Обзор свободного фреймворка ROS — операционной системы для роботов

Валерий Касьяник, Брест, Беларусь*

ROS (Robotics Operation System) is an open-source framework designed to ease developing software for robots. Report covers it's main advantages, use cases, existing shortcomings and future development roadmap.

Введение

Робототехника сейчас одна из самых активных областей развития технологий. Множество компаний и энтузиастов собирают своих собственных роботов, разрабатывают новые аппаратные и программные решения, накапливая критическую массу для возниконовения новых технологий. Так программное обеспечение роботов эволюционирует огромными темпами от простейших прошивок контроллеров устройств до сложнейших программных систем управления для решения широкого круга задач в робототехнике. И как в любой сложной системе здесь возникают архитектурные проблемы, проблемы повторного использования кода, разработки и отладки, коллективной разработки.

Для решения этих проблем был предложен фреймворк ROS (Robotics Operation System)

Фреймворк ROS как операционная система

Фреймворк ROS — это набор утилит, библиотек, соглашений, который упрощает разработку сложного программного обеспечения для различных робототехнических платформ. ROS работает поверх классической операционной системы, обеспечивая службы мета-операционной системы робота: аппаратную абстракцию различных устройств робота, низкоуровневый контроль этих устройств, реализацию часто используемых функций и задач, передачу сообщений между процессами, и управление пакетами. Можно сказать,

*val.tut@gmail.com, http://lvee.org/ru/abstracts/116

что ROS управляет взаимодействием вычислительных устройств с реальным миром.

Как и другие ОС, ROS состоит из двух частей: непосредственно ядра фреймворка **ros**, и **ros-pkg**, набора поддерживаемых сообществом пакетов, которые реализуют различные функции робототехники: позиционирование, планирование, восприятие, моделирование и др.

ROS выпускается в соответствии с условиями BSD- лицензии и с открытым исходным кодом. Кроме того, ROS бесплатен для использования не только в исследовательских, но и в коммерческих целях. Пакеты из **ros-pkg** распространяются на условиях различных открытых лицензий и авторы сами решают какую лицензию использовать.

Ключевые архитектурные особенности ROS

Основными концепциями ROS являются узлы, сообщения, темы, сервисы. Архитектурно все вычислительные задачи выполняются в узлах ROS, которые обмениваются между собой информацией посредством сообщений. Эти сообщения узлами публикуются в темах, которые разделяют эти сообщения на группы интересов. Когда некоторому узлу необходимо получать сообщения с определенными данными, этот узел подписывается на определенную тему. Для реализации синхронной передачи сообщений, которая необходима в определенных случаях, ROS определяет сервисы — механизм, который работает по принципу вопрос-ответ.

Основные способы применения ROS

Как фреймворк ROS подразумевает наличие методологии его применения для решения задач робототехники.Рассмотрим его основные сценарии применения.

Тестирование одной задачи. Этот способ применения хорошо подходит для исследовательских экспериментов. Обычно даже простая исследовательская задача требует большого количества вспомогательного кода, который работает с оборудованием или реализует связанные задачи. ROS преоставляет разработчику возможность сосредоточиться только на своем коде.

Логирование и воспроизведение данных. ROS обеспечивает логирование и воспроизведение записанных результатов. Механизм реализован через единый канал **rosout**.

Обмен разработками и повторное использование кода. Также ROS реализует свой собственный пакетный менеджер, который предоставляет пользователю возможность использовать уже отлаженные и готовые алгоритмы, искать уже реализованные идеи, удобно хранить их и использовать в своих проектах. Для этого существуют утилиты **rospack**, **rosbash**, которые реализуют функциональность пакетного менеджера unix операционных систем для быстрой навигации и работы с пакетами ROS. Кроме того, такой подход позволяет эффективно организовать совместную разработку и перенос стороннего ПО в пакеты ROS.

Тестирование и отладка. Для тестирования и отладки в ROS имеются специальные утилиты **rviz**, **rxplot**. Первая позволяет динамически отображать граф связей запущенных узлов, публикуемые темы, сервисы. Вторая утилита предоставляет инструмент для анализа передаваемых сообщений, отображения временных данных, показаний датчиков и т.д.

Управление сложностью. Как отмечалось выше, архитектура ROS состоит из узлов, которые реализуют некую функциональность робота. Так как сложные задачи навигации или управления манипулятором требуют множества вычислений, то они как правило реализуются множеством связанных узлов, которые могут выполнять одни и те же действия для разных роботов. ROS решает проблему дублирования функциональности с помощью кластеров. Так например, собрав и отладив узлы управления манипулятором, разработчик может запустить несколько копий такого кластера для управления несколькими манипуляторами, а координировать эти кластеры будет отдельный высокоуровневый узел.

Преобразование данных. Потоки данных между подсистемами робота разнородные и требуют соответственно различных преобразований. Для упрощения разработки и отладки информационных потоков в ROS был предложен отдельный модуль **tf**, который отвечает за все информационные преобразования между узлами и сервисами. Таким образом, при написании узла разработчик не заботится о согласовании своего модуля с другими, а выносит эту функциональность в модуль **tf**.

Недостатки и проблемы ROS

Основной проблемой ROS является ее главное достоинство. и продуманая архитектура является проблемой для новичков, только начинающих знакомство с системой или для исследователей, плохо знакомых с разработкой ПО. На решение этой проблемы сообщество выделяет значительные ресурсы: ведется портал с документацией по каждой версии ROS, выпускается серия книг, множество туториалов по использованию ROS с другими популярными инструментами.

Вторым недостатком считается отсутствие поддержки ОС Windows, т. к. многие исследователи и разработчики по-прежнему продолжают использовать эту ОС. Над этим недостатком также ведется работа сообществом, и есть предварительные результаты.

Развитие ROS

ROS продолжает активно развиваться во многих направлениях, что обусловлено философией коллективной разработки. Активно добавляются новое аппаратное обеспечение, сделаны драйвера для множества популярных устройств: роботов-пылесосов iRobot, Microsoft Kinect, Arduino, Raspberry Pi, Leaf Maple и многих других. Сейчас одним из популярных направлений является портирование основных инструментов ROS на ARM, чтобы у разработчиков появилась возможность запускать узлы на самых разных устройствах. Также расширяется выбор инструментов разработки: C++, Python, Matlab.

Выводы

В результате всего вышеперечисленного на ум приходит аналогия с разработкой Linux, когда ядро Линуса Торвальдса объединило и дало новый толчок разработке свободного программного обеспечения. Похоже, что бурно растущая область робототехники тоже получила тот центральный элемент, который способен объединить многих разработчиков на пути достижения мечтаний Азимова.

Автору также представляется, что фреймворк ROS в будущем, благодаря своим достоинствам, может превратиться из мета-операционной системы роботов в операционную систему «умных вещей», которая объединит в себе роботов, умные дома, бытовую технику и т. д. Все возможности для этого в ней уже заложены.