Towards a Robot App Store

Brian Gerkey
Willow Garage
The power of apps

• Shared platform provides system services, distribution & installation mechanism
• Creative users develop and publish novel applications
• Platform functionality explodes (and some people even make money)
What's a robot app?

• In the near future

• Eventually

• For now:
  – demonstrations
  – experiments
  – challenge entries (!)

MapIt!
Autonomous exploration and mapping for any indoor environment.

Click to Buy ($49.99)
Outline

• ROS
• Open Source
• Licensing
• Libraries
• Modularity
• Federated development
ROS (http://ros.sf.net)

- What is ROS?
  - Meta operating system for robotics
  - System for obtaining, building, writing and running code across multiple computers
  - Designed around mobile manipulation systems
ROS (http://ros.sf.net)

Example: opening doors and plugging in

http://pr.willowgarage.com/wiki/Milestone2/Results_2009-05-29_Integrated_(Trial_Procedure)
Open Source

• Core components should be Open
  • much research to be done, and researchers need to see (and change) how things work
  • core system not perfect; users' patches are efficient fixes

• Example core components:
  – build [cmake, pkg-config | rospack, rosbuil]
  – launch [bash | roslaunch]
  – communication [glibc | roscpp, rospy]
  – analysis [top, netstat | rostopic, rxgraph]
  – debugging [gdb | roswtf]
Open Source

• Code used to make claims in papers should be Open
  - key part of experimental design
  - necessary to replicate, refute, or extend results

• How? (*)
  - include versioned download details in the paper
    • SVN URL + revision; Git ref + hash
  - can't share physical state?
    • share configuration info for a well-known simulator

[*] See Wawerla & Vaughan, RSS 2009 workshop on experimental practice
Licensing

• Core components should support commercial use, without license constraints on applications
  – glibc: LGPL
  – ROS core: BSD
• Mid-level components will be more widely used if they follow suit
  – more people will improve upon them, too
  – most ROS packages: BSD or Apache
• Applications: license as appropriate
Libraries

• Implement useful functionality as a library, independent of any robot framework
  – imagine the developer who likes your functionality but doesn't like your framework
• Bind your library into the framework(s) you use
  – bindings should be thin
Libraries

• Issues
  – dependencies
  – data structures
  – control loops / state machines
  – version hell
Modularity

• Break functionality up into small pieces
• Plan for reuse of each piece
  – expose a well-defined interface
• Modules provide natural license boundaries
• Issues:
  – maintenance, QA, release burden, dependency hell
Federated development

• Q: “How do I contribute?”
  – A: Publish your code in a publicly-accessible place (e.g., SourceForge, Google Code)

• Avoid single gateway for (re)distribution of code
  – authors retain control, get credit
  – authors choose licenses, development policies, release schedules
  – scale to worldwide development
Federated development

- Known ROS repositories (12)
Federated development

• Issues:
  – finding available code
  – avoiding duplication of work
  – working from multiple repositories
  – quality control
Hypothesis

• Shared, Open infrastructure + modular libraries + commercial-friendly licensing + federated development =
  – shared engineering burden
  – accelerated system development
  – better scientific practice
  – transferable challenge results
  – vibrant business ecosystem
• and, eventually...a RobotApp Store.
Acknowledgements

• The ROS team:
  – Lead: Morgan Quigley
  – Josh Faust, Ken Conley, Rob Wheeler, Tully Foote, Jeremy Leibs, Bhaskara Marthi, Rosen Diankov

• The STAIR project: Andrew Ng & Co.

• Everybody at Willow

• The fledgeling ROS community