Creating ROS Launch Files Using A Visual Programming Interface

Aditya Narayanamoorthy, Li Renjun, Huang Zhiyong
Robotics Department
Institute for Infocomm Research
Singapore
{adityan, li-r, zyhuang}@i2r.a-star.edu.sg

***Abstract*—Visual programming is widely used to allow users to create programs by manipulating program elements graphically rather than by specifying them textually. It is intuitive and effective. However, it only starts to be used in robotic programming. For example, in the Robot Operating System (ROS), a popular framework used for developing robotic applications, in order to run multiple modules together, a ROS launch file needs to be created and used. The files are in XML format and are difficult to write and understand for non-technical users. To address this problem, in this paper, we propose a visual programming software tool that helps in the creation and visualization of these ROS launch files. This tool enables non-experienced operators to program a robot at a modular level. The tool is one among a set of software tools in the Robot Application Development and Operating Environment (RADOE), which aims to ease the development of robot applications in ROS.**

Keywords—Visual Programming, Robotics, ROS, Launch File

I. Introduction

In computer science and engineering, visual programming is a programming tool that lets users create programs by manipulating program elements graphically rather than by specifying them textually. It allows programming with visual expressions, spatial arrangements of text and graphic symbols, used either as elements of syntax or secondary notation. For example, many visual programming tools (known as dataflow or diagrammatic programming) [1] are based on the idea of boxes and arrows, where boxes or other screen objects are treated as entities, connected by arrows, lines or arcs which represent relations between these entities.

The Robot Operating System (ROS) [2] is an open-source framework that is used to write robot software. It includes libraries and other tools that help in simplifying various aspects of robot software development, such as low-level device control, combining various modules seamlessly etc.

The basic building blocks of ROS are nodes (C++ / Python executables), which talk amongst each other by passing messages through a central server over named buses called topics; nodes can publish / subscribe to these topics to broadcast / receive the messages. The nodes also make use of user-defined arguments, or parameters that are stored on a separate server.

While the nodes are usually designed as stand-alone, they often need to be used in conjunction with other nodes for doing any useful task. For example, in the context of a robot simulation module, different ROS packages are needed for displaying a GUI, performing motion planning, and visualising robot motion; all of these need to be run in parallel and communicate with each other (Figure 1). While this can be achieved by starting each node and adding the required arguments manually, this can be a cumbersome task. Instead, ROS provides a simpler method to do this - using a ROS launch file.

A ROS launch file is an XML-based file that specifies all nodes, arguments and other parameters that need to be launched together. This is a convenient way to launch multiple nodes that work together. However, XML may not be an intuitive format for a non-technical user to follow.

This paper introduces a set of software tools called the Robot Application Development and Operating Environment (RADOE), which aims to ease the development of robot applications in ROS. In particular, the paper focuses on a tool within this framework that helps the user create this via visual programming, which lets the user create the ROS launch files through graphical representations of the launch file elements. The tool has an easy-to-use interface which handles the creation of the ROS launch file in the background, without any further involvement needed from the user.

II. Literature Review

This section covers a review of existing visual programming software, which helped in seeing whether there were any existing software products that did the same task, and selecting features that were necessary for the development of RADOE's visual programming tool.

In terms of commercial software for visual programming, the Microsoft Visual Programming Language [3] (Figure 2) is one of the more widely-used ones. It comes as part of the Microsoft Robotics Developer Studio, and can be used for both general-purpose programming, as well as robotics-specific services. It represents lower-level sensors and devices through

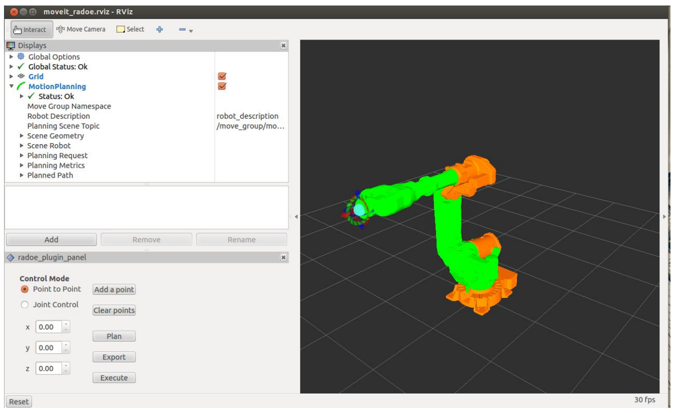

Fig. 1. Robot model displayed in rviz, a visualisation tool in ROS

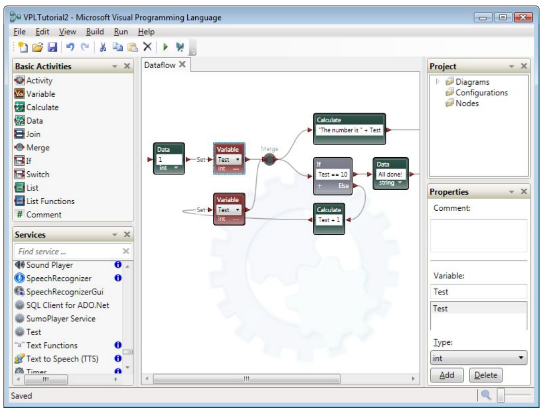

Fig. 2. Microsoft Visual Programming Language (MVPL) Interface

abstractions in the visual interface, which need to later be implemented with specific drivers (in C#). It also has the option of specifying pre-loaded controllers for hardware through a manifest file. Even though there is nothing ROS-specific about this software, it has a good working visual interface and hence was reviewed.

Another set of robotics-based software that uses visual programming is for educational purposes. The most commonly known in this category are the series of software products released by LEGO; more specifically, their MINDSTORMS EV3 software [4]. This consists of block-based programs, where each sensor / functionality is represented by a graphical block, which can be plugged into other blocks to determine the flow of the program. The programming constructs are simple and only deal with the limited hardware that comes with the LEGO toolkits.

There are various other examples of robotics software that have a visual programming interface for educational/recreational use of robots, usually aimed at children. These include Modkit [5] (to connect to Arduino hardware components), ROBO Pro [6] (for programming fischertechnik robots), CiMPLE [7] (used in programming ThinkLabs robots) etc. While these are not connected to ROS programming, they help in understanding the various aspects of visual programming software that has good usability, and is easy to pick up - even amongst non-technical users, like children.

The third category of visual programming tools is those more directly related to ROS, developed as third-party software. One of these is rxDeveloper [8], a tool developed for the same purpose as RADOE's visual programming tool - to generate ROS launch files using a visual programming interface. It is written in C++, in the QT programming development environment. Nodes and other components are represented as boxes, which can be dragged and dropped onto a canvas (Figure 3). Nodes can also specify if they communicate with other nodes by connecting them with lines (which remap topic names, so that both nodes communicate using the same topic). The launch file can be generated and reloaded, and also run directly from the tool. While this tool has most of the features required for the creation of ROS Launch files, it works with an older version of ROS (Electric Emys), and can possibly be improved in terms of user-friendliness.

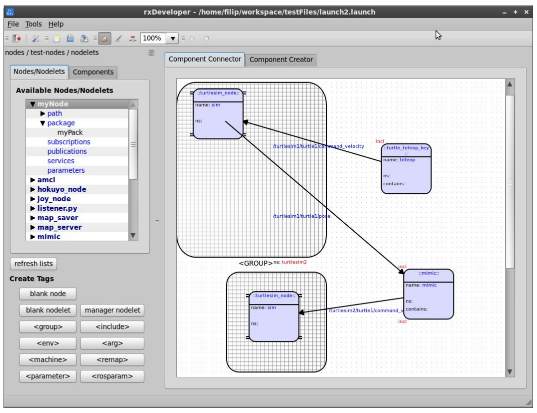

Fig. 3. rxDeveloper Interface

Another tool that works well with ROS is the ROS Toolkit for LabView [9], created by Clearpath Robotics. It helps LabView [10] (a visual programming software by National Instruments intended for system design and instrument control) access ROS nodes through TCP / IP and web sockets using a ROS package called rosbridge [11]. This was made to work specifically with the Baxter Research Robot series, but also has more general-purpose ROS functionality. While this serves well to integrate the visual programming interface of LabView with ROS, it is more cumbersome to construct and run ROS-only programs in comparison to ROS Launch files.

Most of the above-mentioned software products are good for developing robotics applications. However, there are few software products that could be used to develop applications in ROS. Therefore, in this paper, we propose a new tool for generating ROS launch files to program robots.

III. OUR WORK

Our software environment, RADOE, consists of various software tools that ease the development of robot applications. These include using the QT programming environment (used for C++ programming), and ROS packages such as rviz (used for visualisation and simulation of robots). To help the user understand and organise ROS-based applications easily, a visual programming tool was developed within this environment. This software tool deals with ROS components at the node or launch file level.

On starting the tool, the user gets an interface (Figure 4) that includes a list of buttons, which show all the possible elements that can be added into a ROS launch file, and a blank canvas on which the various elements are represented as rectangles. On being clicked, the buttons open up an editor window where various properties of that element can be specified. On completion, the element is represented on the canvas. Some of the fields have been made more user-friendly: for instance, in the editor for the Include element, a file name needs to be specified; this is done using a separate window for file selection, rather than having the name manually typed out (which is more cumbersome and might cause typing errors) (Figure 5).

Elements can be dragged and dropped around the canvas, which can help in easily visualizing the content of the launch file. Double-clicking on an element re-opens the editor window with all the properties of the element filled in; it thus becomes easy to change the values for these properties after creation. They can also be resized easily by clicking and dragging on the bottom-right corner of the rectangle. Certain elements can also have other elements dragged and dropped onto them, to create nested structures. This is similar to the structure that can be found in ROS launch files - for example Group elements can hold any other type of element within themselves.

The toolbar on the top contains a button called Remap Mode. This option enables a node's topic name to be remapped, so that it can communicate with another node that uses a different topic name. In the remap mode, this is done by clicking on the source node and dragging to the destination node. On completing the drag, an editor window shows up that lets the user specify the topic name to be remapped, and its new name. A line drawn between the nodes represents the remapping when completed successfully.

Once the launch file has been designed, it can be saved by clicking on the Create Launch File button on the top toolbar, which generates a launch file with all the elements on the canvas, created according to the specifications laid down by ROS. The launch file can also be loaded back into the program using the Open Launch File button to select the required file, so that it can be further edited.

Before saving the launch file, the user might want to test how it runs. For this, there is a Run Launch File option, which creates a temporary launch file and runs this on a terminal emulator.

From the software architecture point-of-view, each element class inherits from a base class called BaseDragWidget. This contains the common behaviour for all widgets, as well as common properties shared by all ROS Launch elements (such as the if and unless attributes). Each element class has the corresponding properties that are contained in that ROS launch element. Each element also has a corresponding dialog widget that shows the edit window, which helps in setting property values, and editing them later on.

The main canvas is represented by a CanvasWidget, which implements drag-and-drop functionality for the widgets, and also draws lines to represent the remaps. The main widget, which includes the canvas, and the buttons for initialising each

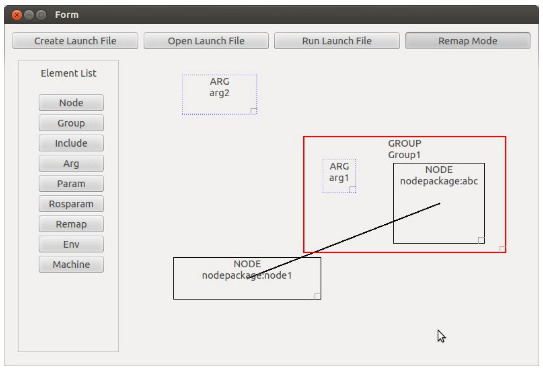

Fig. 4. Proposed tool's interface

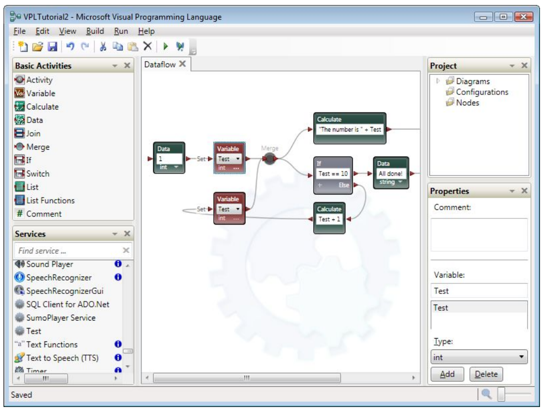

Fig. 5. Example of Editor Window for individual components

element, also implements the main toolbar functionalities of saving / loading / running the launch files. On saving the launch files, each element present on the canvas is invoked to return its ROS launch file text. Each element also recursively calls each of its children in turn, to return the nested ROS launch file text.

IV. COMPARATIVE STUDY

Since the rxDeveloper tool is the closest in intent and functionality to our visual programming tool, the following is a comparison between the two:

A. Functionality

Both rxDeveloper and our tool have mostly similar functionalities, including dragging and dropping of elements, connecting them via line-drawing etc. rxDeveloper has the option of specifying a node's topic publications/subscriptions in a node specification file, which is stored in the YAML format. This can help in other additional features, such as generating source code templates. However, since our tool is mainly concerned with creation of ROS launch files, this is considered as a feature that is additional to the requirements of our tool. Moreover, our visual programming tool is a supplement to the RADOE software tools, and is meant to simplify the development of other ROS-based applications.

B. *Usability*

While both tools have many similar features that improve the user experience in creating ROS launch files, there are some aspects to the rxDeveloper tool that make it slightly less user-friendly. For instance, the height and width of elements can only be changed by manually typing in the pixel value, whereas our tool has the option of dragging to resize. In addition, our tool has some other features that make it easier to use, such as selecting a ROS launch file to include by using a file selector which automatically fills in the file name.

We started a user study to gauge the usefulness of the tool, with subjective feedback from the user helping in deciding how the tool can be further improved. The initial results are positive. We will report quantitatively in future.

V. CONCLUSION AND FUTURE WORK

In this paper, we proposed and implemented a visual programming software tool that helps in the creation and visualization of ROS launch files. It enables easy and intuitive programming of a robot at a modular level. The initial user study has confirmed the usefulness of our design and implementation. The tool has been integrated in a set of software tools in the Robot Application Development and Operating Environment (RADOE).

While our tool covers all the basic features needed to create ROS launch files, there are additional features that could be added in the future to improve it further. These include finding a way to automatically extract publication/subscription information of a node using just its package name and node name; while the current API from ROS does not allow this to be done, a workaround may be found with further research.

Another aspect would be to make the interface more generic to use, so that it is not restricted to the creation of ROS launch files. This would enable the tool to be used to structure ROS applications in our RADOE environment at a higher level, such as visualising modules and the connections between them, without relying on ROS-specific terminology. This would make it even easier for general users to understand and design robotics applications.

REFERENCES

[1] W.M Johnston, J.R.P. Hanna, and R. J. Millar, Advances in dataflow programming languages. ACM Computing Surveys 36 (1): 1–34, 2004.

[2] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, “ROS: an open-source Robot Operating System,” in Proc. of IEEE International Conference on Robotics and Automation (ICRA), Kobe, Japan, 2009.

[3] https://msdn.microsoft.com/en-us/library/bb483088.aspx

[4] http://www.lego.com/en-us/mindstorms/learn-to-program

[5] A. Milner and E. Baafi, “Modkit: Blending and Extending Approachable Platforms for Creating Computer Programs and Interactive Objects,” in IDC , Ann Arbor, USA, 2011.

[6] http://www.fischertechnik.de/en/Home/info/computing/ROBO-Pro-Software.aspx/usetemplate-1_column_no_pano/

[7] N. Karwall, “Visual Programming Application for Children to program Robotic Toys,” in ‘Designing for Children’, IDC, IIT Bombay, India, 2010.

[8] F. Müllers, D. Holz, and S. Behnke, “rxDeveloper: GUI-Aided Software Developmemt in ROS,” in SDIR VII - ICRA, St. Paul, Minnesota, USA, 2012.

[9] https://sites.google.com/site/rosforlabview/

[10] http://www.ni.com/labview/

[11] https://github.com/RobotWebTools/rosbridge_suite