Presentation Outline

Who am I?
Background on ROS
  Motivations
  History
Best Practices Motivated by Above
“...to support the development, distribution, and adoption of open source software for use in robotics research, education, and product development.”

http://osrfoundation.org
The Goal: (Open-Source) Rapid-Prototyping

web 2.0

robotics

Open Source Robotics Foundation
The Goal: (Open-Source) Rapid-Prototyping

web 2.0

robotics

ROS and friends

OpenCV

CloudSim

GAZEBO

pcl

MoveIt!
Movie: 5 years of ROS

https://www.youtube.com/watch?v=PGaXiLZD2KQ
Robot Software Challenges

**technical**

- hardware abstraction
- scalability
- distributed debugging

**social**

- interpersonal
- inter-institutional
- inter-disciplinary
Ancient History

Switchyard

Player | Stage

Personal Robot 1 (PR1)

c.2007-2009

ROS architecture
Recent History

**ROS distributions**

distribution = stable target for applications

- **ROS 1.0 M3** (2010)
- **Box Turtle** (2010)
- **Diamondback Launch** (2011)
- **Electric Dawn** (2011)
- **Ros Fuerte** (2012)
- **ROS Groovy Gaiapogae** (2012)
- **Hydro Medusa** (2013)
- **Indigo Gloo** (2014)
ROS is not...

...a “real” OS

...monolithic

...tied to any particular robot
ROS is...

- Plumbing
- Tools
- Ecosystem
Community: ROSCon

Total: 288
(37% increase from 2012)
ROS users (self-reported)
<table>
<thead>
<tr>
<th>Rank</th>
<th>Country</th>
<th>Visitors</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>United States</td>
<td>53,530</td>
<td>23.40%</td>
</tr>
<tr>
<td>2</td>
<td>Germany</td>
<td>25,133</td>
<td>10.99%</td>
</tr>
<tr>
<td>3</td>
<td>China</td>
<td>17,812</td>
<td>7.79%</td>
</tr>
<tr>
<td>4</td>
<td>Japan</td>
<td>14,619</td>
<td>6.39%</td>
</tr>
<tr>
<td>5</td>
<td>Spain</td>
<td>9,192</td>
<td>4.02%</td>
</tr>
<tr>
<td>6</td>
<td>Canada</td>
<td>8,611</td>
<td>3.76%</td>
</tr>
<tr>
<td>7</td>
<td>France</td>
<td>8,399</td>
<td>3.67%</td>
</tr>
<tr>
<td>8</td>
<td>United Kingdom</td>
<td>7,434</td>
<td>3.25%</td>
</tr>
<tr>
<td>9</td>
<td>India</td>
<td>7,360</td>
<td>3.22%</td>
</tr>
<tr>
<td>10</td>
<td>Italy</td>
<td>6,623</td>
<td>2.90%</td>
</tr>
<tr>
<td>11</td>
<td>Brazil</td>
<td>5,837</td>
<td>2.55%</td>
</tr>
<tr>
<td>12</td>
<td>South Korea</td>
<td>4,988</td>
<td>2.18%</td>
</tr>
<tr>
<td>13</td>
<td>Australia</td>
<td>4,828</td>
<td>2.11%</td>
</tr>
<tr>
<td>14</td>
<td>Singapore</td>
<td>4,688</td>
<td>2.05%</td>
</tr>
<tr>
<td>15</td>
<td>Russia</td>
<td>3,534</td>
<td>1.54%</td>
</tr>
<tr>
<td>16</td>
<td>Portugal</td>
<td>3,194</td>
<td>1.40%</td>
</tr>
<tr>
<td>17</td>
<td>Netherlands</td>
<td>2,556</td>
<td>1.12%</td>
</tr>
<tr>
<td>18</td>
<td>Switzerland</td>
<td>2,454</td>
<td>1.07%</td>
</tr>
<tr>
<td>19</td>
<td>Poland</td>
<td>2,279</td>
<td>1.00%</td>
</tr>
<tr>
<td>20</td>
<td>Taiwan</td>
<td>2,252</td>
<td>0.98%</td>
</tr>
</tbody>
</table>
Metrics (for July 2014)

Downloads:
   Total packages: 3,570,374 (+123% over last year)
   Unique IP addresses: 49,153 (+343%)
   Does not count mirrors (8 public and many private)

Web activity:
   ros.org (wiki):
      Unique visitors: 87,820 (+48%)
      Page views: 976,031 (31.5K / day)
   answers.ros.org (Q&A forum):
      Total questions to date: 18,144 (+38%)
      Answer rate: 68%

Research use (as of Oct 2014):
   Citations of “ROS: an open-source Robot Operating System” (Quigley et al., 2009): 1327
Best Practices Categories

Efficient Development
Reproducibility
Reusability
Community Conventions
File management

- Never edit files in `/opt/ros`
- Always use version control
  - Useful tools:
    - `rosws`, `rosinstall`, `vcs`
- Use overlays
  - To override or customize packages
  - Be careful of all dependencies
3rdparty libraries

- Leverage existing 3rdparty libraries
  - If it’s in debian/ubuntu use that version
    - If not in rosdep submit a rule
  - if it’s not, look for a proposed upstream version
  - If not proposed, consider proposing it.
  - Final resort write a ROS wrapper package

- If you can develop a ROS independent library and release a parallel ROS wrapper
Rules for working in vcs

- Don’t check in large binaries
- Don’t check in generate code
  - Build out of source
- Set ignores (.gitignore, svn:ignore, etc.)
- Create small atomic changes and commit often
- Limit one feature per commit
Package Design

- Packages are cheap, use them to provide semantic grouping
  - dependencies
  - functional purpose
  - interdependency
- **CHOOSE GOOD NAMES!**
- **Write a description!**
- Create separate message/srv/action packages
Debuggability

- Use appropriate console logging
  - Debug, info, warn, error, fatal
- Provide introspection/debug topics
- Use the right concepts
  - topics for streaming
  - services for short/atomic RPC
  - actions for long running RPC
  - dynamic parameters for runtime adjustable parameters
Finding solutions

http://answers.ros.org

- Search for questions
- Ask questions
- Answer questions

- If a bug is found file a ticket
Testing

- Write unit tests
  - They will save you time in the future!
- Setup continuous integration
  - The faster mistakes are caught, the less pain they will cause.
  - (Needs unit tests...)

Open Source Robotics Foundation
Code Reviews

- If developing in a group consider using pull-request code reviews.
  - A second set of eyes is valuable.
- Pair programming can be effective.
- Conduct design reviews
  - Design reviews are more important than code reviews
Release reusable packages

- Release numbers are cheap
  - Release early, release often
- Be clear about stability for downstream developers
- **Write documentation!**
- Provide example launch files (separate packages if different dependencies)
- Group using metapackages
Use good names!

- Packages have specific semantic meaning
  - No “utils” or other catchalls.
  - Don’t over scope, “planners” is a bad name.

- Define/document interfaces for modules
  - sets of topics and parameters expected
Community Conventions

**Naming Conventions**

- package names
- topic names
- node names
  - local parameters
  - local topics
Don’t optimize too early
Nodelets can give much better performance
Community Conventions

- Use standard units, **REP 103**
- Leverage standard interfaces
  - Mobile bases, **REP 105**
  - Diagnostic System, **REP 107**
  - Humanoid systems, **REP 120**
- Review REPs in general
- Propose your own
Logging

- Leverage ROS bags
  - design to be able to record all inputs
  - enable launch of processing pipeline with different parameters from bag files
  - Build and distribute datasets.
- Tag specific versions of things run in experiments (preferably versioned releases)
Follow the style guides

- C++ Style Guide
- Python Style Guide
Communicate

- Write documentation!
- Research other products out there already.
  - Don’t reinvent the wheel.
- Announce your work
  - On ros-users
- Look for collaborators
- Create a blog
  - Submit it to planet.ros.org
Participate

Stay connected
 Norte Subscribe to the ros-users mailing list
 Norte Read the blog
 Norte Follow planet.ros.org
 Norte Join/make special Interest groups
Future Work -- ROS 2.0

- Enable better separation of computation and execution
- Promote and collapse primitives developed in ROS 1.0
  - parameters vs dynamic parameters
  - services vs actions
  - nodes vs nodelets
  - master vs multi-master
- Get involved join [ros-sig-ng](#)
Questions?