STOL Automation Project – Investigation of feasibility and Identification of Best Approach

Objective: To analyze and investigate the possibility of automatically converting ASN.1 document to ROS structures

The following areas were investigated.

1. Understanding the way ASN1C converts ASN.1 programming structures to C files
2. Understanding the Structures of ROS
3. Generation of ROS structures

ASN1C

While analyzing program ASN1C, the locations where the ASN1 program is parsed, compiled into C structures and printed out to .c and .h files have been identified. It was found that ASN1C uses long and unsigned long types for all INTEGER range. This will force any C to ROS parser to generate variables that are no smaller than 4 bytes. Even if the variable takes values 0, and 2, where a byte is enough to represent the whole range of the variable, this will consume 3 extra unused bytes per such variable.

ROS

ROS has structure types that are similar to those in Java - Each file itself acts like an implicit structure type that can be referred to in another ROS file. In the current manual conversion of C structures to ROS, instead of using int32 (long equivalent) and uint32 (unsigned long equivalent) for all integer variables, wherever it is possible to use shorter variables, int16, uint16, int8 and uint8 are being used, and it is a desirable effort, too.

Generation of ROS structures

1. C to ROS conversion

This is one option looked into during this investigation as C syntax is thoroughly understood. As a demo, a parser for C-to-C was created. This parser can be directly converted to C-to-ROS parser with some changes. This C-to-ROS parser will take the output C files given by ASN1c program as input. The disadvantage of this approach is the parser will have to remove unnecessary structures, macro definitions and variables that may not have any meaning in a ROS program.

As a proof of concept of this method, initially a program that parses C to C conversion has been generated. While the parser is primitive, it is a work in progress. As the syntax of ROS is being investigated and understood thoroughly, this parser can be readily converted to a C-to-ROS parser.

1. Generate ROS structure from ASN.1

The syntax of ASN.1 can be directly converted to ROS by creating a parser like asn1c that converts ASN.1 structures to ROS structures.

1. Modify ASN1C to generate ROS structures together with C structures

As a part of this investigation, the locations where the current asn1c parses and creates core structures of ASN.1 document have been identified. Also, the locations where the C structures are generated have been identified. This knowledge allows a programmer to add hooks to the current asn1c program to generate ROS structures along with the generation of C structures.

Of the 3 options, the best option would be the third option as the first option involves only one tool and so one pass. But from the implementation point of view, it is perhaps the most complicated one. It does involve a lot less work, albeit it requires a very good skillset in programming.

Challenge

As mentioned above, one drawback of the C files generated by asn1c is that all integer variables are defined as long or unsigned long. While ASN.1 clearly allows programmer to specify the range a variable can take, ASN1C is clearly ignoring this range while declaring variables. This needs to be rectified to produce less overhead in generated messages that ROS will need to deal with. This will be an added bonus by the end of this project. This step is necessary for either of the two options enumerated above.