ROS Node Configuration #3260

Locked **kaspermeck-arm** started this conversation in **Design**

kaspermeck-arm on Feb 8, 2023 Collaborator

The Open AD Kit WG is working on improving the process of improving how to configure the ROS node parameters, see Issue 2636 for details and discussion.

Decisions which have been made:

- single source parameter file
  - which will be used for: ROS node configuration, web documentation and ROS launch
- path: autoware.universe/.../config/*.param.yaml
- *default_value* not to be used in *declare_parameter()*
- each parameter has following attributes:
  - name
  - type
  - default value
  - description
  - bounds

(Pending) Consensus on solution:

1. Adopt PickNikRobotics generate parameter library

- Pros:
  - sophisticated solution
  - verifies that the value input to ROS node is within the bounds
  - removes the need for the *declare_parameter(...)* function in the ROS nodes
  - can be no discrepancy between *declare_parameter(...)* function and parameter file
- Cons:
  - requires many changes in the Autoware source code
  - creates a dependencies on a 3rd party repo

2. Create our parameter yaml structure for parameter files

- Pros:
  - we own the parameter file structure
  - simple to implement
- Cons:
  - no bounds checking for input value

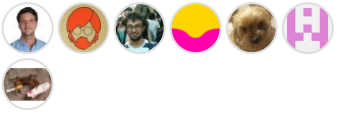
Category

Design

Labels

None yet

7 participants



This is not an extensive list of pros/cons, just what came to mind. Please give your feedback on which approach is preferred!

Thanks.

↑ 1 👍 3

9 comments · 24 replies

Oldest | Newest | Top

kaspermeck-arm on Feb 8, 2023 Collaborator Author

Also, see slides 9 and 10 on Open AD Kit - DevOps Dojos.

↑ 1 2 replies

angry-crab on Feb 10, 2023 Maintainer

Hi, I cannot access the slides.

kaspermeck-arm on Feb 10, 2023 Collaborator Author

Please request access and it'll be granted!

angry-crab on Feb 10, 2023 Maintainer edited

~~Personally, I think 2 is better. Maybe it could be done with a simple parser and reflection.~~

Please correct me if I'm wrong. For option 2, are you suggesting that we should implement a parameter parser and wrap the `declare_parameter(...)` calls?

btw, 1 still uses `declare_parameter(...)` anyway. I think the only difference is the functions calls are automatically generated.

↑ 1 2 replies

kaspermeck-arm on Feb 10, 2023 Collaborator Author

For Option 2, the idea is to remove *default_value* from *declare_parameter(...)* to enforce that the parameter file contains all parameters which the node expects. No wrapper should be needed.

I haven't yet tried Option 1 myself, I'll update the initial comment with corrections if I got anything wrong.

kenji-miyake on Feb 10, 2023

Oh, if you meant so, I misunderstood one thing. I thought as if we're going to define the parameter structures in some original format.
In that case, it's a good start as the first step.
And we can try out PickNik's library in parallel to find better solutions.

doganulus on Feb 10, 2023 Collaborator

Option 1 is brittle. My experience is that the code generation approach (as in Option 1) is harder to maintain and test for all parties involved. Also complicates the build process even when `cmake` scripts are provided. I would be more enthusiastic to use it if implemented as a plain `C++` library.

↑ 2 1 reply

xmfcx on Feb 11, 2023 Maintainer

Option 1 eliminates a lot of manual labor of:

- declaring parameters
- defining and enforcing bounds
- automatically generating set-callbacks for parameters
- generating yaml config files from config, this one is possible but it's in the issue stage

To achieve all these, one would need to either

- make fundamental changes to ROS2's internal parameter handling infrastructure (too costly)
- use c++ macros in very challenging ways (uglier than current solution, restrictive(outputing a .yaml file would make it awkward)) to get the same result as this library.

The fact that it can eliminate a lot of boilerplate code, I think it is a quite valuable tool.

And it can have many more features like these listed in its issues:

- option to print params
- async set-callbacks for parameters

I also understands the concerns about the code maintainability. But most of the hard parts are already done and updating it shouldn't be that hard at this stage, just editing and playing around with these jinja_templates should be enough to maintain some ROS 2 version cycles.

And for what it brings, the organization that is actively maintaining it, its readily availability, to me, the pros outweigh the cons. There will always be trade-offs, as long as we weigh our options carefully, we should be on the right path.

For this, @kasperornmeck will create a small demo using this library for a simple package (stop_filter) in the Autoware Universe and see how it performs.

And of course we can coordinate and help PickNikRobotics team with the features we also want there to be too.

👍 1

kenji-miyake on Feb 10, 2023 edited

I think option 1 is ideal as a long-term approach because option 2 sounds like reinventing the wheel.
(Edited: There was a misunderstanding about option 2. It was not as complicated as I thought, so it's not reinventing the wheel but a minimal solution. See https://github.com/orgs/autowarefoundation/discussions/3260#discussioncomment-4941063.)

However, there are several caveats:

- If we adopt PickNikRobotics's library, we should contribute to it and push the library to become the de facto standard of ROS 2.
- There is no need to force all nodes to use the library. We can still allow raw `declare_parameter` styles as well, especially for Autoware Universe (not Core).

↑ 1 👍 1 11 replies

Show 6 previous replies

doganulus on Feb 12, 2023 Collaborator edited

I will quote from the recent interview with the CEO of Intrinsic (Tan White) and the ex-CEO of the OpenRobotics (Brian Gerkey):

> ... what Brian was finding at Open Robotics is that he was starting to get that pull to build to that level, and then ended up building loads of customized things on top of ROS when really what was needed was a more industrialized platform. But he didn't want to force the ROS community into that alone.

... ended up building loads of customized things on top of ROS
... loads of customized things on top of ROS
... loads of customized things

This is the problem.

And PickNik's solution is another customized thing, another layer of complications. Really you gonna lose more developer time when adopting such custom tooling. Already lost many hours on the Autoware project because of such dependency problems. This must be one of the main concerns of the software architecture team.

See the PickNik's repo and what the issues are. Python dependency, CMake dependency, soon Jinja dependency, more dependency to 'clang-tidy' Cmake macros, or perhaps also it will work with any code analyzers because code is generated on the fly. Discussions to add generated code or not. Again losing precious developer time. Please show me how the tool is tested. How error is handled? Code generation is a super hard problem once you want to make it right, do you think Picknik is a compiler maker?

So what is the solution? The solution is to do proper C++ development and libraries as everyone does. Custom tooling is detrimental and unstable, especially if you are talking about developer resources. Believe me that even doing nothing is better than adopting fragile tooling.

Please don't be more royalist than the Queen.

xmfcx on Feb 12, 2023 Maintainer

> ... loads of customized things

Software world is built upon dependencies, like anything else in life. Reducing overall coupling and connascence are very important while developing good software. But so is DRY. Right now doing nothing corresponds to a lot of repeated, boilerplate code created by the developers.

> The solution is to do proper C++ development and libraries as everyone does.

Implementing the functionalities provided in these jinja templates would require heavily templated c++ codes and a lot of c++ macros, which fall into the same code generation category. And it wouldn't be a clean solution too.

If you have an idea on how this can be done elegantly, I expect the ROS2 community could benefit from it too.

> Please show me how the tool is tested. How error is handled?

As it is in their plans, PickNikRobotics/generate_parameter_library#25 some integration tests can solve this problem. And since what code does it well known, well defined, it should be fairly easy to cover most of the test cases.

> Believe me that even doing nothing is better than adopting fragile tooling.

Doing nothing would be to keep writing boilerplate code. I think to see how much this tool can simplify, you'd need to write a node

- which takes in multiple types of parameters
- all their ranges are validated
- they can be updated asynchronously via parameter service calls
  And this would make you realize how this tool can simplify your

workflow.

As long as a tool is properly tested, and its functionality can be improved with clear pathways, I see no problem in adopting it.

> Please don't be more royalist than the Queen.

Sorry, I don't understand what you mean in here.

👍 1

esteve on Feb 12, 2023 Maintainer

@doganulus it'd be really helpful if you took the time to answer **@xmfcx** questions with something more specific that we can look into. I know nothing about web development and it seems you have more experience. Please try to focus on this discussion so that we can understand better your case and what you're proposing, so I'm pasting **@xmfcx** 's questions here again. Thanks.

> > But it is important to recognize robotics and ROS libraries are lagging behind in this business.
>
> I think it can be constructive if you can provide
>
> ```
> * what prospects/features of ROS 2 are lagging behir
> * how these affect us negatively,
> * what can be done about them,
> * what are the alternatives, competitor solutions?
> ```
>
> > When it comes to parameter management, I prefer the experience of web developers rather than roboticists.
>
> What aspects of the web developers' parameter management experiences do you like compared to roboticists? Could you provide more concrete examples on your expectations?

👍 1

doganulus on Feb 12, 2023 Collaborator

> As long as a tool is properly tested, and its functionality can be improved with clear pathways, I see no problem in adopting it.

No. The functionality is just one part of the system design. But-it-works approach will ruin the project sooner or later. I see it in the Autoware project and many other ROS projects. ROS2 crashed 2-3 years ago as far as I can see. And now the team has exited. I mean these decisions kill projects slowly. Better prepared.

> Doing nothing would be to keep writing boilerplate code.

Yes, writing boilerplate code is better than code generation. Much less headache. Keep an eye on Picknik's repos, they will withdraw or reimplement it as a proper library. As I said above, I would not say much if they had it as a C++ library. This is what robotics or C++ needs actually.

> Please don't be more royalist than the Queen.

ROS is the queen. But it is important to switch a view that ROS is just a C++ library (or better an interface) for pub-sub and nothing more. I am saying you give too much credit to anything coming from the ROS community and that affects the sanity of design decisions. Otherwise, I do not understand why we discuss a repo so much, which has a single unit test only (and promises from months ago) and create a dependency on it from every part of Autoware.

doganulus on Feb 12, 2023 Collaborator edited

@xmfcx @esteve @kenji-miyake

> what prospects/features of ROS 2 are lagging behind

- Explained in the discussion, in Picknik's library, and in the motivation of the Dojo. It is better to read them if there is any question.

> how these affect us negatively,

- Rigid parameter configuration would prevent us to streamline the development, build and test processes.
- Adding software dependencies carelessly would kill the project.

> what can be done about them,

- For example, study the config management of Hugo static site generator: https://gohugo.io/getting-started/configuration/
- Making a website would be a good weekend project. Everything will be clearer regarding your questions.
- And think twice or thrice when proposing a new dependency on critical paths. Do not add them if there is a slightest question.

> what are the alternatives, competitor solutions?

- This is the part 3 if I understand Kasper correctly. Doing nothing is an alternative too. We should also look at JSON schema as an alternative. Consistency with JSON schema specification might be a goal.

isamu-takagi on Feb 10, 2023 Maintainer

> single source parameter file
> which will be used for: ROS node configuration, web documentation and ROS launch

This is a parameter definition file created for each node. The existing parameter files are created for each configuration and will continue to be used. Is this right?

↑ 1 1 reply

kaspermeck-arm on Feb 10, 2023 Collaborator Author

Each node will have a parameter file with populated default values, along with some other attributes. The parameter file structure will be the same across all nodes, so that the parameter file can rendered into a table to be used in the web documentation.

kaspermeck-arm on Feb 13, 2023 Collaborator Author

Thanks everyone for contributing to the discussion! :)

My assessment of how to proceed is to keep it simple but not to prevent potential future more complex/sophisticated implementations. I am leaning towards that we go with Option 2 and define a yaml file layout in such a way which doesn't prohibit Option 1 (or other solution) further down the road. I see Option 2 as a stepping stone and allows us to make progress!

What has been decided:

- single source node parameter file (one file per node), which will be used for:
  - node configuration
  - web documentation
  - launch files
- parameter file path: *autoware.universe/.../config/*.param.yaml*
  - "..." is feature/function and package
  - "*" is the node
  - E.g., autoware.universe/localization/ndt_scan_matcher/config/ndt_scan_matcher.param.yaml
- *default_value* not to be used in *declare_parameter(...)* and all parameters in the parameter yaml file are declared in the node
  - this enforces 1-to-1 alignment between node and parameter file
- each parameter in the node parameter file has following attributes:
  - *name*
  - *type*
  - *default_value*
  - *description*
  - *bounds*

Pending:

- the node parameter yaml file layout
  - requirements:

- 1-to-1 alignment with the node parameter declaration(s)
- parameter attributes should be rendered/interpreted into an MD table for web documentation

Please have a think of the proposal above and we can discuss in the next Open AD Kit meeting!

@armaganarsln, @HamburgDave could you add this to the agenda for next week? (as we're skipping this week due to the TSC) Thanks!

↑ 1 👍 5 6 replies

Show 1 previous reply

xmfcx on Feb 15, 2023 Maintainer edited

- Enforcing the `param.yaml` files to be used instead of launch file arguments
- Making parameter declarations' default values empty in nodes
- Rendering parameters in markdown automatically

These can be done regardless of whether we use PickNik's library or not. And I support these.

So it's ok to take action to realize them.

And for PickNick's library, in the last Software WG meeting, @kasperornmeck agreed to create a small demo using generate_parameter_library for a simple package (stop_filter) in the Autoware Universe and see how it performs.

👍 3

esteve on Feb 15, 2023 Maintainer

@xmfcx ah, sorry, I forgot about that, makes total sense.

👍 1

isamu-takagi on Feb 16, 2023 Maintainer

How can node users configure parameters when this is introduced? Is it possible to use conventional way (param tags and param files), or does it need to create a dedicated parameter file?

xmfcx on Feb 16, 2023 Maintainer edited

@isamu-takagi the reason we are coming up with this enforcement comes from launch file chains. They can override arguments and it is hard to track what parameters passed to the node.

So there should be at least one default `params.yaml` for the node. And the launch file (if it exists) should pass it to the node.

As for the user running the node on their local machine, they can do it however they like, for example:

- `ros2 run demo_nodes_cpp parameter_blackboard --ros-args --params-file demo_params.yaml`
- `ros2 run demo_nodes_cpp parameter_blackboard --ros-args -p some_int:=42 -p "a_string:=Hello world" -p "some_lists.some_integers:=[1, 2, 3, 4]" -p "some_lists.some_doubles:=[3.14, 2.718]"`

Also there can be multiple `params.yaml` files in the params/config(?) folder for testing the node or for different configurations.

👍 3

kaspermeck-arm on Feb 16, 2023 Collaborator Author edited

@xmfcx

I know I committed to creating a working example for the *stop_filter* using the PickNikRobotic's generate parameter library. I don't have bandwidth for this right now and would like to postpone that effort to focus on getting things done faster. Is this OK?

@ambroise-arm and I looked at how we can adopt the yaml file layout which is used in PickNikRobotic's library. This proposed approach will work with minor modifications to *declare_parameter(...)*. For example using lidar_apollo_segmentation_tvm_nodes as a base and making modifications:

lidar_apollo_segmentation_tvm.param.yaml

```
/**:
  ros__parameters:
    range: {
      type: int,
      default_value: 90,
      description: "The range of the 2D grid with respect
to the origin.",
      validation: {
        bounds<>: []
      }
    }
...
```

lidar_apollo_segmentation_tvm_node.cpp

```
...
    declare_parameter("range.default_value",
DEFINE_HOW_TO_INSERT_TYPE),
...
```

Notice that we need to add *.defalt_value* in the *declare_parameter(...)* function. Also, as we're removing the default value which also defined the type, we need to replace **DEFINE_HOW_TO_INSERT_TYPE** above with a way to align *range.type*. This can be done in various ways.

👍 2

doganulus on Feb 17, 2023 Collaborator edited

Here are some comments on the layout design started at https://github.com/orgs/autowarefoundation/discussions/3260#discussioncomment-5002893.

1. Better layout keys and parameter names use standard identifier naming rules. I think PickNik's `<>` suffix for some keys would be problematic for parsers and text editors. This doesn't look good in dot notation.
2. Currently single scalar values must be passed as arrays to validation keys in PickNik's layout. It may ease the implementation but is inconvenient for users and error-prone. The key-value pair `gt<>: [0]` is cleaner when written `gt: 0`.
3. A top-level `version` key might be helpful to ease migrations later.

Btw do you know anyone who uses JSON schema for `node_name.ros__parameters` by chance?

↑ 1 1 reply

kenji-miyake on Feb 18, 2023

> I think PickNik's <> suffix for some keys would be problematic for parsers and text editors. This doesn't look good in dot notation.

Could you show some concrete examples to discuss it more clearly?

> The key-value pair gt<>: [0] is cleaner when written gt: 0.

It might be true. I guess it's coming from the simplicity of implementation.
If you create an issue and send a pull request, the maintainers might consider accepting it.

> A top-level version key might be helpful to ease migrations later.

It might be good as well. How about proposing it in their repository?

> Btw do you know anyone who uses JSON schema for node_name.ros__parameters by chance?

I'm sorry, I'm not familiar with that. 😢

kenji-miyake on Feb 21, 2023

FYI: There is a little related discussion https://github.com/orgs/autowarefoundation/discussions/3281, which is about the directory structure.

↑ 1 0 replies

kaspermeck-arm on Feb 24, 2023 Collaborator Author

Closing this discussion. Please find final proposal here:
https://github.com/orgs/autowarefoundation/discussions/3288

↑ 1 0 replies