ARC Centre of Excellence for

Autonomous Systems

Orca: Components for Robotics

Alexei Makarenko,
Alex Brooks,
Tobias Kaupp

ARC Centre of Excellence for Autonomous Systems
The University of Sydney

Uni. of Sydney: where we are coming from

- **Distributed applications**
  - Sensor networks, decentralized information fusion
  - Naturally leads to components
- **Industry funded projects**
  - Need for robust implementations
- **Complexity of robotic software**
  - Robotic software – the bottleneck of what is possible today (not computing hardware, sensors, algorithms)
  - “The PhD problem”

Component-Based Software Engineering (CBSE)

- **Advantages**
  - Modularization: “collection of small problems is easier to solve than one big problem”
  - Minimizing source-code cross-dependencies
  - Maximizing software reuse
  - Parallel development by large distributed teams

Component Model

- **Fundamentals**
  - Interface definition
  - Communication mechanism
- **Options**
  - .NET (Microsoft)
  - Enterprise Java Beans (Sun)
  - CORBA (OMG and implementers)
  - Ice (ZeroC)
  - Custom (e.g. Player, Orca-1)

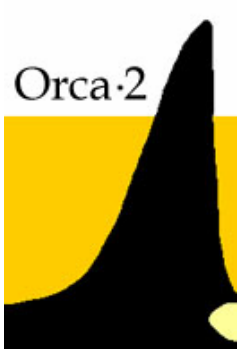
Real Choices for Robotics

- **Middleware: Off-the-shelf vs. Custom**
  - Player uses custom and is very successful
  - We believe there is a limit to what can be done with custom middleware (reliability, features, performance)
- **Off-the-shelf Middleware: CORBA vs. Ice**
  - The concept is similar if not identical
  - For comparison of design, features, performance, etc. see ZeroC.
  - We've used Ace/Tao and found it difficult
  - We find Ice much more straightforward

Ice – Internet Communication Engine

- **Background**
  - Company: ZeroC, since 2001
  - http://www.zeroc.com
  - Open-source, proprietary
- **Support**
  - OS: Linux, Win, MacOS X, Solaris
  - Language: C++, Java, C#, Visual Basic, Python, PHP
  - Transport: TCP/IP, UDP/IP, SSL
- **Licensing**
  - GPL for open-source projects
  - LGPL for Orca
  - Commercial license otherwise

Orca Project: Components for Robotics

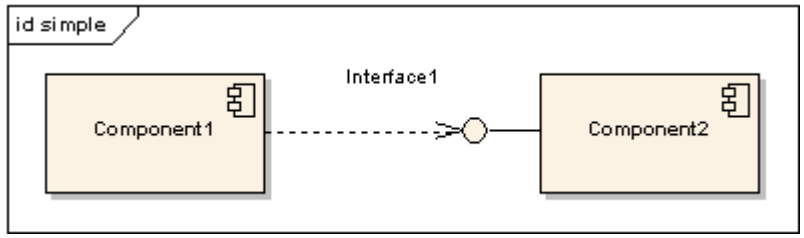


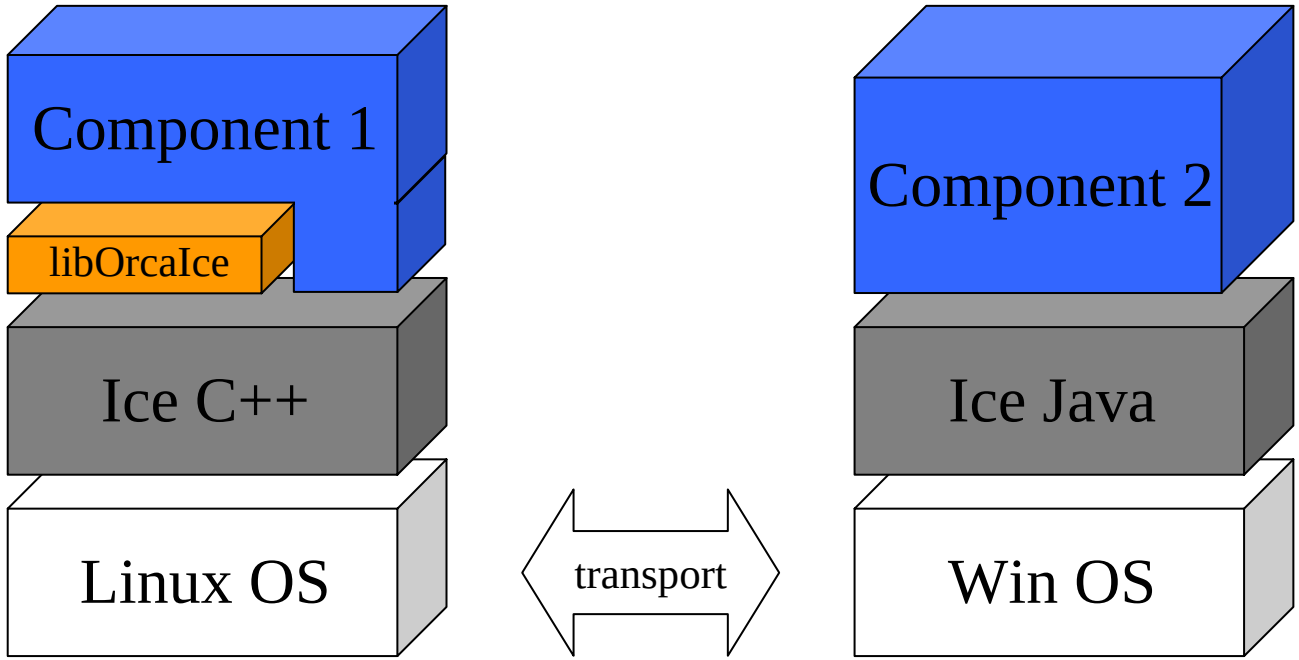
- **Main objective: Software Reuse**
  - *Enable*: by defining a set commonly-used interfaces
  - *Simplify*: by providing libraries with a high-level convenient API
  - *Encourage*: by maintaining a repository of components
- **What makes it different?**
  - adopts a Component-Based Software Engineering approach
  - uses an industrial-strength library (Ice) for communication and interface definition
  - aims to be general, flexible and extensible
  - provides optional tools to assist in the development of individual components and the management of large systems
  - maintains a repository of free, reusable components

“An object-oriented middleware platform”

- **Developer**
  - Defines interfaces in an Interface Definition Language (IDL)
    - A bunch of remote procedure calls
  - Writes server application
    - Implements RPC’s
  - Writes client application
    - Calls RPC’s
- **Ice**
  - Takes care of communication, serialization, etc.

Software Architecture





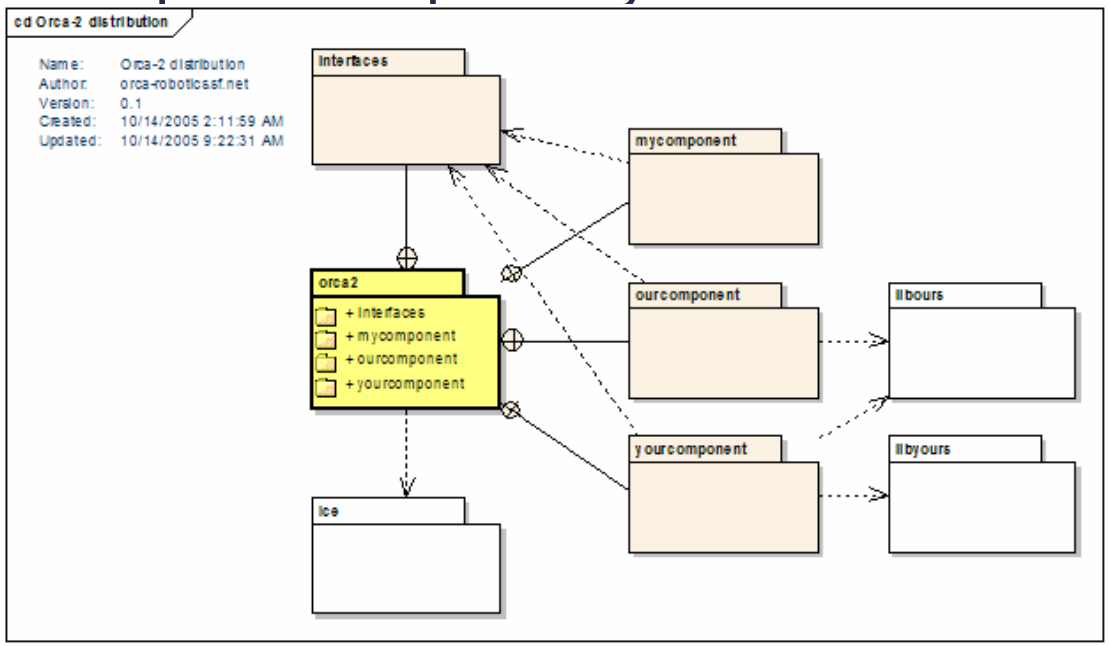
e.g. TCP/IP

Ice Services

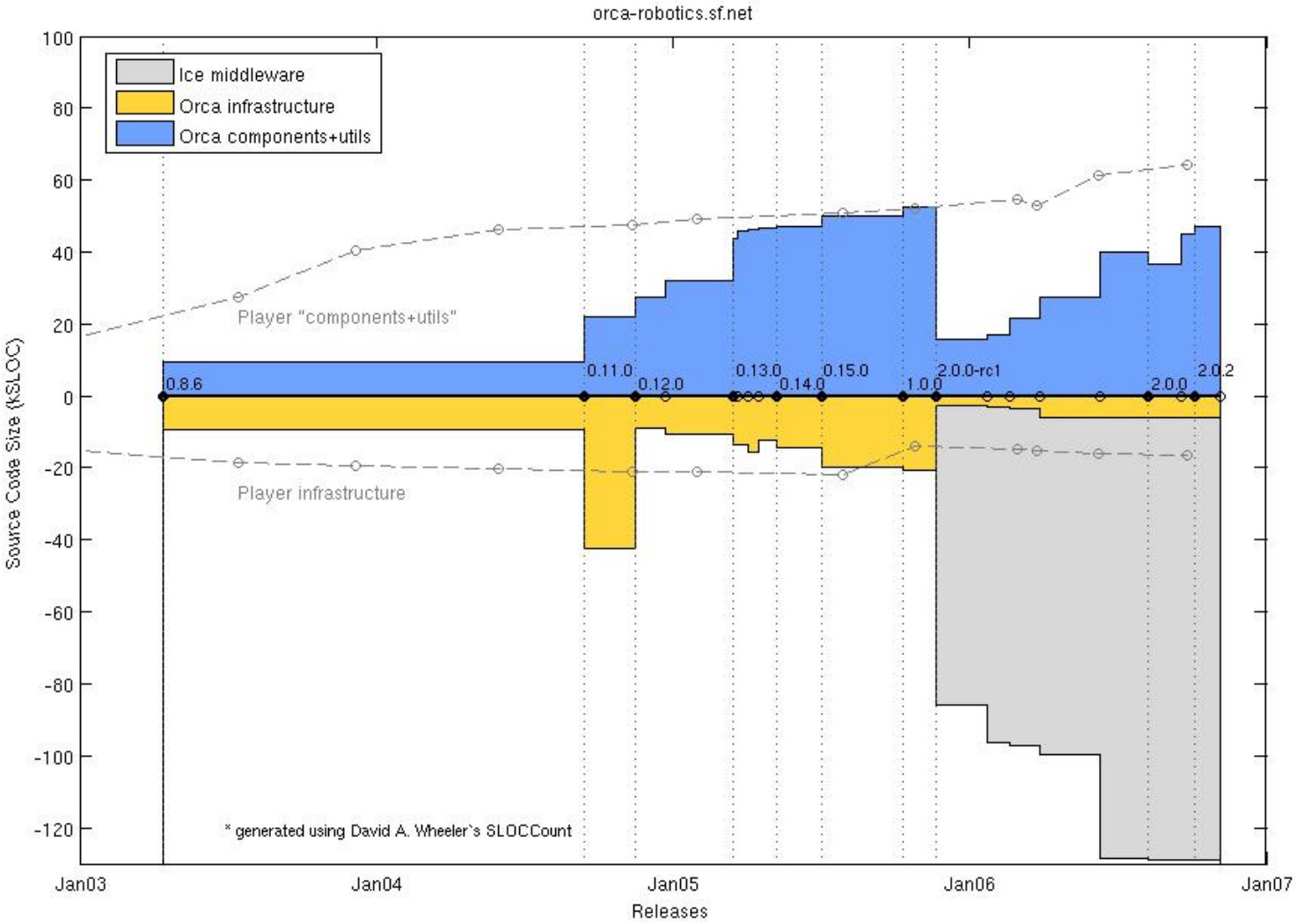
- **IceGrid – grid computing**
  - Registry (centralized)
  - On-demand activation, reactivation
  - Replication, etc.
- **IceBox – application server**
  - Internal comms are optimized to a function call (optional)
- **IceStorm – event service (an IceBox service)**
  - Publish/subscribe of any interface
  - Federation
- **IcePatch2 – file distribution**
- **Freeze – persistence service**
- **Glacier2 – firewall**
- **Ice-E – “embedded Ice”**
  - Light-weight version for PDA’s.

Orca Project

- **Hosted on SourceForge**
- **Distribution includes**
  - Common interface definitions (Slice IDL)
  - Utility libraries
  - Component repository



Project History and Size



Interface Definitions

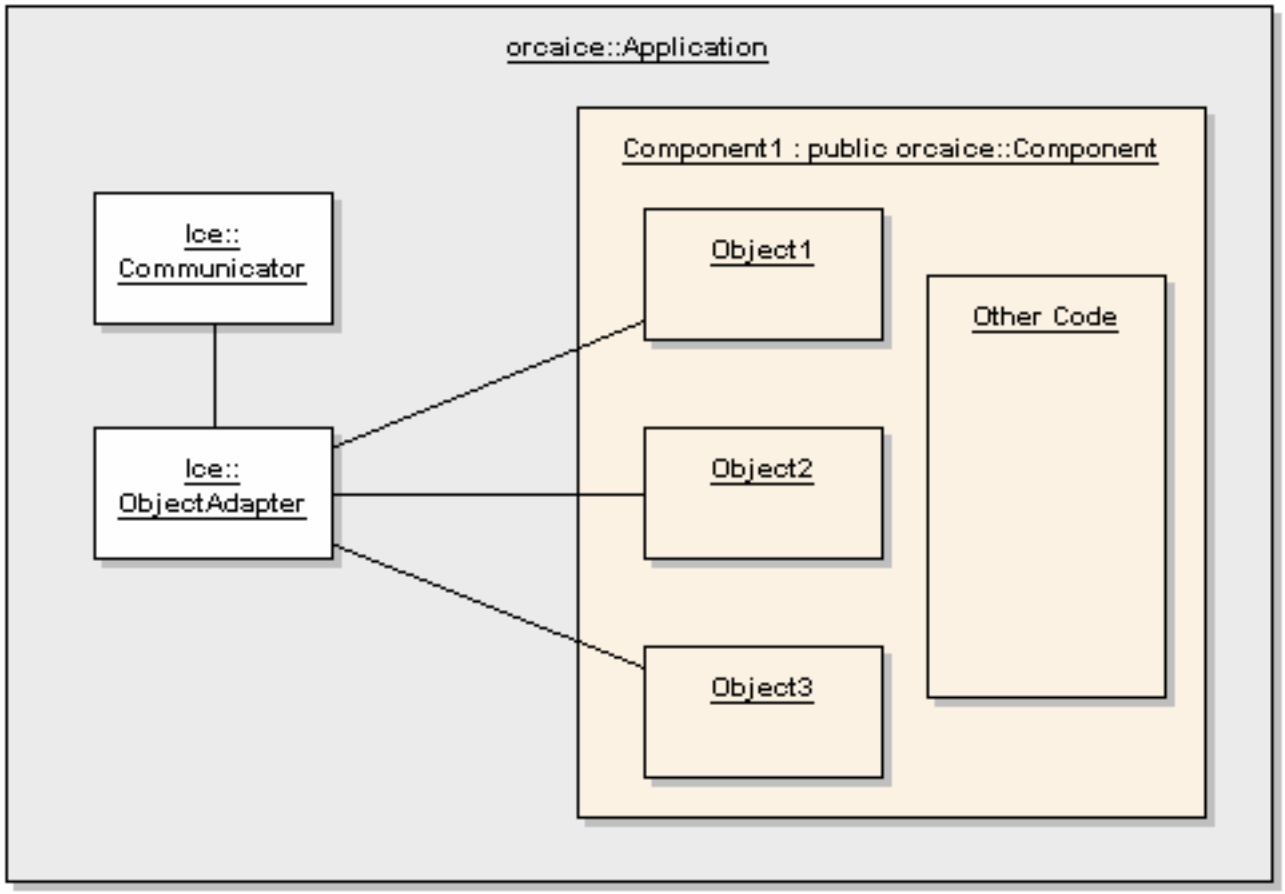
- **Slice: an Interface Definition Language (IDL)**

```
// Slice

class Position2dData extends OrcaObject
{
  Frame2d pose;
  Twist2d motion;
};


interface Position2d
{
  nonmutating Position2dData getData()
            throws DataNotExistException, HardwareFailedException;

  void subscribe( Position2dConsumer* subscriber )
            throws SubscriptionFailedException;

  idempotent void unsubscribe( Position2dConsumer* subscriber );
};
```

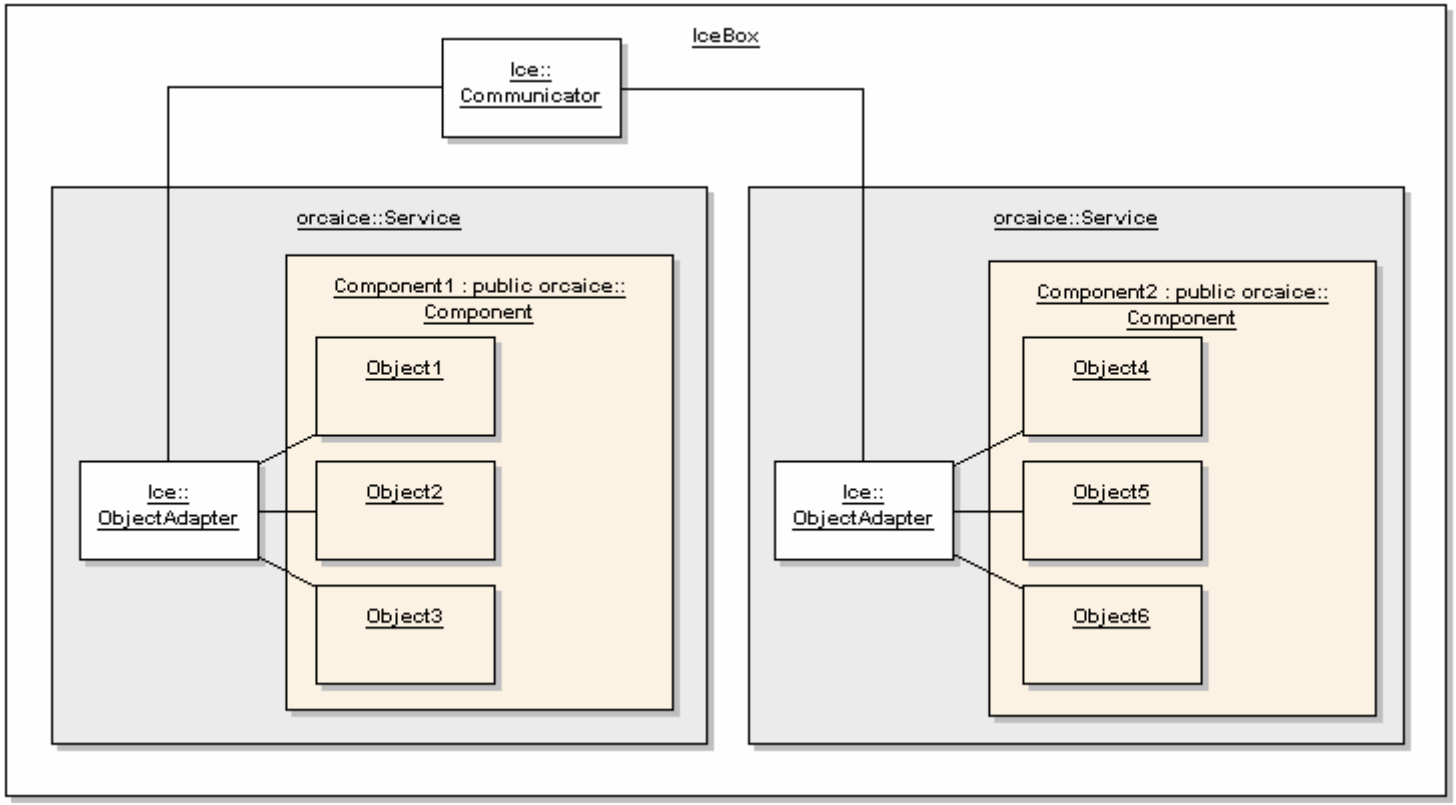
libOrcaIce

- **Aims**
  - To set naming conventions
  - To simplify component development
- **Provides**
  - Classes to derive from
  - Classes to use
  - Helper functions
- **Strictly optional**
  - Can choose not to use it and call **libIce** directly

Stand-Alone Component



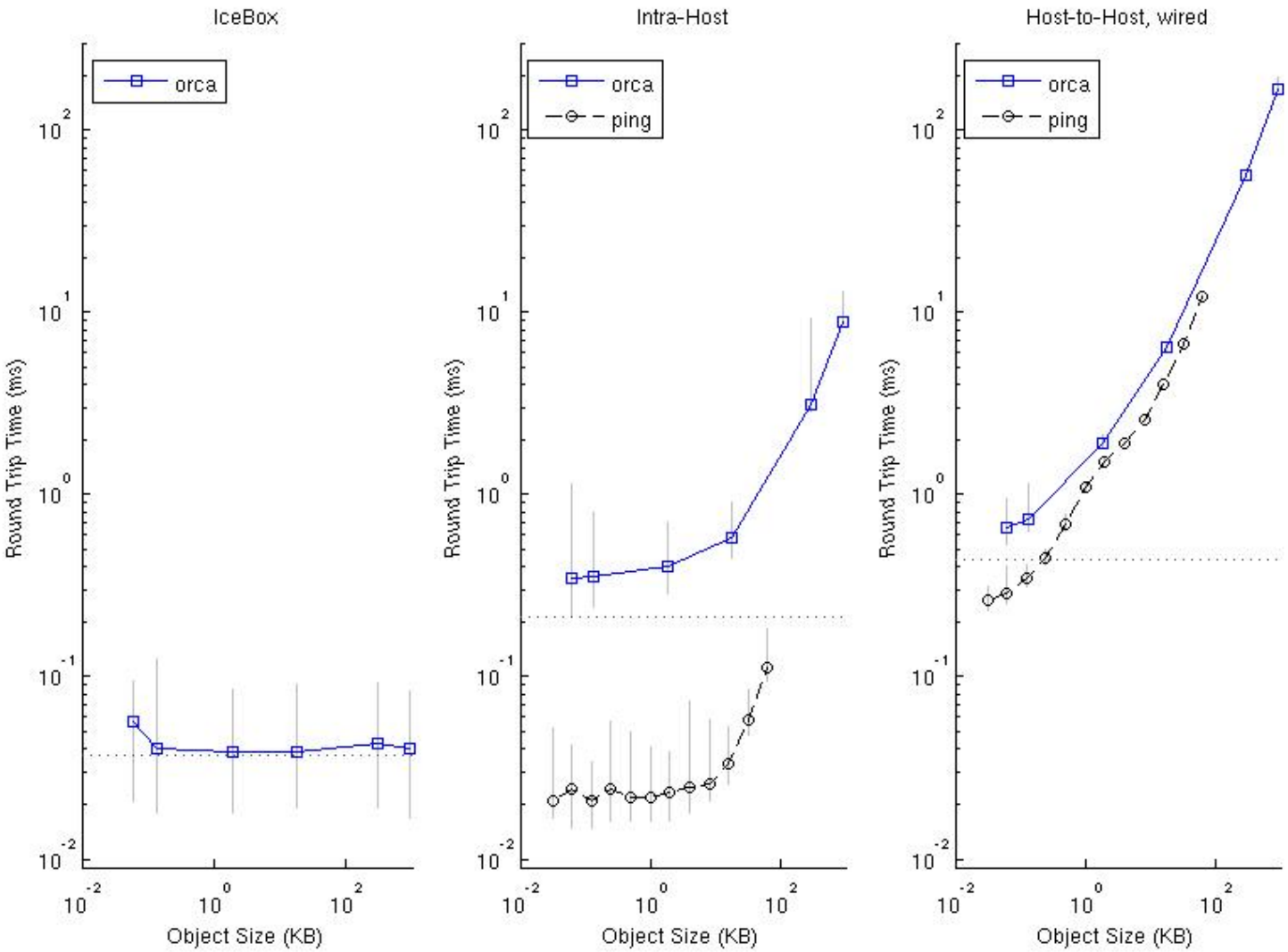
Component as an IceBox Service



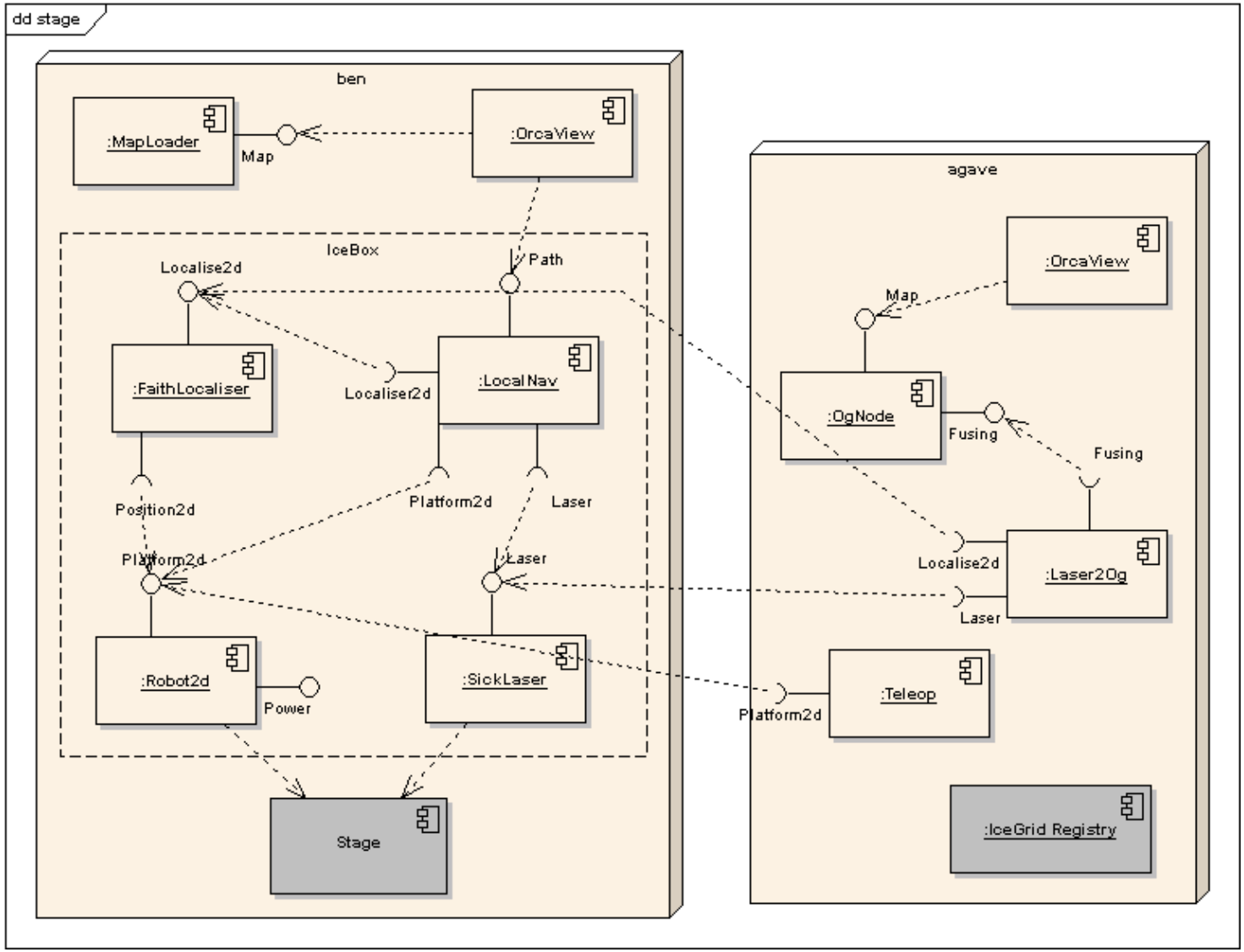
Component Repository

- FaithLocaliser
- FeatureMapLoader
- LaserFeatureExtractor
- LocalNav
- Logger
- LogPlayer
- OgMapLoader
- OrcaView
- RegistryView
- SegwayRmp
- SickLaser
- SimLocaliser
- Teleop

Performance



Example: Simple 2-D Demo with Stage



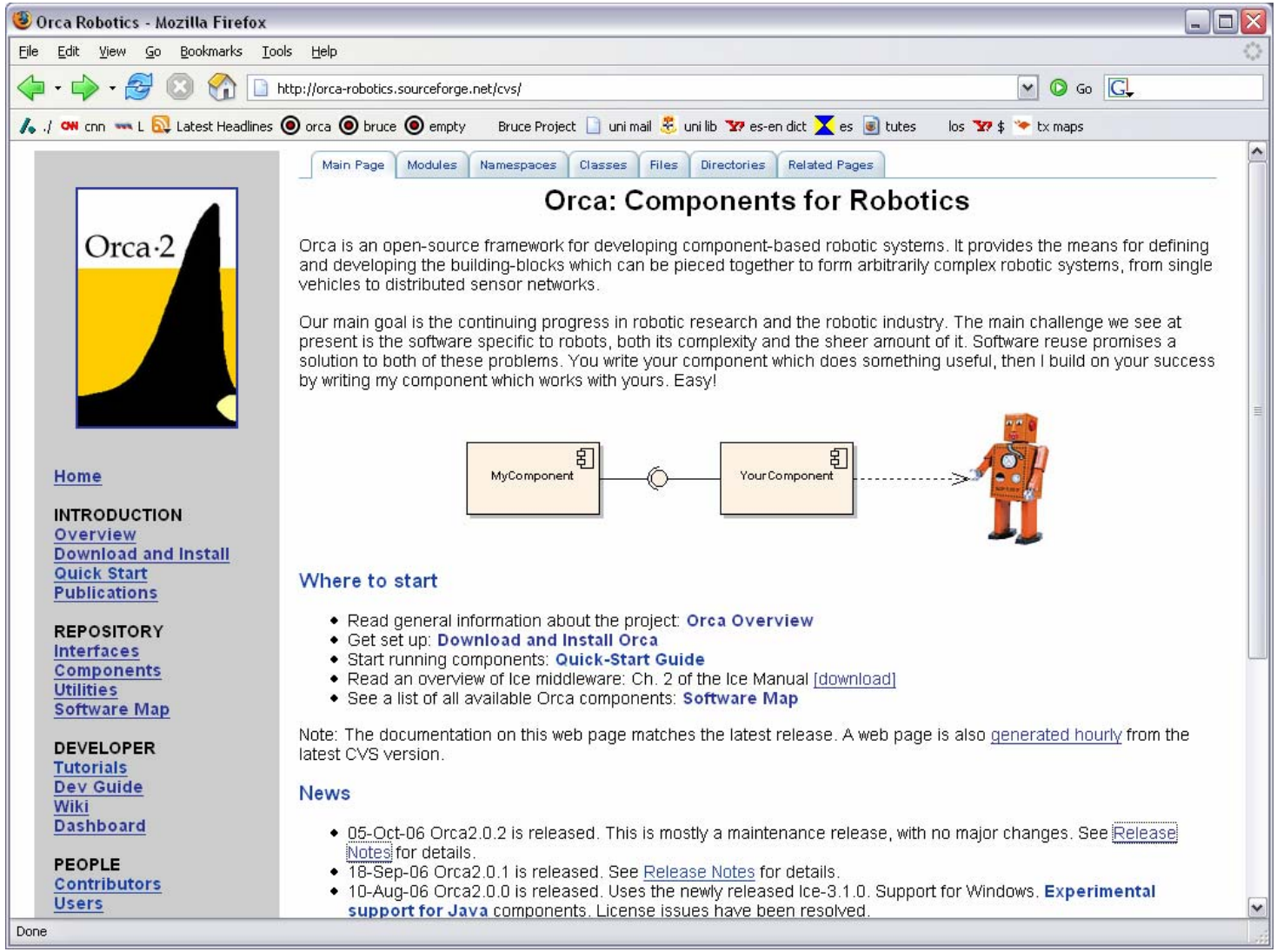
Current work : ACFR Segway Project

Current work: ACFR/Berkeley Team for 2007 Urban Grand Challenge

Potential Issues to Be Aware of

- **Developer skills**
  - Harder than monolithic applications
  - Requires certain discipline in
    - Structuring the project
    - Writing individual components
  - But … for large projects, CBSE is a solution not a problem
- **Testing**
  - Harder than monolithic applications
  - But … using a middleware package removes many testing issues.

http://orca-robotics.sf.net



ARC Centre of Excellence for

Autonomous Systems