ROS NODE / PACKAGE GENERATOR

MAKE IT SIMPLE  (AND WELL...)  

Anthony Remazeilles, Jon Azpiazu, Tecnalia Research & Innovation

ROS-Industrial EU Spring’18 Workshop, May 29 2018

Rosin: rosin-project.eu
On ROS node / package generator

Typical situation:

Our application uses an Inertial Measurement Unit. Now the sensor output is a bit noisy, we would like to apply some filtering on the data before using it in the rest of the application.

Can you do it (asap)?

Yes I can!

or…

This is not as easy, let me think a bit…
On ROS node / package generator

- catkin_create_pkg  imu_filtering  sensor_msgs  roscpp
- #... and then ???
On ROS node / package generator

- catkin_create_pkg  imu_filtering  sensor_msgs  roscpp
- #... and then ???

- Nodes usually created from scratch
  - Loss of time

- Implementation strongly relies on Developer’s expertise (& time)

- Interface hidden in the code itself
  - Separation of concerns (communication vs core intelligence)

- Life-cycle hidden in the code itself
  - How the node starts / stops, update frequency?

- Reuse, update, collaboration in team not always easy
ROS package / node generator needs identified

- Node creation based on the interface

- Automatic code generation, including node life-cycle patterns

- Clear separation Interface / implementation

- Plurality of life-cycle patterns

- Keep it simple
Implemented work

- cd [my_ros_workspace]/src
- gedit imu_filtering.ros_package
- rosrin package_generator generate_package imu_filtering.ros_package cpp_node_update
- ....
- Package generated!
## Implemented work

- `cd [ros_workspace]/src`
- `gedit imu_filtering.ros_package`
- `rosrun package_generator generate_package imu_filtering.ros_package` _cpp_node_update_

Package generated!

```xml
<?xml version="1.0" encoding="utf-8"?>
<package name="imu_filtering"
  author="Anthony Remazeilles"
  author_email="anthony.remazeilles@tecnalia.com"
  description="IMU data filtering"
  license="GPL">

  <node frequency="100.0" name="imu_filter">
    <publisher name="imu_filtered"
               type="sensor_msgs::Imu"
               description="IMU data filtered"/>
    <subscriber name="imu_in"
               type="sensor_msgs::Imu"
               description="IMU data to filter"/>
  </node>

  <depend>sensor_msgs</depend>
  <depend>roscpp</depend>
</package>
```

- **Package description**
- **Nodes definition**
- **Interface description**
- **Dependency definition**
**Implemented work**

- cd [ros_workspace]/src
- gedit imu_filtering.ros_package
- rosrun package_generator generate_package imu_filtering.ros_package

... Package generated!

- **Available ones:**
  - Python / C++
  - Separation of concerns

- **Definition of new templates possible**

```
    cd [ros_workspace]/src
    gedit imu_filtering.ros_package
    rosrun package_generator generate_package imu_filtering.ros_package

    ....
    Package generated!
```

```
    >>> tree cpp_node_update/
    cpp_node_update/    (to be filled)
    ├── config
    │   └── dictionary.yaml
    ├── template
    │   ├── cfg
    │   │   └── node.cfg
    │   └── model
    │       ├── CMakeLists.txt
    │       └── package.xml
    └── README.md

    >>> tree src
    src/    (don’t touch)
    ├── common
    │   └── src
    │       └── node_common.cpp
    └── ros
        └── src
            └── node_ros.cpp
```
Developer contribution

Snapshot of the `imu_filter_common.cpp` (where the Developer inserts the core intelligence of the node)

FYI

Expected contribution within tagged areas
Update procedure

The filter is great, but it could be good to be able to adjust the filter online.

Can you do it (asap)?
Update procedure

1. Update the xml package description

```xml
<?xml version="1.0" encoding="utf-8"?>
<package name="imu_filtering"
  author="Anthony Remazeilles"
  author_email="anthony.remazeilles@tecnalia.com"
  description="IMU data filtering"
  license="GPL">

  <node frequency="100.0" name="imu_filter">
    <publisher name="imu_filtered"
      type="sensor_msgs::Imu"
      description="IMU data filtered"
    />

    <subscriber name="imu_in"
      type="sensor_msgs::Imu"
      description="IMU data to filter"
    />

    <dynParameter name="filter_order"
      description="order of the filter to be applied"
      type="int"
      value="1"/>
  </node>

</package>
```

2. Recall the package generator

```bash
rosrun package_generator generate_package imu_filtering.ros_package cpp_node_update
```
Update procedure

3. Update node intelligence

FYI

4. Recompile

All code within tagged areas is maintained on update.
Node documentation

Readme file is automatically filled (thanks to the template) (before /after dynamic parameter insertion, as generated by GitLab)
ROS Package Generator

- Model inspired by BRIDE from BRICS
- Generator fully implemented in python

- All standard ROS communication interface available
  - Subscriber, publisher, services, actions, tf listener & broadcaster, parameters
- Able to handle interface update while maintaining Developer contribution

- First templates available for python & C++
  - Handle the whole package creation, communication definition and management
  - Developer focuses on the real intelligence of the node
  - Possibility to provide its own template

- Currently focused on node packages
  - Will to extend generation to other ROS components
ROS package generator

- Code to be publically released soon (end of June?)
  - Willing to get feedback on code generator, templates proposed

- Check rosin news

- Can’t wait? → Contact me  anthony.remazeilles@tecnalia.com
Our system architecture

- `package_generator.py`
- `package_xml_parser.py`
- `code_generator.py`
- `pkg_name.ros_package (xml)`
- `template_package directory`
- `{nodeName}_ros.cpp`

Legend:
- Orange: Input file (either a model / template, or just a package instance)
- Green: Python scripts
- Blue: Generated files