xtUML in the Field

Robotic soil sampling



Levi Starrett
September 2019

Presentation goals

- Give context and overview of the business and technology at Rogo
- Demonstrate how xtUML is currently being used to add value at Rogo
- Describe and report on MC-ROS a new model compiler project being used at Rogo
- Discuss where xtUML can continue to be used at Rogo and Rogo’s place in the community

Design: mechanical



Design: mechanical



- 60 HP hydraulic skid steer platform (Bobcat T450)

Design: mechanical



- 60 HP hydraulic skid steer platform (Bobcat T450)
- Hydraulic lift, carriage, and auger with automatic cleaning collar

Design: mechanical



- 60 HP hydraulic skid steer platform (Bobcat T450)
- Hydraulic lift, carriage, and auger with automatic cleaning collar
- Closed-loop vacuum system with positive pressure

Design: mechanical



- 60 HP hydraulic skid steer platform (Bobcat T450)
- Hydraulic lift, carriage, and auger with automatic cleaning collar
- Closed-loop vacuum system with positive pressure
- Automatic packager with forming collar and electric heat sealers, barcode reader

Design: mechanical



- 60 HP hydraulic skid steer platform (Bobcat T450)
- Hydraulic lift, carriage, and auger with automatic cleaning collar
- Closed-loop vacuum system with positive pressure
- Automatic packager with forming collar and electric heat sealers, barcode reader
- Control box houses all control systems (PLC, Linux nav computer, HMI, LTE and WiFi radio)

Design: mechanical



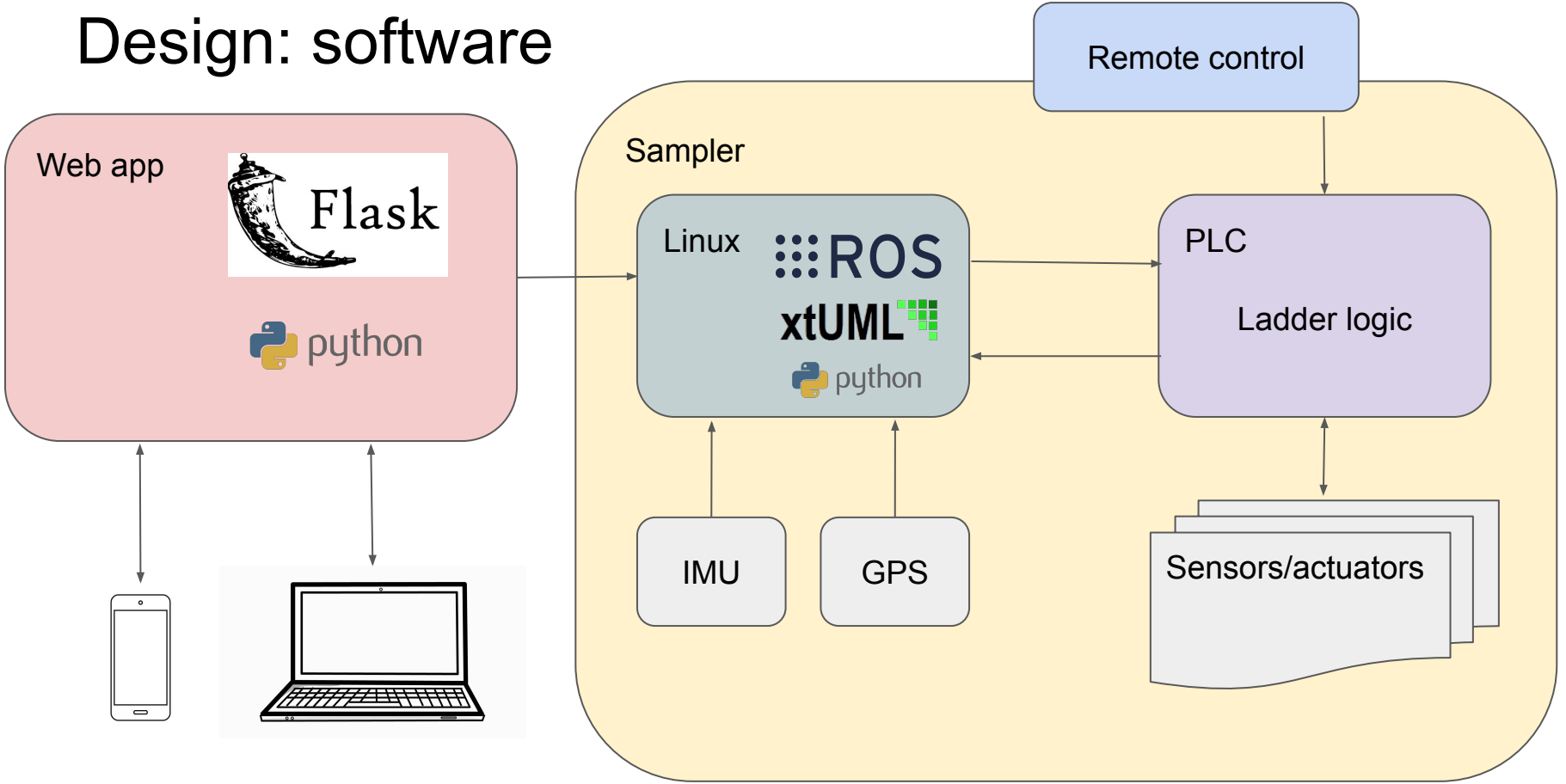
- 60 HP hydraulic skid steer platform (Bobcat T450)
- Hydraulic lift, carriage, and auger with automatic cleaning collar
- Closed-loop vacuum system with positive pressure
- Automatic packager with forming collar and electric heat sealers, barcode reader
- Control box houses all control systems (PLC, Linux nav computer, HMI, LTE and WiFi radio)
- GPS receiver with RTx correction

Design: mechanical

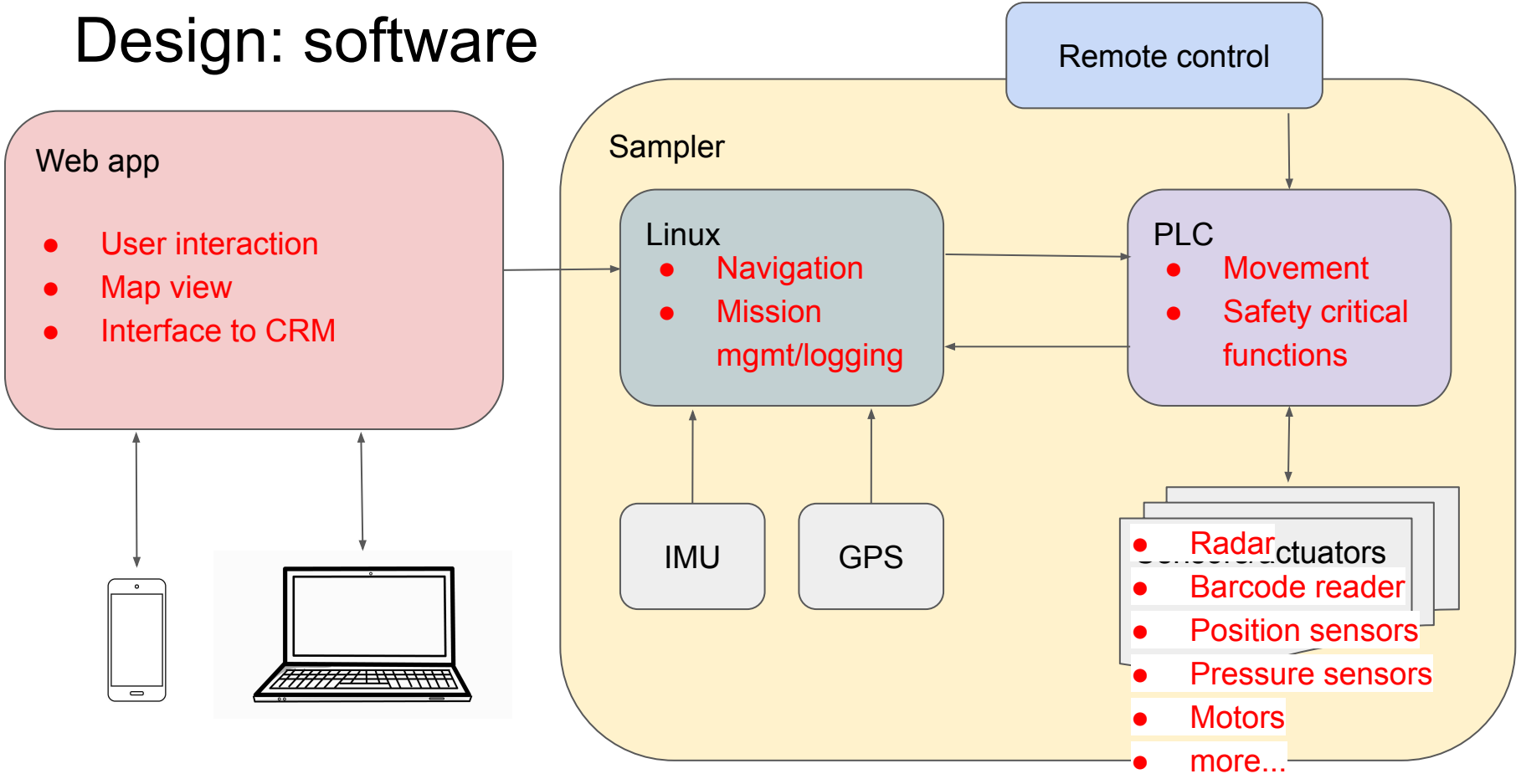


- 60 HP hydraulic skid steer platform (Bobcat T450)
- Hydraulic lift, carriage, and auger with automatic cleaning collar
- Closed-loop vacuum system with positive pressure
- Automatic packager with forming collar and electric heat sealers, barcode reader
- Control box houses all control systems (PLC, Linux nav computer, HMI, LTE and WiFi radio)
- GPS receiver with RTx correction
- Inertial measurement unit

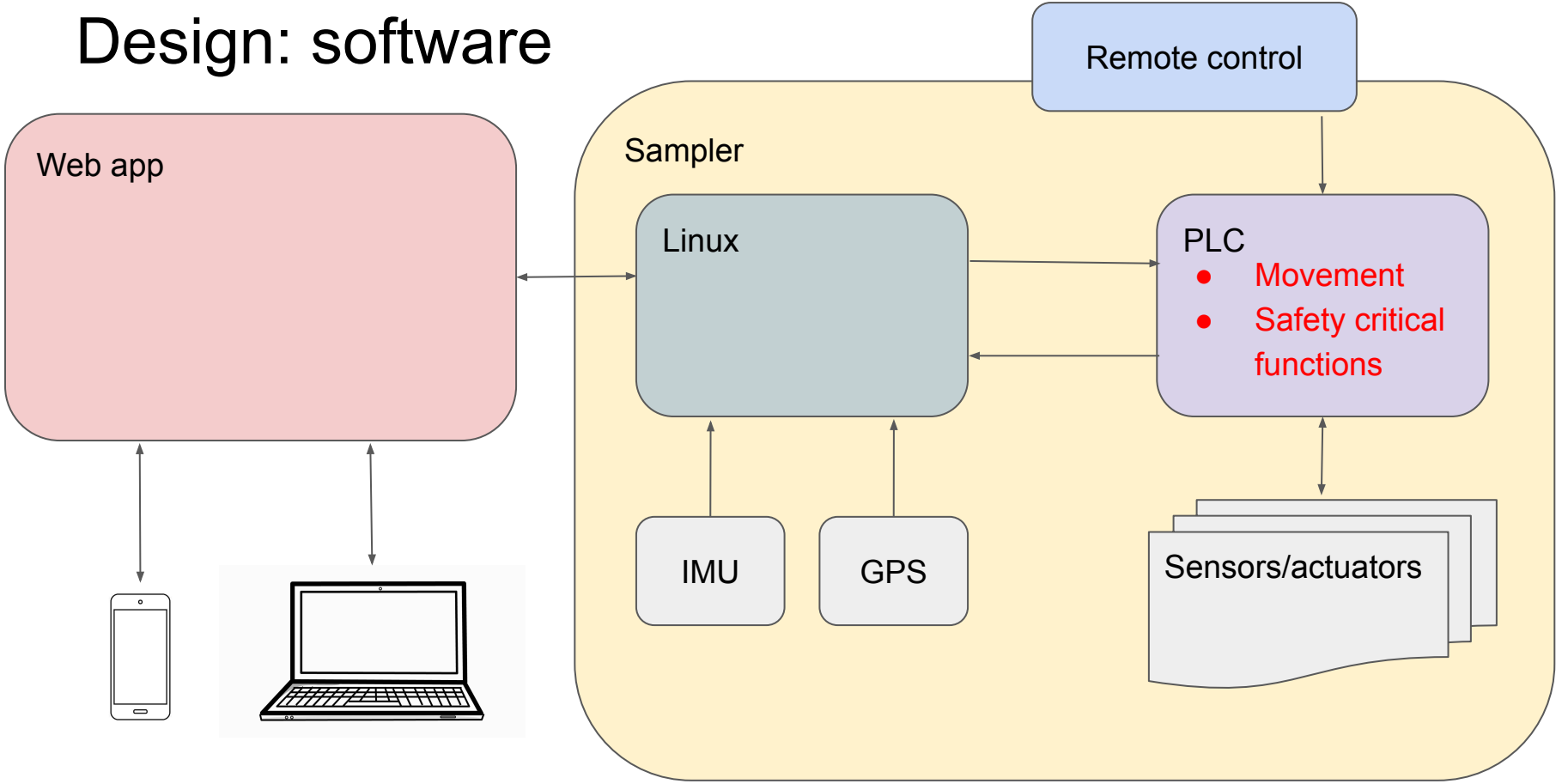
Design: software



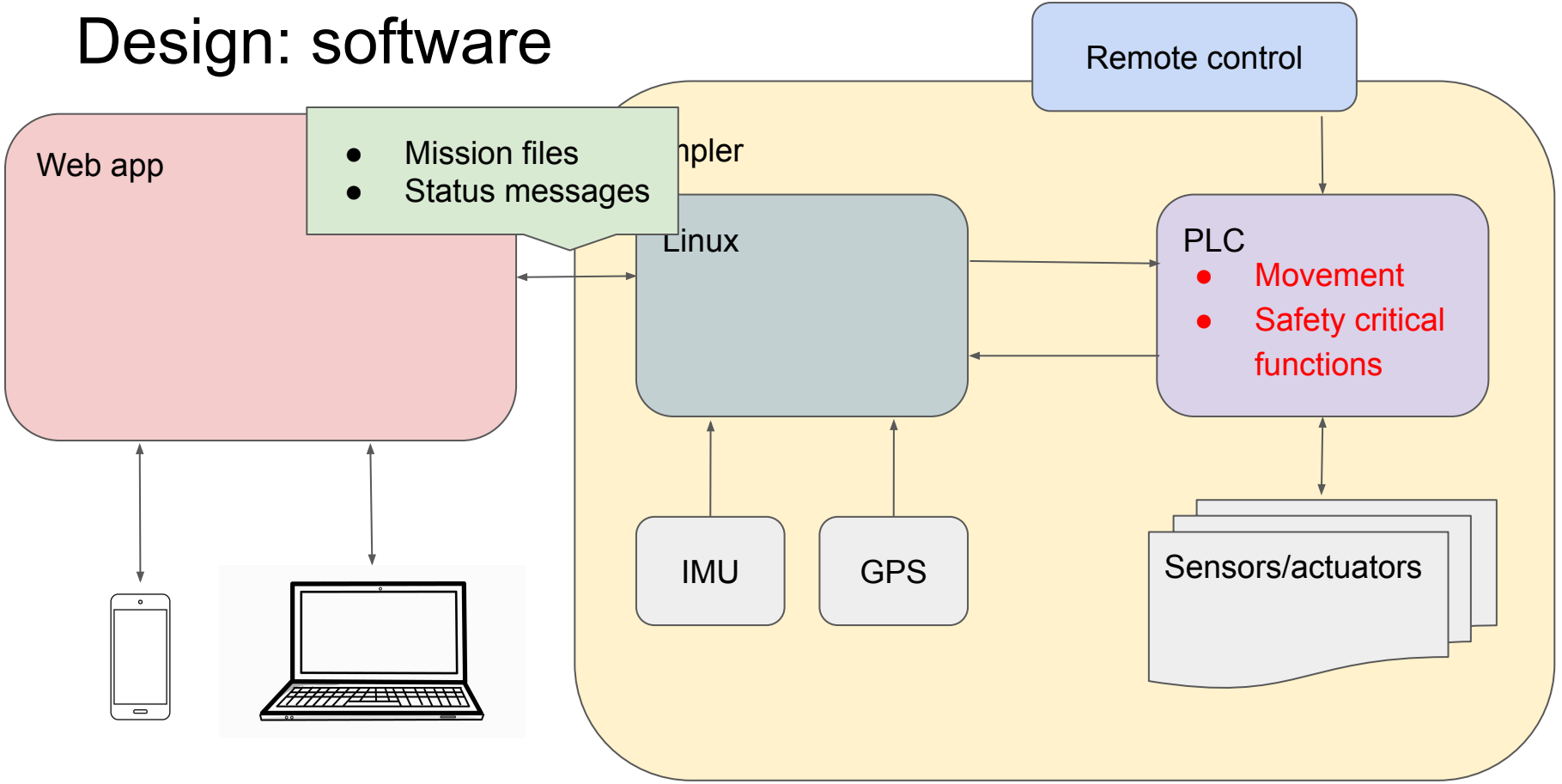
Design: software



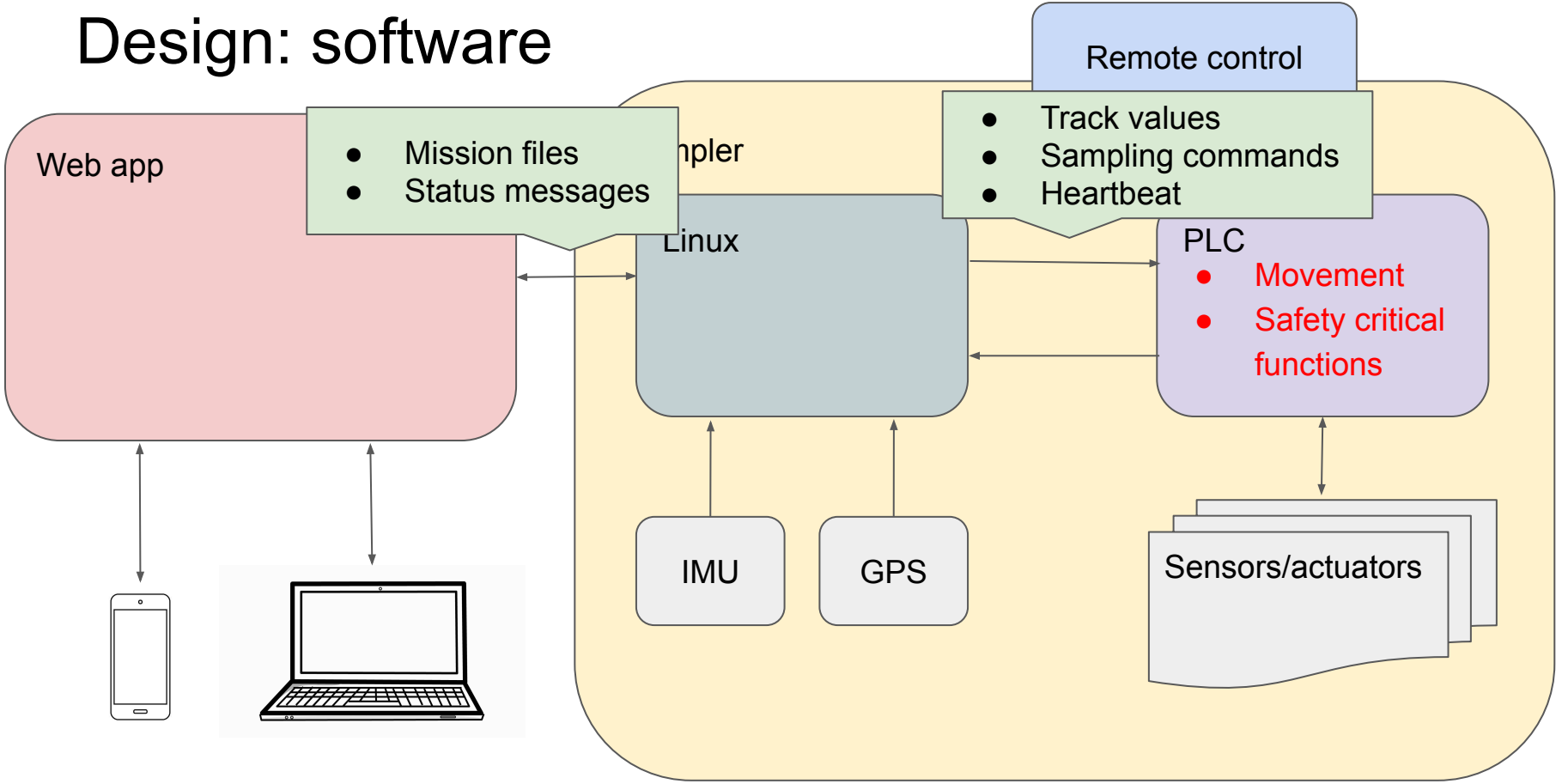
Design: software



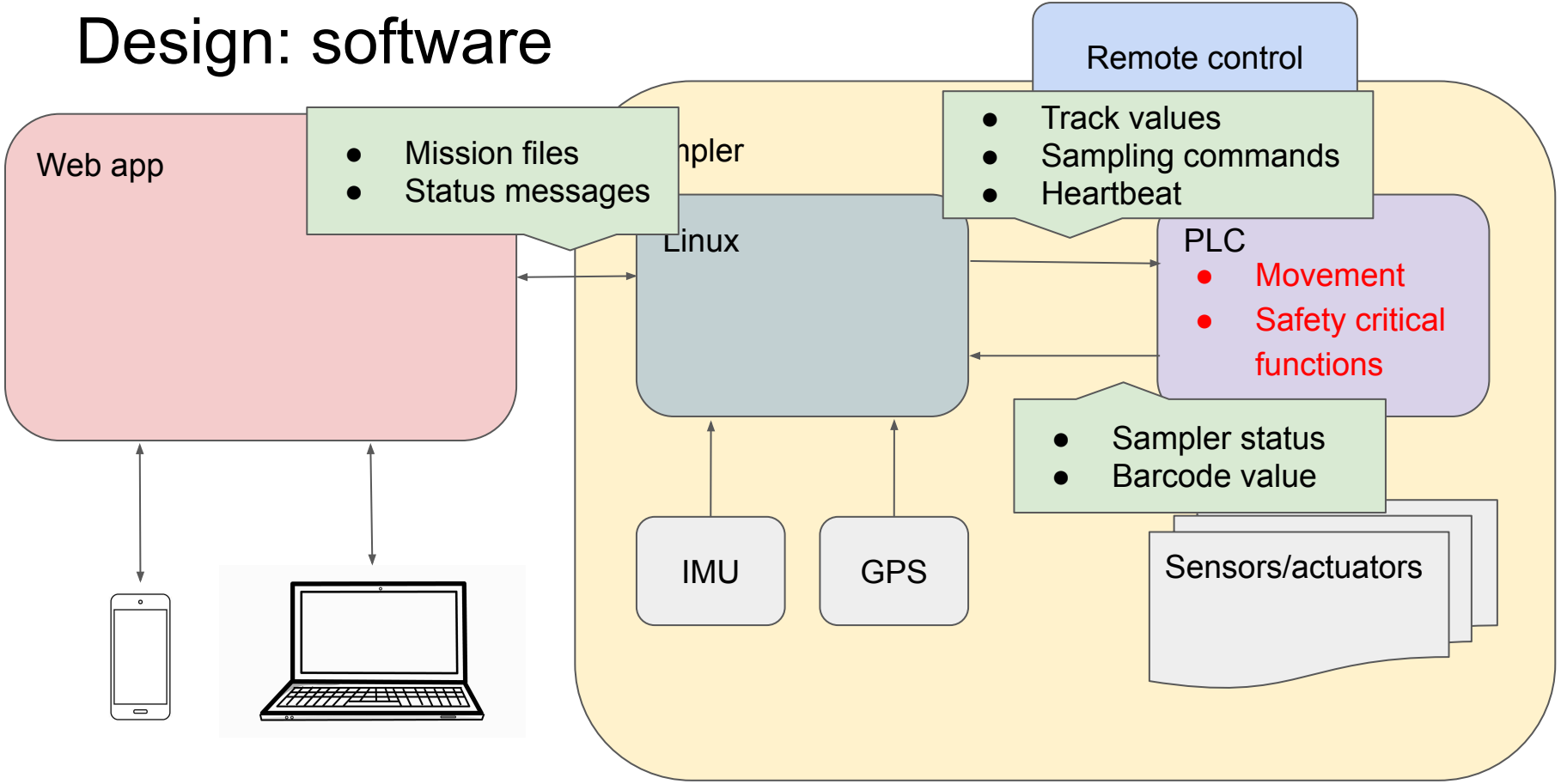
Design: software



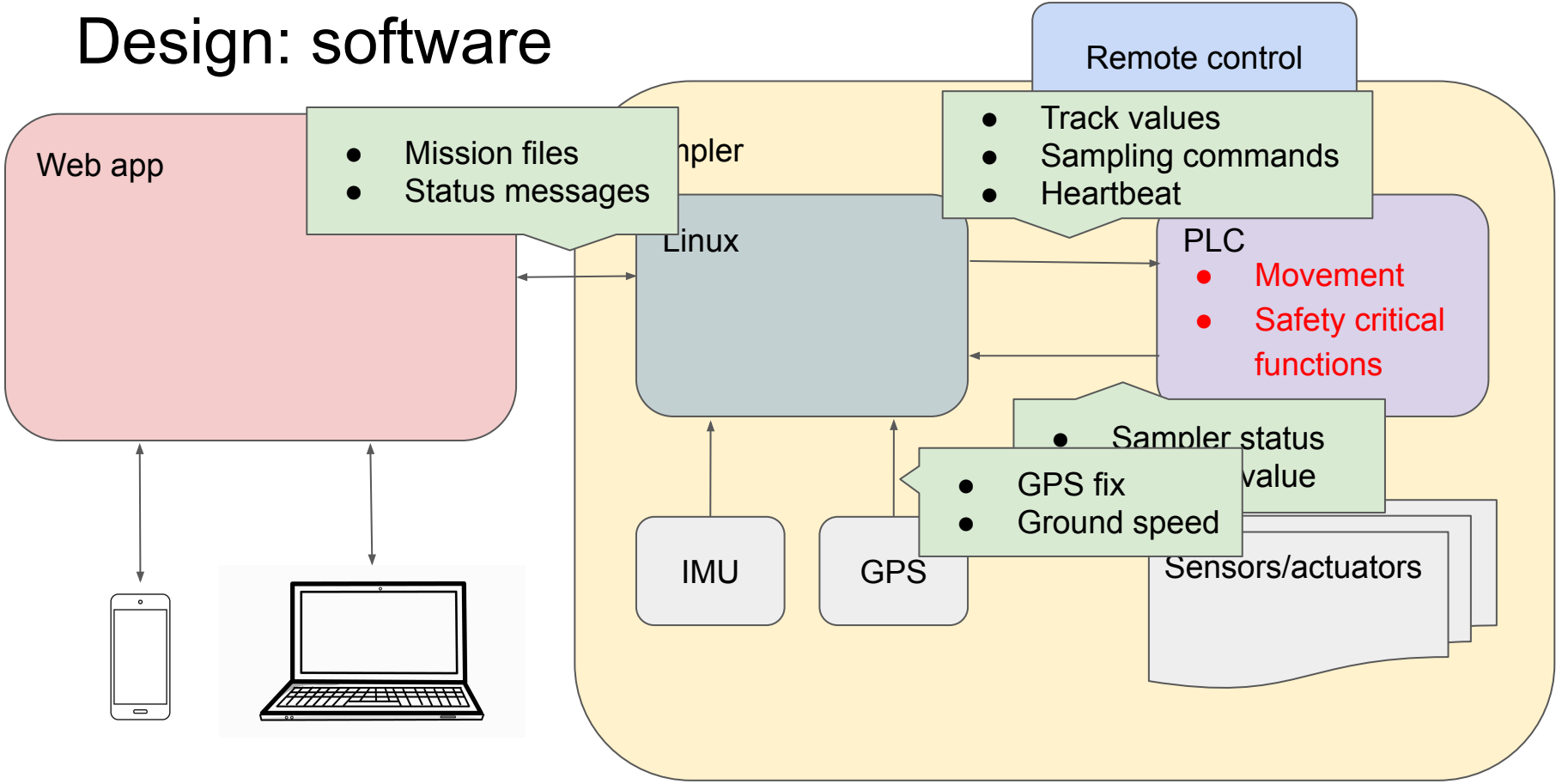
Design: software



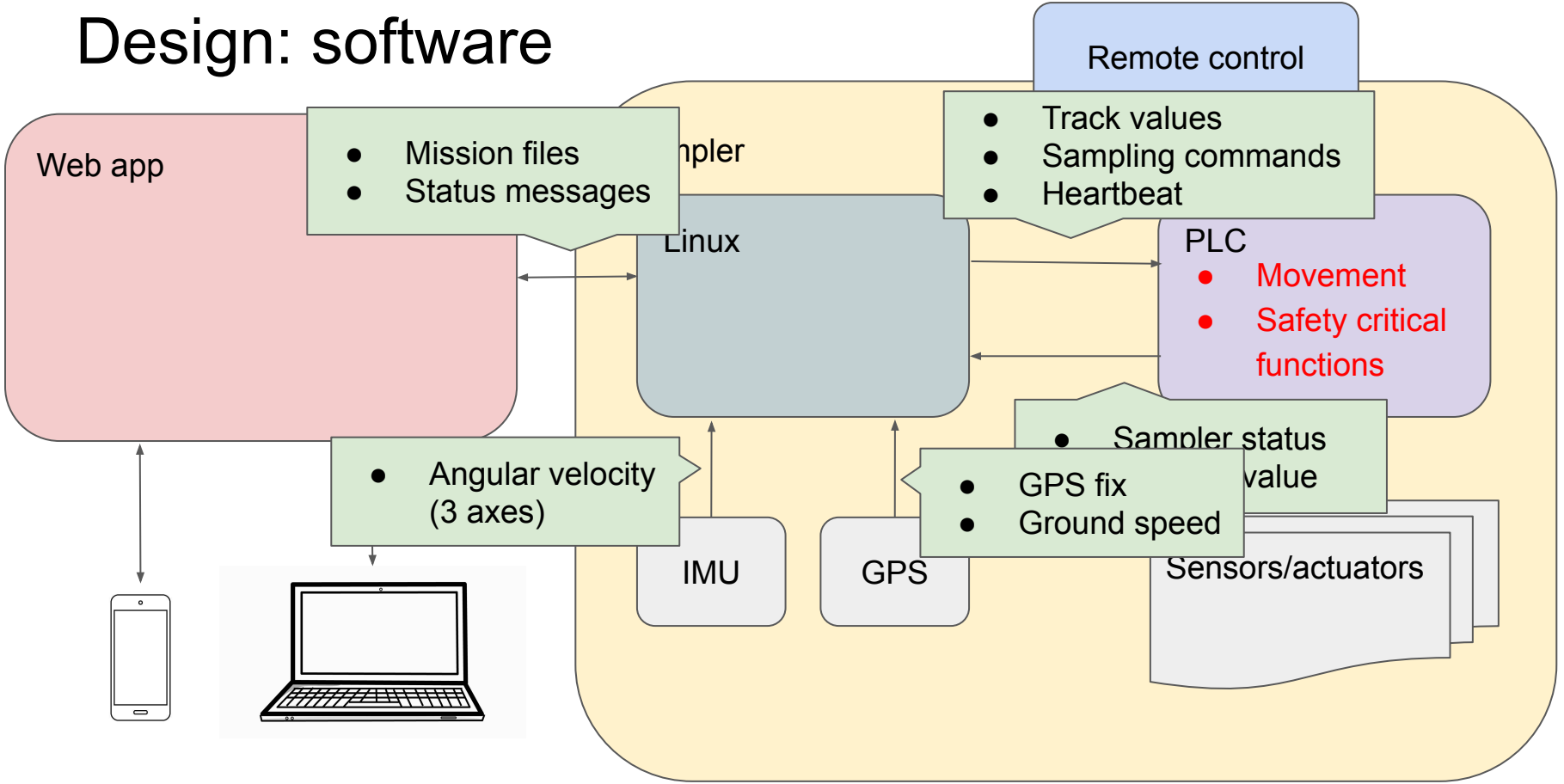
Design: software



Design: software



Design: software



Design: software

Safety mechanisms:

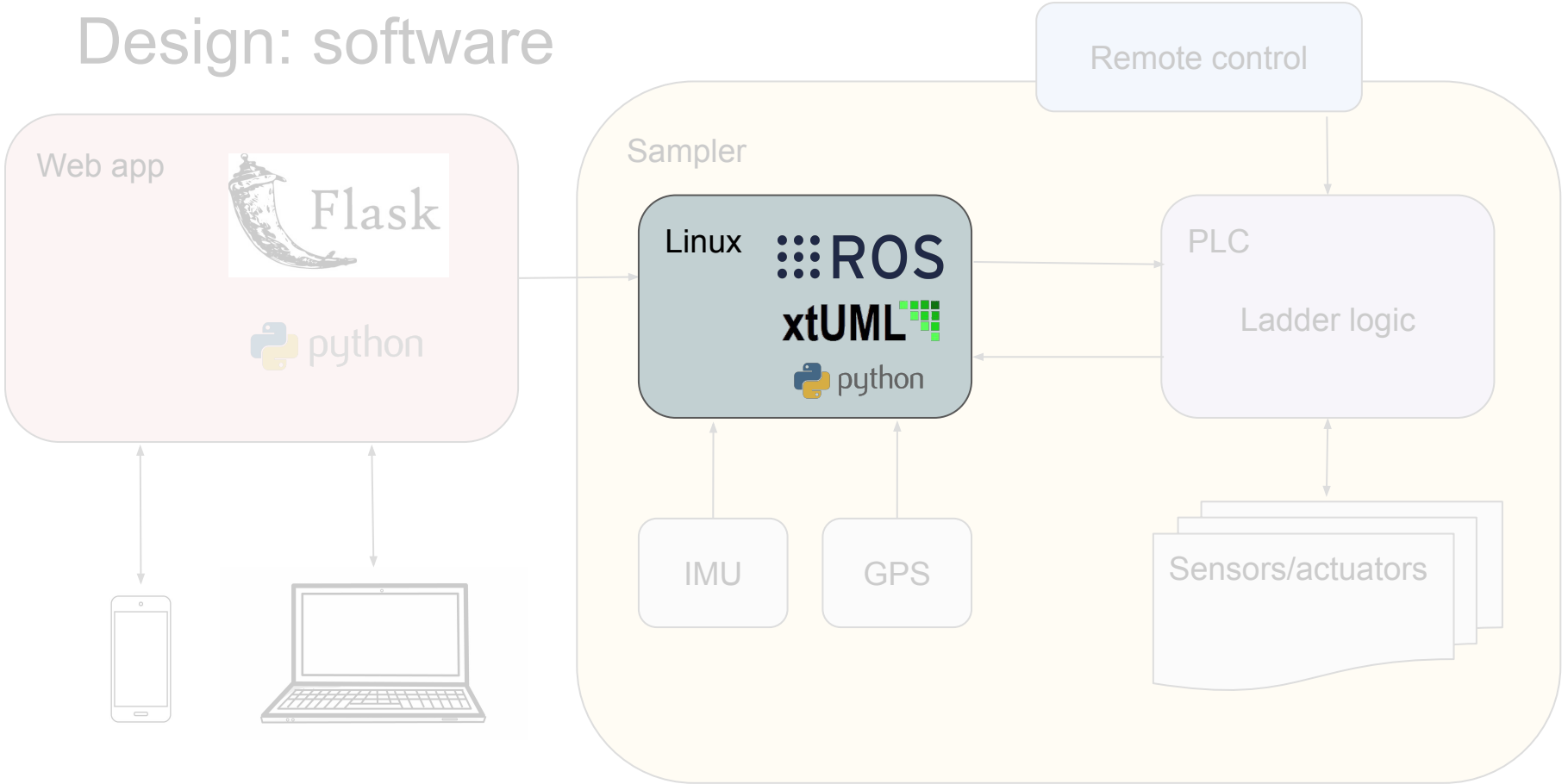
- Software defined boundaries
- Nav computer heartbeat
- Radar obstacle detection (coming this Fall)
- Remote pause, remote e-stop
- Operator worn proximity vest (coming this Fall)
- Safety line
- Physical e-stop buttons (x3)



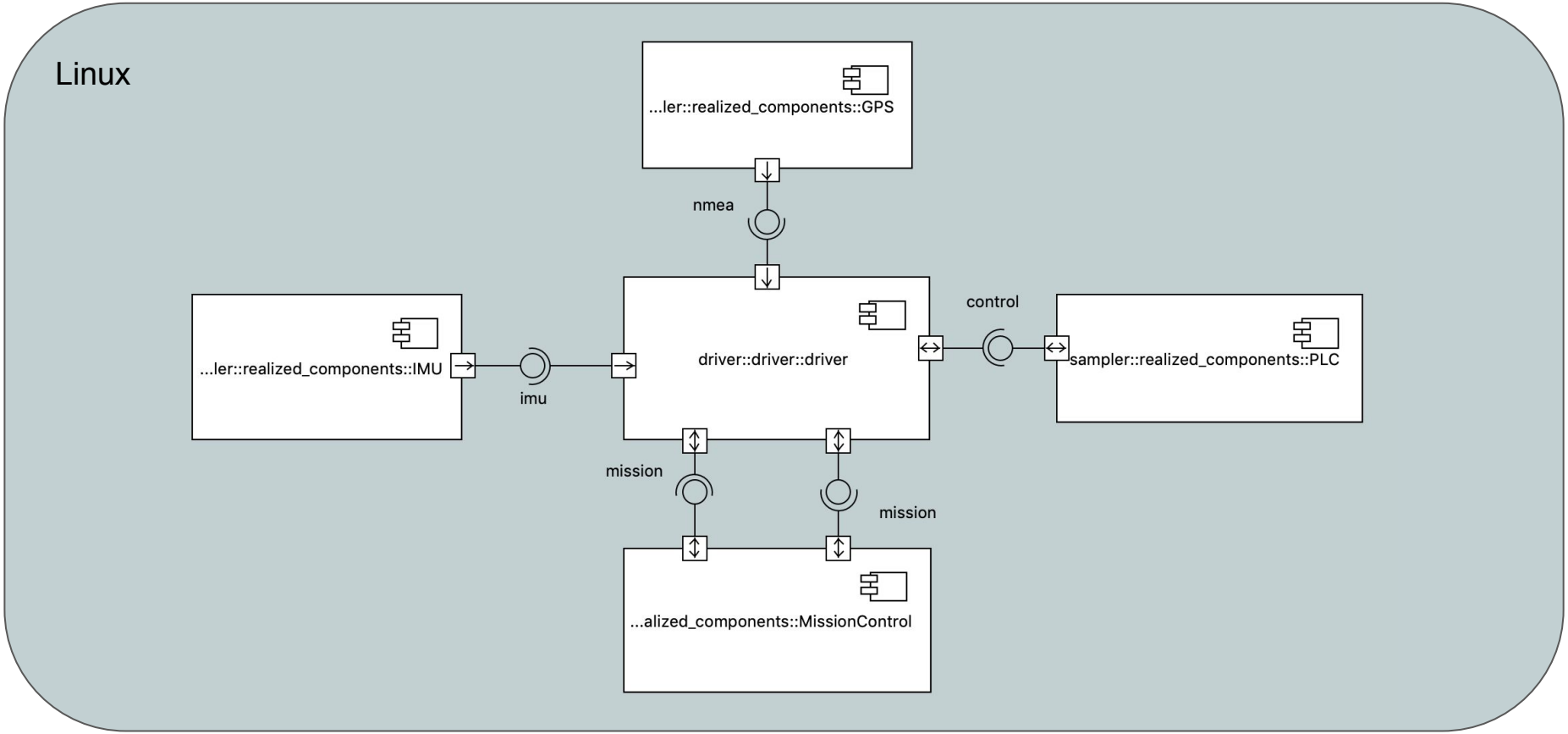
Where does 

fit in?

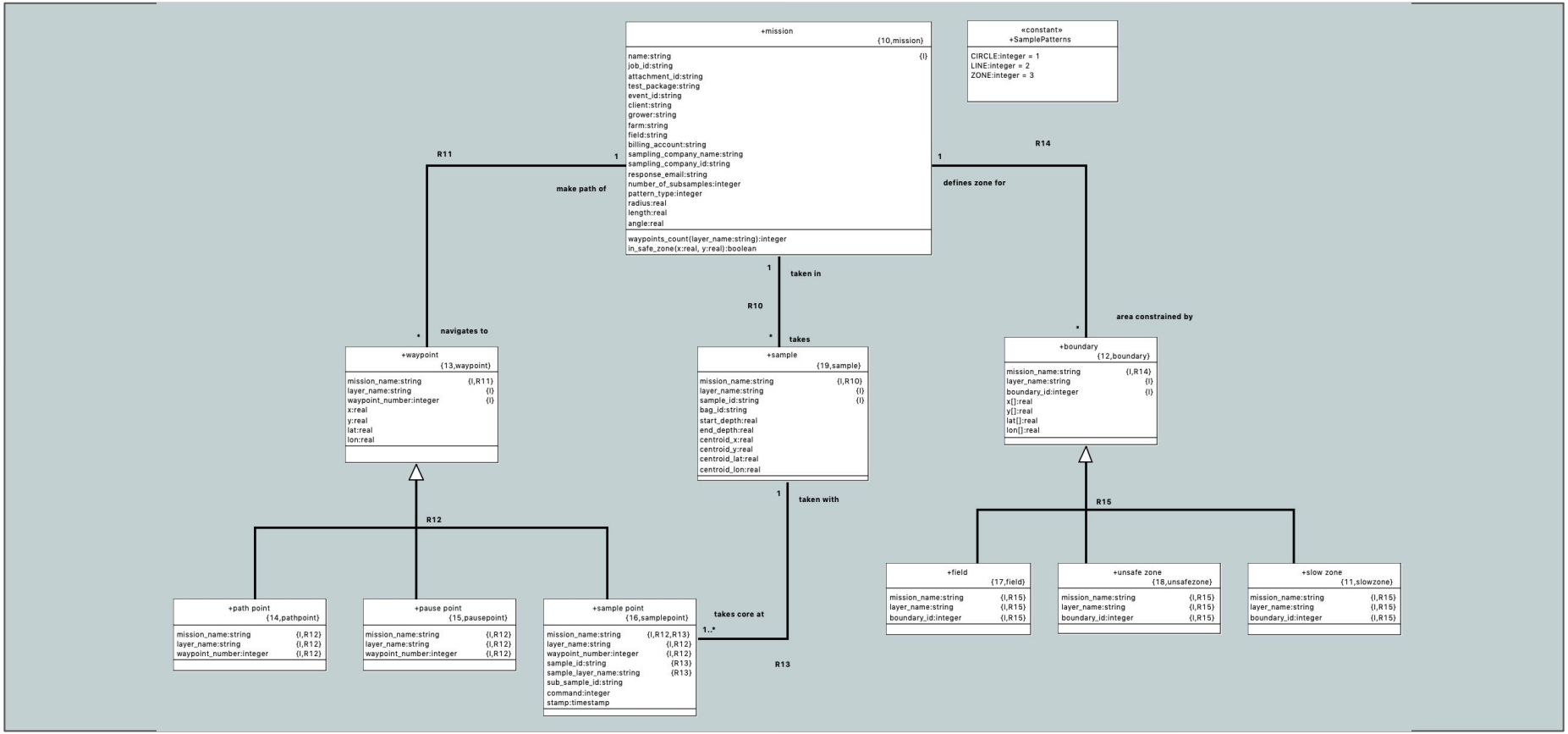
Design: software



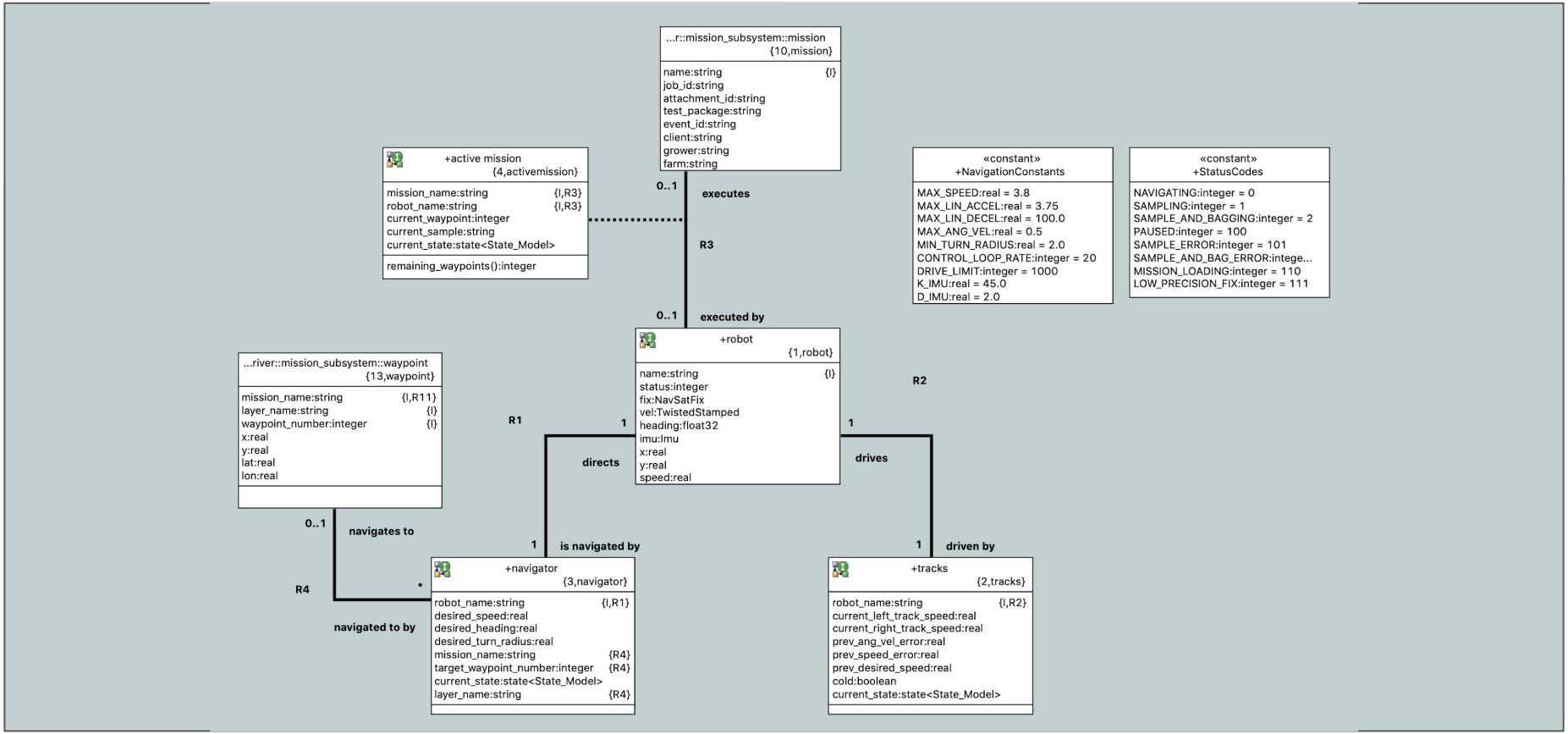
xtUML diagrams: system



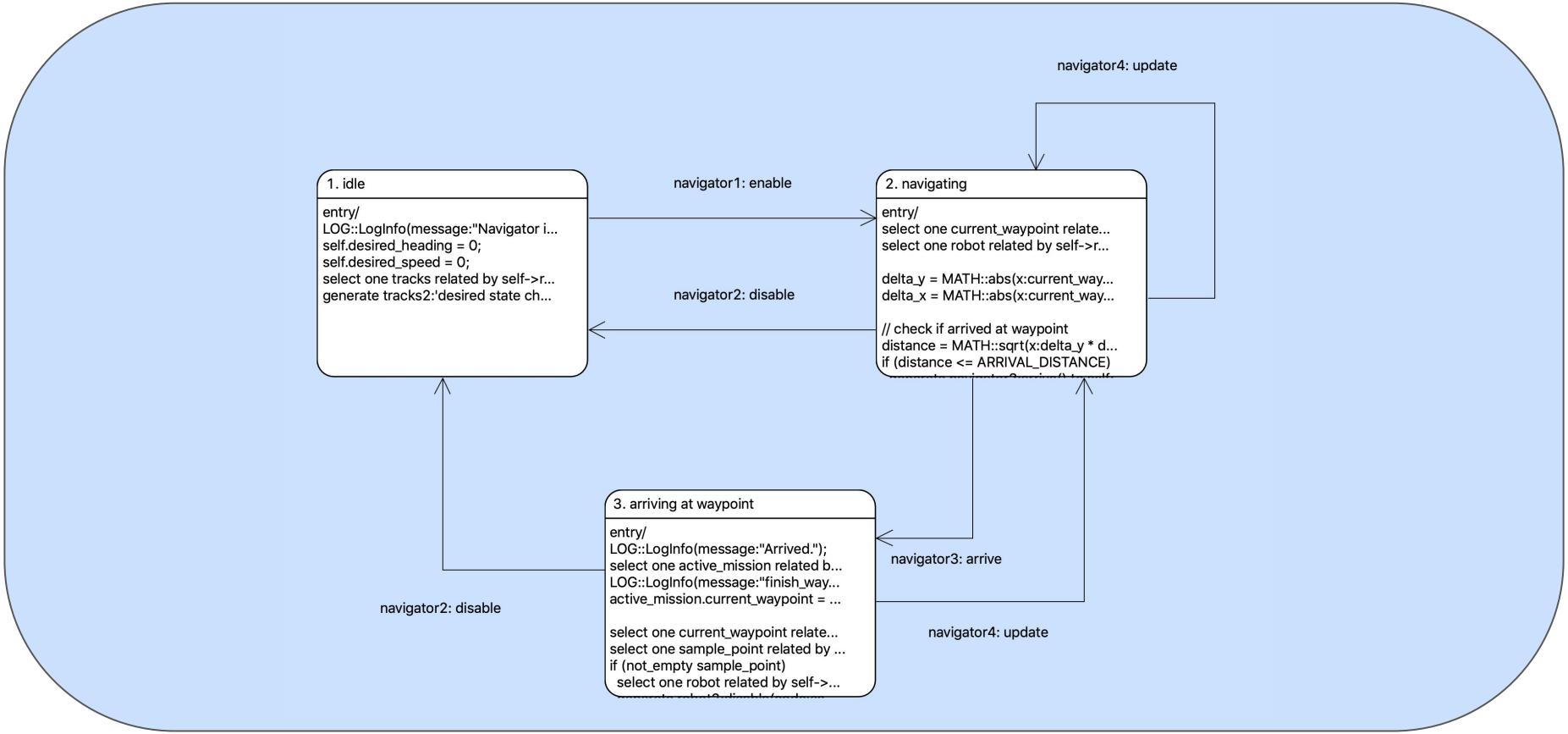
xtUML diagrams: mission subsystem



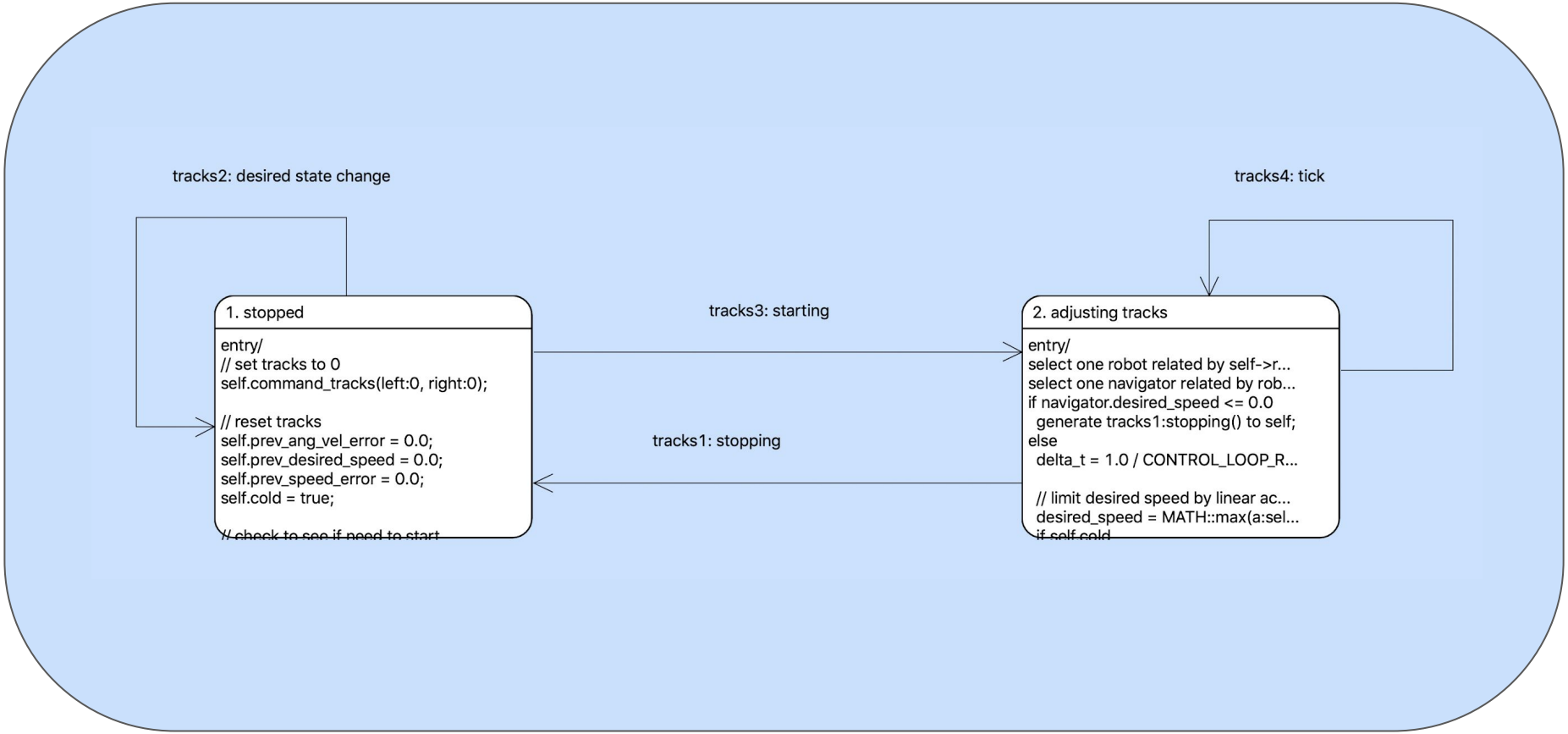
xtUML diagrams: navigation subsystem



xtUML diagrams: navigator state machine



xtUML diagrams: tracks state machine



ROS overview: what is ROS?

“ROS is an open-source, meta-operating system for your robot. It provides ... hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers...The ROS runtime "graph" is a peer-to-peer network of processes (potentially distributed across machines) that are loosely coupled using the ROS communication infrastructure. ROS implements several different styles of communication, including synchronous RPC-style communication over services, asynchronous streaming of data over topics, and storage of data on a parameter server.”



ROS overview: what is ROS?

“ROS is an open-source, meta-operating system for your robot. It provides ... hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers...The ROS runtime "graph" is a peer-to-peer network of processes (potentially distributed across machines) that are loosely coupled using the ROS communication infrastructure. ROS implements several different styles of communication, including synchronous RPC-style communication over services, asynchronous streaming of data over topics, and storage of data on a parameter server.”

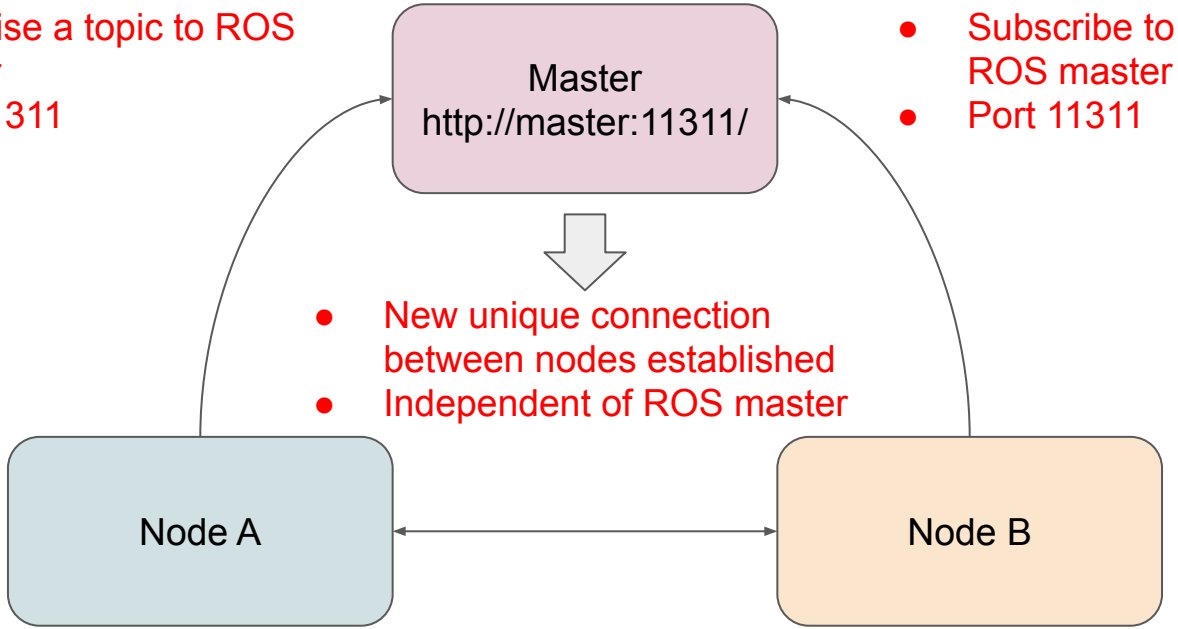


ROS overview: topics

- ROS nodes can advertise as publishing or subscribing to topics
- Each subscriber to a particular topic receives a copy of the message from each publisher
- ROS generates code from message definitions which serialize and deserialize messages

ROS overview: topics

- Advertise a topic to ROS master
- Port 11311



- Subscribe to a topic via ROS master
- Port 11311

- New unique connection between nodes established
- Independent of ROS master

- Live demo!... with simulated robot
    - Simulator inputs: track left, track right, command
    - Simulator outputs: lat, lon, heading, speed, angular velocity (z)
    - Speed and angular velocity estimated from track values
    - Location and heading accumulated over time
- Video demo

MC-ROS: model compiler for ROS (Python)

- Model based model compiler
- Built with pyxtuml and ciera
- ciera:
  - Java model compiler built to build model compilers
  - Side project to be productized… stay tuned.
- pyxtuml 2.2.0 (Sept. 2019)
  - Enhanced support for OAL parsing
  - Updated schema
  - ZIP archive loading

MC-ROS: mappings

- Node == component
- Outbound port message == publisher
- Inbound port message == subscriber
- Built for simplicity: utilized as much ROS infrastructure as practical
  - Timers handled by ROS
  - Logging handled by ROS

MC-ROS: system modeling

- “launch” files allow a group of nodes to be defined as a system
- Topic remapping allows each node to maintain its own topic namespace
  - Allows for distinct satisfactions between components even if interfaces are used multiple times in the system
  - Topic format: “<component name>/<port name>/<message name>”
  - Inbound message topics are remapped to the name of the corresponding outbound message topic across the satisfaction
- Specified in XML format

MC-ROS: system modeling

```
<node pkg="tractobot" name="ruser" type="ruser.py" output="screen"/>

<group if="$(optenv ENABLE_MODBUS true)">
  <remap from="PLC/control/command" to="driver/CTL/command"/>
  <remap from="PLC/control/left_track" to="driver/CTL/left_track"/>
  <remap from="PLC/control/right_track" to="driver/CTL/right_track"/>
  <remap from="PLC/control/waypoint_count" to="driver/CTL/waypoint_count"/>
  <node pkg="tractobot" name="plc" type="plc_main.py" respawn="true" respawn_delay="5" output="screen">
    <param name="do_not_sample" value="false"/>
    <param name="track_min_move" value="$(env TRACK_MIN_MOVE)"/>
    <param name="track_max_speed" value="$(env TRACK_MAX_SPEED)"/>
    <param name="track_tol" value="$(env TRACK_TOL)"/>
  </node>
</group>

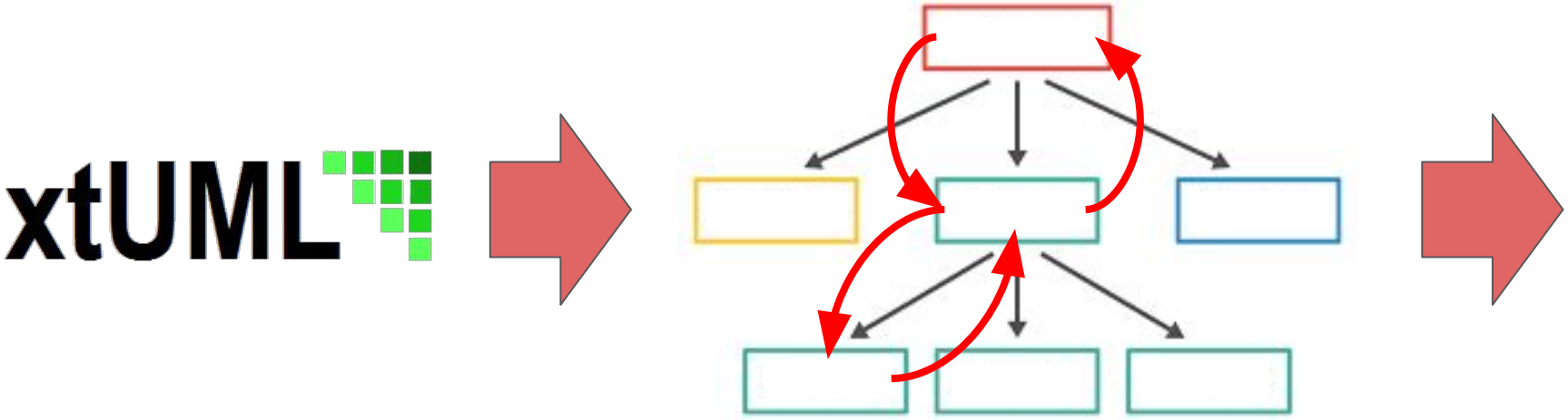
<node pkg="tractobot" name="driver" type="driver_main.py" respawn="true" respawn_delay="5" output="screen">
  <remap from="driver/CTL/status" to="PLC/control/status"/>
  <remap from="driver/GPS/fix" to="gps/fix"/>
  <remap from="driver/GPS/vel" to="gps/vel"/>
  <remap from="driver/IMU/heading" to="heading"/>
  <remap from="driver/IMU/imu" to="imu"/>
</node>

<node pkg="tractobot" name="ipcam" type="ipcam_main.py" respawn="true" respawn_delay="5" output="screen">
```

Model-based model compiler

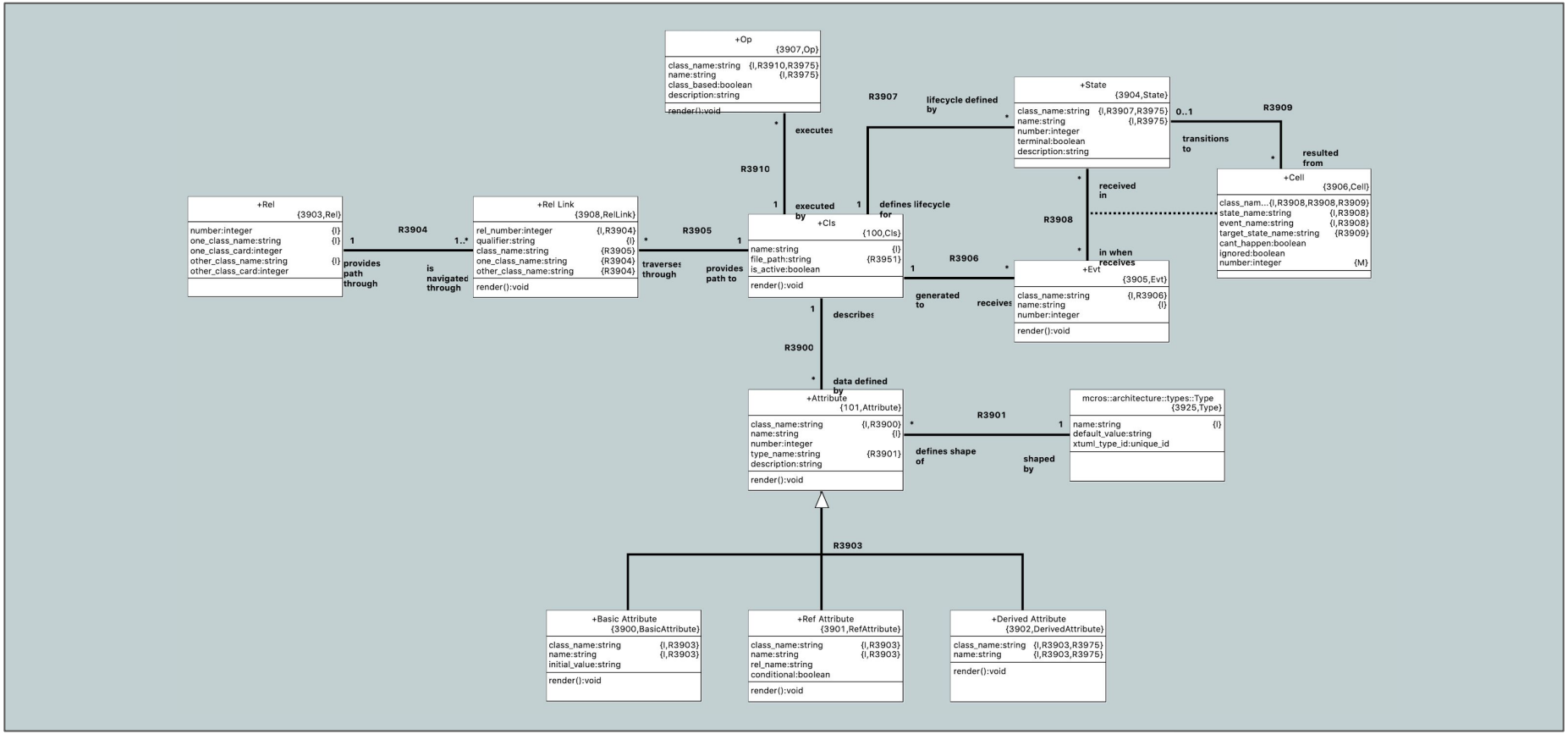
Step 1: load xtUML instance population with SQL instance loader

Step 3: Walk the architectural model generating code with templates (RSL) and “rolling up” along the way

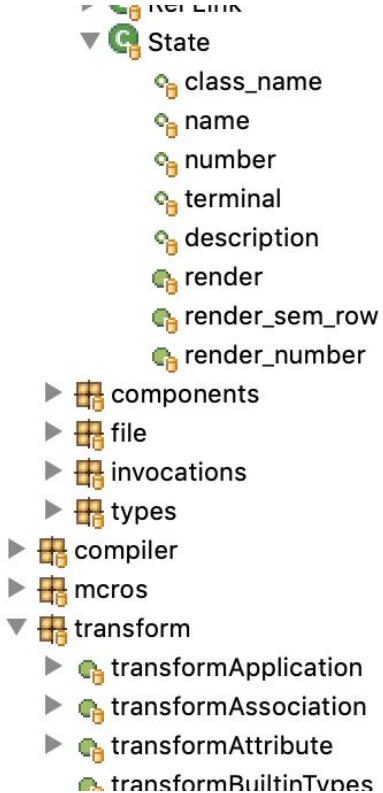


Step 2: Transform xtUML MM instances to instances of an architectural model using OAL actions

MC-ROS diagrams: classes model



MC-ROS diagrams: transforms and templates




```
 3
 4 .end if
 5 class ${self.name}(mcros.ModelClass):
 6 .if (self.is_active)
 7
 8 ${event_numbers}\
 9
10 ${state_numbers}\
11 .end if
12
13     def __init__(self):
14         super(${self.name}, self).__init__()
15 .if (self.is_active)
16         self.current_state = ${self.name}.$u{initial_state}
17 .end if
18 ${attribute_initializers}\
19 ${relationships}\
20 .if ("" != referential_attributes)
21
22 ${referential_attributes}\
23 .end if
24 .if ("" != derived_attributes)
25
26 ${derived_attributes}\
```

MC-ROS: Does it generate code?

- …. Kinda….
- Class definitions, relationships, state machines ~90% done
- Component definitions, ports ~50% done
- Types, enumerations, constants, structures ~20% done
- OAL translation 0% done
- System modeling 0% done
- Runtime support 100% functional!
- Currently use code generation to put a “wireframe” in place, write action language and missing bits by hand

The future of xtUML at Rogo?

- Clear next steps:
  - Full MC-ROS code generation
  - MC-ROS test generation
  - Runtime execution of models with BridgePoint Verifier
- Off the wall ideas:
  - State machine generation for PLC ladder logic
  - Python/ROS tool for monitoring instance populations of running models using ROS topics (much like Inspector for MASL)
  - xtUML web app generation tool

Summary

- The xtUML methodology is being used and successfully adding value in a real world application at Rogo
  - Vastly increased ability to understand and communicate about the application at Rogo
  - Increased traceability and debug
  - Modular forward path
- The xtUML technology is moving forward at Rogo
  - MC-ROS, ciera
  - Bug identification and resolution (I know a guy…)



Thank you