SW-Architektur und –Integration für ROS-basierte Produkte

Dr.-Ing. Ingo Lütkebohle, Bosch

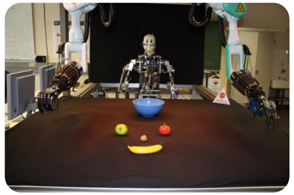
Für die ROSCon Deutschland, 23. November 2023

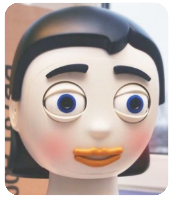
Agenda

- Intro
- Open Source Management
- Practices for teams
- Scaling
- Diagnostics
- Testing

Dr.-Ing. Ingo Lütkebohle

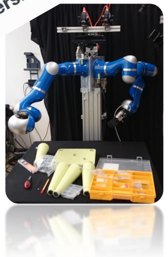
UNIVERSITÄT BIELEFELD





2005 – 2013

Universität Stuttgart



2010 & 2013

BOSCH

µROS

2014-

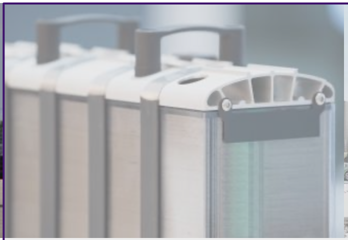
From system integration to dependability

From HRI to embedded systems

Bosch Research Focus Areas

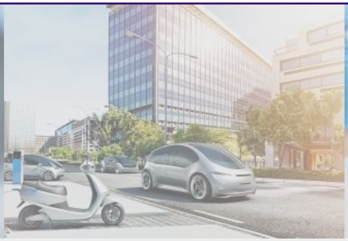
Autonomous Intelligent Driving (AID)

Sensing, perception, prediction, planning
Systems & infrastructure for L2-L4



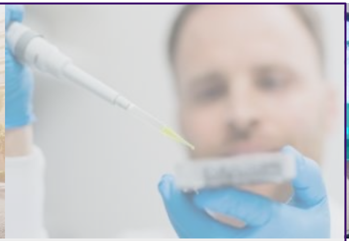
Chemical Energy Converters (CEC)

Hydrogen electrolysis
PEM fuel cell for EVs
Solid Oxide Fuel Cell



Electrified Mobility and Systems (EMY)

Electric drives
Power electronics
Integrated electrified products



Healthcare Solutions (HCS)

Point-of-care lab diagnostics
Liquid biopsy
Next gen sequencing

IoT @ Life (IOT)

Enabling AIoT for mobile, residential, tools and multi-domain applications

Production Systems (PRS)

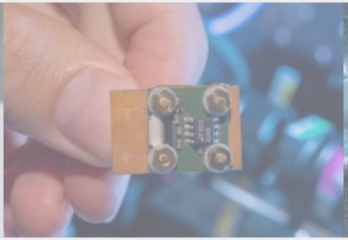
Production technologies
AI in production
Internet of production

Robotic Systems (ROB)

Consumer robots
Industrial service robots
Industrial robot arms

Sustainability (SST)

Sustainability big picture
Climate change mitigation
Circular economy
Carbon dioxide removal



Smart Sensors & HW Systems (SSY)

MEMS
Quantum sensors
HMI technology
Smart systems
Embedded AI hardware

Artificial Intelligence Methods (AIM)

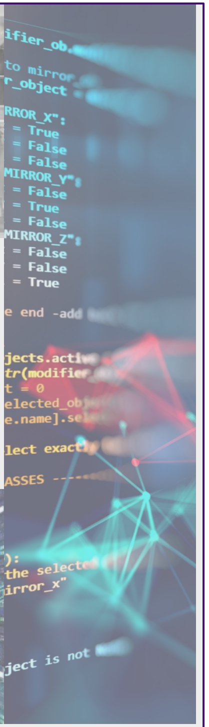
AI data loop enablers
Natural language processing
Computer vision
AI method incubator

Information and Communication Tech (ICT)

Software & systems engineering
Distributed infrastructure
Intelligent functions & services

Modeling, Simulation, Optimization (MSO)

Sustainable engineering
Virtual product design
Virtual validation
Use phase monitoring



Industrialization of AI & SW (BIS)

Providing mature AI & SW artefacts

Bosch’s 12 Year Journey with ROS

Bosch participates
in PR2 beta program

ROS industrial

Founding member
of ROS-I Europe

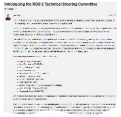
Bosch Research and Technology Center Joins Forces with Open Source Robotics Foundation to Advance the Development of ROS

October 7, 2016 by Tully Foote

Bosch Underwrites Full-Time Developer for ROS 2 Research

A full-time developer sponsored by Bosch's Research and Technology Center in North America will begin working with Open Source Robotics Foundation (OSRF) this month to advance the development of Robotics Open Source 2, the two organizations announced today.

Bosch sponsors
development of ROS 2



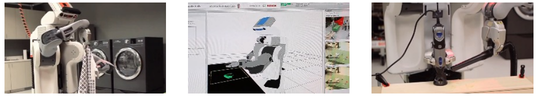
Founding member
of ROS TSC

EU project
micro-ROS

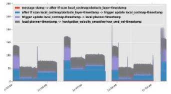
EU ITP
MROS

EU project
CONVINCE

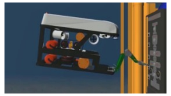
2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022



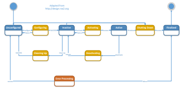
Tools and basic algorithms from PR2 beta program



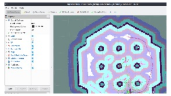
Tracepoints



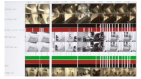
UUV simulator



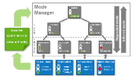
Node lifecycle



rviz2



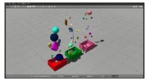
rosbag2



System Modes

fmi Functional Mock-Up Interface

fmi_adapter



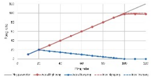
pcg_gazebo

ROS

Client library for C

iceoryx

Middleware adapter

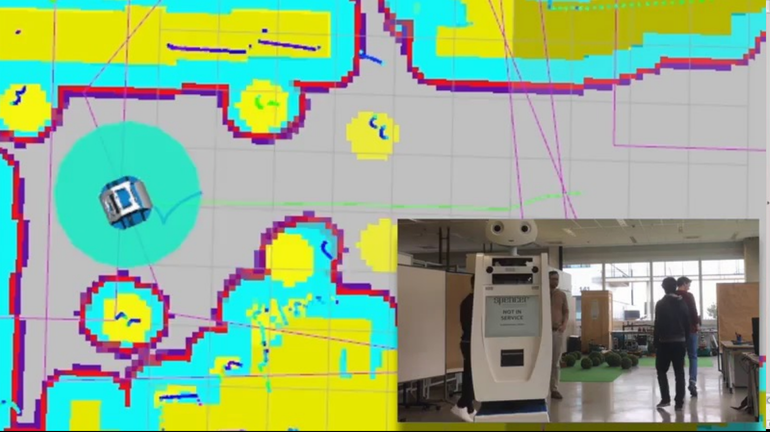


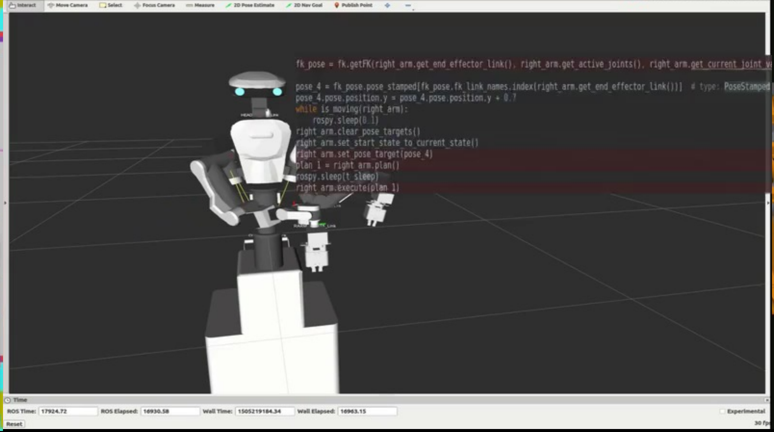
Refined Executor

ros2_control

Diagnostics

From a small research team to hundreds of developers using ROS

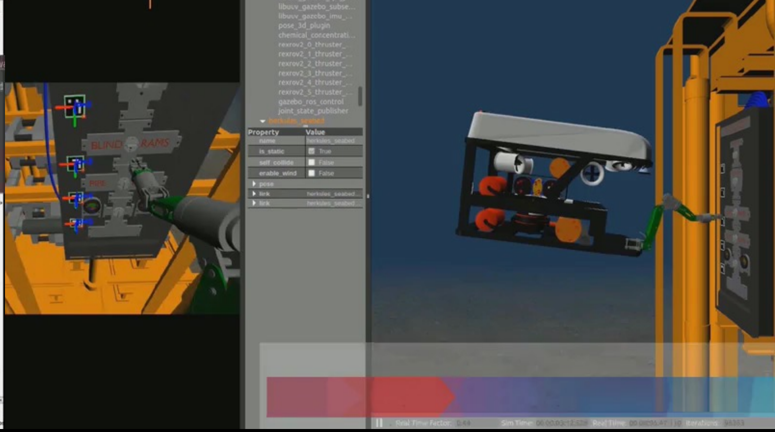


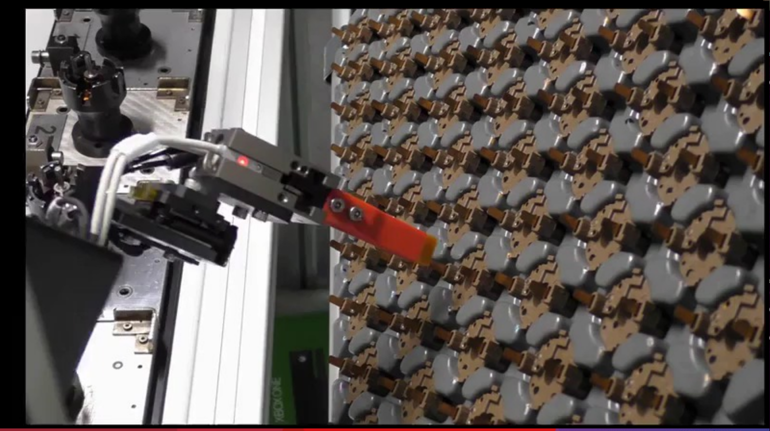


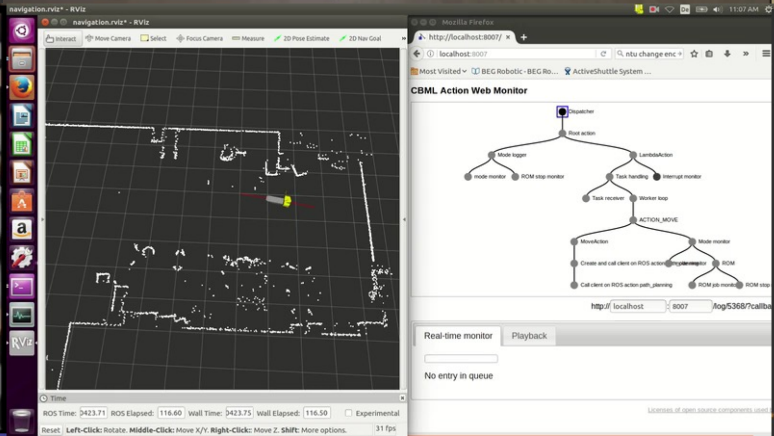
```
fk_pose = fk.getFK(right_arm.get_end_effector_link(), right_arm.get_active_joints(), right_arm.get_current_joint_va

pose_4 = fk_pose.pose_stamped[fk_pose.fk_link_names.index(right_arm.get_end_effector_link())]  # type: PoseStamped
pose_4.pose.position.y = pose_4.pose.position.y + 0.7
while is_moving(right_arm):
    rospy.sleep(0.1)
right_arm.clear_pose_targets()
right_arm.set_start_state_to_current_state()
right_arm.set_pose_target(pose_4)
plan_1 = right_arm.plan()
rospy.sleep(t_sleep)
right_arm.execute(plan_1)
```

ROS Time: 17924.72 ROS Elapsed: 16930.58 Wall Time: 1505219184.34 Wall Elapsed: 16963.15







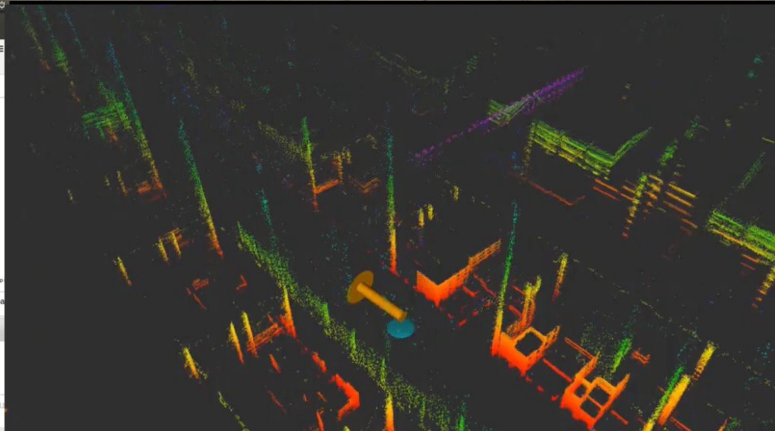
ROS Time: 3423.71 ROS Elapsed: 116.60 Wall Time: 3423.75 Wall Elapsed: 116.50

Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click: Move Z. Shift: More options.

CBML Action Web Monitor

http:// localhost : 8007 /log/5368/?callba

No entry in queue



Open Source Management

What to Open Source and what not

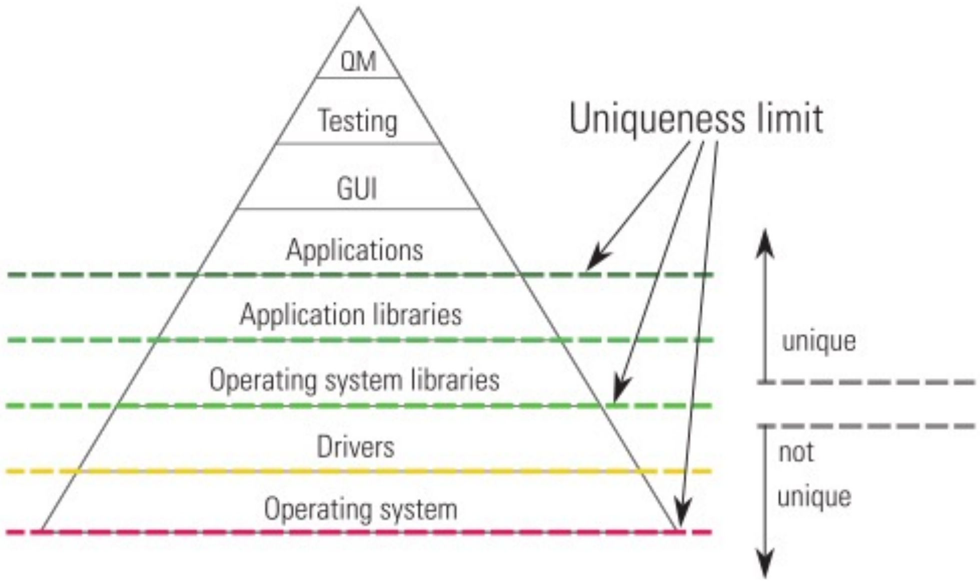
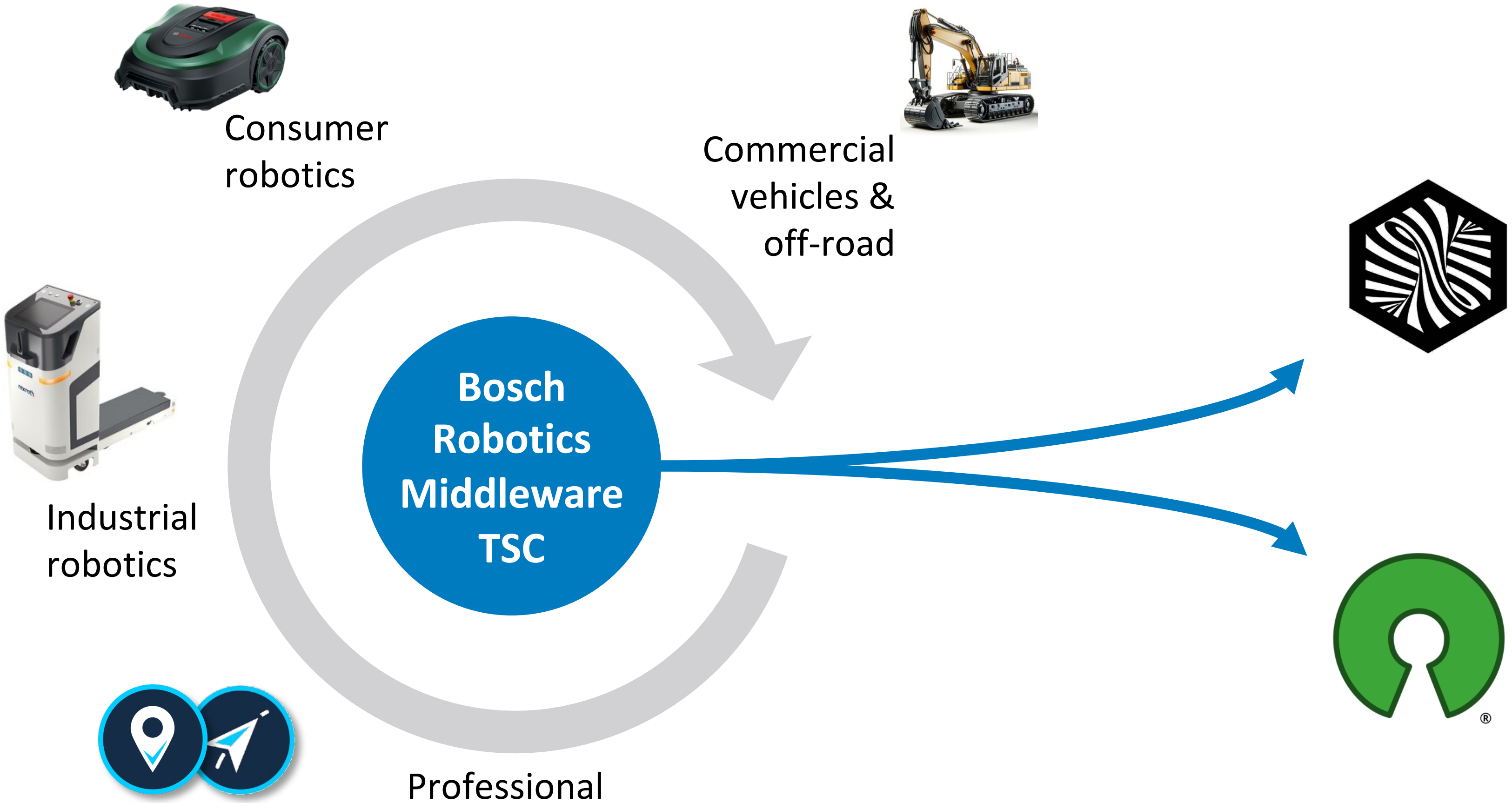


Image Source: OSADL
https://www.osadl.org/fileadmin/dam/pictures/paperless/Whitepaper-Free-and-Open-Source-Software-FOSS-ENG-Preview.png

- Common strategy for embedded companies
- Our limits
  - ROS 2 framework is not unique
  - Nav 2 *framework* is not unique
  - Motion planning: Largely unique
  - Perception algorithms: Largely unique (on top of OpenCV etc)
  - Deliberation (e.g., behavior trees): Unique
  - A lot of optimization is unique

Steering of Our ROS Strategy and Infrastructure



Consumer robotics

Commercial vehicles & off-road

Industrial robotics

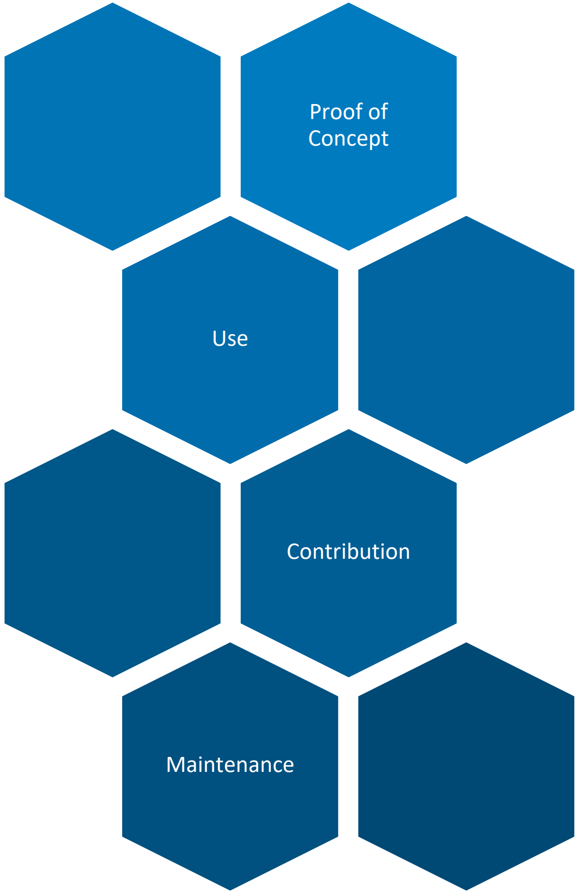
Professional service robotics

Bosch Robotics Middleware TSC

Inner source

Open source

Working with OSS



- Expectation management
  - What can others reasonably expect
- Intellectual property
  - Implicit patent licenses
- License compliance
  - Compatibility, copyleft
  - Ensuring licenses are correct

  → Training & Tooling
- Maintenance
  - Disclaiming liability is not always possible
  - Maintenance comes with responsibilities

  → We prefer contribution

Practices for team alignment

From individual to team

Essentials for effective team-work

- Shared tools
- Shared understanding
- Shared practices

Shared Tools

ROS!

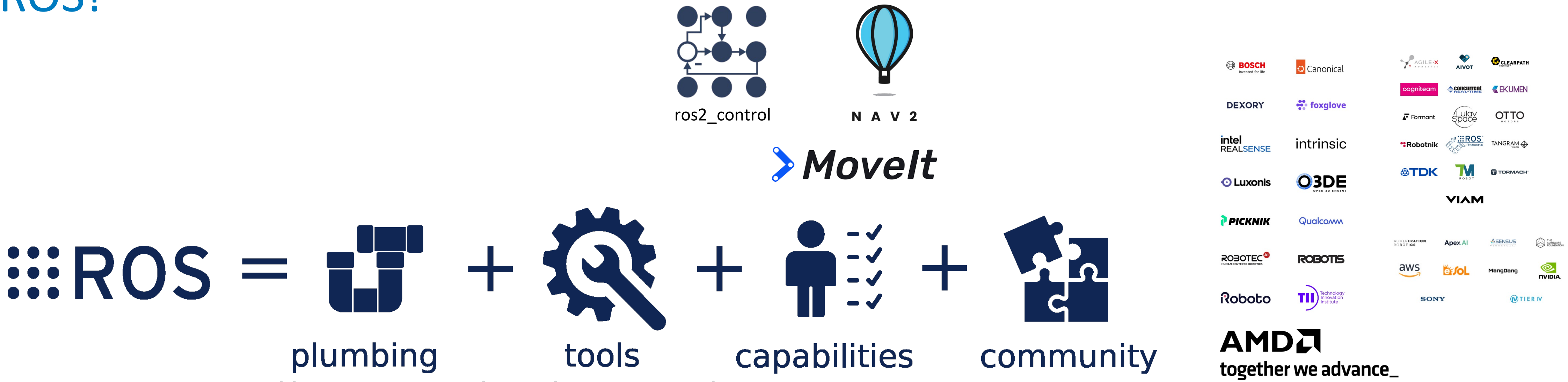


Image Source: https://www.ros.org/blog/ecosystem/

Shared Tools

DevOps Tooling (and a lot of custom tooling)

docker

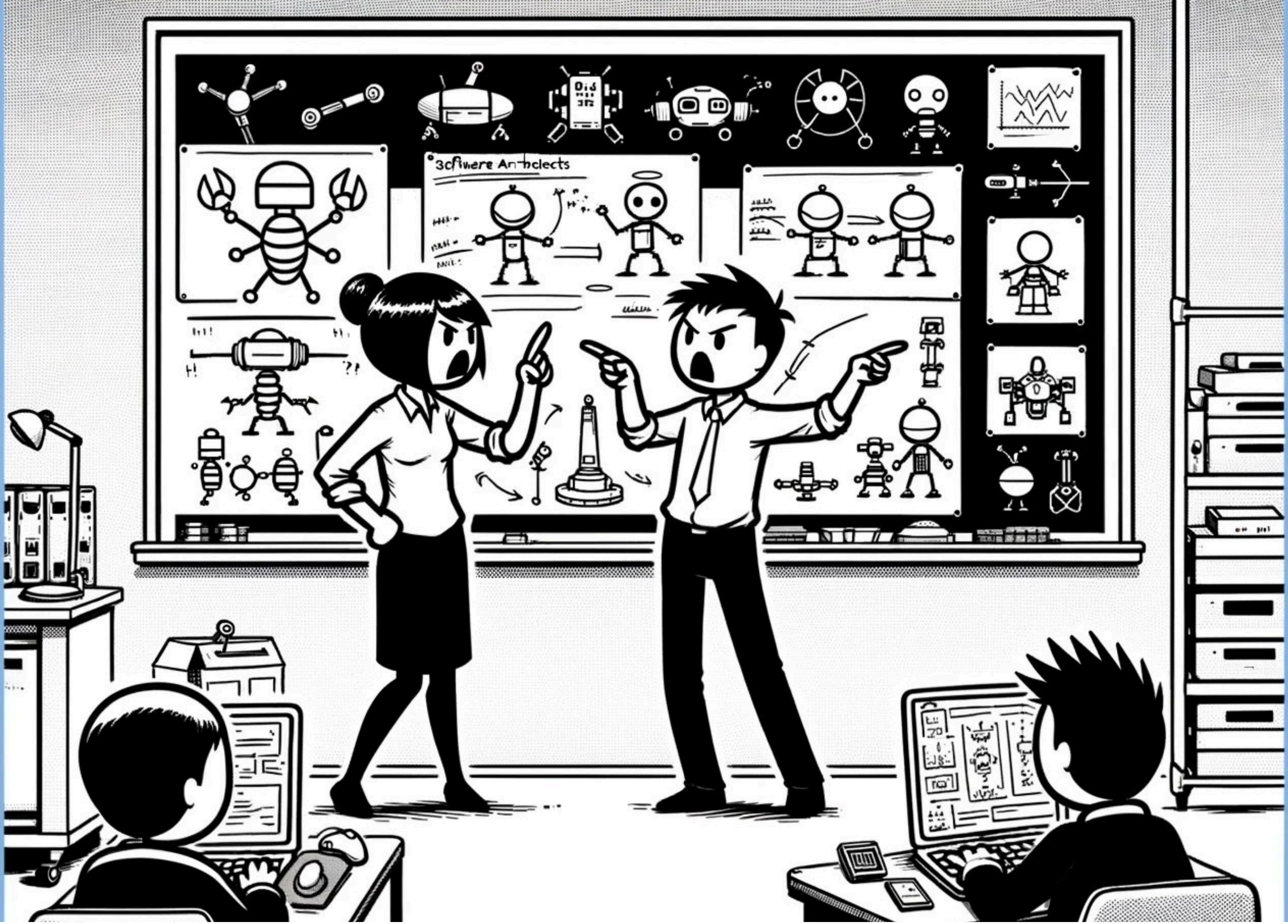
GAZEBO

GitHub

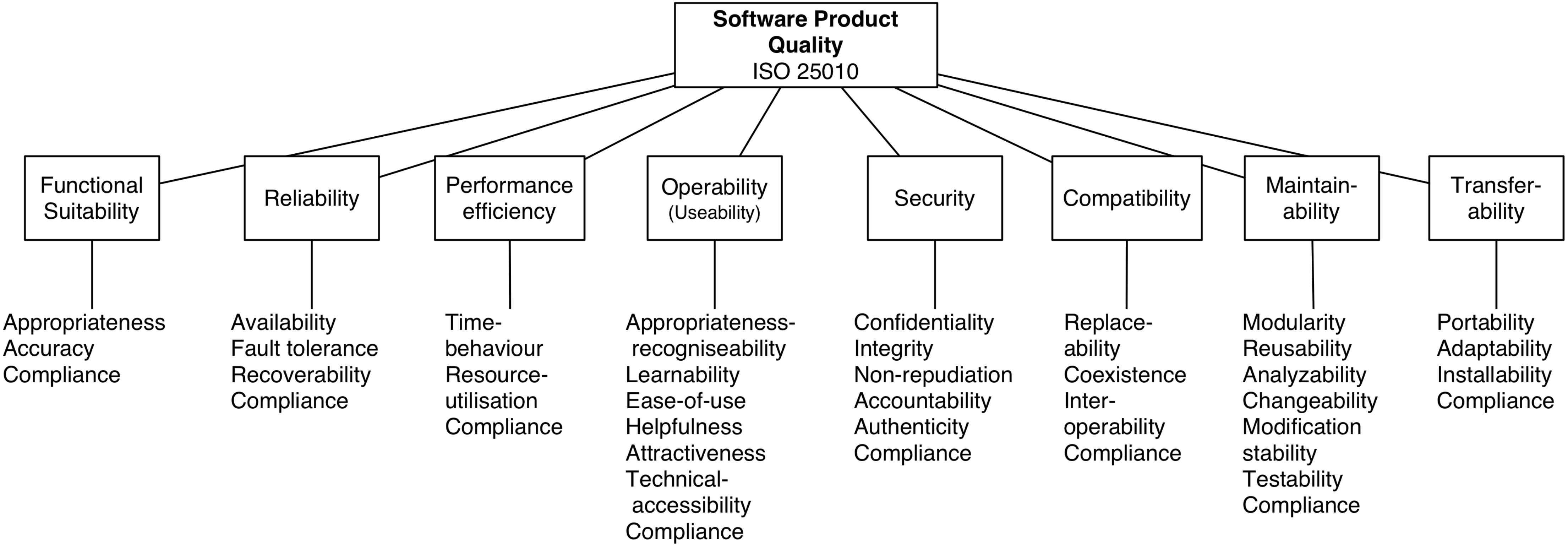
ANSIBLE

sphinx-needs

Jenkins


3cfiwere Anhclects

Product Qualities

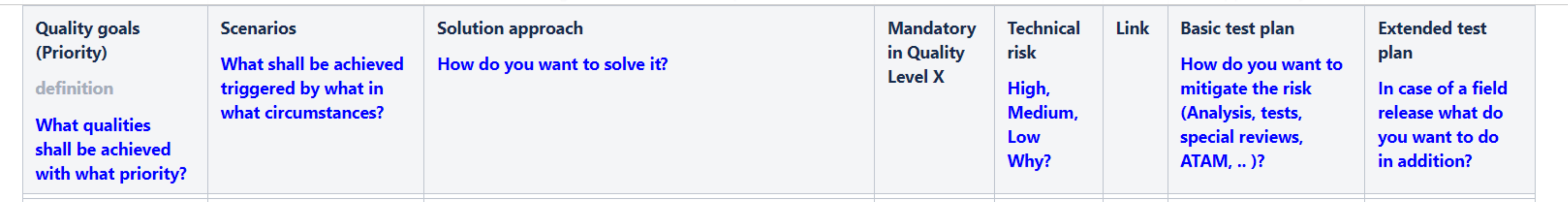


Software Product Quality
ISO 25010

- Functional Suitability
  - Appropriateness
  - Accuracy
  - Compliance
- Reliability
  - Availability
  - Fault tolerance
  - Recoverability
  - Compliance
- Performance efficiency
  - Time-behaviour
  - Resource-utilisation
  - Compliance
- Operability (Useability)
  - Appropriateness-recogniseability
  - Learnability
  - Ease-of-use
  - Helpfulness
  - Attractiveness
  - Technical-accessibility
  - Compliance
- Security
  - Confidentiality
  - Integrity
  - Non-repudiation
  - Accountability
  - Authenticity
  - Compliance
- Compatibility
  - Replaceability
  - Coexistence
  - Interoperability
  - Compliance
- Maintainability
  - Modularity
  - Reusability
  - Analyzability
  - Changeability
  - Modification stability
  - Testability
  - Compliance
- Transferability
  - Portability
  - Adaptability
  - Installability
  - Compliance

Making Qualities Tangible and Assessable

- Quality Scenarios „What do we need in our software to be sucessfull?“



Quality goals (Priority) definition What qualities shall be achieved with what priority?	Scenarios What shall be achieved triggered by what in what circumstances?	Solution approach How do you want to solve it?	Mandatory in Quality Level X	Technical risk High, Medium, Low Why?	Link	Basic test plan How do you want to mitigate the risk (Analysis, tests, special reviews, ATAM, ..)?	Extended test plan In case of a field release what do you want to do in addition?

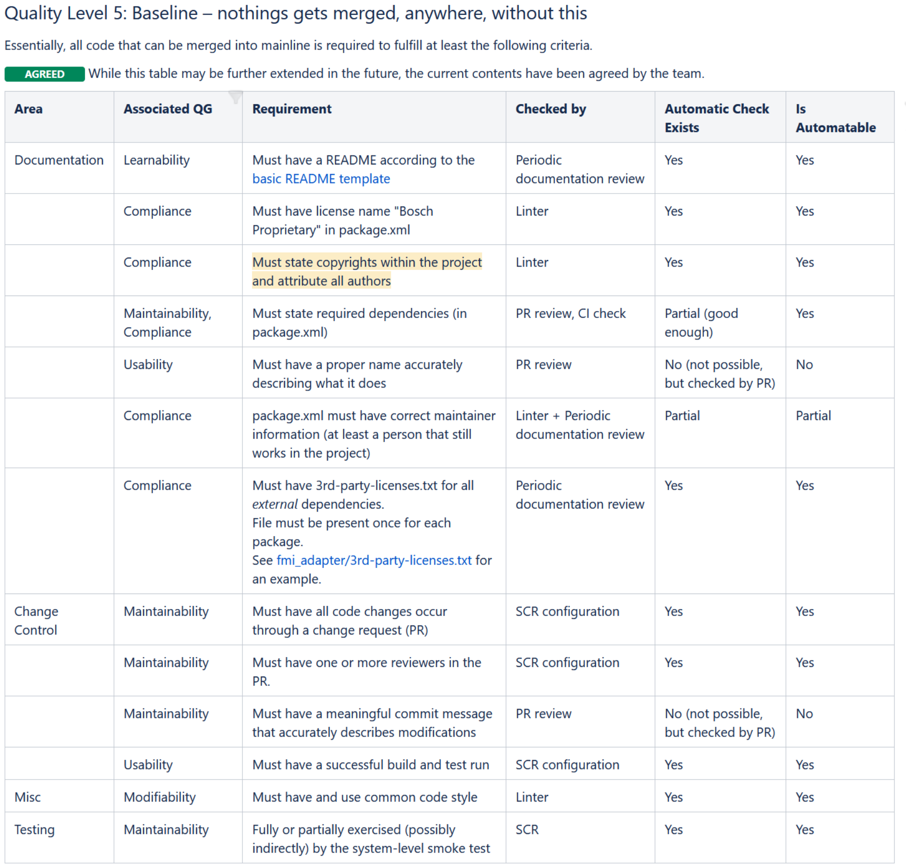
- Architecture fitness functions
  „ Any mechanism that performs an objective integrity assessment of some architecture characteristic or combination of architecture characteristics.” (“Evolutionary Architecture”, Ford et al, 2017)
- Examples given later

From individual to team

Quality Levels: *Agreement* on practices

- ROS 2 added „Package Quality Categories“
  - REP 2004
- Level 1: Production
- Level 4: Demos, tutorials experiments
- Level 5: Baseline

- We adopted and added to this →



Quality Level 5: Baseline – nothings gets merged, anywhere, without this

Essentially, all code that can be merged into mainline is required to fulfill at least the following criteria.

AGREED While this table may be further extended in the future, the current contents have been agreed by the team.

Area	Associated QG	Requirement	Checked by	Automatic Check Exists	Is Automatable
Documentation	Learnability	Must have a README according to the basic README template	Periodic documentation review	Yes	Yes
	Compliance	Must have license name "Bosch Proprietary" in package.xml	Linter	Yes	Yes
	Compliance	Must state copyrights within the project and attribute all authors	Linter	Yes	Yes
	Maintainability, Compliance	Must state required dependencies (in package.xml)	PR review, CI check	Partial (good enough)	Yes
	Usability	Must have a proper name accurately describing what it does	PR review	No (not possible, but checked by PR)	No
	Compliance	package.xml must have correct maintainer information (at least a person that still works in the project)	Linter + Periodic documentation review	Partial	Partial
	Compliance	Must have 3rd-party-licenses.txt for all *external* dependencies. File must be present once for each package. See fmi_adapter/3rd-party-licenses.txt for an example.	Periodic documentation review	Yes	Yes
Change Control	Maintainability	Must have all code changes occur through a change request (PR)	SCR configuration	Yes	Yes
	Maintainability	Must have one or more reviewers in the PR.	SCR configuration	Yes	Yes
	Maintainability	Must have a meaningful commit message that accurately describes modifications	PR review	No (not possible, but checked by PR)	No
	Usability	Must have a successful build and test run	SCR configuration	Yes	Yes
Misc	Modifiability	Must have and use common code style	Linter	Yes	Yes
Testing	Maintainability	Fully or partially exercised (possibly indirectly) by the system-level smoke test	SCR	Yes	Yes

From individual to team

Adopt DevOps Practices

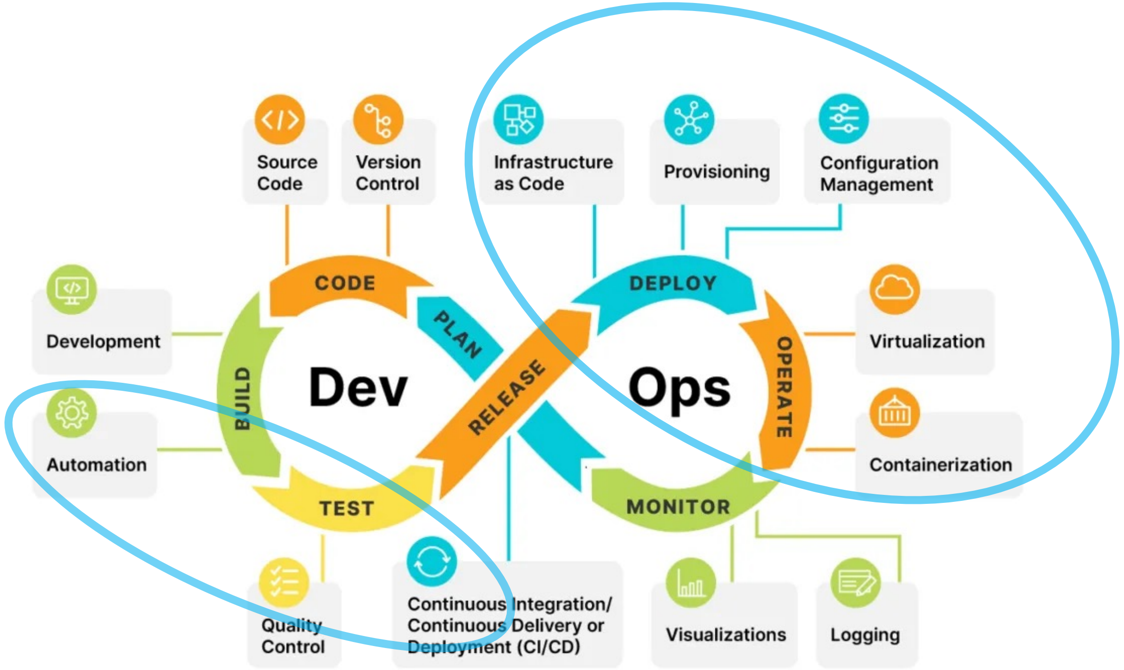
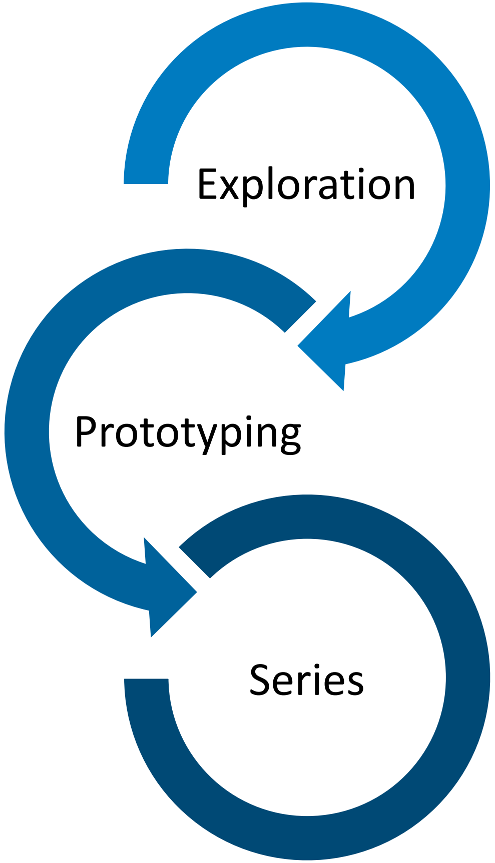


Image Source: https://productcoalition.com/12-top-devops-best-practices-for-a-successful-transition-in-2023-b73b54014d0d

- Getting rid of „it works for me“
  - Automated devel environment setup
  - Automated deployment
  - Configuration management
- Enabling refactoring
  - Automatic quality control

Scaling

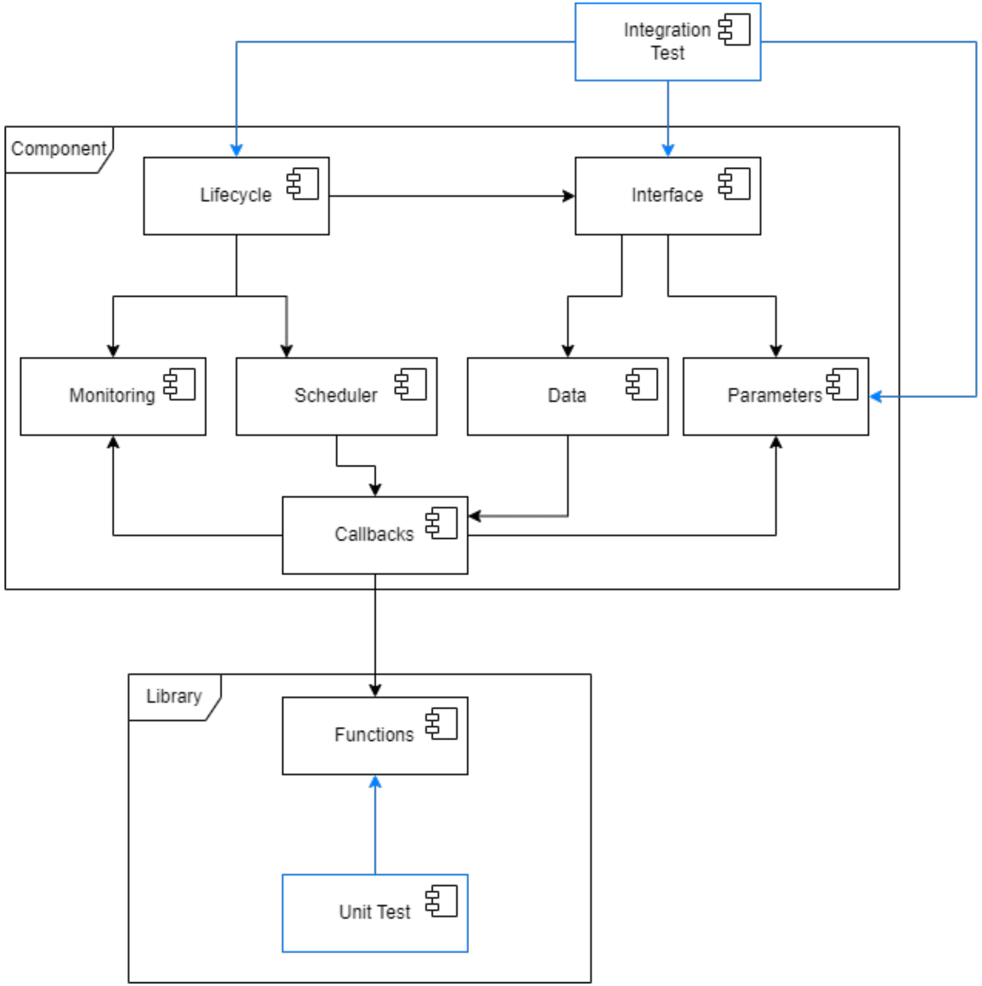
Overarching phases for novel products



- Exploration: Before deciding to do it
  - Something like 10% time
- Prototyping: Small team builds a core
  - Can be stopped at any time
- Series: Ramp-up to full product setup
  - Can still be stopped, but less likely

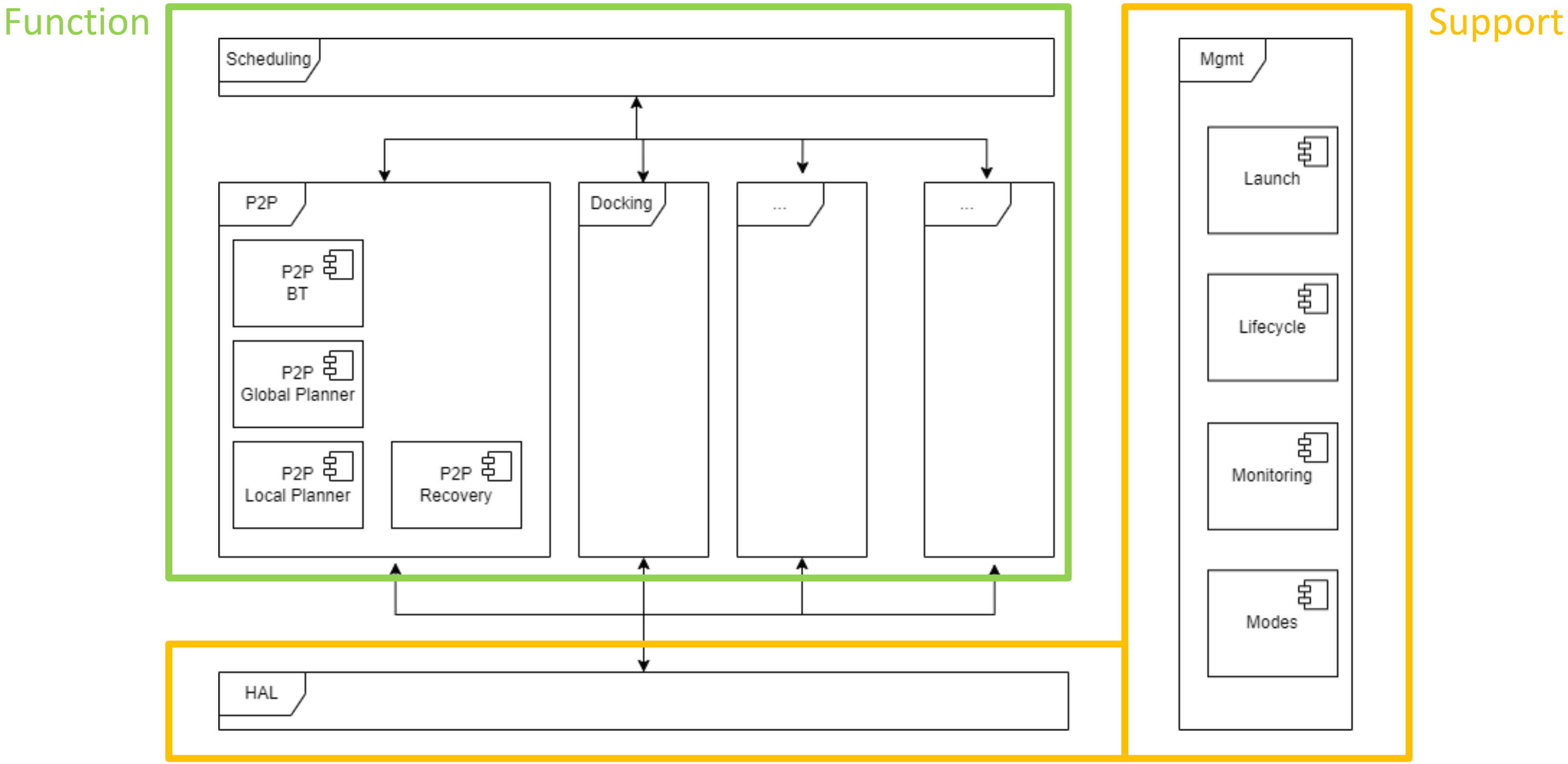
Functions → Components

- Architecture must partially be ahead
  - Essential aspects present in common frameworks
- Separation of Concerns
  - Modularize for testability
  - Separate function and component
  - Factor out execution, bringup, monitoring

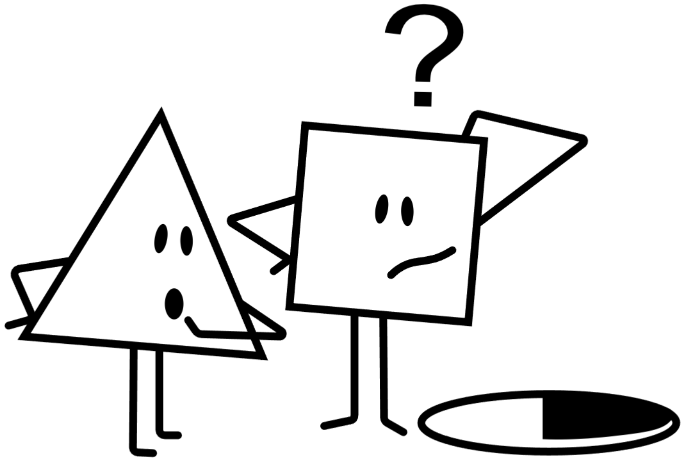


BOSCH

Robot Software Architecture: Within device



The Integration Problem



Created by R Diepenheim
from the Noun Project

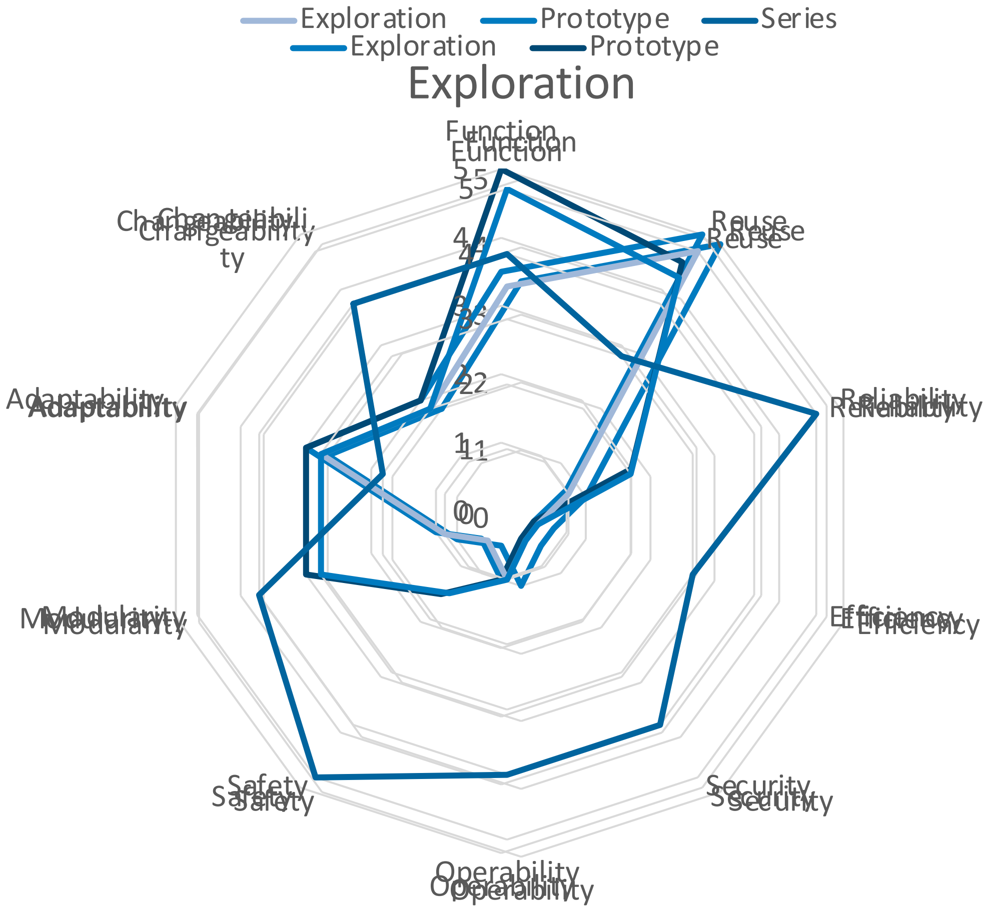
- Software Integration as well as SW/HW Integration have long been recognized as major challenges
- Major causes
  - Mismatched interfaces
  - Implicit assumptions / bad documentation
  - Bad software quality in components
  - “silo-thinking”
  - Lack of competences
- Main strategies:
  1. Reduce complexity
  2. Start small, grow judiciously and consciously (iteration…)
  3. Use supporting frameworks

Recurring issue categories in our projects

- Distributed systems
  - Lack of synchronization
  - Lossy networks
  - In DDS: Badly connected remote subscribers pulling down local communication
- Interfaces
  - Misuse of message fields
  - Static indexes for named arrays
  - Wrong reference frames
  - Time-sync causing jumps
- Complex system descriptions
  - Cf. complex, hierarchical launch issues mentioned on Monday
- Hardware
  - Variability
  - Hardware changes not communicated
  - Hardware different from simulation
- Quality
  - Bad timestamping in drivers
  - Lack of lifecycle in drivers
  - Drivers using binary-only SDKs
- Dependency management
  - Lack of version pinning
  - Difficulty in upstreaming
  - Sheer number of dependencies

Process Overview

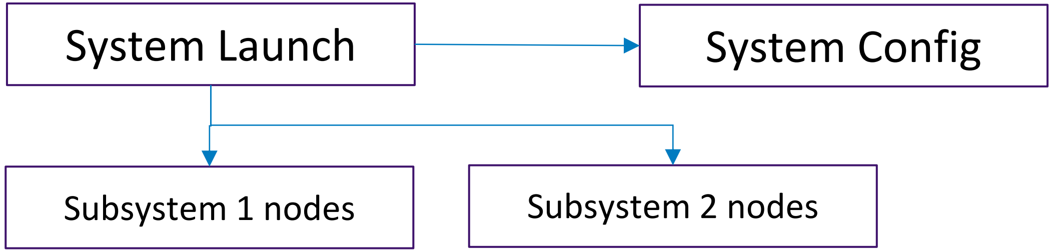
Quality characteristics change



Simple bringup helps: 2-layer Launch

- Many ROS systems use a deep launch-file inclusion hierarchy
  - This causes a great deal of duplication for passing arguments
- Much content in launch files is also for handling arguments
  - Unecessary complexity

- 2-layer Launch
  - Bottom layer: Nodes for one subsystem
  - Top layer: Only includes from bottom layer
  - All arguments in single YAML file


System Launch

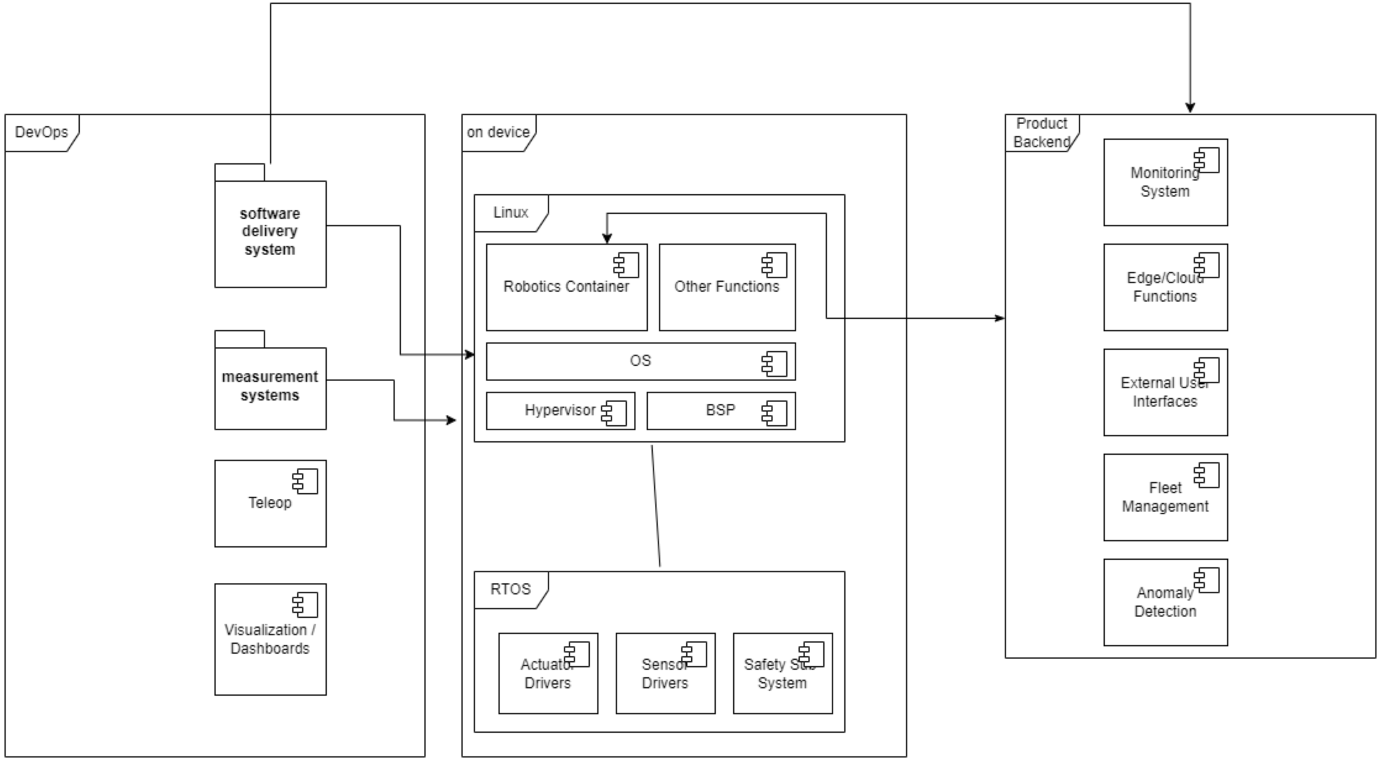
System Config

Subsystem 1 nodes

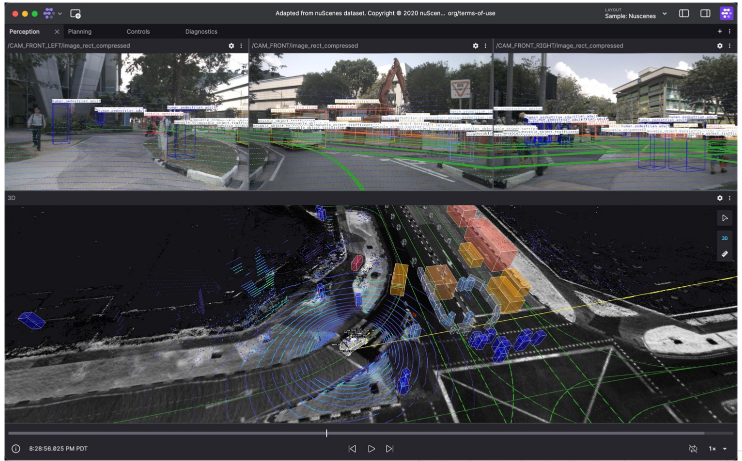
Subsystem 2 nodes

Digitisation in robotics

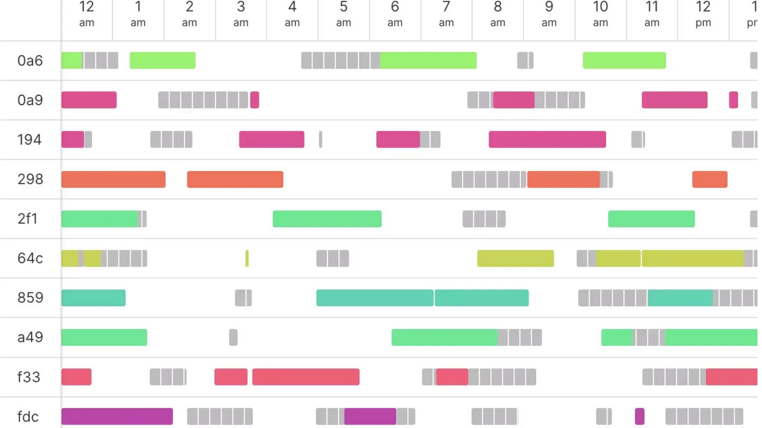
- Robotics is now, more than ever, an eco-system of eco-systems
  - Machine learning-based functions
  - Advanced, model-based control
  - Monitoring and analytics
  - Edge/Cloud-based functions
  - Novel programming languages (e.g., Rust)
  - DevOps-oriented delivery



Local Bag Storage → Data Management Platforms



Adapted from nuScenes dataset. Copyright © 2020 nuScen... org/terms-of-use
Sample: Nuscenes
Perception
Planning
Controls
Diagnostics
/CAM_FRONT_LEFT/image_rect_compressed
/CAM_FRONT/image_rect_compressed
/CAM_FRONT_RIGHT/image_rect_compressed
3D
8:28:56.025 PM PDT
1x



foxglove

Image Source: https://foxglove.dev/

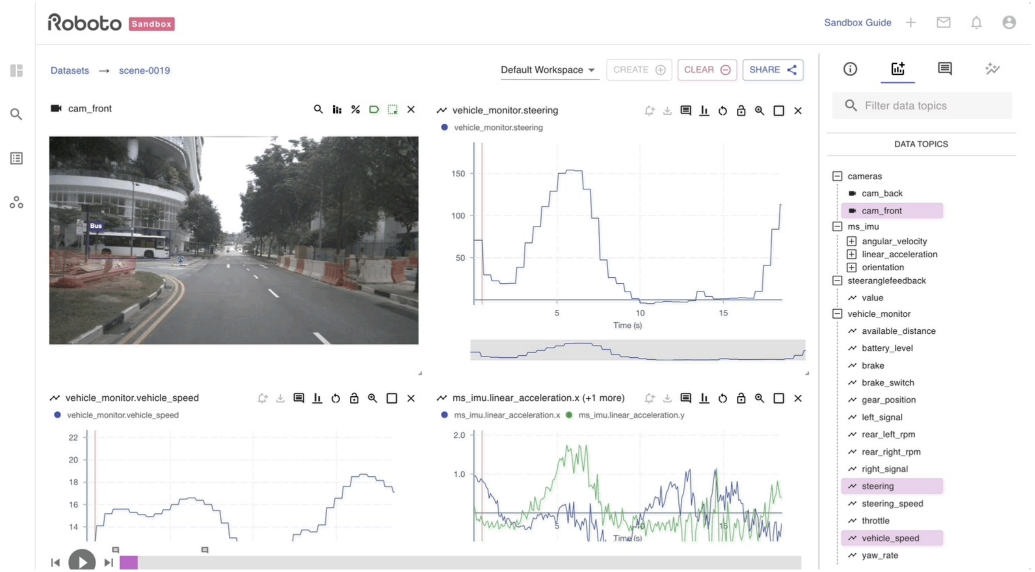


Image Source: https://www.roboto.ai/

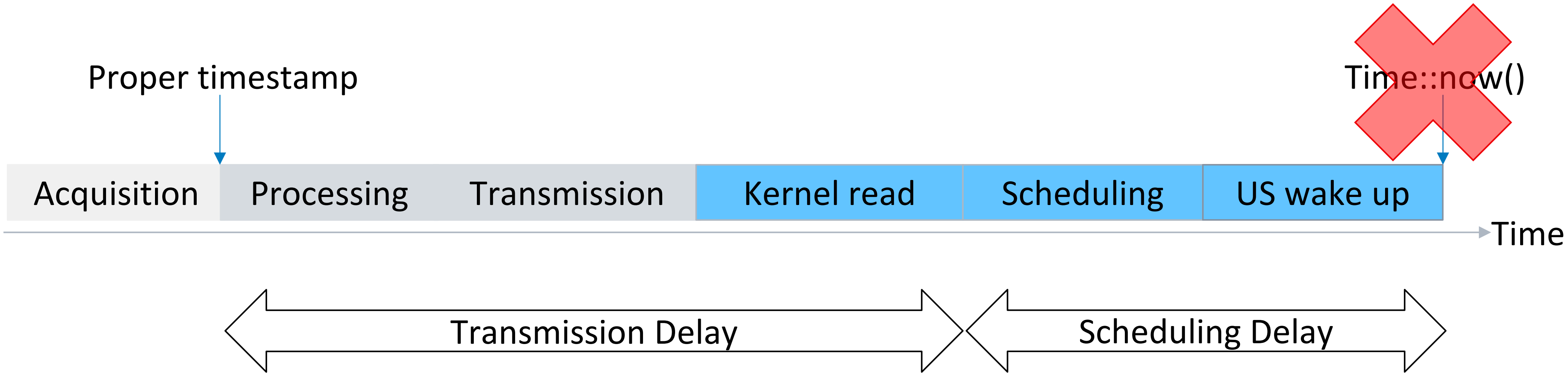
BOSCH

Diagnostics

From prototype to product
Getting it working *every time*

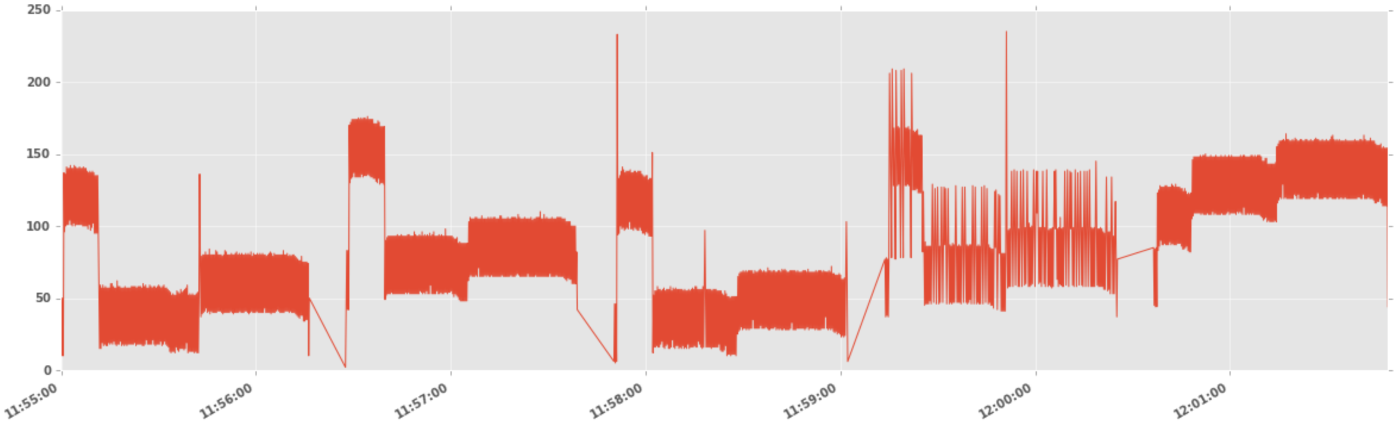
- Requirements
  - System Engineering & Integration
  - Handling diverse situations
- How do we get there?
  - Explicit requirements, architecture, design
  - Establish a *useful* feedback loop (testing, diagnostics, metrics, real-world recordings, …)
- What is "useful"?
  - Results are relevant for delivering a better product
  - Problems are quick to diagnose
- How do we diagnose quickly?
  - Break it down into pieces
  - Look at both pieces and whole

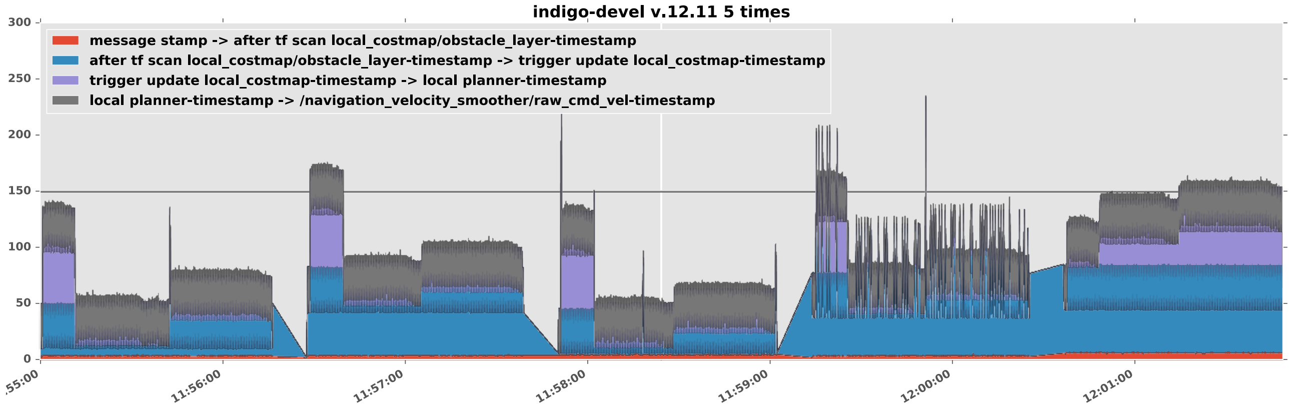
Watch out for sensor timestamping



- time::now → easily ~10-100ms error!
- Transmission delay correction can be used when sensor data has no timestamp
- Scheduling delay without RT – easily 10ms
- Use real-time scheduling!

Monitor for determinism

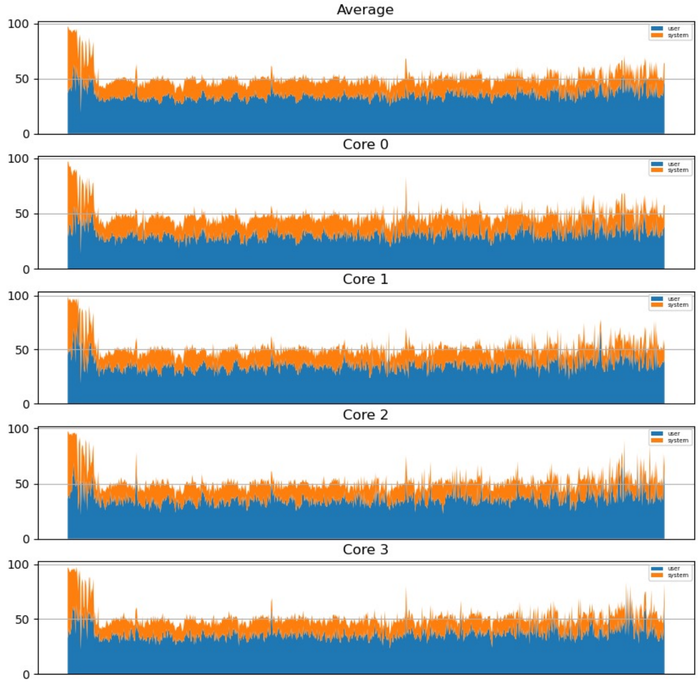



indigo-devel v.12.11 5 times

Helpful tools

- Diagnostics
  - TopicDiagnostic → Message age/frequency
- Standard monitoring tools
  - Resource usage (e.g., perf, flamegraphs)
  - Crashes
- ros2_tracing
  - Callback durations
  - Callback sequences
  - Queue latencies
  - IO latencies
  - Scheduling latencies
  - Many more things…

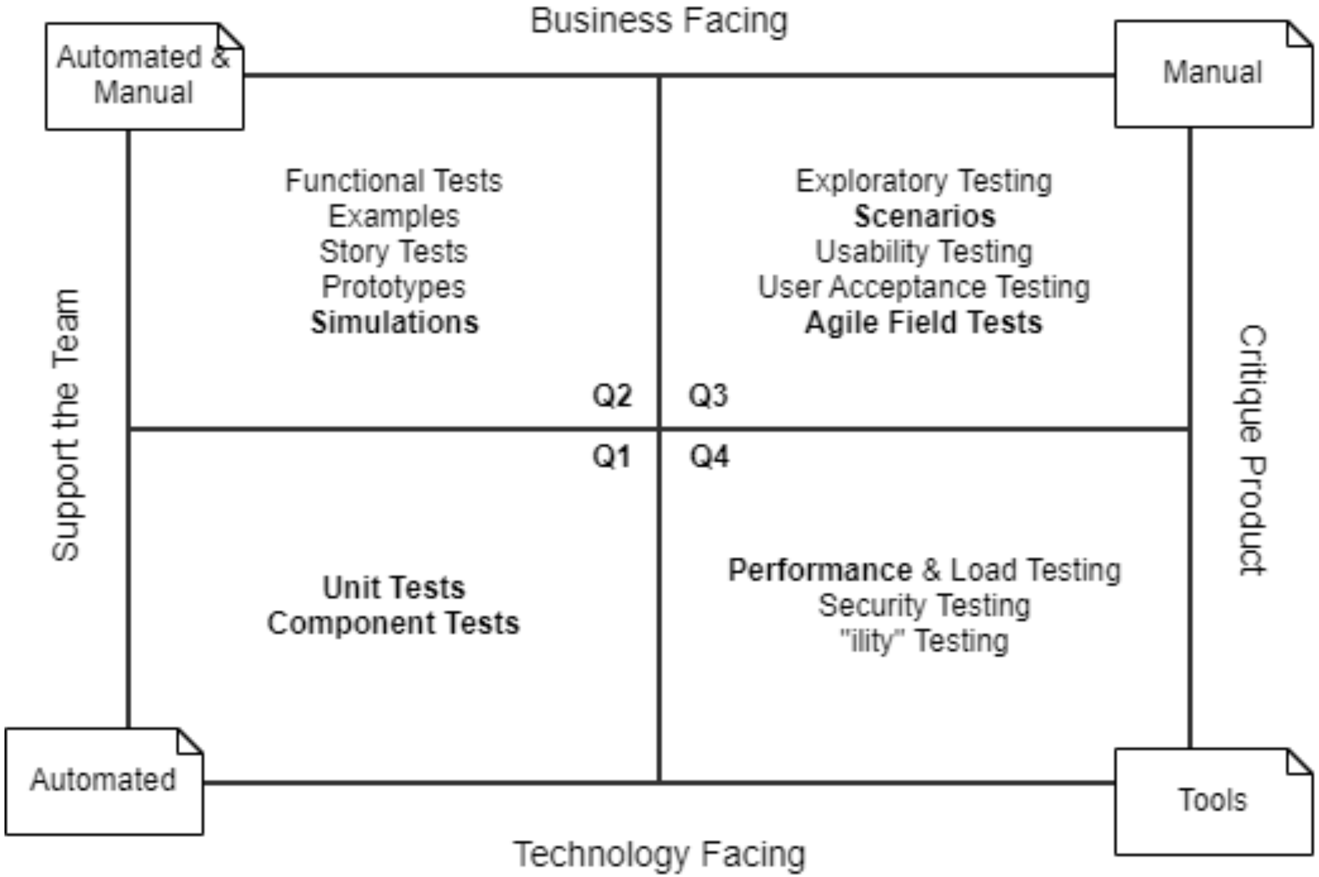
- Compare different versions



Testing

Process Overview

Agile Test Quadrants



Business Facing

Automated & Manual

Manual

Support the Team

Critique Product

Functional Tests
Examples
Story Tests
Prototypes
Simulations

Q2

Exploratory Testing
Scenarios
Usability Testing
User Acceptance Testing
Agile Field Tests

Q3

Q1

Unit Tests
Component Tests

Q4

Performance & Load Testing
Security Testing
"ility" Testing

Automated

Tools

Technology Facing

After: „Agile Testing“, Crispin & Gregory, 2008

Testing

Test Levels

Level	Tools	Entry Criteria	Exit Criteria	Coverage Criteria
Unit-testing	Gtest, pytest, etc.	Every commit	Success	Mutation testing
Interface tests	e.g., launch_test	Every commit	Success	Architecture match
Integration tests	Launch_test, gazebo, etc.	On pull request	Endurance testing success	Architecture match
SW/HW integration tests	Xray, etc.	On HAL change	Endurance success	custom
Performance tests	Various	Weekly	KPIs okay	By KPI

Testing
How to come up with good tests?

- Basic software engineering tradecraft
  - Some standards, such as ISO 29119
- Driven by coverage criteria
- Property-based testing
- Regression testing (if there was a bug…)
  - Often also based on recorded data
- Experience

- Generally, major source of concern and in dire need of improvement
- Areas in need of improvement
  - Mocking
  - Behavior interface tests
  - Speeding up tests
  - Test reporting/visualization
  - Integration of more formal tools

Testing

Random data collection gets you only so far

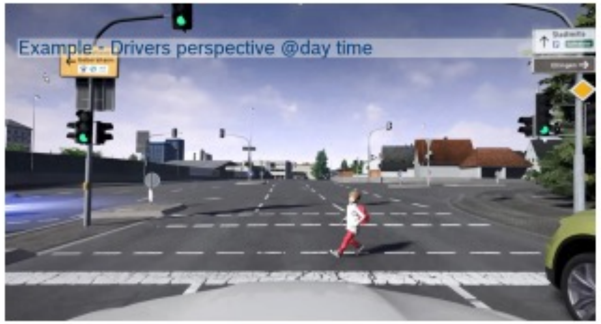
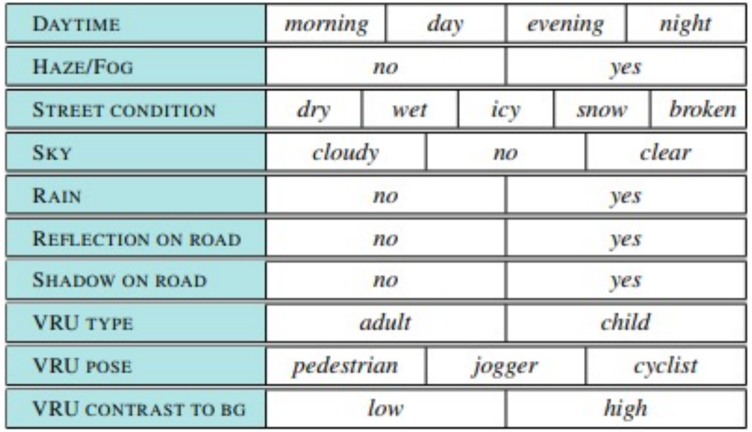
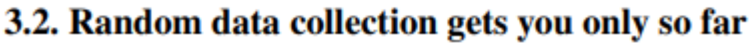


Figure 1: Intersection scenario at daytime (VRU example)

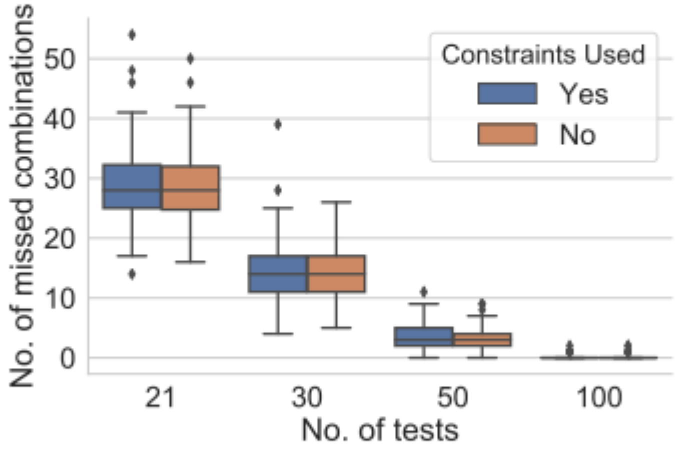


DAYTIME	morning	day	evening	night	
HAZE/FOG	no	yes			
STREET CONDITION	dry	wet	icy	snow	broken
SKY	cloudy	no	clear		
RAIN	no	yes			
REFLECTION ON ROAD	no	yes			
SHADOW ON ROAD	no	yes			
VRU TYPE	adult	child			
VRU POSE	pedestrian	jogger	cyclist		
VRU CONTRAST TO BG	low	high			

- Scenario has 9866 combinations
- Correctly chosen, 21 tests can cover
- Random choice → ~5 times more effort!



3.2. Random data collection gets you only so far



Source: "Leveraging Combinatorial Testing for Safety-Critical Computer Vision Datasets", Gladisch et al, CVPR WS 2020

ROS 2 Architecture Working Groups

- Primarily, at present
  - Middleware working group
  - Client library working group
- Future: Production Working Group

Conclusions

- Shared tools, understanding and practices are essential for sucessful product development
- Architecture methods contribute to this on several levels
  - Quality alignment
  - Scaling
  - Test selection
- Common architectural building blocks in ROS 2 contribute to
  - Shared interfaces
  - Diagnostics and Data Analysis
  - Simulation and testing

Vielen Dank für die Aufmerksamkeit!

Fragen?