Orca: A Component Model and Repository

Alex Brooks\(^1\), Tobias Kaupp\(^2\), Alexei Makarenko\(^3\), Stefan Williams\(^4\), and Anders Oreback\(^5\)

\(^1\) ACFR, University of Sydney, AUSTRALIA a.brooks@acfr.usyd.edu.au
\(^2\) ACFR, University of Sydney, AUSTRALIA t.kaupp@acfr.usyd.edu.au
\(^3\) ACFR, University of Sydney, AUSTRALIA a.makarenko@acfr.usyd.edu.au
\(^4\) ACFR, University of Sydney, AUSTRALIA s.williams@acfr.usyd.edu.au
\(^5\) Royal Institute of Technology, SWEDEN oreback@nada.kth.se

Summary. This Chapter describes Orca: an open-source project which applies Component-Based Software Engineering principles to robotics. It provides the means for defining and implementing interfaces such that components developed independently are likely to be inter-operable. In addition it provides a repository of free re-useable components. Orca attempts to be widely applicable by imposing minimal design constraints. This Chapter describes lessons learned while using Orca and steps taken to improve the framework based on those lessons. Improvements revolve around middleware issues and the problems encountered while scaling to larger distributed systems. Results are presented from systems that were implemented.

1 Introduction

This Chapter describes Orca: an open-source project which applies Component-Based Software Engineering (CBSE) principles to robotics. Many robotic software projects are presented in this book, however the Orca project stands out by both

- explicitly adopting a component-based approach from the outset, and
- acknowledging the importance of a component market.

While it is not alone in either respect, the Orca project is, to the authors’ knowledge, the only framework to combine the two.

The introduction to this Part defined Component-Based Software Engineering, identified its benefits and motivated its application to robotics. The benefits of a component-based approach include

- the ability to build systems quickly by incorporating existing third-party components,
- the ability to build more reliable systems by incorporating components which have been tested across multiple projects,
• the software engineering benefits of having a modular system with controlled, explicit dependencies only, and
• the ability to build flexible systems in which individual components can be replaced.

In non-commercial settings, component markets entail the trading of independently-developed components. The recognition of the importance of this has both a practical and a technical impact. Practically, it drives the maintenance of the Orca component repository\(^6\): an online selection of open-source, interoperable, re-useable robotic components. Significant effort has been devoted to providing sufficient documentation for both the framework itself and for each component, to ensure that components can indeed be deployed by third parties. Technically, the recognition of the importance of component markets drives Orca to make as few assumptions as possible about the systems to which it may be applied, in order to be applicable across as broad a market as possible. Section 2 develops these ideas further, presenting the basic philosophy behind the Orca approach.

The two main principles have remained unchanged since the start of the project in 2001 (under the name of Orocos@KTH). Some implementation details have undergone radical changes, warranting a recent increment of the version number from Orca1 [BKM05] to Orca2. A number of lessons were learned based on experiences with Orca1, resulting in improvements that were made or planned for Orca2. These revolve around middleware-related issues and the problems discovered when scaling to large distributed systems, as discussed in Sections 3 and 4 respectively.

To demonstrate the effectiveness of the framework, Section 5 describes some of the component-based systems that have been built and the re-use that has occurred between those systems. Section 6 compares Orca with the Player project. The comparison is included because Player is the only robotics software project to have established a significant market, and hence has become a de-facto standard in robotics. Finally, Section 7 concludes.

2 Design Philosophy

2.1 Design Minimalism: Impose as Few Constraints as Possible

The design of the Orca framework is conceptually simple. A system consists of a set of components which run asynchronously, communicating with one another over a set of well-defined interfaces. Each component has a set of interfaces it provides and a set of interfaces it requires. The fundamental purpose of the framework is to provide the means for defining and implementing these interfaces. Standardising the definitions and implementations of interfaces ensures that components are likely to be inter-operable, and hence re-useable.

\(^6\) available from http://orca-robotics.sourceforge.net