

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

Filter by item

Hosts ☒ Show Nodes ☒ Show Topics ☒ Show Connections ☒ Also show Subscribers ☐ Erroneous Only + - ▼ Apply

	Name	State	Data
st	h!virtubuntu	unknown	Average CPU Usage: Currently no value available % - Average CPU Temperature: Currently n...
no...	n!/snow_owl	unknown	Average CPU Usage: Currently no value available % - Average RAM Usage: Currently no val...
▼	t!/once_upon_1984	Ok	Dropped Messages: 0.0 - Traffic: 71907.0 Byte - Average Message Age: 0.0 s
	c!/mole!/once_upon_...	Ok	Dropped Messages: 0.0 - Traffic: 5108.0 Byte - Average Messages Interval: 0.000000 s
	c!/mole!/once_upon_...	unknown	Dropped Messages: 0.0 - Traffic: 19.0 Byte - Average Messages Interval: 0.000000 s
▼	no... n!/mole	Error	Average CPU Usage: Currently no value available % - Average RAM Usage: Currently no val...
	c!/mole!/once_upon_...	Error	Dropped Messages: 0.0 - Traffic: 5108.0 Byte - Average Messages Interval: 0.000000 s
	c!/mole!/once_upon_...	unknown	Dropped Messages: 0.0 - Traffic: 19.0 Byte - Average Messages Interval: 0.000000 s

Current Status: Unk
Unknown

Information Graphs Log

Gliederung

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

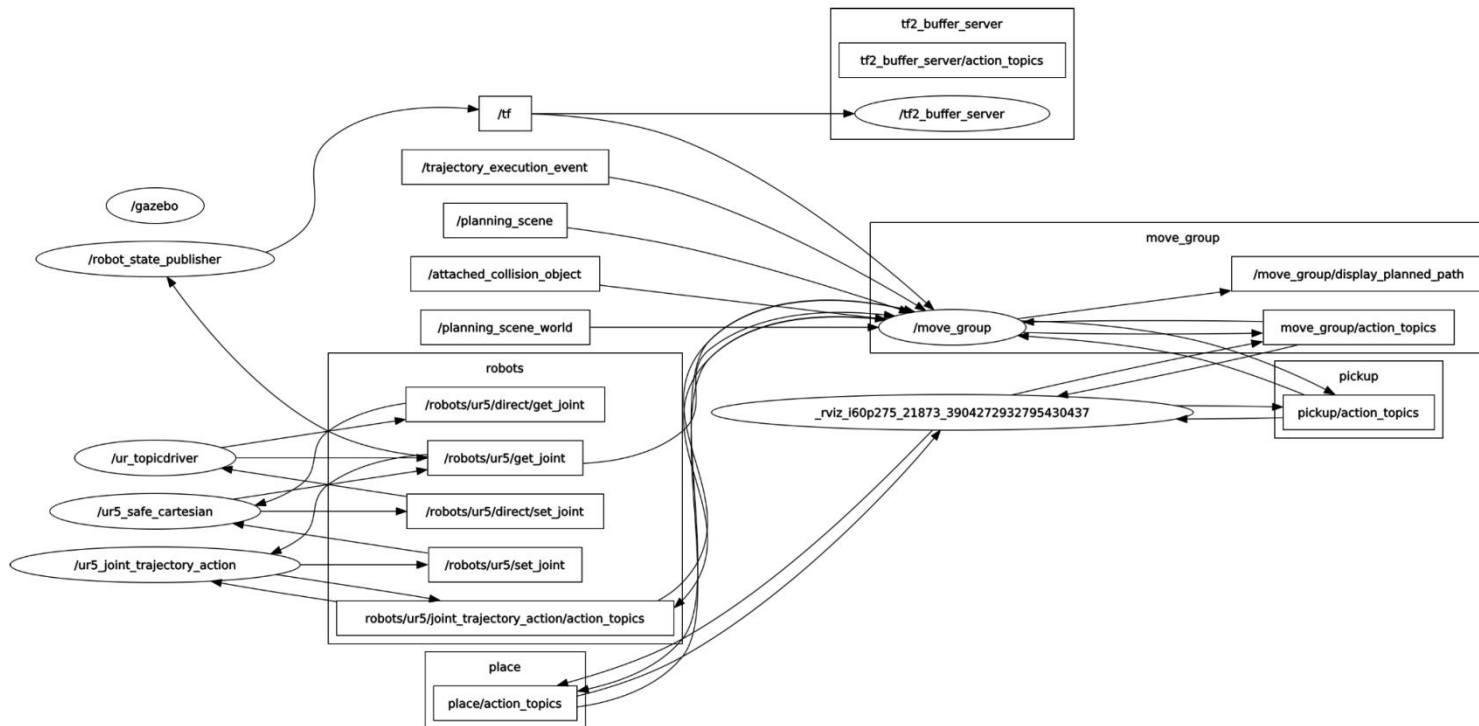
Robot Operating System

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

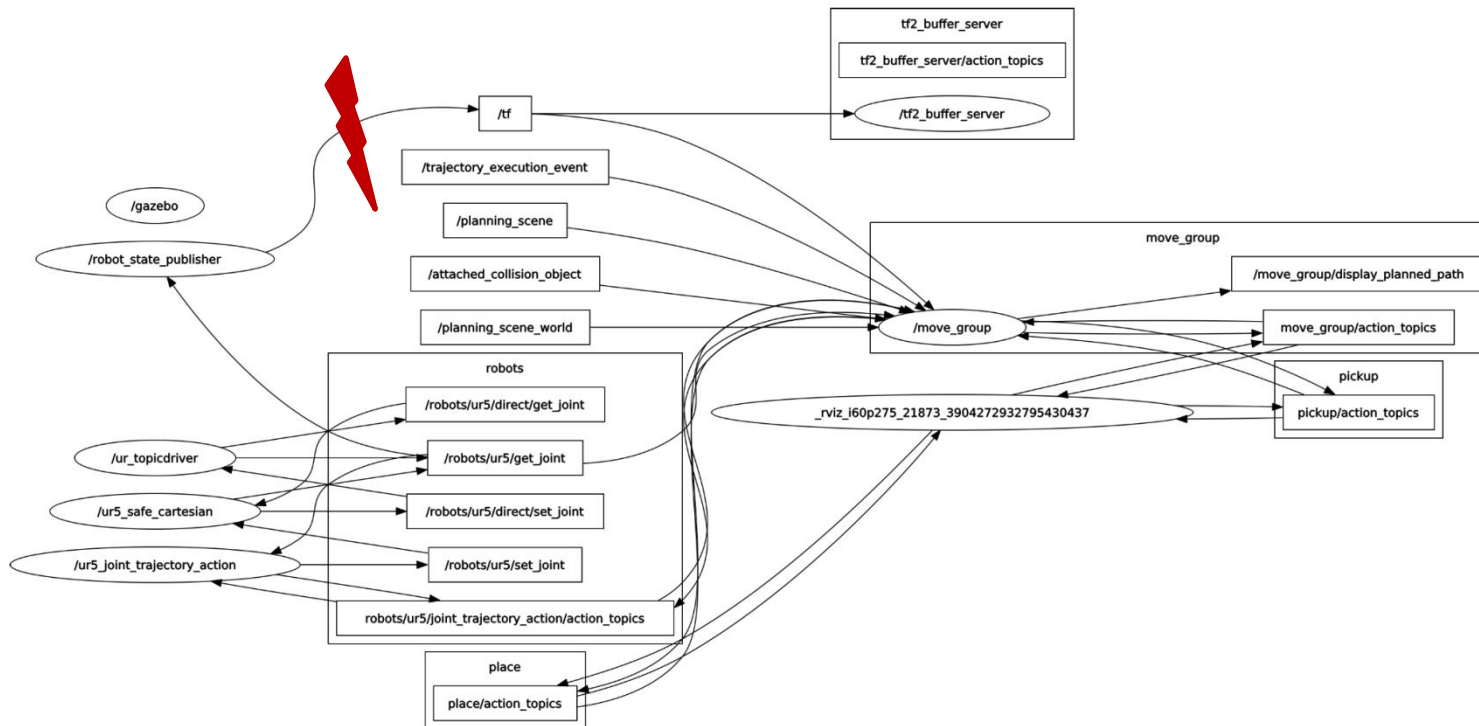
 **ROS.org**



Ausgangsproblem



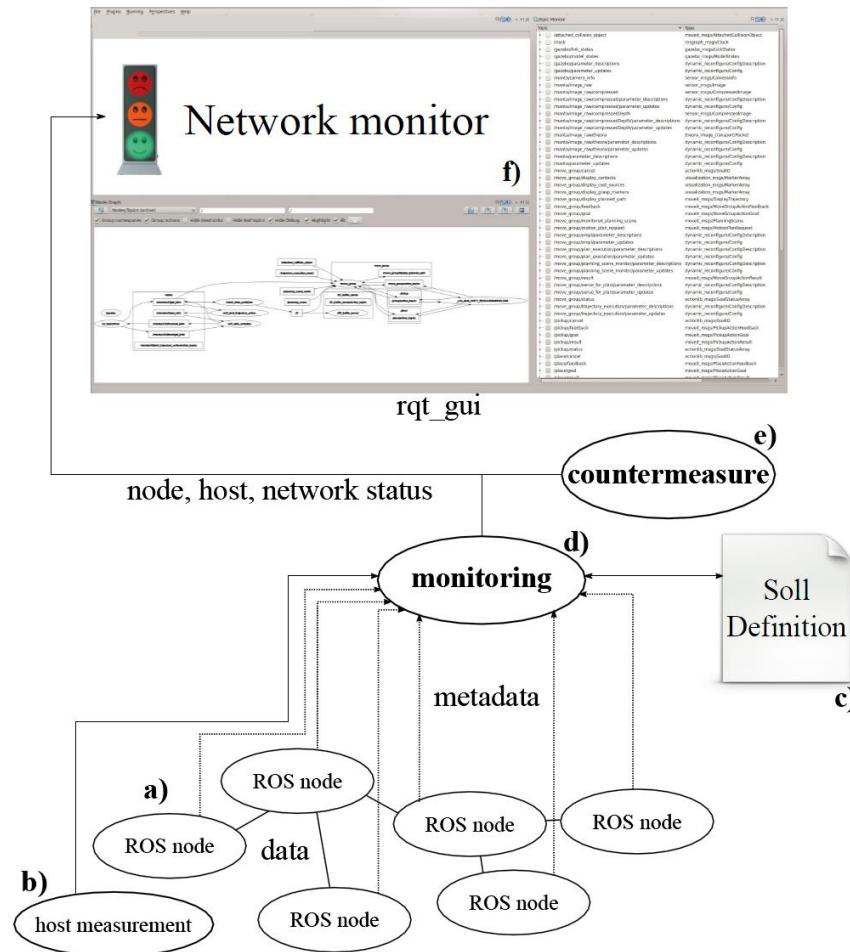
Ausgangsproblem



→ Problem: Fehlersuche



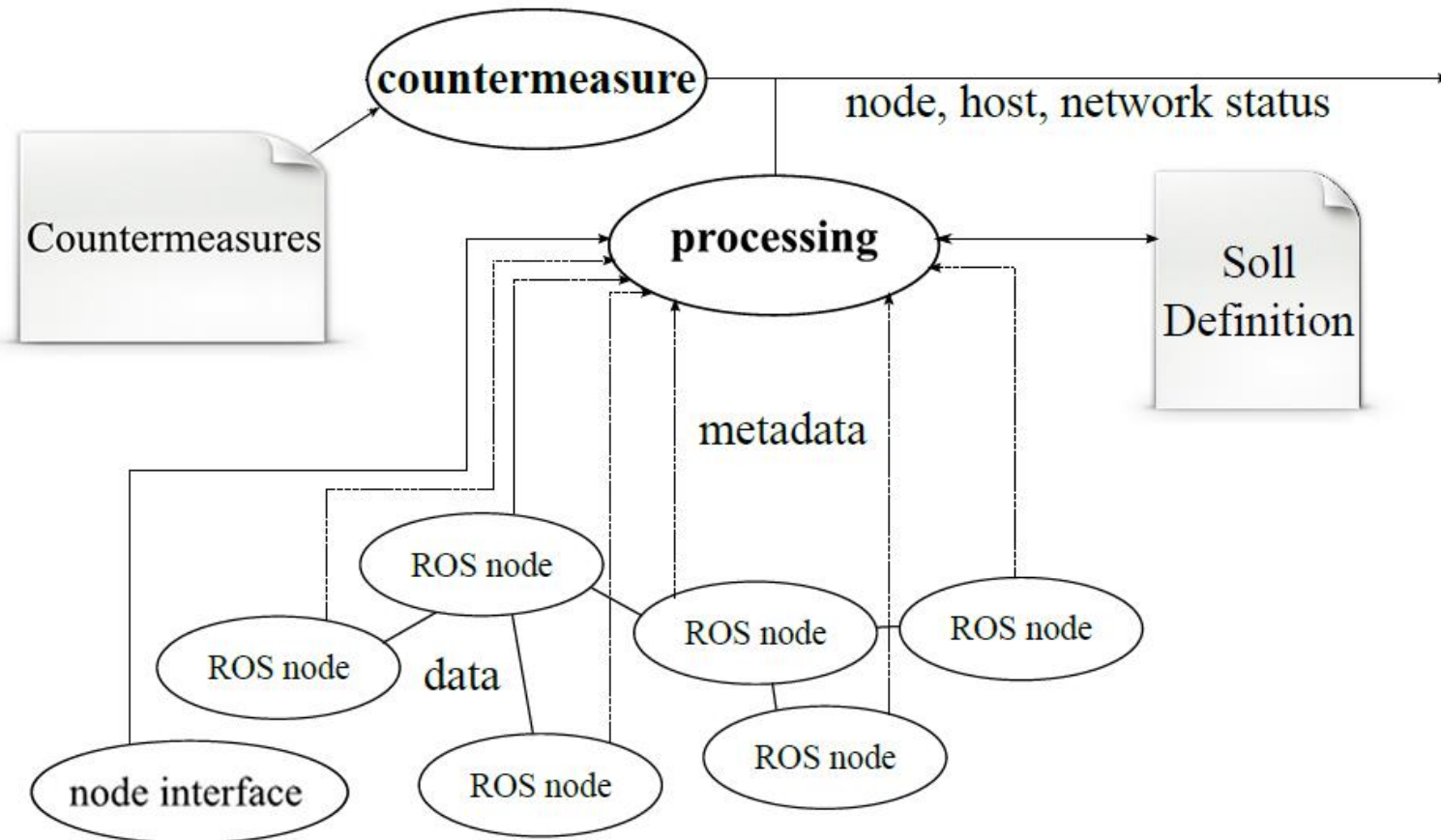
Aufgabenstellung



Aufgabenstellung

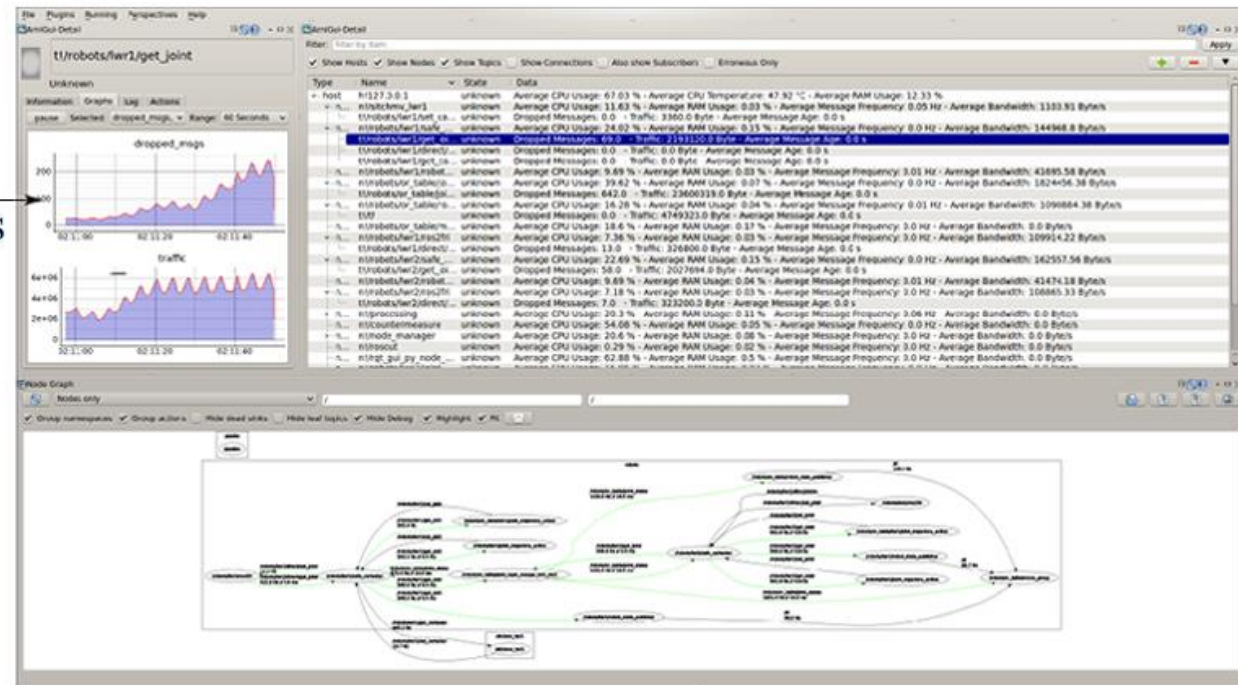
- Erfassung des Systemzustandes
- Definition von Soll-Werten
- Visualisierung der erhobenen Werte
mit farblicher Darstellung von Fehlerzuständen
- Definition von Maßnahmen für Fehlerfälle

Funktionsschema (Teil 1)



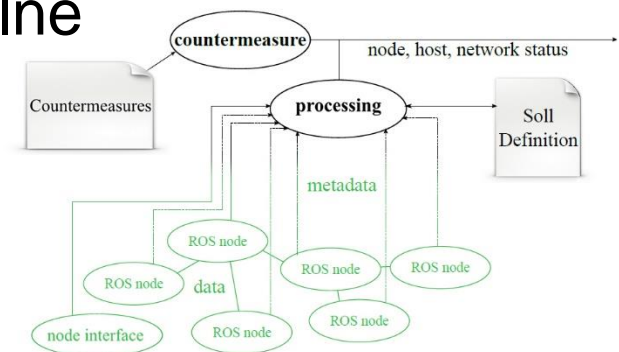
Funktionsschema (Teil 2)

node, host,
network status



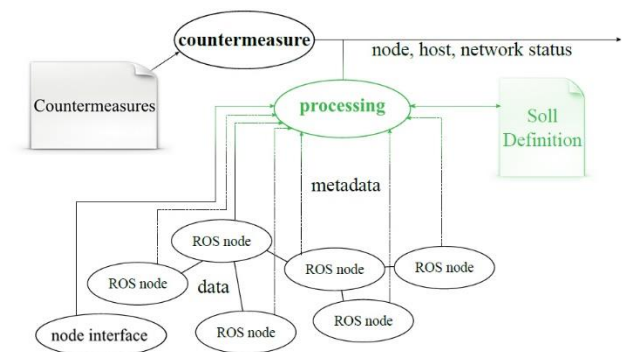
Datenerfassung

- 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
-
- ```
graph LR; CM[Countermeasures] --> C(countermeasure); C -- "node, host, network status" --> P((processing)); P -- "metadata" --> SD[Soll Definition];
```
- The diagram illustrates the system architecture. It shows a flow from a document icon labeled 'Countermeasures' to an oval labeled 'countermeasure'. From 'countermeasure', a line labeled 'node, host, network status' connects to a central oval labeled 'processing'. From 'processing', a line labeled 'metadata' connects to a document icon labeled 'Soll Definition'.



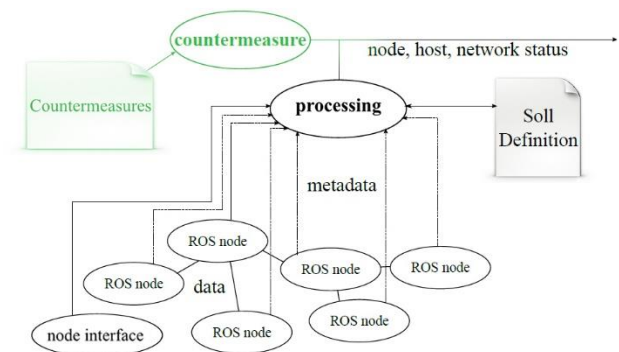
# Datenverarbeitung

- Laden von Spezifikationen auf den Parameterserver
- Aggregation 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 <sup>1</sup>
- 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

<sup>1</sup> u.A. [https://github.com/ros/ros\\_comm/issues/501](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)



## 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

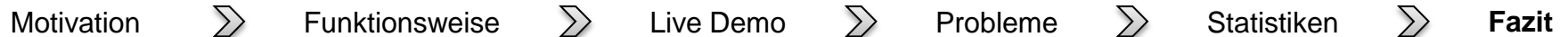


Und mehr:

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



- 
- ARNI**  
Advanced ROS Network Introspection



# Vielen Dank für die Aufmerksamkeit!

## Noch Fragen?

