* 1. robotic\_interviews

## Nodes

| Name | Description | Files | References |
| --- | --- | --- | --- |
| Background |  | 0 | 0 |
| Current role |  | 16 | 23 |
| Hardware |  | 2 | 2 |
| Robots |  | 0 | 0 |
| Autonomous vehicle robots |  | 1 | 1 |
| Bomb disposal robots |  | 1 | 1 |
| Cleaning robots |  | 1 | 1 |
| Climbing robots |  | 1 | 1 |
| Consumer robots |  | 1 | 5 |
| Exercise assistance robots |  | 1 | 1 |
| Forklift |  | 2 | 2 |
| Hospital robots |  | 1 | 1 |
| Humanoid robots |  | 3 | 3 |
| Inspection robots |  | 0 | 0 |
| Floor Inspection |  | 1 | 2 |
| Nuclear inspection |  | 1 | 1 |
| Refineries inspection |  | 1 | 1 |
| Lawnmower robots |  | 1 | 1 |
| Line marking robots |  | 1 | 3 |
| Mink farm robots |  | 1 | 1 |
| Mobile manipulators |  | 7 | 7 |
| Mobile platforms |  | 3 | 3 |
| Modular robots |  | 1 | 1 |
| Radiation monitoring robots |  | 1 | 1 |
| Service robots |  | 1 | 1 |
| Transportation robots |  | 3 | 3 |
| Underwater robots |  | 3 | 3 |
| Vacuum cleaner |  | 1 | 1 |
| Software |  | 0 | 0 |
| Controllers |  | 2 | 2 |
| Development Tools |  | 1 | 1 |
| Front-end |  | 1 | 1 |
| Middleware |  | 1 | 1 |
| Perception |  | 1 | 1 |
| Simulator |  | 1 | 1 |
| Studies and company |  | 10 | 21 |
| Future plans |  | 1 | 1 |
| Cool things |  | 0 | 0 |
| Ethical motivation |  | 1 | 1 |
| Use in education |  | 2 | 2 |
| RQ1 State of Robotic Software Development | How is software developed for mobile service robots? | 0 | 0 |
| RQ1.1 Principles | What software engineering principles are used, incl. processes or QA techniques? | 0 | 0 |
| Activities |  | 0 | 0 |
| Common activity - Evolution and maintainance |  | 2 | 2 |
| Common activity - Implementation |  | 2 | 4 |
| Common activity - Real-world experimentation |  | 3 | 3 |
| Common activity - Testing |  | 4 | 7 |
| Pressure |  | 2 | 3 |
| Consulting |  | 1 | 1 |
| Integration |  | 1 | 2 |
| Less common activity - Requirements engineering |  | 1 | 1 |
| How RE is done |  | 3 | 9 |
| Business-oriented colleagues |  | 1 | 1 |
| Customer Interviews |  | 5 | 5 |
| Customers |  | 2 | 7 |
| Demonstrations |  | 4 | 6 |
| Dealing with customers who don't know what they want |  | 1 | 1 |
| HW and SW bundled |  | 1 | 1 |
| From high-level to technical |  | 2 | 2 |
| Sales |  | 1 | 2 |
| Test markets |  | 1 | 1 |
| Workshops |  | 1 | 1 |
| RE done by majority, but not called RE |  | 3 | 4 |
| Reuse or no-reuse is a central question |  | 5 | 9 |
| Types of requirements |  | 0 | 0 |
| Autonomy |  | 1 | 1 |
| Hardware |  | 1 | 1 |
| Price |  | 1 | 2 |
| Simulation |  | 2 | 2 |
| Somewhat common activity - Design |  | 4 | 5 |
| Uncommon activity - Automatic code generation |  | 2 | 2 |
| Paradigms |  | 0 | 0 |
| Common paradigm - Component-based |  | 6 | 13 |
| Common paradigm - Object oriented programming |  | 4 | 4 |
| Consequent application of paradigms complicated by high turnover |  | 1 | 1 |
| No strict guidelines on paradigms, developer-dependent |  | 1 | 1 |
| Substantial awareness but little use of MBSE and SPLE |  | 5 | 7 |
| Actual MBSE use |  | 7 | 12 |
| Drawbacks |  | 0 | 0 |
| Cost - Automation not seen as investment-worthy |  | 1 | 1 |
| Risk |  | 1 | 1 |
| Tools |  | 2 | 3 |
| More common paradigm in academia - MBSE |  | 3 | 4 |
| More common paradigm in industry - SPLE |  | 1 | 1 |
| More common in automotive department |  | 1 | 1 |
| Open to it but products currently too diverse |  | 1 | 1 |
| Widespread ROS use enforces component-based paradigm |  | 5 | 8 |
| Processes |  | 0 | 0 |
| Common processes - Agile |  | 7 | 16 |
| Customers involved in development cycles |  | 1 | 2 |
| Robotic software engineering dominated by experienced engineers who experiment a lot |  | 1 | 1 |
| Scrum taken with grain of salt |  | 4 | 5 |
| Hard to estimate task duration estimation |  | 1 | 1 |
| Hard to match tasks to user stories |  | 1 | 1 |
| Scrum vs extreme programming |  | 1 | 1 |
| Small teams, frequent switching |  | 1 | 1 |
| Something like Kanban |  | 1 | 2 |
| Something like Scrum |  | 2 | 6 |
| It depends |  | 0 | 0 |
| Business department more rigorous than research dpt |  | 1 | 1 |
| Size of company matters |  | 1 | 2 |
| Waterfall for HW centric stuff |  | 2 | 3 |
| Waterfall for more traditional companies |  | 1 | 3 |
| Sometimes no clear process applied |  | 2 | 2 |
| Mixing of bottom-up and top-down |  | 1 | 4 |
| Process is money-driven |  | 1 | 1 |
| Stategic long-term planning |  | 1 | 2 |
| Quality Assurance |  | 0 | 0 |
| Identification of failure root rauses |  | 1 | 1 |
| Failure cause identification mostly by debugging and testing |  | 2 | 2 |
| QA technqiues |  | 0 | 0 |
| Common QA technique - Code reviews |  | 3 | 4 |
| Programmers mostly do code reviews |  | 0 | 0 |
| Common QA technique - Integration testing |  | 1 | 1 |
| Component-based development creates need of integration testing |  | 0 | 0 |
| Lead technical roles and academics do less testing than programmers |  | 0 | 0 |
| Lead technical roles and academics mostly do integration testing |  | 0 | 0 |
| Common QA technique - Unit testing |  | 3 | 3 |
| Continous integration |  | 2 | 2 |
| Guidelines |  | 3 | 5 |
| Less common QA technique - Performance testing |  | 1 | 1 |
| Lack of PT surprising - safety-critical domain |  | 0 | 0 |
| Reason for lack of PT - efficient hardware |  | 1 | 1 |
| Reason for lack of PT - service is simulation |  | 0 | 0 |
| Runtime monitoring |  | 1 | 1 |
| How is runtime monitoring done |  | 0 | 0 |
| Simulation |  | 0 | 0 |
| How is simulation done |  | 2 | 2 |
| Uncommon QA technique - Formal methods |  | 0 | 0 |
| Model checking |  | 1 | 2 |
| People don’t know and apply formal methods |  | 0 | 0 |
| Sources of test data |  | 0 | 0 |
| Common source of test data - Experience |  | 0 | 0 |
| Common source of test data - Runtime monitoring |  | 1 | 1 |
| Common source of test data - Simulation |  | 0 | 0 |
| RQ1.2 Technological space | What technological spaces are used? | 0 | 0 |
| Frameworks |  | 1 | 2 |
| Most common framework - ROS |  | 10 | 12 |
| Benefits |  | 0 | 0 |
| Community availability |  | 3 | 3 |
| Ease of use |  | 5 | 6 |
| Interopability with different languages |  | 1 | 1 |
| Modularity |  | 1 | 1 |
| Not tied to a company |  | 1 | 1 |
| Onboarding of new engineers |  | 2 | 2 |
| Openness |  | 4 | 5 |
| Pre-existing components |  | 8 | 9 |
| Quality can be an issue |  | 2 | 2 |
| Tooling |  | 3 | 3 |
| Understanding component behavior |  | 3 | 3 |
| Visualization |  | 2 | 3 |
| Companies create and use own ROS distributions |  | 1 | 1 |
| Drawbacks |  | 0 | 0 |
| Efficiency |  | 1 | 2 |
| Handling of communication |  | 2 | 4 |
| Interfaces and contracts not well-exposed |  | 1 | 1 |
| Lack of documentation |  | 1 | 1 |
| Lack of life-cycle control (ROS1) |  | 2 | 3 |
| Lack of real-time support |  | 5 | 8 |
| Lack of robustness |  | 2 | 2 |
| Lack of standard |  | 1 | 1 |
| Need to understand third-party code |  | 2 | 3 |
| Not applicable for company's mission |  | 1 | 3 |
| Rapid version changes |  | 1 | 1 |
| Security |  | 3 | 3 |
| Steep learning curve |  | 2 | 3 |
| History of ROS |  | 1 | 1 |
| Industrial engineers contribute to ROS |  | 3 | 3 |
| Companies are incentivized with EU money |  | 1 | 2 |
| ROS used in pre-development but not actual products |  | 1 | 1 |
| Widespread use of ROS also thanks to social aspects |  | 1 | 1 |
| OROCOS used by small community sector due to real-time support |  | 1 | 1 |
| Other |  | 0 | 0 |
| Autosar |  | 1 | 1 |
| CMQ |  | 1 | 1 |
| FMI |  | 1 | 1 |
| Google Protobufs |  | 1 | 1 |
| Oberon (confidentiality requested) |  | 1 | 2 |
| Xamarin |  | 1 | 1 |
| ROS2 not yet widely used |  | 2 | 2 |
| Fixes issues of ROS1 |  | 2 | 3 |
| Most ROS2 users are those related to the ROS community |  | 0 | 0 |
| Never change a running system |  | 1 | 2 |
| People consider moving to ROS2 |  | 2 | 2 |
| People wait for ROS2 to become stable |  | 2 | 2 |
| Steeper learning curve |  | 1 | 1 |
| Languages |  | 0 | 0 |
| Common language - C++ |  | 6 | 6 |
| C++ allows to protect code |  | 1 | 1 |
| C++ is performance-efficient |  | 3 | 3 |
| C++ is structured |  | 1 | 1 |
| C++ used as implementation language in MBSE |  | 2 | 2 |
| C++ used for low-level components with hard time constraints |  | 1 | 1 |
| Works with ROS |  | 2 | 2 |
| Common language - Python |  | 4 | 4 |
| Machine learning components |  | 1 | 1 |
| Python used for high-level components without hard time constraints |  | 0 | 0 |
| Python used for prototyping since it is simpler than C++ |  | 2 | 3 |
| Python used for scripting |  | 1 | 1 |
| Works with ROS |  | 2 | 2 |
| Uncommon language - C |  | 4 | 5 |
| Uncommon language - DSLs |  | 4 | 5 |
| Behavior trees |  | 3 | 5 |
| Mission specification DSL |  | 1 | 1 |
| State machines |  | 1 | 1 |
| Uncommon language - Java |  | 0 | 0 |
| Uncommon language - MATLAB-Simulink |  | 1 | 2 |
| Uncommon language - UML |  | 3 | 3 |
| Uncommon languages - C# |  | 1 | 1 |
| Uncommon languages - JavaScript |  | 1 | 1 |
| Tools |  | 0 | 0 |
| Architectural and detailed design |  | 0 | 0 |
| Papyrus |  | 1 | 3 |
| Word |  | 1 | 1 |
| Automatic code generation |  | 0 | 0 |
| Kinematics - Modelica - Dymola - Matlab Simulink |  | 1 | 1 |
| Deployment |  | 0 | 0 |
| Self-developed |  | 1 | 2 |
| Implementation |  | 0 | 0 |
| IDEs |  | 2 | 2 |
| Developer's choice |  | 4 | 4 |
| Eclipse |  | 2 | 2 |
| QT-Creator |  | 4 | 5 |
| Integrated with ROS |  | 1 | 1 |
| Visual Studio |  | 2 | 2 |
| Text editors |  | 1 | 1 |
| Project management |  | 1 | 1 |
| Excel |  | 1 | 1 |
| Jira |  | 1 | 1 |
| Real-world experimentation |  | 0 | 0 |
| Requirements engineering |  | 0 | 0 |
| Papyrus |  | 1 | 1 |
| PowerPoint |  | 1 | 1 |
| Text editors |  | 1 | 1 |
| Trello |  | 1 | 2 |
| Software maintenance and evolution |  | 0 | 0 |
| GitHub |  | 1 | 1 |
| GitLab |  | 1 | 1 |
| Jenkins |  | 1 | 1 |
| Testing and simulation |  | 1 | 1 |
| Eclipse |  | 1 | 1 |
| Gazebo |  | 2 | 2 |
| ROS bag |  | 1 | 1 |
| rviz |  | 2 | 3 |
| Unity |  | 1 | 1 |
| Visualization |  | 0 | 0 |
| Self-developed |  | 1 | 1 |
| RQ1.3 OS, Reuse, Libraries | What is the role of open source resources and reusable libraries? | 0 | 0 |
| Interoperability |  | 0 | 0 |
| Interoperability mostly handled by relying on experience of team |  | 2 | 2 |
| Interoperability rarely handled by following a precise architecture or a standard |  | 0 | 0 |
| ROS infrastructure is most common standard |  | 0 | 0 |
| Is ROS actually a standard - Controversial |  | 1 | 1 |
| Libraries |  | 0 | 0 |
| Almost everyone uses OS libraries |  | 7 | 7 |
| Enthusiam for OS |  | 2 | 3 |
| Vision as major usecase for libraries |  | 1 | 1 |
| Industrial engineers contribute to OS |  | 2 | 2 |
| Companies are incentivized with EU funding |  | 1 | 2 |
| OS is complicated from business perspective |  | 1 | 4 |
| Self-criticism for lack of OS contributions |  | 1 | 2 |
| Library selection is mix of experience and google |  | 1 | 1 |
| Proprietary libraries sometimes used |  | 0 | 0 |
| Release models - OS and proprietary both common |  | 3 | 3 |
| What we sell is a service |  | 1 | 1 |
| Reuse types |  | 0 | 0 |
| Copy-paste-modify reuse is not most common |  | 0 | 0 |
| But people still do it |  | 3 | 3 |
| Systematic reuse is most common |  | 1 | 1 |
| Configuration processes |  | 2 | 7 |
| Developers make efforts towards systematic reuse |  | 3 | 9 |
| Well-designed mix of reuse and non-reuse |  | 1 | 1 |
| Reused artifacts |  | 0 | 0 |
| Code, libraries, and components often reused |  | 1 | 1 |
| Programmers more aware of issue of documentation reuse |  | 1 | 1 |
| Test cases rarely reused |  | 0 | 0 |
| Roadblocks to reuse |  | 0 | 0 |
| Common reuse problem - lack of documentation |  | 1 | 2 |
| Common reuse problem - licensing issues |  | 1 | 1 |
| Common reuse problem - safety certification |  | 1 | 1 |
| Common reuse problem - technical issues |  | 1 | 1 |
| Some people prefer self-developed solutions |  | 0 | 0 |
| RQ2 Differences to other SE domains | How is SE in mobile service robotics different from SE in other domains? | 0 | 0 |
| Doesn't really differ |  | 2 | 3 |
| 'Not invented here' syndrome |  | 1 | 2 |
| Some problems from robotics now manifest in other domains |  | 1 | 2 |
| Heterogenous components and systems |  | 3 | 4 |
| Mission specification |  | 1 | 1 |
| Building a map |  | 2 | 2 |
| Custom vs generic parts |  | 2 | 3 |
| Methods |  | 0 | 0 |
| API-based |  | 1 | 2 |
| Behavior trees |  | 4 | 8 |
| DSLs |  | 2 | 4 |
| Formal methods |  | 0 | 0 |
| GUI |  | 7 | 11 |
| Hard-coded |  | 3 | 3 |
| Hard-coded with parameters ('press a button') |  | 3 | 6 |
| Hybrid methods |  | 1 | 12 |
| Probabilistic methods |  | 1 | 3 |
| State machines |  | 1 | 2 |
| Roles |  | 1 | 3 |
| Non-functional concerns |  | 0 | 0 |
| Realtime support |  | 3 | 3 |
| Resource efficiency |  | 1 | 2 |
| Safety |  | 5 | 10 |
| Bigger concern in industrial robotics, vs. service robotics |  | 1 | 1 |
| Even a shutdown is tricky |  | 2 | 2 |
| Role of academia & research |  | 2 | 2 |
| Commercial robotics is research-oriented |  | 1 | 2 |
| Commercial robotics is unlike research |  | 1 | 1 |
| Role of hardware |  | 0 | 0 |
| Companies do integrated HW and SW design |  | 1 | 1 |
| Hardware can behave unexpectly & need calibration |  | 3 | 3 |
| Hardware is the bottleneck for the system's capabilities |  | 3 | 6 |
| HW variability |  | 2 | 2 |
| New HW developments can enable new features |  | 1 | 1 |
| Unreliable HW makes debugging tough |  | 1 | 1 |
| Theoretical background and vocabulary |  | 3 | 3 |
| Uncertain environments |  | 1 | 2 |
| Classical debugging tools not applicable |  | 2 | 2 |
| Dynamic adaptation |  | 2 | 2 |
| Interaction with people |  | 1 | 2 |
| Interaction with the physical world |  | 6 | 9 |
| Noisy sensors |  | 4 | 4 |
| Reliability is key |  | 2 | 3 |
| The environment changes a lot |  | 2 | 2 |
| Variability of the physical world |  | 1 | 1 |
| RQ3 Challenges | What challenges do practitioners face when engineering mobile service robots? | 0 | 0 |
| Additional challenges |  | 0 | 0 |
| Access to hardware |  | 1 | 2 |
| Political factors |  | 1 | 1 |
| Adoption and accessibility of solutions |  | 2 | 4 |
| Reinventing the wheel |  | 1 | 3 |
| Architecture |  | 4 | 5 |
| Debugging |  | 4 | 6 |
| Distributed development |  | 1 | 1 |
| Doing things faster and cheaper |  | 2 | 2 |
| Engineers are expensive |  | 1 | 3 |
| Expectation management (customer-facing) |  | 1 | 1 |
| Focus on tech instead of use cases |  | 1 | 2 |
| Hardware, drivers |  | 4 | 6 |
| Manual tasks |  | 1 | 1 |
| Quality, lack of proper abstractions |  | 1 | 4 |
| Memory |  | 1 | 2 |
| Safety vs. performance trade-off |  | 1 | 1 |
| Security |  | 1 | 1 |
| Setup, installation (a.k.a. localization) |  | 5 | 8 |
| Simulation |  | 2 | 5 |
| Technical debt |  | 2 | 4 |
| Time pressure |  | 1 | 1 |
| UI development |  | 2 | 4 |
| Visual feedback |  | 1 | 2 |
| Error-prone activities |  | 0 | 0 |
| Behavior design |  | 1 | 1 |
| Testing |  | 1 | 2 |
| Time-consuming activities |  | 0 | 0 |
| Getting something physical up and running |  | 1 | 1 |
| Testing |  | 1 | 1 |
| Top 10 |  | 0 | 0 |
| Top 1 Challenge - Robustness |  | 5 | 13 |
| Failure handling for real-world situations |  | 5 | 7 |
| Simulation vs. real world |  | 8 | 12 |
| Top 10 Challenge - AI |  | 4 | 5 |
| Applications |  | 0 | 0 |
| Arm movement |  | 1 | 1 |
| Automating assembly steps |  | 1 | 1 |
| Emotion detection - confidentiality requested |  | 1 | 1 |
| Learning complex actions |  | 1 | 1 |
| Obstacle detection, object detection |  | 5 | 5 |
| Signal and image processing |  | 1 | 1 |
| Problems |  | 0 | 0 |
| Adequate datasets needed |  | 1 | 1 |
| Balancing autonomy |  | 1 | 1 |
| Big datasets needed |  | 1 | 2 |
| Top 2 Challenge - Validation (a.k.a. Testing) |  | 7 | 13 |
| Need to involve various HW devices |  | 1 | 2 |
| Top 3 Challenge - Lack of documentation |  | 3 | 8 |
| Top 4 Challenge - Dynamic adaptation |  | 9 | 12 |
| Top 5 Challenge - Interaction of heterogenous components |  | 4 | 5 |
| Top 6 Challenge - Safety certification |  | 3 | 4 |
| In the age of machine learning |  | 2 | 2 |
| Top 7 Challenge - Lack of standards |  | 2 | 6 |
| Top 8 Challenge - Software reuse |  | 3 | 4 |
| Top 9 Challenge - Mission Specification |  | 1 | 1 |