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-project.eu

tecnalia Inspiring Business

This project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 732287

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

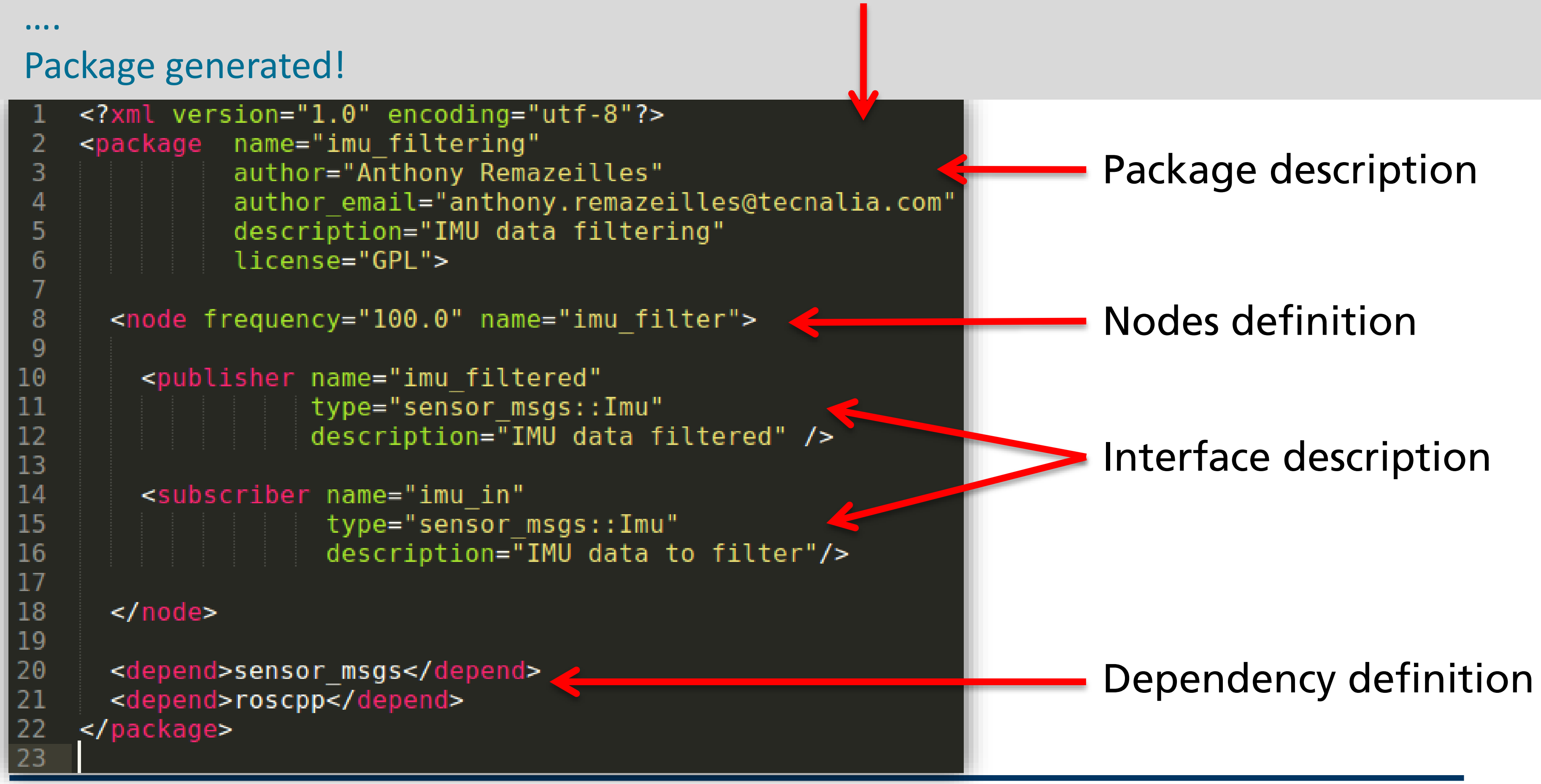
```
➢ cd [my_ros_workspace]/src
➢ gedit imu_filtering.ros_package
➢ rosrun  package_generator  generate_package  imu_filtering.ros_package   cpp_node_update
    ….
    Package generated!
```

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>
```

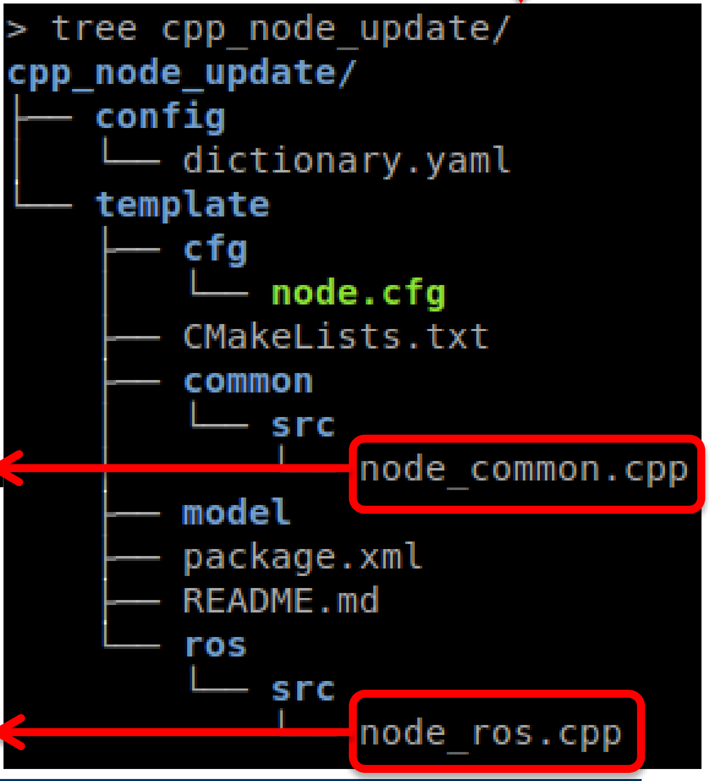
Implemented work

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

  ….

  Package generated!

- Available ones:
  - Python / C++
  - Separation of concerns
- Definition of new templates possible

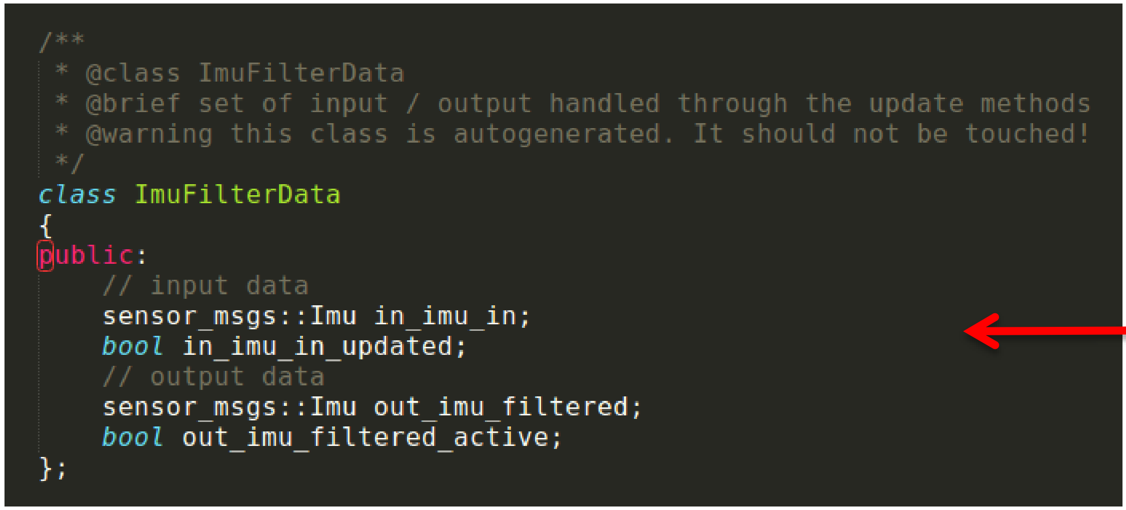


```
> tree cpp_node_update/
cpp_node_update/
├── config
│   └── dictionary.yaml
└── template
    ├── cfg
    │   └── node.cfg
    ├── CMakeLists.txt
    ├── common
    │   └── src
    │       └ node_common.cpp
    ├── model
    ├── package.xml
    ├── README.md
    └── ros
        └── src
            └ node_ros.cpp
```

Implementation
(to be filled)

Interface
(don't touch)

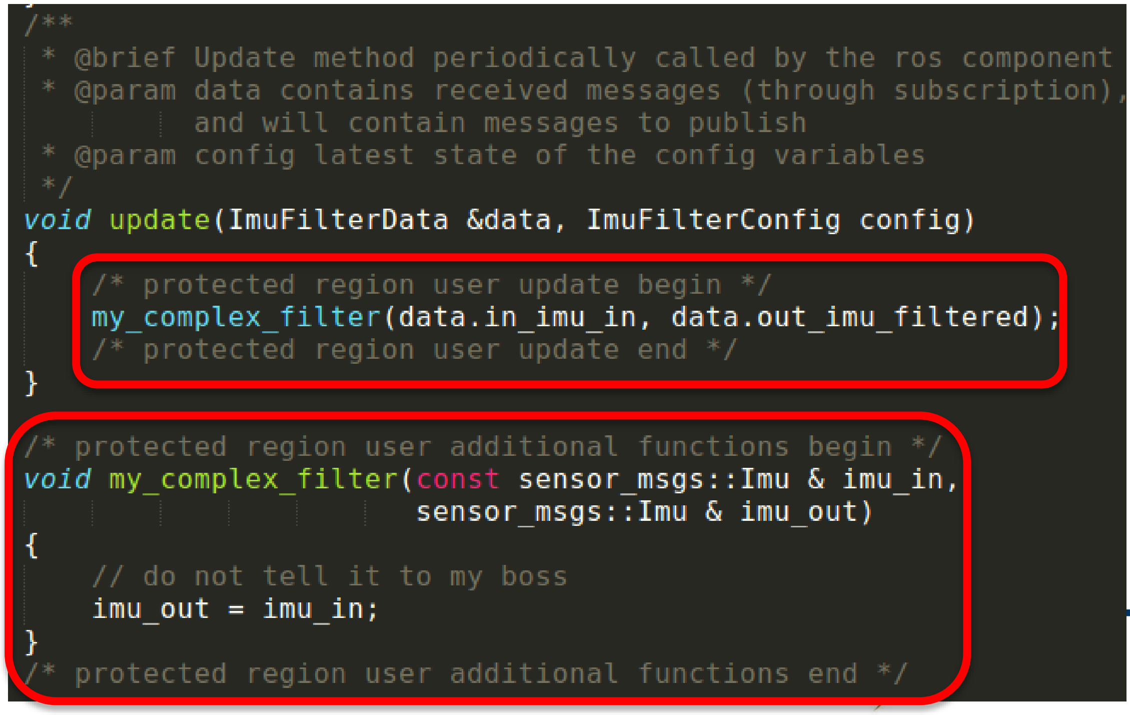
Developer contribution



```
/**
 * @class ImuFilterData
 * @brief set of input / output handled through the update methods
 * @warning this class is autogenerated. It should not be touched!
 */
class ImuFilterData
{
public:
    // input data
    sensor_msgs::Imu in_imu_in;
    bool in_imu_in_updated;
    // output data
    sensor_msgs::Imu out_imu_filtered;
    bool out_imu_filtered_active;
};
```

FYI

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



```
/**
 * @brief Update method periodically called by the ros component
 * @param data contains received messages (through subscription),
            and will contain messages to publish
 * @param config latest state of the config variables
 */
void update(ImuFilterData &data, ImuFilterConfig config)
{
    /* protected region user update begin */
    my_complex_filter(data.in_imu_in, data.out_imu_filtered);
    /* protected region user update end */
}

/* protected region user additional functions begin */
void my_complex_filter(const sensor_msgs::Imu & imu_in,
                       sensor_msgs::Imu & imu_out)
{
    // do not tell it to my boss
    imu_out = imu_in;
}
/* protected region user additional functions end */
```

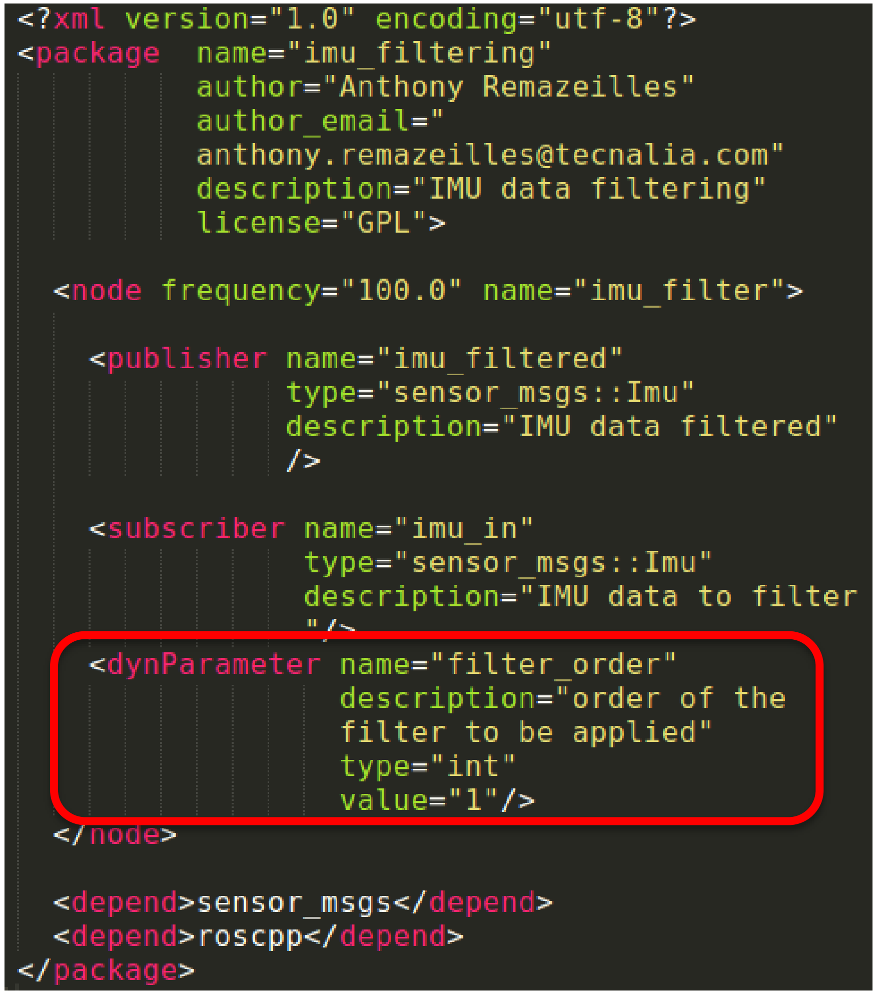
Expected contribution within tagged areas

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



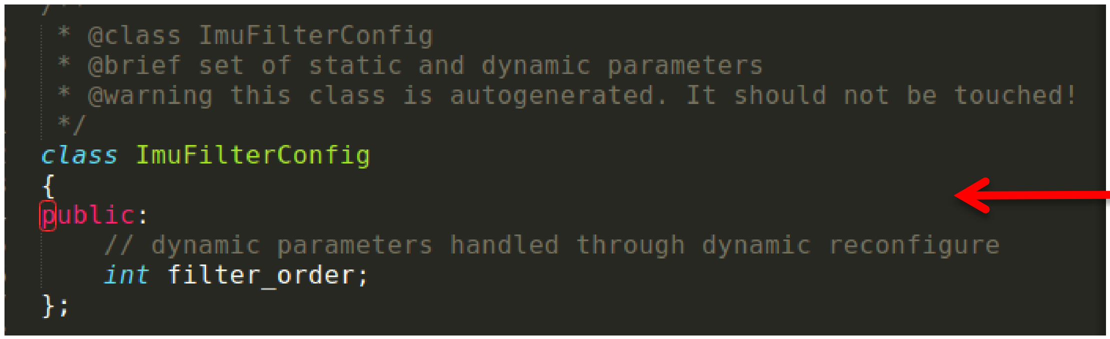
```
<?xml version="1.0" encoding="utf-8"?>
<package  name="imu_filtering"
          author="Anthony Remazeilles"
          author_email="
          anthony.remazeilles@tecnalia.com"
          description="IMU data filtering"
          license="GPL">

  <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>

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

1 Update the xml package descriptio

2 Recall the package generator

➢ rosrun package_generator generate_package imu_filtering.ros_package cpp_node_update

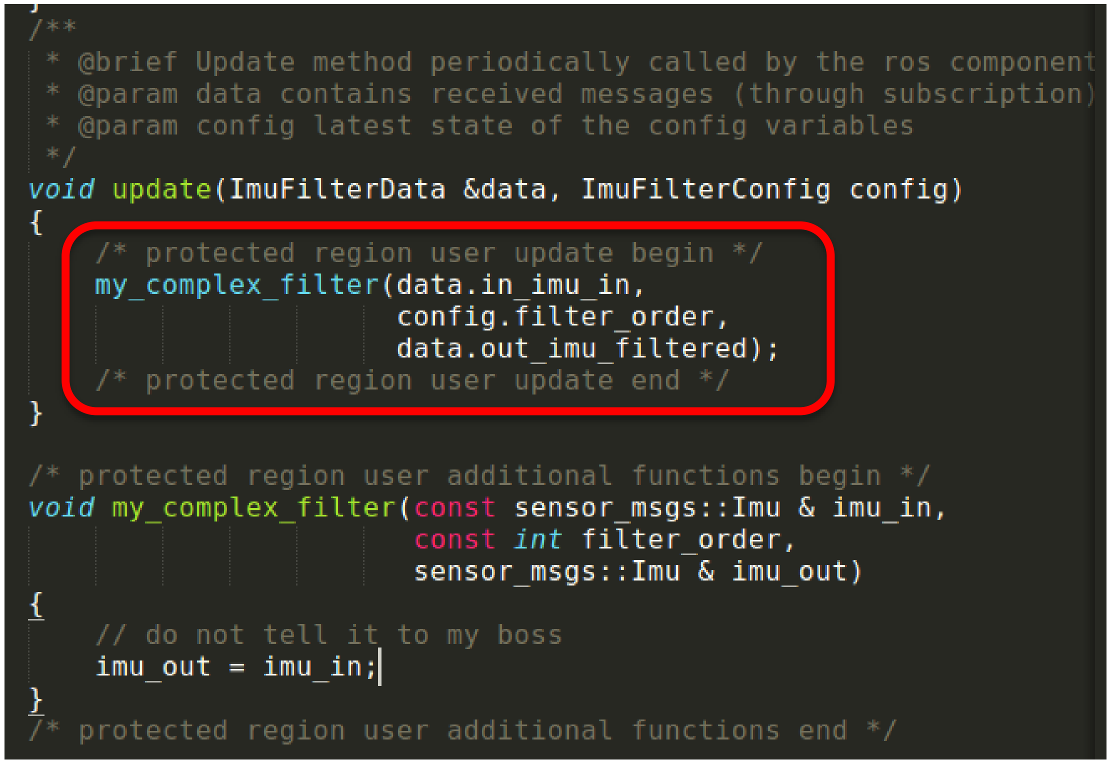
Update procedure



```
 * @class ImuFilterConfig
 * @brief set of static and dynamic parameters
 * @warning this class is autogenerated. It should not be touched!
 */
class ImuFilterConfig
{
public:
   // dynamic parameters handled through dynamic reconfigure
   int filter_order;
};
```

3 Update node intelligenc

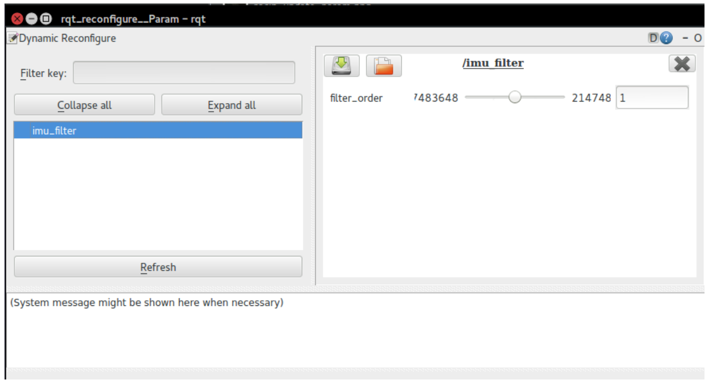
FYI



```
/**
 * @brief Update method periodically called by the ros component
 * @param data contains received messages (through subscription)
 * @param config latest state of the config variables
 */
void update(ImuFilterData &data, ImuFilterConfig config)
{
   /* protected region user update begin */
   my_complex_filter(data.in_imu_in,
                     config.filter_order,
                     data.out_imu_filtered);
   /* protected region user update end */
}

/* protected region user additional functions begin */
void my_complex_filter(const sensor_msgs::Imu & imu_in,
                       const int filter_order,
                       sensor_msgs::Imu & imu_out)
{
   // do not tell it to my boss
   imu_out = imu_in;
}
/* protected region user additional functions end */
```

All code within tagged areas
is maintained on update

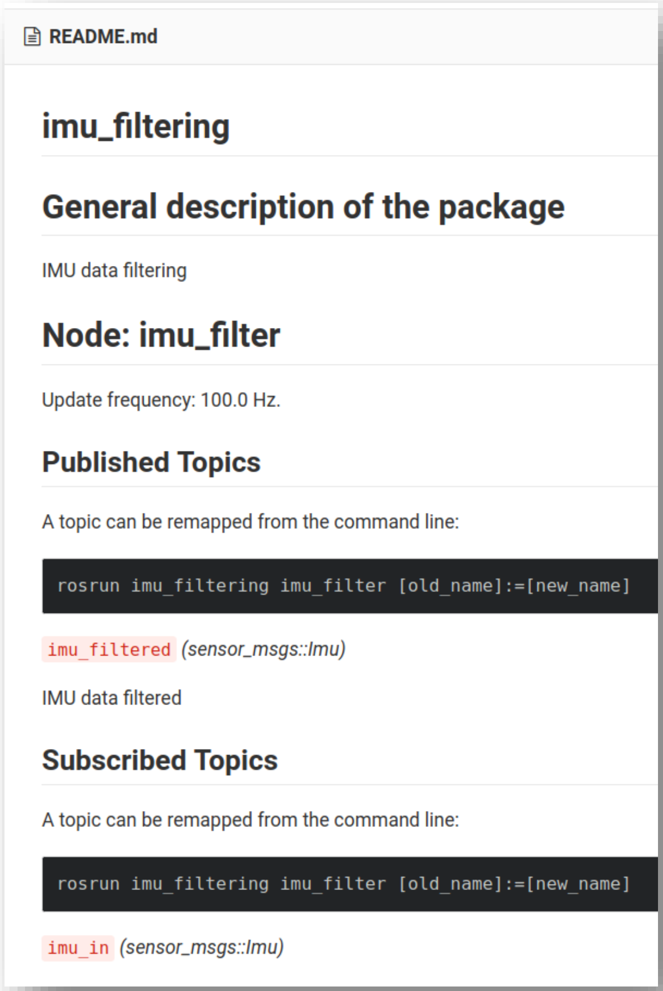
4 Recompile



filter_order ⁷483648 214748 1

(System message might be shown here when necessary)

Node documentation



README.md

imu_filtering

General description of the package

IMU data filtering

Node: imu_filter

Update frequency: 100.0 Hz.

Published Topics

A topic can be remapped from the command line:

```
rosrun imu_filtering imu_filter [old_name]:=[new_name]
```

`imu_filtered` *(sensor_msgs::Imu)*

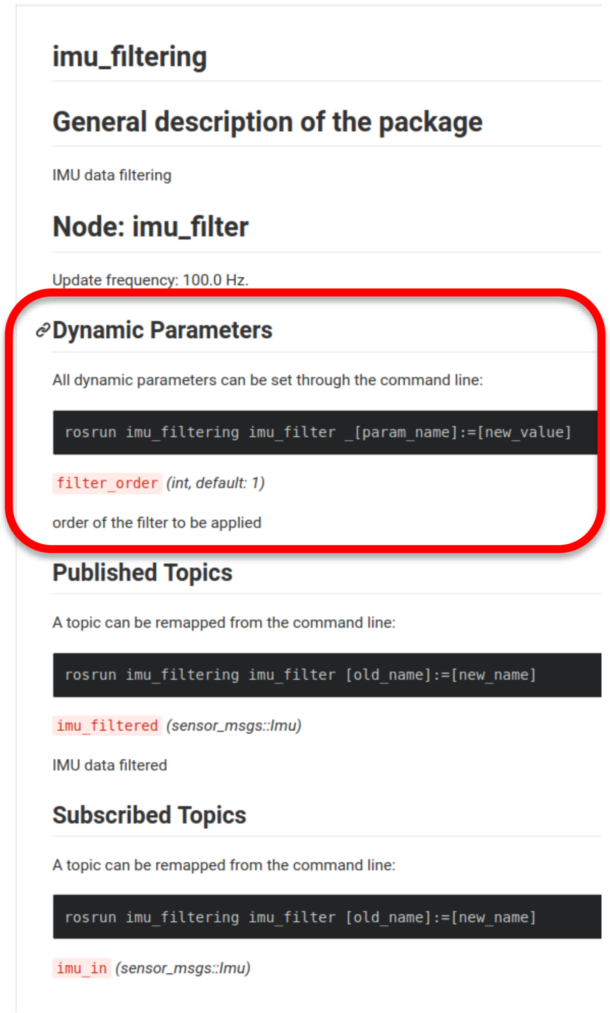
IMU data filtered

Subscribed Topics

A topic can be remapped from the command line:

```
rosrun imu_filtering imu_filter [old_name]:=[new_name]
```

`imu_in` *(sensor_msgs::Imu)*



imu_filtering

General description of the package

IMU data filtering

Node: imu_filter

Update frequency: 100.0 Hz.

Dynamic Parameters

All dynamic parameters can be set through the command line:

```
rosrun imu_filtering imu_filter _[param_name]:=[new_value]
```

`filter_order` *(int, default: 1)*

order of the filter to be applied

Published Topics

A topic can be remapped from the command line:

```
rosrun imu_filtering imu_filter [old_name]:=[new_name]
```

`imu_filtered` *(sensor_msgs::Imu)*

IMU data filtered

Subscribed Topics

A topic can be remapped from the command line:

```
rosrun imu_filtering imu_filter [old_name]:=[new_name]
```

`imu_in` *(sensor_msgs::Imu)*

Readme file is automatically filled (thanks to the template)
(before /after dynamic parameter insertion, as generated by Gi[illegible]o
)

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

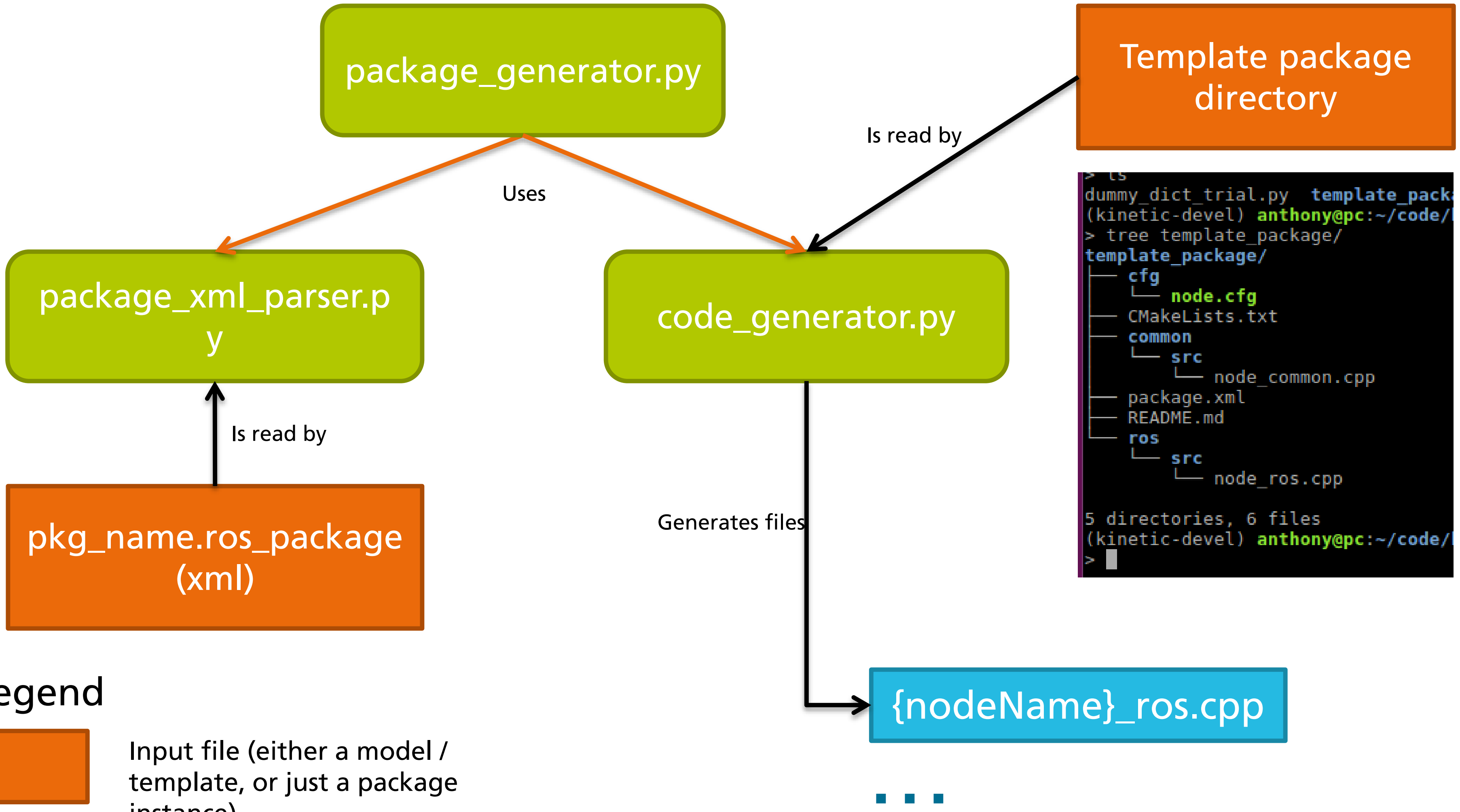
ROS package generator

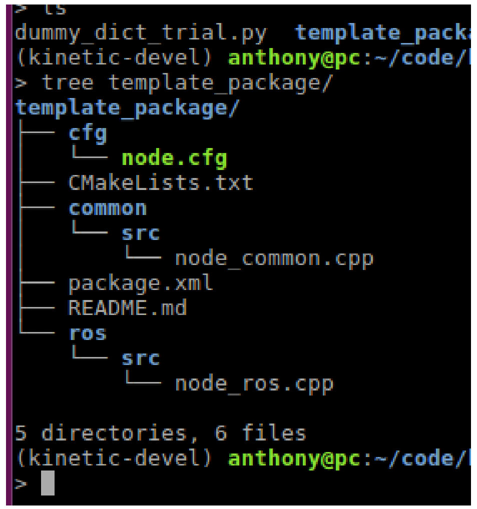
- Code to be publically released soon (end of June?)
    - Willing to get feedback on code generator, templates proposed

- → Check rosin news

- Can't wait ? → Contact me anthony.remazeilles@tecnalia.com

Our system architecture





```
> ls
dummy_dict_trial.py  template_packa
(kinetic-devel) anthony@pc:~/code/
> tree template_package/
template_package/
├── cfg
│   └── node.cfg
├── CMakeLists.txt
├── common
│   └── src
│       └── node_common.cpp
├── package.xml
├── README.md
└── ros
    └── src
        └── node_ros.cpp

5 directories, 6 files
(kinetic-devel) anthony@pc:~/code/
> 
```

Legend

🟧 Input file (either a model / template, or just a package instance)

🟩 Python scripts

🟦 Generated files