Current Status of ROS 2
Hands-on Feature Overview

Dr. Ralph Lange
Robert Bosch GmbH, Corporate Research
7 May 2019
ROS-I EU Spring ’19 Workshop
Current Status of ROS 2 - Hands-on Feature Overview

Agenda

- Why, Roadmap and Governance
- Installation and Command Line Tools
- ROS 2 Architecture and Layer-by-Layer Walkthrough
- Launch
- System Modes
- Real-Time
- Bridge to ROS 1
- Security
- FMI Adapter
- Build Tooling
- Contributing to ROS 2
Why, Roadmap and Governance
Current Status of ROS 2 - Hands-on Feature Overview

Why ROS?

Use-case / basic requirements
- Teams of multiple robots
- Small embedded platforms
- Real-time systems
- Non-ideal networks
- Production environments
- Prescribed patterns for building and structuring systems

History
- Development started in 2014
- First stable release in December 2017
- First LTS release planned for May 2019

http://design.ros2.org/articles/why_ros2.html
https://index.ros.org/doc/ros2/Roadmap/
Current Status of ROS 2 - Hands-on Feature Overview

Key Features of ROS 2

- Data Distribution Service (DDS) as middleware
  - No central ROS master by DDS discovery
  - Built-in security: authentication, access control, encryption
- Run multiple nodes in one process
  - No differentiation between nodelets and nodes
  - Efficient zero-copy communication
- Node lifecycle and (more) deterministic launch
- ROS core functionality in C for easier support of different programming languages
- Real-time ready core algorithms
- New build system ament+colcon
- Windows-support
- Use of C++14 and Python 3 standards

Example from https://github.com/ros2/sros2

http://design.ros2.org/
Current Status of ROS 2 - Hands-on Feature Overview

ROS 2 Governance

- Sep 2018: Formation of ROS 2 Technical Steering Committee
  “broaden participation to accelerate ROS 2 delivery, starting with these areas: determining the roadmap, developing core tools and libraries, and establishing working groups to focus on important topics”

- Monthly meetings
- Primarily status reports and roadmap development
- Members provide feedback/priorities on OSRF tasks and contribute own work

Current working groups

https://index.ros.org/doc/ros2/Governance/
https://discourse.ros.org/t/introducing-the-ros-2-technical-steering-committee/6132
Current Status of ROS 2 - Hands-on Feature Overview

ROS 2 Roadmap


- Roadmap for next release Dashing Diademata:
Installation
Current Status of ROS 2 - Hands-on Feature Overview

Installation

- Binary packages for
  - Ubuntu 18.04 (Bionic)
  - OS X
  - Windows

See https://index.ros.org/doc/ros2/Installation/

- ROS 2 Crystal can be also build directly on Ubuntu 16.04 Xenial – with exception of rqt packages

See https://index.ros.org/doc/ros2/Installation/Linux-Development-Setup/
Current Status of ROS 2 - Hands-on Feature Overview
Command Line Tools

▶ Don’t forget to source!
  ▶ local_setup.bash applies settings from current workspace only, i.e. without parent

▶ All run-time tools integrated in ros2
▶ https://github.com/ros2/ros2cli
  ▶ Every verb is in a separate package
▶ Possible to extend, cf. for example https://github.com/ros2/sros2/

▶ Meta build tool: colcon
  ▶ Later more …
Hands-On

source /opt/ros/crystal/setup.bash
ros2 run demo_nodes_cpp talker

source /opt/ros/crystal/setup.bash
ros2 run demo_nodes_py listener

ros2 topic list
ros2 topic hz/chatter
ros2 topic pub/chatter std_msgs/String " data: 'test' "
Hands-On

Make and enter workspace folder + src

```
git clone https://github.com/micro-ROS/micro-ROS_kobuki_demo.git
```

cd ...

colcon build

source install/local_setup.bash

```
ros2 launch kobuki_bringup kobuki_bringup.launch.py
```

Second window:

```
ros2 run odom_to_tf circle_odom_publisher
```
Architecture and Layer-by-Layer Walkthrough
Current Status of ROS 2 - Hands-on Feature Overview

ROS 2 Architecture

**rcl** – language-specific ROS client libraries

**rcl** – C library
  ▶ Ensures same core algorithms in all language-specific client libraries

**rmw** – ROS middleware interface
  ▶ Hide specifics of DDS implementations
  ▶ Streamline QoS configuration

**rmw_*** – DDS adapters

Diagram:
- User code
- `rclcpp (-py, ...)`
- `rcl`
- `rmw`
- Specific `rmw_adapter`
- Specific DDS implementation
Current Status of ROS 2 - Hands-on Feature Overview

ROS 2 Layer by Layer: DDS and rmw

- You should be the experts after yesterday’s talks :-)

- FastRTPS is the default distributed with ROS 2

- Instructions for OpenSplice and RTI Connext:
  [https://index.ros.org/doc/ros2/Installation/Linux-Install-Debians/#install-additional-rmw-implementations](https://index.ros.org/doc/ros2/Installation/Linux-Install-Debians/#install-additional-rmw-implementations)

  - Adapters for those both are provided as binary packages

- rmw defines interface to be implemented by adapters
  - E.g., search `rmw_get_publisher_names_and_types_by_node` in
Current Status of ROS 2 - Hands-on Feature Overview

ROS 2 Layer by Layer: Other Middlewares

rmw defines few basic concepts only:

- some endpoint naming
- publish-subscribe
- request-response

Several non-DDS implementations have been developed, for example:

- OPC UA – an M2M communication protocol
  https://www.elektrotechnik.vogel.de/mit-opc-ua-gelingt-die-nahtlose-integration-mobiler-robotersysteme-a-537411/ (in Germany only)

- Intel’s Distributed Publish & Subscribe for IoT
  https://github.com/ros2/rmw_dps
Current Status of ROS 2 - Hands-on Feature Overview

ROS 2 Layer by Layer: rcl – ROS Client Support Library

**rcl** implements major ROS concepts in C

- node, naming, publisher, subscription, service, action, lifecycle, time, logging, ...
- Very well documented API: [https://github.com/ros2/rcl/](https://github.com/ros2/rcl/)
- Allows use of custom allocator (e.g., TLSF)
  - [https://github.com/ros2/rcutils](https://github.com/ros2/rcutils) -> include/rcutils/allocator.h
- Lacks of execution management – no threads on rcl layer

int main(int argc, char* argv[])
{
    rcl_init_options_t initOptions = rcl_get_zero_initialized_init_options();
    rcl_init_options_init(&initOptions, rcutils_get_default_allocator());
    rcl_context_t context = rcl_get_zero_initialized_context();
    rcl_init(argc, argv, &initOptions, &context);
    rcl_init_options_fini(&initOptions);

    rcl_node_t node = rcl_get_zero_initialized_node();
    rcl_node_options_t nodeOptions = rcl_node_get_default_options();
    rcl_node_init(&node, "rcl_int32_subscriber", ",", &context, &nodeOptions);

    const rosidl_message_type_support_t * typeSupport = ROISIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32);
    rcl_subscription_t subscription = rcl_get_zero_initialized_subscription();
    rcl_subscription_options_t subscriptionOptions = rcl_subscription_get_default_options();
    rcl_subscription_init(&subscription, &node, typeSupport, "std_msgs_msg_Int32", &subscriptionOptions);

    rcl_wait_set_t waitSet = rcl_get_zero_initialized_wait_set();
    rcl_wait_set_init(&waitSet, 1, 0, 0, 0, rcl_get_default_allocator());
    size_t index;
    rcl_wait_set_add_subscription(&waitSet, &subscription, &index);
    rcl_wait(&waitSet, RCL_MS_TO_NS(1000));

    // Check waitSet for available messages
    std_msgs__msg__Int32 msg;
    rmw_message_info_t messageInfo;
    rcl_take(&subscription, &msg, &messageInfo);
    printf("Message data is %ld\n", msg.data);

    rcl_subscription_fini(&subscription, &node);
    rcl_node_fini(&node);
    return 0;
}
Current Status of ROS 2 - Hands-on Feature Overview

ROS 2 Layer by Layer: Waitsets

- SELECT-like mechanism to query for messages on subscribed topics

- Does not provide information on the message count, i.e. current queue size
rclcpp wraps rcl into C++ data types and brings

- Execution management
- Intraprocess communication
- Parameters
- Callback_groups
- Listeners for graph, parameters

- Allows dynamic creation and deletion of almost everything
- Elaborate architecture
Current Status of ROS 2 - Hands-on Feature Overview

ROS 2 Layer by Layer: rclcpp Architecture

https://github.com/ros2/rclcpp/ → include/rclcpp/node_interfaces
class MyNode : public rclcpp::Node {
public:
    MyNode() : Node("my_node") {
        auto callback = [this](const std_msgs::msg::String::SharedPtr msg) -> void {
            RCLCPP_INFO(this->get_logger(), "I heard: [%s]", msg->data.c_str());
        }
        sub_ = create_subscription<std_msgs::msg::String>("/chatter", callback);
    }

private:
    rclcpp::Subscription<std_msgs::msg::String>::SharedPtr sub_;}

int main(int argc, char * argv[]) {
    rclcpp::init(argc, argv);
    auto node = std::make_shared<MyNode>(topic);
    rclcpp::spin(node);
    rclcpp::shutdown();
    return 0;
}
Current Status of ROS 2 - Hands-on Feature Overview

ROS 2 Layer by Layer: rclcpp Parameters

- Managed with node – no parameter server
- Handy tooling with ros2 param
- Typed (and to be declared from Dashing on)
  - ... but nice template in rclcpp/node.hpp
- Old semantics – no declaration – still available via NodeOptions

```cpp
template<typename ParameterT>
auto
declare_parameter(
    const std::string & name,
    const ParameterT & default_value,
    const rcl_interfaces::msg::ParameterDescriptor & parameter_descriptor = rcl_interfaces::msg::ParameterDescriptor());

rclcpp::Parameter parameter;
if (nodeInterface->get_parameter(name, parameter)) {
    value = parameter.as_double();
}
```
Current Status of ROS 2 - Hands-on Feature Overview
ROS 2 Layer by Layer: rclcpp – Node Composition and Executor

User code

Node 1
onNewLaserData(…)
add_node

Node 2
onNewLaserData(…)
onCameraImage(…)
add_node
add_node

Node 3
processOdometry(…)
add_node
message

ROS client library

Executor
(Single or multi-threaded)
- Wait for DDS queue
- Take item from queue
- Call corresponding callback with this item

rmw
wait
rmw_take
message

Type conversion, wait sets

DDS

Topic ‘laser’

 DDS client

Topic ‘camera’

Topic ‘odometry’

User code

ROS client library

Executor

rmw

Type conversion, wait sets

DDS

Topic ‘laser’

Topic ‘camera’

Topic ‘odometry’
Current Status of ROS 2 - Hands-on Feature Overview

ROS 2 Layer by Layer: rclcpp – Node Composition and Executor

```cpp
int main(int argc, char * argv[]) {
  rclcpp::init(argc, argv);
  auto node = rclcpp::Node::make_shared("listener");
  auto sub = node->create_subscription<std_msgs::msg::String>("/chatter", callback);
  rclcpp::spin(node);
  rclcpp::shutdown();
}
```
Current Status of ROS 2 - Hands-on Feature Overview

ROS 2 Layer by Layer: rclcpp – Node Composition and Executor

```cpp
int main(int argc, char * argv[]) {
    rclcpp::init(argc, argv);
    rclcpp::executors::SingleThreadedExecutor executor;
    auto talker_node = std::make_shared<Talker>();
    executor.add_node(talker_node);
    auto listener_node = std::make_shared<Listener>();
    executor.add_node(listener_node);
    executor.spin();
    rclcpp::shutdown();
}
```
Current Status of ROS 2 - Hands-on Feature Overview
ROS 2 Layer by Layer: Executor Semantics

- Analyzed and described current semantics of Executor implementation with models from real-time community
  - https://github.com/boschresearch/ros2_response_time_analysis

- Complicated semantics consisting of
  - Priority scheduling on first level and
  - Round-robin for subscriptions on second level

- See https://github.com/ros2/rclcpp/blob/master/rclcpp/src/rclcpp/executor.cpp#L540
- On-going discussion in several GitHub issues and real-time working group
Current Status of ROS 2 - Hands-on Feature Overview

ROS 2 Layer by Layer: Managed Nodes (aka Lifecycle)

Lifecycle design
by Biggs & Foote, 2015
- Inspired by OMG RTC
- Managed Lifecycle

Concepts:
- Primary states
- Transition states
- Transitions

Tool: `ros2 lifecycle`

Adapted from
http://design.ros2.org

https://design.ros2.org/articles/node_lifecycle.html
https://github.com/bosch-robotics-cr/ros1_lifecycle
Current Status of ROS 2
ROS 2 Layer by Layer: rclpy

```python
class OdomToTf(Node):
    def __init__(self):
        super().__init__('odom_to_tf')

        self.last_position = Vector3()
        self.last_orientation = Quaternion()
        self.last_orientation.x = 0.0
        self.last_orientation.y = 0.0
        self.last_orientation.z = 0.0
        self.last_orientation.w = 1.0

        self.sub_robot_pose = self.create_subscription(Vector3, '/robot_pose', self

        self.pub_tf = self.create_publisher(TFMessage, '/tF')

        self.tf_timer = self.create_timer(0.05, self.tf_timer_callback)

    def robot_pose_callback(self, self, msg):
        self.last_position.x = msg.x;
        self.last_position.y = msg.y;
        self.last_orientation.z = math.sin(msg.z / 2.0);
        self.last_orientation.w = math.cos(msg.z / 2.0);

Example from https://github.com/micro-ROS/micro-ROS_kobuki_demo
```

- [https://github.com/ros2/rclpy/](https://github.com/ros2/rclpy/)
- Well-developed and maintained
  ... and used for most command line tools
Current Status of ROS 2

ROS 2 Layer by Layer: rclpy

- [https://github.com/ros2/rclpy/](https://github.com/ros2/rclpy/)

- Well-developed and maintained
  ... and used for most command line tools

- No need of CMakeLists.txt with colcon
  but real Python package
Launch
Current Status of ROS 2 - Hands-on Feature Overview

Launch – Overview

- Launch files are Python scripts, i.e. very little prescribed structure
- `ros2 launch`
- Little documentation available
- On-going discussion and developments ... also regarding more prescribed mechanisms and structures
- Major know concepts from ROS 1 implemented
  - Nested launch files
  - Arguments
  - Parameter files

Example from https://github.com/boschresearch/fmi_adapter_ros2

```python
def generate_launch_description():
    fmu_path = (ament_index_python.packages.get_package_share_directory('fmi_adapter_examples') + '/share/DampedPendulum.fmu')

    fmi_adapter_description = launch.actions.IncludeLaunchDescription(
        launch.launch_description_sources.PythonLaunchDescriptionSource(ament_index_python.packages.get_package_share_directory('fmi_adapter') + '/launch/fmi_adapter_node.launch.py'),
        launch_arguments={'fmu_path': fmu_path}.items())

    description = launch.LaunchDescription()
    description.add_action(fmi_adapter_description)

    return description
```
Launch – New Concepts

Support for Managed Nodes (Lifecycle)

- Launch allows to register event handlers – e.g., on lifecycle state changes
- Nodes can be started in synchronized stages or as a pipeline or ...

```python
activate_delay_node = launch.actions.EmitEvent(event=launch_ros.events.lifecycle.ChangeState(
    lifecycle_node_matcher=launch.events.process.matches_action(delay_node),
    transition_id=lifecycle_msgs.msg.Transition.TRANSITION_ACTIVATE))

pendulum_fmu_path = (  
    smt_index_python.packages.get_package_share_directory('fmi_adapter_examples') +  
    '/share/DampedPendulum.fmu')

pendulum_node = launch_ros.actions.LifecycleNode(  
    package='fmi_adapter',
    node_executable='fmi_adapter_node',
    node_name='damped_pendulum',
    node_namespace='example',
    parameters=[
        {'fmu_path': pendulum_fmu_path,
         'l': 25.0,  # set pendulum length to 25m.
         'd': 0.01  # Reduce damping ratio (default is 0.1).
        }],
    output='screen')

configure_pendulum_node = launch.actions.EmitEvent(  
    event=launch_ros.events.lifecycle.ChangeState(  
        lifecycle_node_matcher=launch.events.process.matches_action(pendulum_node),
        transition_id=lifecycle_msgs.msg.Transition.TRANSITION_CONFIGURE))
```

Example from https://github.com/boschresearch/fmi_adapter_ros2
Current Status of ROS 2 - Hands-on Feature Overview
Launch – New Concepts

Support for Managed Nodes (Lifecycle)

- Launch allows to register event handlers – e.g., on lifecycle state changes
- Nodes can be started in synchronized stages or as a pipeline or ...

Node composition from shared libraries

- https://index.ros.org/doc/ros2/Tutorials/Composition/
- Example: https://github.com/boschresearch/fmi_adapter_ros2

Branch Dashing → examples → launch
System Modes

On-going activity, in the EU project OFERA for micro-ROS
Current Status of ROS 2 - Hands-on Feature Overview

Extended Lifecycle and System Modes

We observed three different but closely interwoven aspects to be handled on the deliberation layer:

1. **Task Handling**: Orchestration of the actual task, the straightforward, error-free flow

2. **Contingency Handling**: Handling of task-specific contingencies, e.g., expectable retries and failure attempts, obstacles, low battery.

3. **System Error Handling**: Handling of exceptions, e.g., sensor/actuator failures.

The mechanisms being used to orchestrate the skills are service and action calls, re-parameterizations, set values, activating/deactivating of components, etc.

**Function-oriented calls vs. system-oriented calls intermixed and implicit in code**
Current Status of ROS 2 - Hands-on Feature Overview
Extended Lifecycle and System Modes

Assumption: System is built from nodes with lifecycle

Introducing:

1. Modes
   - Specialize ACTIVE state
   - Set of values or ranges for node parameters

2. Notion of (sub)system
   - Collection of nodes and (sub)systems (system of system)

3. Modes for (sub)systems
   - (Sub)system get same lifecycle as nodes
   - Modes of (sub)systems specify modes of their parts

https://micro-ros.github.io/system_modes/
https://github.com/microROS/system_modes
# Clone system_modes
colcon build & source install/local_setup.bash
ros2 run system_modes_examples drive_base

# New bash
ros2 run system_modes_examples manipulator

# New bash
ros2 run system_modes mode-manager --modelfile
    src/system_modes/system_modes_examples/example_modes.yaml

# New bash
ros2 run system_modes mode-monitor --modelfile
    src/system_modes/system_modes_examples/example_modes.yaml

# New bash
ros2 service call /actuation/change_state lifecycle_msgs/ChangeState
    "{transition: {id: 1, label: 'configure'}}"
ros2 service call /actuation/change_state lifecycle_msgs/ChangeState
    "{transition: {id: 3, label: 'activate'}}"
ros2 service call /actuation/change_mode system_modes/ChangeMode
    "{node_name: 'actuation', mode_name: 'PERFORMANCE'}"
ros2 service call /drive_base/change_mode system_modes/ChangeMode
    "{node_name: 'drive_base', mode_name: 'SLOW'}"
ros2 service call /manipulator/change_mode system_modes/ChangeMode
    "{node_name: 'manipulator', mode_name: 'WEAK'}"

https://github.com/micro-ROS/system_modes/tree/master/system_modes_examples
Real-Time

On-going activity, partially in the EU project OFERA for micro-ROS
Current Status of ROS 2 - Hands-on Feature Overview

Callback-group-level Executor

Drive-Base node:
- onEmergencyStopMsg
- onCmdVel
- publishWheelTicks
- reportDiagnosticsData

Other node:
- ...
- ...

Executor Executor Executor

OS Scheduler
Current Status of ROS 2 - Hands-on Feature Overview

Callback-group-level Executor

User code

Node 1

Node 2

Node 3

Meta-Executor

Single- or Multi-threaded Executor
Homogeneous scheduling properties for all threads

ROS client library

Single- or Multi-threaded Executor
Homogeneous scheduling properties for all threads

rmw

Wait sets are defined on granularity of callback groups

Topic 'laser'

Topics: 'camera', 'odometry'

Type conversion, wait sets

Node 1

Node 2

Node 3

DDS client
Current Status of ROS 2 - Hands-on Feature Overview

Callback-group-level Executor

Ping Node

- Configurable `rt_send_period`
- Measured `rt_receive_rate`

Pong Node

- Configurable `rt_send_period`
- Measured `rt_receive_rate`

With SCHED_FIFO and core pinning:

- High priority real-time thread
- Low priority real-time thread

Short string with simple counter

Configurable `rt_busy_loop` burns cpu cycles

Configurable `be_busy_loop` burns cpu cycles

rt: real-time       be: best effort
Current Status of ROS 2 - Hands-on Feature Overview

Callback-group-level Executor

Ping Node

Pong Node

Executor

Executor

github.com/boschresearch/ros2_examples → meta-executor
Bridge to ROS 1
Current Status of ROS 2 - Hands-on Feature Overview

Bridge to ROS 1
Bridge to ROS 1

Current Status of ROS 2
Hands-On Feature Overview

**Hands-On**

```
sudo apt install ros-crystal-ros1-bridge
```

```
source /opt/ros/crystal/setup.bash
source /opt/ros/crystal/local_setup.bash
ros2 run demo_nodes_cpp talker
```

```
ros2 run ros1_bridge dynamic_bridge
```

```
source /opt/ros/crystal/setup.bash
source /opt/ros/melodic/setup.bash
roscore
```

```
rosrun roscpp_tutorials listener
```

Press Ctrl+C to interrupt

```
source /opt/ros/melodic/setup.bash
roscore http://training-vm:11311
```

```
rosrun roscpp_tutorials listener
```

© Robert Bosch GmbH 2019. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.
Security
Security

Current Status of ROS 2 - Hands-on Feature Overview

- Use of DDS Security mechanisms
- [https://www.omg.org/spec/DDS-SECURITY/](https://www.omg.org/spec/DDS-SECURITY/)
  - Version 1.1 is from July 2018

- SROS2 put convenience tooling on top: ros2 security
- [https://github.com/ros2/sros2](https://github.com/ros2/sros2)
- Requires rebuild with colcon build --cmake-args -DSECURITY=ON

FMI_Adapter
Current Status of ROS 2 - Hands-on Feature Overview

FMI_Adapter

- Example: ActiveShuttle DevKit at Bosch
- Problem with lever to caster wheels
  - Slipping drive wheels
  - Blocking caster wheels
- Root cause: Bore friction
Current Status of ROS 2 - Hands-on Feature Overview

FMI_Adapter

- Model-based approach: **Path Filter** reduces bore friction
- Realized with generic **ROS-fmi** wrapper
- Reduction of peak currents, jerk, oscillations

![Diagram showing path filter and motor currents]

Schröder et al.: “Enhanced Motion Control of a Self-Driving Vehicle Using Modelica, FMI and ROS”, Modelica Conference 2019
Current Status of ROS 2 - Hands-on Feature Overview

FMI_Adapter

http://fmi-standard.org/
Current Status of ROS 2 - Hands-on Feature Overview

FMI_Adapter

Shared library
- Equations
- Solver

modelDescription.xml
(C sources)
Current Status of ROS 2 - Hands-on Feature Overview

FMI_Adapter

https://github.com/boschresearch/fmi_adapter_ros2
Current Status of ROS 2 - Hands-on Feature Overview

**FMI_Adapter**

```
FMIAdapter adapter("MyModel.fmu");

vector<string> names = 
    adapter.getParameterNames();

adapter.setInputValue("phi", 4.5);
adapter.doStepsUntil(ros::Time::now());
double y = adapter.getOutputValue("y");
```

[GitHub Link](https://github.com/boschresearch/fmi_adapter_ros2)
Hands-On
Build Tooling
Current Status of ROS 2 - Hands-on Feature Overview

Build Tooling (very brief overview)

Colcon + Ament + Cmake

- Colcon – meta-build system
  - Can be extended for other, non-ROS environments
    
    https://colcon.readthedocs.io/

- Ament – “iteration on catkin”
  - Many handy CMake macros for linting, package declaration, testing, ...

- colcon build, test, test-results

- **but**: ros2 pkg create
Contributing
Current Status of ROS 2 - Hands-on Feature Overview

Contributing

ROS 2 core packages and features

New packages
- Bloom as with ROS 1:
  - Separate build farm instance at http://build.ros2.org/

Porting packages
- Contact ROS 1 maintainer early
- Decide about refactoring, coding guidelines, ... – maybe start with reformatting PR first
THANK YOU

Dr. Ralph Lange
Robert Bosch GmbH, Corporate Research
Robert-Bosch-Campus 1
71272 Renningen, Germany
ralph.lange@de.bosch.com | github.com/ralph-lange