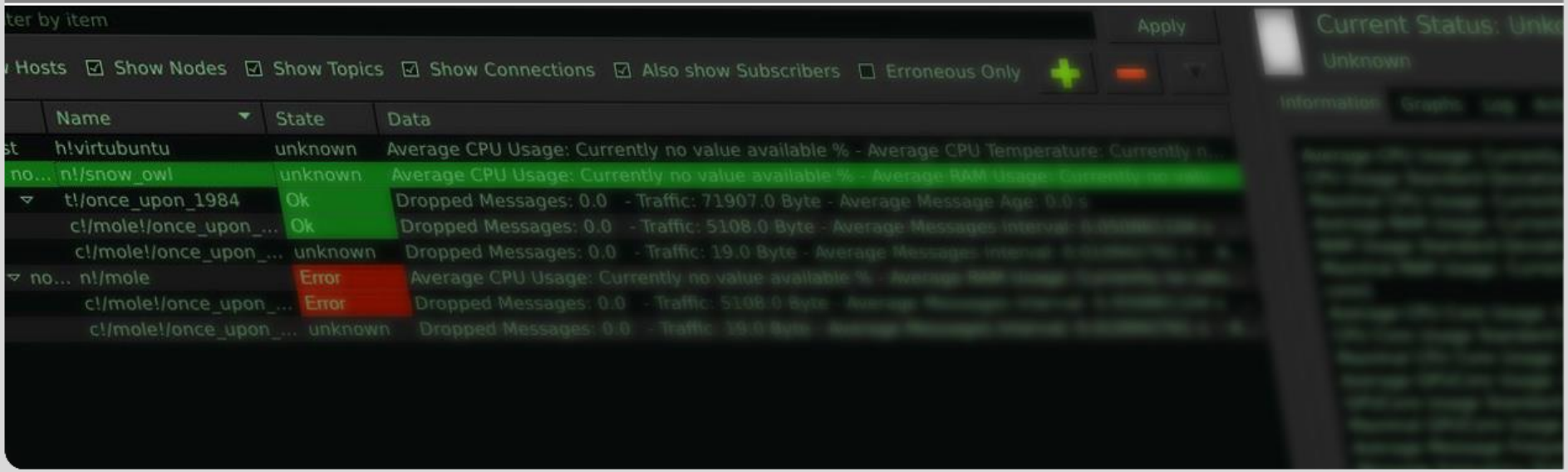
Karlsruher Institut für Technologie

ARNI – Advanced ROS Network Introspection

Abschlusspräsentation

Alex Weber, Matthias Hadlich, Matthias Klatte, Micha Wetzel, Sebastian Kneipp

Praxis der Softwareentwicklung SS 2014
Advanced ROS Network Introspection



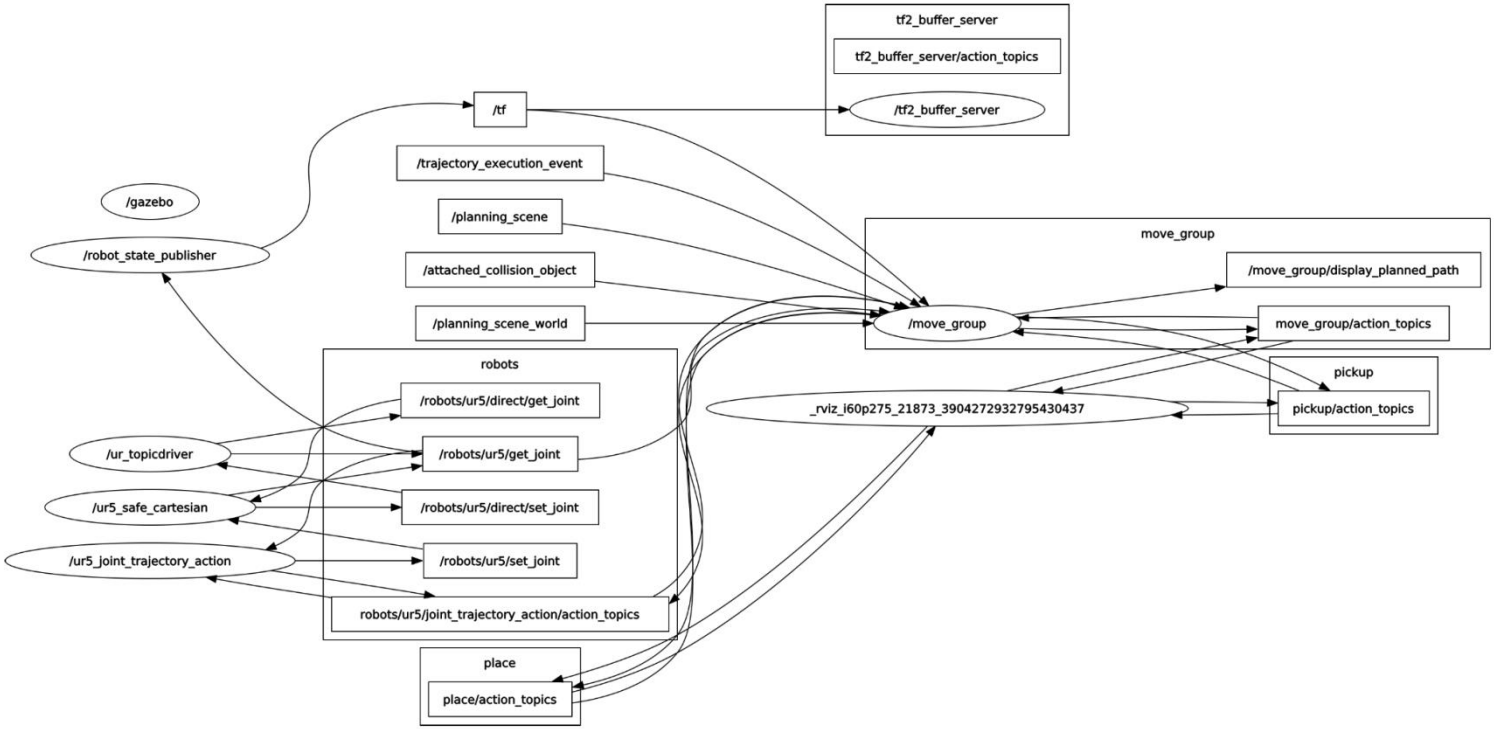
Gliederung

- Motivation
- Funktionsweise
- Live Demo
- Probleme
- Statistiken
- Fazit

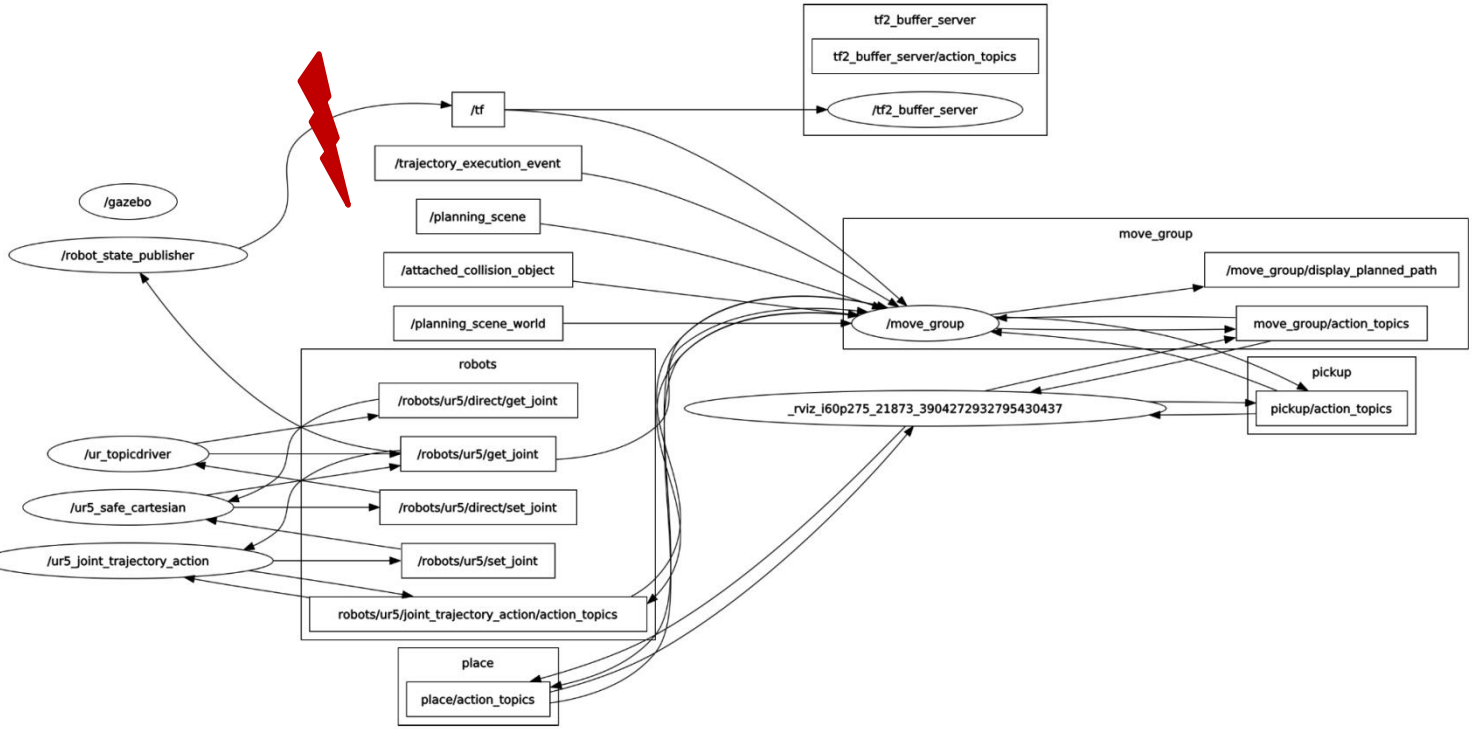
Robot Operating System

- Entwicklung 2007
- teilen von Funktionalität in Nodes
- hauptsächlich Forschung
- bald: ROS-Industrial

Ausgangsproblem

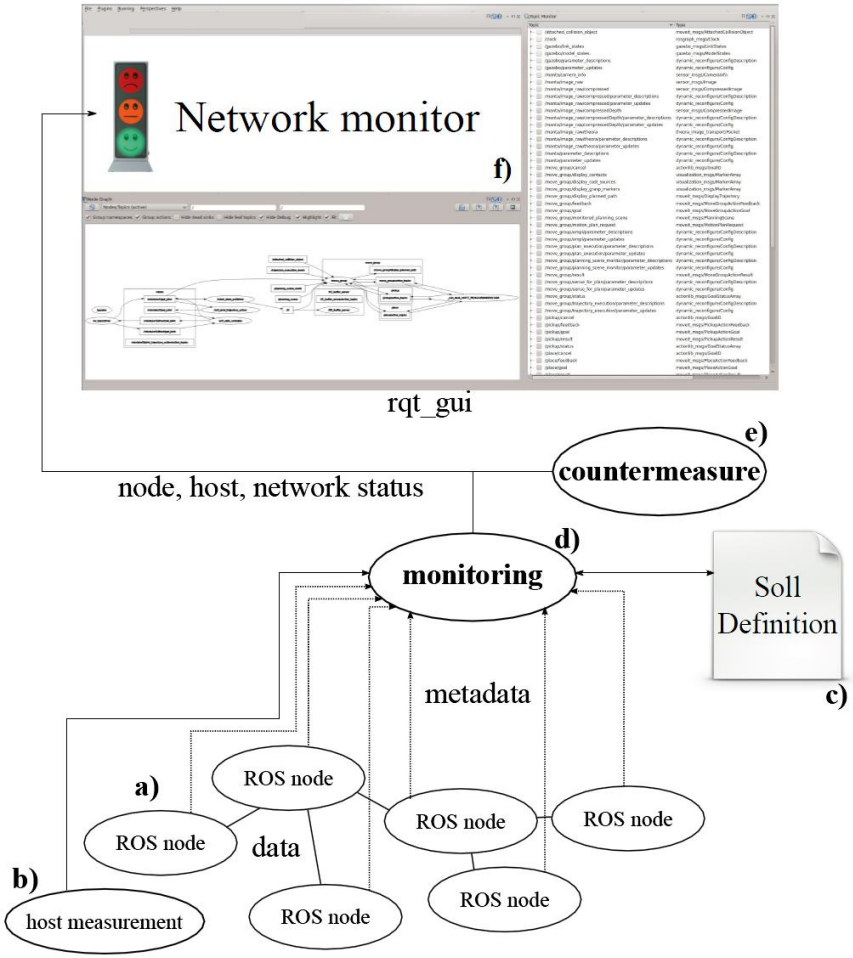


Ausgangsproblem



→ Problem: Fehlersuche

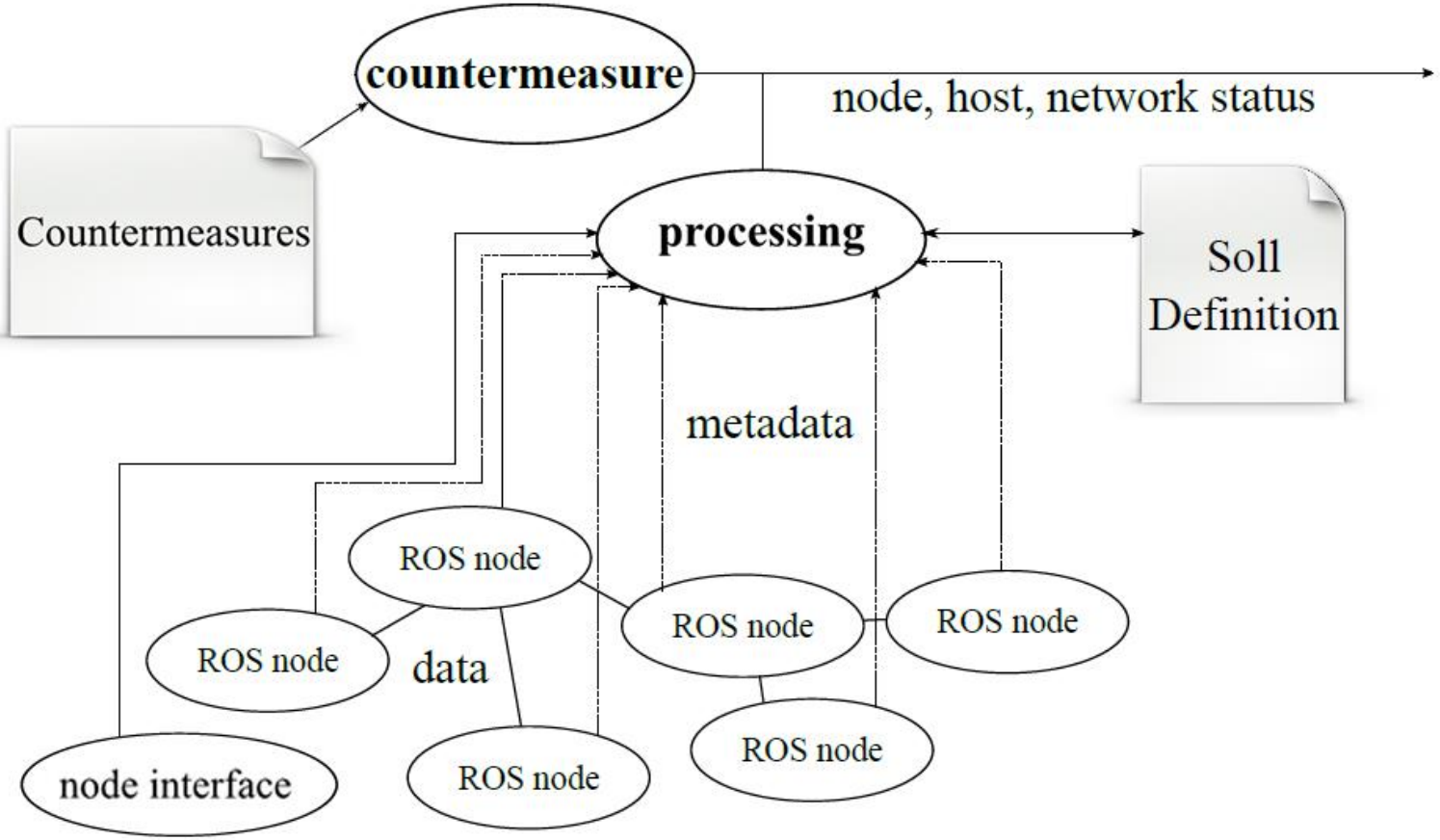
Aufgabenstellung



Aufgabenstellung

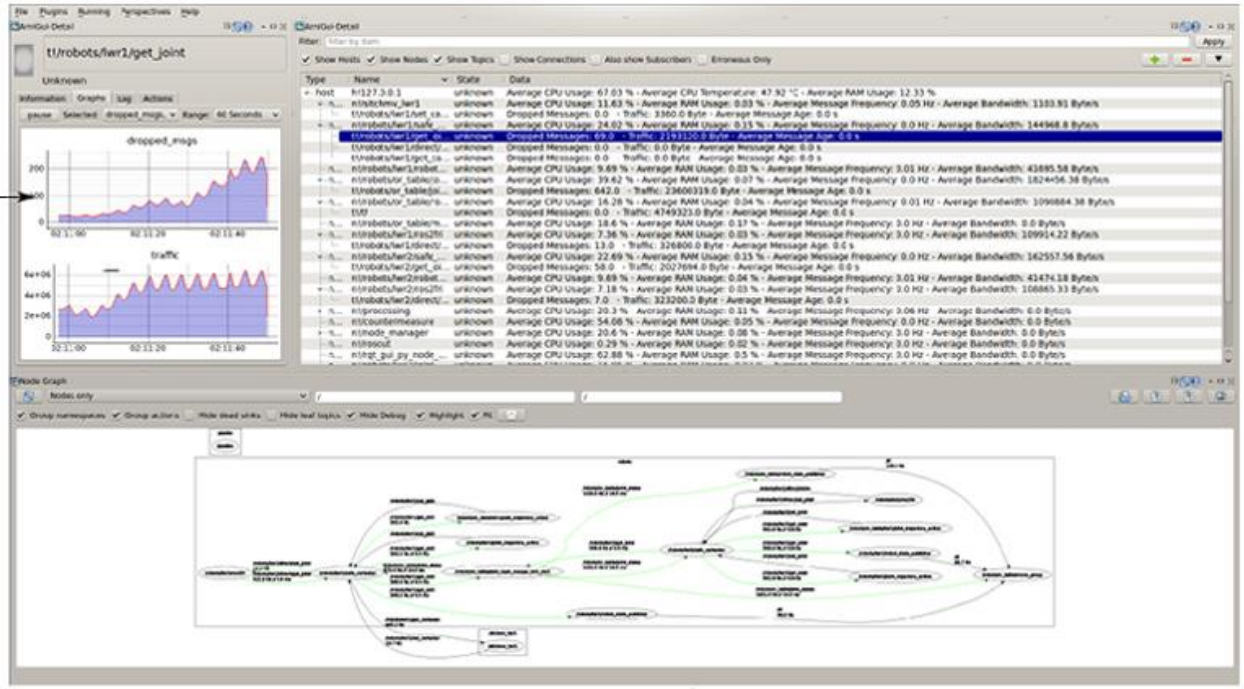
- Erfassung des Systemzustandes
- Definition von Soll-Werten
- Visualisierung der erhobenen Werte
  mit farblicher Darstellung von Fehlerzuständen
- Definition von Maßnahmen für Fehlerfälle

Funktionsschema (Teil 1)



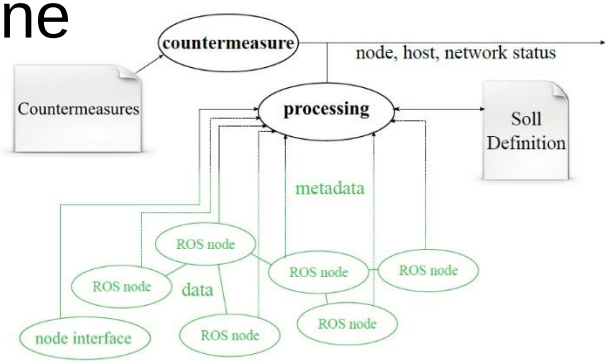
Funktionsschema (Teil 2)

node, host,
network status



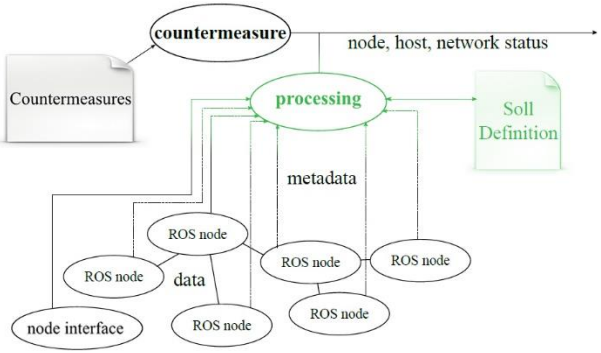
Datenerfassung

- Erweiterung der Publish/Subscribe-Funktionalität: Statistiken zum Sendeverhalten erheben
  - Nachrichtengröße
  - Sendezeit, Sendeverzögerung
- Neuer Knoten für Systemstatistiken
  - Läuft auf jedem Host
  - Ressourcenverbrauch durch einzelne Nodes
  - Hardwareauslastung des Systems



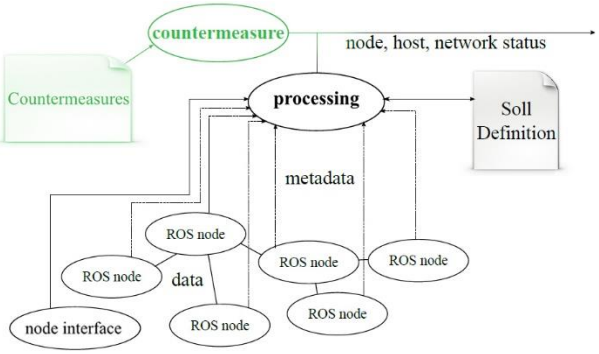
Datenverarbeitung

- Laden von Spezifikationen auf den Parameterserver
- Aggregierung von eingehenden Daten
- Bewertung anhand der Spezifikationen
  - Low, High, Ok, Unknown
- Publizieren auf ein Topic



Gegenmaßnahmen

- Laden von „Constraints“
- Logische Verknüpfungen von Bedingungen
- Ausführen von Gegenmaßnahmen
  - Debug-Ausgaben
  - Neustarten von Knoten
  - Ausführen von Befehlen



Visualisierung

- Auflistung verfügbarer Verbindungen und Host-Systeme
- Darstellung aller erhobener Werte
  - Als lokalisierbarer Text
  - Als Graphen über Zeit
- Farbliche Hervorhebung der Bewertungsergebnisse

Live Demo

Probleme

- 2 Bugs in ros_comm gefunden [1]
- Segmentation Faults mit PySide/PyQt (ein Bug gefunden)
- Schlecht dokumentierte API‘s z.B. pyqtgraph
  (Funktionen nicht aufgeführt oder Funktionsweise unzureichend erklärt)
- Python 3 Features benötigt, aber Python 2 verwendet

[1] u.A. https://github.com/ros/ros_comm/issues/501

Statistiken

- 5895 Lines of Python Code (11239 Zeilen mit Kommentaren und allen Sprachen)
- 56 Klassen
- Über 135 Seiten Dokumente

- Codeabdeckung von über 75%, in manchen Bereichen über 90% (gemessen mit coverage.py)

Tests

Unittests

64 Tests decken weite Teile des Codes ab

Integrationstests

5 umfangreiche Integrationstests simulieren unterschiedliche Situation und testen die Reaktion von ARNI

Produktiveinsatz am IPR

2 Wochen Betrieb an laufenden Systemen inkl. Tests bei hoher Last und bei wechselnden Bedingungen

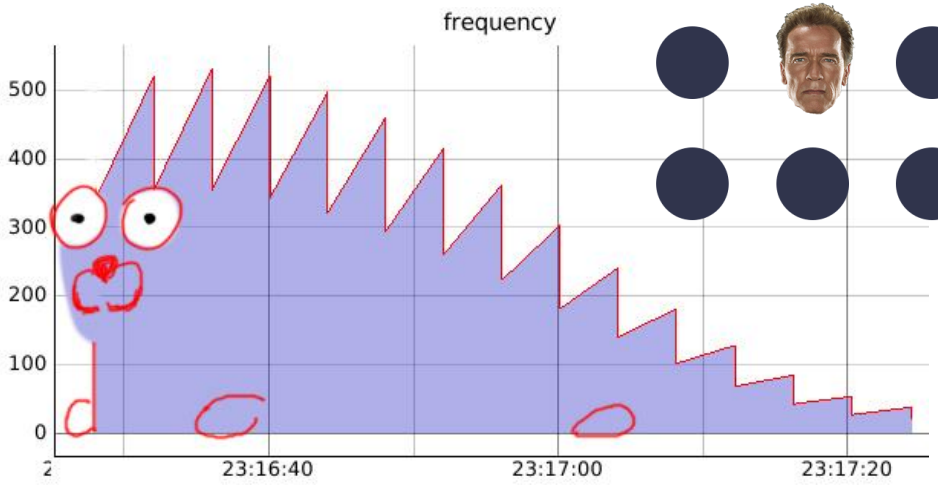
Verwendete Frameworks

Und mehr:

- psutil
- pyqtgraph
- pysensors
- Yaml
- Xml
- Latex

Fazit

- Spannendes Projekt
- Viel Arbeit, aber auch viel gelernt
- Geplante Aufnahme in ROS
- Spaß:



ARNI

Advanced ROS Network Introspection

Vielen Dank für die Aufmerksamkeit!

Noch Fragen?