Nama	Muhammad Abyan Ridhan Siregar
NIM	1103210053
Kelas	TK-45-01

Tugas Week 14

Summary “ROS1 vs ROS2”

Sejarah dan Alasan Pengembangan ROS2

- **Sejarah ROS1**: Diluncurkan pada tahun 2007 oleh Willow Garage, menjadi sangat populer di komunitas robotika open-source.

- **Alasan ROS2**: Dibuat dari awal untuk mengatasi keterbatasan ROS1, seperti kebutuhan akan real-time, sertifikasi keselamatan, dan keamanan. Perubahan besar di ROS1 akan menyebabkan ketidakstabilan, sehingga ROS2 dikembangkan dengan arsitektur yang lebih modern dan stabil.

Perbedaan Utama

1. **API dan Arsitektur**

    - ROS1:
        - Menggunakan `roscpp` untuk C++ dan `rospy` untuk Python.
        - API untuk C++ dan Python dibangun secara independen dan tidak selalu konsisten.
    - ROS2:
        - Menggunakan library dasar `rcl` yang diimplementasikan dalam C dan digunakan oleh `rclcpp` (C++) dan `rclpy` (Python).
        - Memastikan API lebih konsisten antar bahasa dan memudahkan pembuatan library client untuk bahasa lain.

2. **Struktur Node**

    - **ROS1**: Tidak ada struktur khusus, bisa menggunakan fungsi callback atau pemrograman berorientasi objek (OOP).

    - **ROS2**: Menggunakan OOP dengan kelas yang mewarisi objek Node, membuat struktur kode lebih modular dan konsisten.

3. **Komponen dan Nodelet**

    - **ROS1**: Nodelet digunakan untuk komunikasi intra-process.

    - **ROS2**: Menggunakan komponen yang terintegrasi langsung dalam core ROS2, memungkinkan banyak node dalam satu executable.

4. **Lifecycle Node**

   ROS2 : Memperkenalkan node dengan lifecycle yang memiliki status seperti unconfigured, inactive, active, dan finalized. Ini sangat berguna untuk fase setup sebelum menjalankan fungsi utama node.

5. **Parameter dan Server Parameter**

   - **ROS1**: Menggunakan server parameter global dalam ROS master.
   - **ROS2**: Setiap node memiliki parameter sendiri yang dikelola dan bisa diubah secara dinamis dengan callback parameter, mirip dengan dynamic reconfigure di ROS1 tetapi lebih terintegrasi.

6. **Layanan (Services)**

   - **ROS1**: Layanan sinkron.
   - **ROS2**: Mendukung layanan asinkron dengan callback, memberikan lebih banyak fleksibilitas dalam pengembangan.

7. **Aksi (Actions)**

   - **ROS1**: Aksi dibangun di atas topik dan bukan bagian dari core.
   - **ROS2**: Aksi menjadi bagian dari core dan menggunakan topik serta layanan asinkron, dengan API yang mirip dengan ROS1 tetapi lebih terintegrasi.

8. **Quality of Service (QoS)**

   ROS2: Memperkenalkan QoS untuk mengatur bagaimana node menangani komunikasi, memastikan pesan diterima dengan cara tertentu sesuai kebutuhan aplikasi.

Lingkungan Pengembangan

1. Sistem Build

   - **ROS1**: Menggunakan `catkin`.
   - **ROS2**: Menggunakan `ament` dan `colcon`, yang lebih modern dan mendukung lebih banyak fitur.

2. **Alat Command Line**

   - **ROS1**: Menggunakan berbagai alat seperti `rosnode`, `rostopic`.

   - **ROS2**: Konsisten menggunakan `ros2` diikuti dengan perintah, misalnya `ros2 node`, `ros2 topic`, membuatnya lebih terstruktur dan mudah diingat.

3. **Paket**

   ROS2: Membedakan antara paket C++ dan Python saat membuat paket, memerlukan build type `ament_cmake` untuk C++ dan `ament_python` untuk Python. Ini membuat struktur paket lebih jelas dan terorganisir.

4. **Sumber Overlay**

   ROS2: Mendukung overlay workspace untuk pengembangan yang lebih fleksibel, memungkinkan pengujian dan pengembangan terisolasi tanpa mengganggu kode yang ada.

5. **Sistem Operasi yang Didukung**

   ROS2: Dapat dijalankan di Ubuntu, Windows, dan MacOS, membuatnya lebih fleksibel dan kompatibel dengan berbagai aplikasi.

Kapan Beralih ke ROS2

1. **Pengembang Baru** : Disarankan untuk belajar dasar-dasar ROS2 terlebih dahulu, tetapi juga mendapatkan sedikit pengalaman dengan ROS1 untuk memahami perbedaannya dan menggunakan paket-paket yang belum di-porting ke ROS2.

2. **Proyek Baru** : Mulai dengan ROS2 untuk mengurangi pekerjaan transisi di masa depan. Konsep inti antara ROS1 dan ROS2 serupa, sehingga pengembang yang berpengalaman dengan ROS1 dapat beralih dengan lebih mudah.

3. **Basis Kode yang Ada** : Pertimbangkan kompleksitas dan waktu yang dibutuhkan untuk migrasi. Gunakan ROS1 untuk proyek lama dan mulai transisi ke ROS2 untuk proyek baru.

ROS1 Bridge

Fungsi : Menghubungkan komunikasi antara ROS1 dan ROS2, membantu dalam proses migrasi bertahap, memungkinkan penggunaan fitur ROS2 sambil mempertahankan kode ROS1 yang ada.

Kesimpulan

Video ini memberikan panduan komprehensif tentang perbedaan utama antara ROS1 dan ROS2, menunjukkan bahwa ROS2 menawarkan arsitektur yang lebih modern dan stabil dengan banyak fitur baru yang mendukung kebutuhan industri. Bagi pengembang, mempelajari dan beralih ke ROS2 merupakan langkah strategis untuk masa depan pengembangan robotika.