Inhaltsverzeichnis

ROS hat drei Konzeptschichten:

- Dateisystem
- Computation Graph
- Community

Diese sind folgend zusammengefasst und werden in weiteren Kapiteln im Detail erläutert.

Zusätzlich zu den Konzeptschichten werden einerseits Package Resource Names und Graph Resource Names definiert (siehe Namen (/Names)).

1. ROS Dateisystem

ROS Ressourcen auf der Harddisk:

- **Packages (/Packages)**: Pakete sind die hauptsächliche Art der Softwareorganisation in ROS. Ein Paket kann folgende Elemente enthalten:
  - Laufzeitprozesse (*Nodes*)
  - Von ROS abhängige Bibliotheken
  - Datensätze
  - Konfigurationsdateien
  - Alles was sonst noch zu diesem Kontext gehört
- **Manifests (/Manifest)**: Das Manifest (`manifest.xml`) enthält Metadaten über das Paket. Dazu gehören Lizenzinformationen und Abhängigkeiten sowie sprachspezifische Einstellungen wie Compilerflags.
- **Stacks (/Stacks)**: Stacks sind eine Sammlung von Paketen, welche zusammen eine Funktionalität bereitstellen (z.B. der *navigation stack*). Zudem wird ROS Software in Form von Stacks ausgeliefert und versioniert.
- **Stack Manifests (/Stack%20Manifest)**: Das Stack Manifest (`stack.xml`) enthält Metadaten zum Stack. Dazu gehören Lizenzinformationen sowie Abhängigkeiten zu anderen Stacks.
- **msg (Message types) (/msg)**: Die Datenstruktur von messages (/Messages) ist in `<package_name>/msg/<message_type>.msg` abgelegt.
- **srv (Service types) (/srv)**: Anfrage- und Antwortdatenstrukturen für services (/Services) sind unter `<package_name>/srv/<service_type>.srv` zu finden.

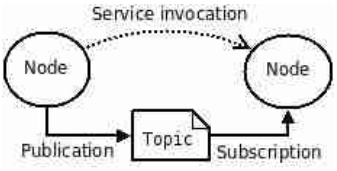
2. ROS Computation Graph

Der *Computation Graph* ist ein Peer-to-Peer Netzwerk von ROS Prozessen, welche gemeinsam eine Aufgabe erfüllen. Der Graph bezieht seine Daten aus folgenden Komponenten (ros_comm (/ros_comm) stack):

- **Nodes (/Nodes)**: Knoten sind Prozesse, welche Berechnungen durchführen. ROS ist modular ausgelegt, deshalb arbeiten in einem Robotersystem normalerweise viele Knoten zusammen. Ein Knoten setzt auf einer ROS client library (/Client%20Libraries) wie roscpp (/roscpp) oder rospy (/rospy) auf. Knoten kommunizieren direkt untereinander, die Verbindungsinformationen erhalten sie vom Master. Das Standardkommunikationsprotokoll ist TCPROS (/ROS/TCPROS).
- **Master (/Master)**: Über den ROS Master können Namen registriert oder nachgeschlagen werden. Ohne den Master wären die Knoten nicht in der Lage zu kommunizieren.
- **Parameter Server (/Parameter%20Server)**: Der Parameterserver ist Teil des Masters und speichert Daten zu einem Schlüssel an zentraler Stelle.
- **Messages (/Messages)**: Knoten kommunizieren über Nachrichten (messages (/Messages)). Eine Nachricht ist eine Datenstruktur mit typisierten Feldern. Standard Datentypen (Integer, Float, Boolean) und Arrays derselben werden unterstützt. Nachrichten können beliebig verschachtelte Strukturen enthalten.
- **Topics (/Topics)**: Nachrichten werden über einen Publisher-Subscriber Mechanismus verteilt. Ein Knoten veröffentlicht Nachrichten zu einem Thema (publish to a topic (/Topics)) identifiziert über einen Namen (/Names). Ein anderer Knoten, welcher an diesen Nachrichten interessiert ist, abonniert das Thema (subscribe). Es können mehrere Knoten gleichzeitig als Publisher und Subscriber fungieren und ein Knoten kann mehrere Themen veröffentlichen oder abonnieren. Allgemein gesagt wissen Publisher und Subscriber nichts von einander. So werden Erzeugung und Konsum von Informationen entkoppelt.
- **Services (/Services)**: Der Publisher-Subscriber Mechanismus erlaubt eine flexible, einseitige n-zu-n Kommunikation. Diese ist jedoch für eine Anfrage-Antwort Interaktion zwischen den Knoten nicht geeignet. Für diese Art der Kommunikation bietet ein Knoten einen Service (/Services) an, identifiziert durch einen Namen (/Names). Der aufrufende Knoten sendet eine Anfrage und wartet auf die Antwort.
- **Bags (/Bags)**: Bags sind zur Ablage und Wiedergabe von ROS messages. Mit Bags können z.B. Sensordaten aufgezeichnet werden, welche für die Entwicklung nötig sind, aber nur mit grossem Aufwand aufgezeichnet werden können.

Namen (/Names) haben eine wichtige Funktion im Computation Graph. Nodes, Topics, Services und Parameter verfügen alle über einen Namen.

Damit die Komponenten auch in anderen Laufzeitumgebungen funktionieren, können Namen über die ROS client library (/Client%20Libraries) während der Laufzeit neu zugeordnet werden.



3. ROS Community

Das Communitykonzept ermöglicht verschiedenen Gemeinschaften Software und Wissen auszutauschen. Dafür bestehen folgende Ressourcen:

- **Distributions (/Distributions)**: Eine ROS Distribution ist eine installierbare Sammlung von versionierten stacks (/Stacks). Distributionen erleichtern die Installation der Software und erhalten konsistente Versionen über eine Menge von Software.
- **Repositories (/Repositories)**: ROS stützt auf ein verteiltes Netzwerk von Code repositories, welche verschiedenen Institutionen ermöglichen ihre eigenen Roboterkomponenten zu entwickeln und auszuliefern.
- **The ROS Wiki (/Documentation)**: Das ROS Wiki ist die zentrale Stelle für Dokumentation rund um ROS. Jedermann kann sich anmelden und seine eigene Dokumentation, Korrekturen, Tutorials und andere Informationen beitragen.
- **Bug Ticket System (/Tickets)**: Mit dem Ticket System können Fehler an zentraler Stelle verwaltet und von der Community verarbeitet werden.
- **Mailing Lists (/Mailing%20Lists)**: Die ROS Users Mailing Liste (/Mailing%20Lists) ist der primäre Kommunikationskanal für Updates zu ROS.
- **ROS Answers (http://answers.ros.org)**: Über ein Portal können Fragen und Antworten zu ROS der Community zur Verfügung gestellt werden.
- **Blog (http://www.willowgarage.com/blog)**: Der Willow Garage Blog (http://www.willowgarage.com/blog) stellt regelmässig Updates, Fotos und Videos bereit.

4. Nächstes Kapitel

Higher-Level Concepts (/ROS/Higher-Level%20Concepts)

Wiki: de/ROS/Concepts (zuletzt geändert am 2013-12-23 15:01:59 durch TullyFoote (/TullyFoote))