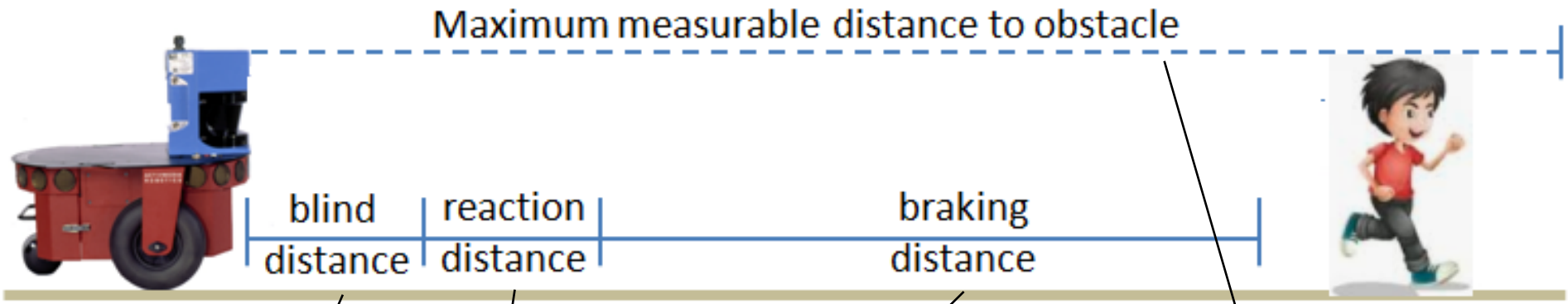
A CASE STUDY OF SELF-RECONFIGURATION AND SELF-ADAPTATION

Reconfigurable Systems



Slide credit: Danny Weyns, "Tutorial: Engineering Self-Adaptive Systems - An Organized Tour"

Safe navigation & Robot Variability



Rover speed & Sensor scan rate

Rover speed & Software System Response Time

Rover speed & Rover max deceleration

Sensor range

Workload of the computing system
- frequency of job activation
- computational cost of each job

Performance Analysis of System Variants

	Software								Hardware					Response Time (sec)
	Obstacle Avoidance		Trajectory Following		Localization		Obstacle Detection		Rover		Sensors			
	DWA	SND	DD	OD	GM	VM	AB	DB	PI	YO	HL	KDS	BSC	
	D	D	-	-	D	D	D	D	-	-	-	D	D	
$Variant_1$	x		x		x		x		x		x			2.17
$Variant_2$	x		x		x		x		x		x			2.15
$Variant_3$	x			x		x	x			x			x	1.96
$Variant_4$	x			x	x			x		x	x			2.16
$Variant_5$	x			x	x		x			x	x			3.91
$Variant_6$	x		x		x		x		x			x		2.15
$Variant_7$	x		x		x		x		x				x	2.16
$Variant_8$		x	x		x		x		x		x			2.19
$Variant_9$		x	x		x		x		x		x			2.18
$Variant_{10}$		x	x			x	x		x		x			1.98
$Variant_{11}$		x	x		x			x	x		x			2.18
$Variant_{12}$		x		x	x		x			x	x			3.89
$Variant_{13}$		x	x		x		x		x			x		2.17
$Variant_{14}$		x	x		x		x		x				x	2.16

Model-based development of QoS-aware Reconfigurable Autonomous Robotic Systems

Davide Brugali, Raffaela Mirandola, Rafael Capilla, Catia Trubiani, IEEE Robotic Computing 2018

Hospital Logistics Scenario

- Scenario: urgent delivery
  - the robot used is OmniDirectional
  - the robot transports medical items
  - the corridors are not crowded
  - obstacles are detected with a stereo camera

	Software								Hardware					Response Time (sec)
	Obstacle Avoidance		Trajectory Following		Localization		Obstacle Detection		Rover		Sensors			
	DWA	SND	DD	OD	GM	VM	AB	DB	PI	YO	HL	KDS	BSC	
	D	D	-	-	D	D	D	D	-	-	-	D	D	
$Variant_3$	x			x		x	x			x			x	1.96

Max speed = 1.5 m/s

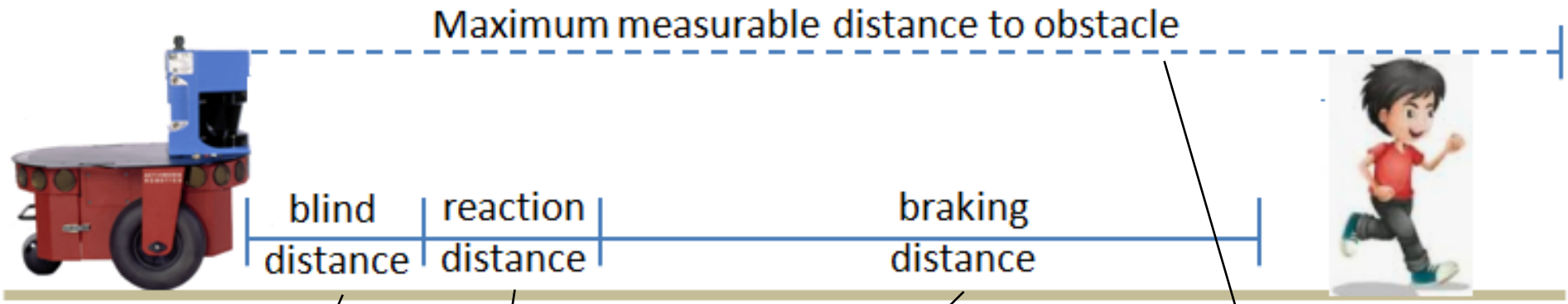
Self-Reconfiguration

- Scenario: urgent delivery
  - the robot used is OmniDirectional
  - the robot transports medical it
  - the corridors are not crowded
  - ~~obstacles are detected with a stereo camera~~
  - Use a laser range finder and a geometric map

SCARCE ILLUMINATION

	Software								Hardware					Response Time (sec)
	Obstacle Avoidance		Trajectory Following		Localization		Obstacle Detection		Rover		Sensors			
	DWA	SND	DD	OD	GM	VM	AB	DB	PI	YO	HL	KDS	BSC	
	D	D	-	-	D	D	D	D	-	-	-	D	D	
$Variant_4$	x			x	x			x		x	x			2.16

Self-Adaptation



Rover speed
&
Sensor scan rate

Rover speed
&
Software System Response Time

Rover speed
&
Rover max deceleration

Sensor range

Self-Adaptation

Increased Response-Time
Reduced sensor rate and range

=> Reduce rover speed

Thank you for your attention

UNIBG

Davide Brugali

University of Bergamo, Italy