(Research for) Creating a ROS 2 Distribution

Dirk Riehle

University of Erlangen

2023-11-23 ROSCON DE

Prof. Dr. Dirk Riehle, M.B.A.

Professor of Computer Science at University of Erlangen, https://oss.cs.fau.de

Main current research projects

- Engineering economics dashboard
- Data engineering, pipeline infrastructure
- Qualitative data analysis research software

Academic and industry experience

- Professor, University of Erlangen, Germany
- Research group leader, SAP Research, California
- Software architect, Skyva Inc., Boston, U.S.A.

https://dirkriehle.com – https://profriehle.com

Commercial services through Bayave Gmbh, https://bayave.com

1. Overview

Distribution [1]

A distribution is

- A non-trivial set of components configured to work well with each other

An open source distribution is

- A distribution mostly or exclusively built from open source components

A distribution often covers a whole layer in the tech stack

- Examples are Linux, Kubernetes, Open Stack

[1] See https://dirkriehle.com/publications/2021-selected/the-open-source-distributor-business-model/

Examples and Classification of Distributions [1]

	Growth stage		Mature stage	
Complex products	**Kafka:** Confluent Platform (Confluent)	**Lucene/Solr:** Elasticsearch (Elastic)	**Hadoop:** Cloudera Distribution (Cloudera)	**Drupal:** Acquia Lightening (Acquia)
Commercial distributions	**Kubernetes:** OpenShift (Red Hat), Mesosphere Kubernetes Engine (D2IQ), Kubermatic Platform (Kubermatic)	**OpenStack:** VMware Integrated OpenStack (VMware), Mirantis Cloud Platform (Mirantis), Ericsson CEE9 (Ericsson)	**Linux:** Red Hat Enterprise Linux (Red Hat), SUSE Linux Enterprise Server (SUSE), Univention Corporate Server	**Python:** ActiveState Python (ActiveState), Anaconda (Continuum Analytics)
Community distributions	**ROS:** ROS (Open Robotics)		**TeX:** TeX Live, MacTeX, MiKTeX **Linux:** Debian, Fedora, OpenSUSE	**Python:** CPython, WinPython

[1] See https://dirkriehle.com/2021/05/29/open-source-distributions-by-life-cycle/

The Robot Operating System (ROS)

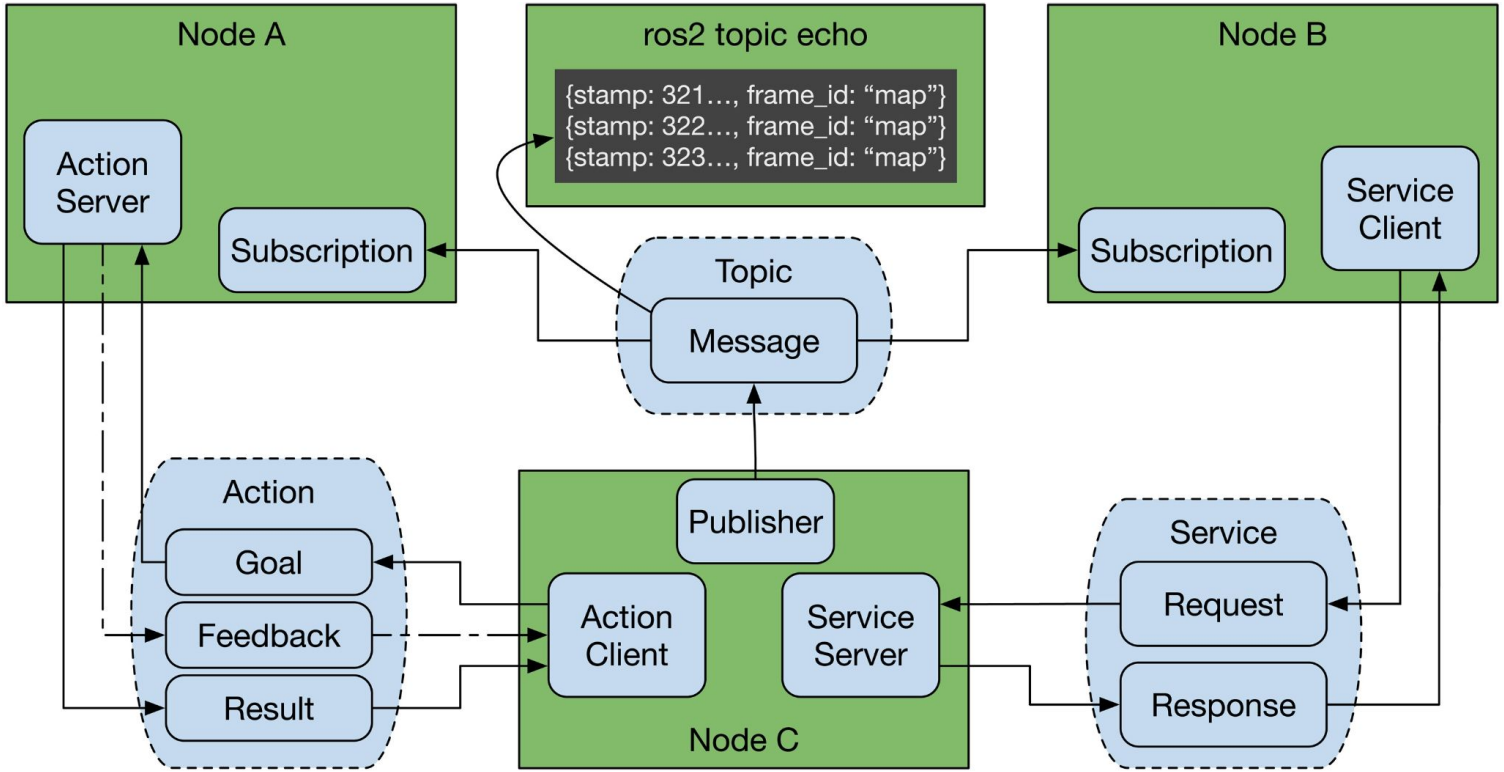
The Robot Operating System (ROS) is a

- Middleware for operating robots

ROS (v1) is mostly used in design and testing of systems

- ROS 2 is intended to be used in production

Design of ROS 2 [1]



[1] Macenski, S., Foote, T., Gerkey, B., Lalancette, C., & Woodall, W. (2022). Robot Operating System 2: Design, architecture, and uses in the wild. Science Robotics, 7(66).

Domains / Industries

- Mobility (vehicles)
  - Automotive (cars)
  - Shopfloor (cf. Turtlebots)
  - Off-highway (e.g. agriculture)
- Industrial (manufacturing robots)
- Medical (biomedical engineering, medical devices)

ROS 2 Distributions

The main open source distribution

- Open Robotics, https://docs.ros.org/en/humble/Releases.html
- About annual releases, currently Humble Hawksbill (“humble”)
- Supports (“tier 1”) Ubuntu 22.04 and Windows 10, no RTOS

One commercial distribution

- Apex.OS (+ Apex.Middleware), https://www.apex.ai/apex-os
- Based on ROS 2 (heavily modified)
- With underlying RTOS (probably QNX)

2. Research

Software Engineering Research Challenges

An open source distribution that is

1. (Well) integrated
2. Reproducible
3. Certifiable

Plus other challenges

2.1 Component Integration

The (original) challenge of any distribution is to

- Integrate well a
  - (possibly large) number of
  - (possibly complex)
  - (binary) components

Key Challenges of Integration

1. Definition and creation of correct (binary) builds
   a. For a given architecture, operational environment
   b. Build configuration for components
2. Definition and consistent use of versioning
   a. Versioned configurations for a release
   b. Overall release management
3. Definition and consistent use of component configurations
   a. Configuration files and their locations
   b. Parameter configuration patterns and mechanisms
4. Definition and consistent use of component interactions
   a. Consistent integration of components into middleware
   b. Patterns for implicit (not typed) dependencies, interactions

Dissertation-Size Research Project “Integration”

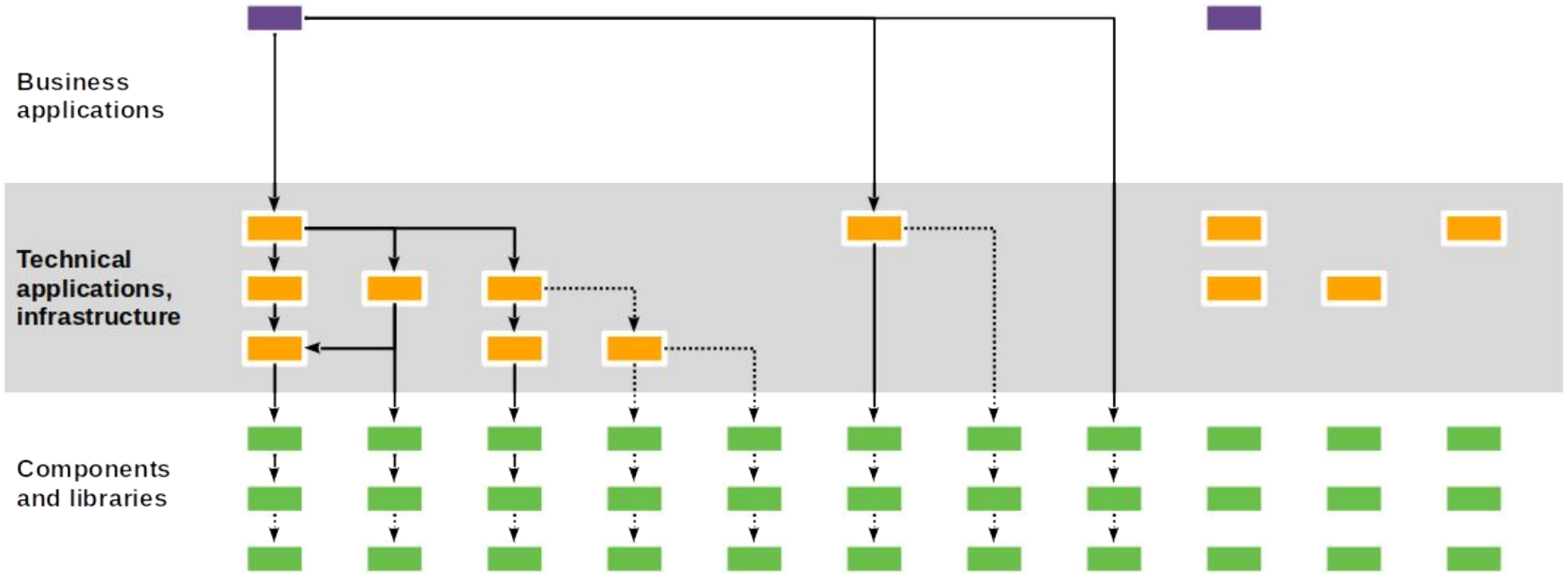
1. Problem identification
   a. Theory building using methodological triangulation using
      i. Structured literature review on distributions
      ii. Qualitative survey on challenges to creating a distribution
      iii. Quantitative descriptive survey
      iv. Artifact analysis
2. Objective definition
   a. For one (or several) problem dimensions, find solution to problem, e.g.
      i. **How should a multi-dimensional compatibility matrix look like?**
3. Solution design
   a. Build out theory, implement tooling for evaluation
4. Demonstration
   a. Ongoing demonstration using ROS 2
5. Evaluation
   a. Perform case study research with three industry partners for focused use case

2.2 Software Reproducibility

A distribution is reproducible if

- Every single open source component can be verified to be true to its source
- The verification must not depend on the distribution

Dependencies and the Software Supply Chain



U.S. Whitehouse Executive Order 2022-05-12 [1]

U.S. Whitehouse Executive Order on Improving the Nation's Cybersecurity [1]

- Sec. 4. Enhancing Software Supply Chain Security
  - **A purchaser must be provided a software bill of materials**

U.S. Cybersecurity & Infrastructure Security Agency

- Software bill of materials: https://www.cisa.gov/sbom

National Telecom and Infrastructure Administration

- https://www.ntia.gov/SBOM

[1] See https://whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

Key Challenges to Reproducibility

1. Create a complete and correct software bill of materials
2. Make component included in distribution open for verification
3. Make component build process open and deterministic
4. Design mechanism to perform verification from source
5. Allow for verification at the scale of a full open source distribution
6. Support incremental updates to components and verification

Dissertation-Size Research Project “Reproducibility” [1]

1. Problem identification
   a. Theory building using methodological triangulation
2. Objective definition
   a. For one (or several) problem dimensions, find solution to problem, e.g.
      i. **How to make a component available to verification?**
      ii. **How to scale verification to whole distribution?**
3. Solution design
   a. Build out theory, extend build systems, create verification mechanism
4. Demonstration
   a. Ongoing demonstration using ROS 2
5. Evaluation
   a. Perform case study research with three industry partners for focused use case

[1] The topic is certainly worth more than one dissertation

2.3 Certifiability

A distribution is certifiable if (with reasonable effort)

- Products built on top of distribution can be certified

Goal is to make certification as easy as possible

In the ROS 2 case, there are at least two confounding factors

- Choice of domain (so many marks to choose from)
- Choice of operating system

Domains / Industries and Their Certifications

Industry	Example Standard	Assessment Levels
Automotive	ISO 26262	ASIL A-D
Industrial	IEC 62061 / ISO 13849-1	–
Shopfloor	–	–
Agriculture	–	–
Medical	IEC 62304 / ISO 13485	–

Choice of Operating System

Depends on choice of domain / industry (including potential industry sponsors)

- BuildRoot
- Yocto Project
- Zephyr RTOS

Key Challenges of Certifiability

Product certification

- Mismatch between existing and needed software architecture

Process certification

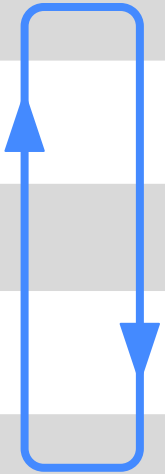
- Mismatch between community process and certification requirements

Dissertation-Size Research Project “Certifiability”

1. Problem identification
   a. Theory building using methodological triangulation
   b. Iterative, including choice of industry / domain
2. Objective definition
   a. For one (or several) problem dimensions, find solution to problem, e.g.
      i. How to resolve product / process mismatch?
      ii. How to perform a large-scale code-base refactoring?
      iii. How to transparently immigrate for a certification purpose?
3. Solution design
   a. Build out theory, participate in development
4. Demonstration
   a. Ongoing demonstration using ROS 2
5. Evaluation
   a. Perform case study research with three industry partners for focused use case

Alternative Research Design for “Certifiability”

#	Activity	Development	Action research
1	Set initial goal	Lead or help refactor code base to become more easily certificable	Build theory of how to make open source projects certifiable
2	Choose focus	Create and revise incremental build-out roadmap	
3	Plan the action	Plan development	Plan project engagement
4	Execute the action	Perform development in incremental steps, prepare patch sequence	Explain steps, submit patches, work with project
5	Observe outcome	Learn for next development steps	Track what works what doesn’t
6	Build out theory	Build out theory of how to make open source projects certifiable	



Other Challenges to (Open Source Distributions)

1. Hardware abstraction
2. Operating system integration
3. Installation and updates
4. LTS and backports
5. Open source license compliance
6. Layering e.g. ROS 2 + ROS Industrial

3. Business

(January 2023)

Who Owns ROS? Interest in ROS?

Multiple answers possible

- Open Robotics
- Nobody / a community

Broad industry support

- Broad industry steering committee
- Broad consortia support around the world

Focus is industrial robots e.g. ROS Industrial

Corporate Structure Behind Open Robotics

Open Source Robotics Foundation, Inc.

- https://www.guidestar.org/profile/45-5230545
- Total pre-pandemic (2019) revenues U$5.5M, operating at breakeven
- Net assets (2019) at US$2.1M according to Open Robotics 2019 Form 990

Open Source Robotics Corporation

- Subsidiary of the OSRF
- Used for consulting services around Gazebo, ROS

Open Robotics is used as a catch-all for both OSRF and OSRC

Open Robotics Employee Demographics

47 employees according to their website

- >50% employees are called (senior) software engineers
- 5 “software engineering supervisors” (= project managers?)
- One CEO, CFO, CHRO, BizDev person

Open Robotics is a consulting organization

- Average engineering salary = US$150K (according to Glassdoor)
- Total costs of 30 engineers, fully loaded = 30* US$225K = US$6.750M

Intrinsic (an Alphabet Company)

Intrinsic, an Alphabet (Google) company

- On a mission “to make industrial robots accessible”

Intrinsic concluded acquisition of OSRC in December 2022

- Intrinsic acquired skilled personnel in key project positions
- Intrinsic leaves a basic OSRF in place to host ROS 2 (and Gazebo)

Intrinsic’s goals are unclear, possible scenarios

- Heavy-handed (worst case): Hamstring ROS 2, start owning it
- Light touch (best case): Foster and support a fair and equal ecosystem

The Tools Available to Intrinsic

Hard mechanisms

- ROS trademark (officially with OSRF) [1]
- Patents: Status unknown
- Committer status: Core committers employed now by Intrinsic

Soft mechanisms

- Community leadership: Core committers employed by Intrinsic
- Secondary trademarks (ROSCON, etc.) [1]

[1] See https://www.ros.org/imgs/ROSBrandGuide.pdf

Apex.AI

Apex.AI provides Apex.OS, a fork of ROS 2 that

- “Has been made real-time, reliable, and deterministic”
- “Is developed in sync with future releases of ROS 2”

Apex.AI business focus is on software-defined vehicles

Any Specific Neglected Domain?

Common

- SDV, mobility
- Industrial robots

Observed

- Space (ROSCON)
- Medical technology

Others?

Thank you! Any questions?

dirk.riehle@fau.de – https://oss.cs.fau.de

dirk@riehle.org – https://dirkriehle.com – @dirkriehle

Legal Notices

License

- No license given, please ask for permission to use

Copyright

-