

Software Components and Frameworks for Medical Robot Control

Ankur Kapoor, Anton Deguet and Peter Kazanzides



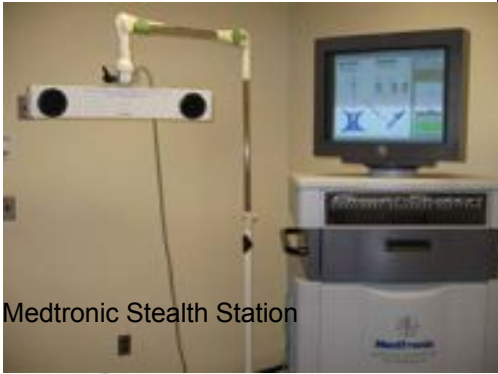
*Department of Computer Science
The Johns Hopkins University*



Context: Surgical Robots and Tracking Devices



Eye Robot



Medtronic Stealth Station



DaVinci



Steady Hand Robot



4mm Snake-Like Robot for Throat



SARRP

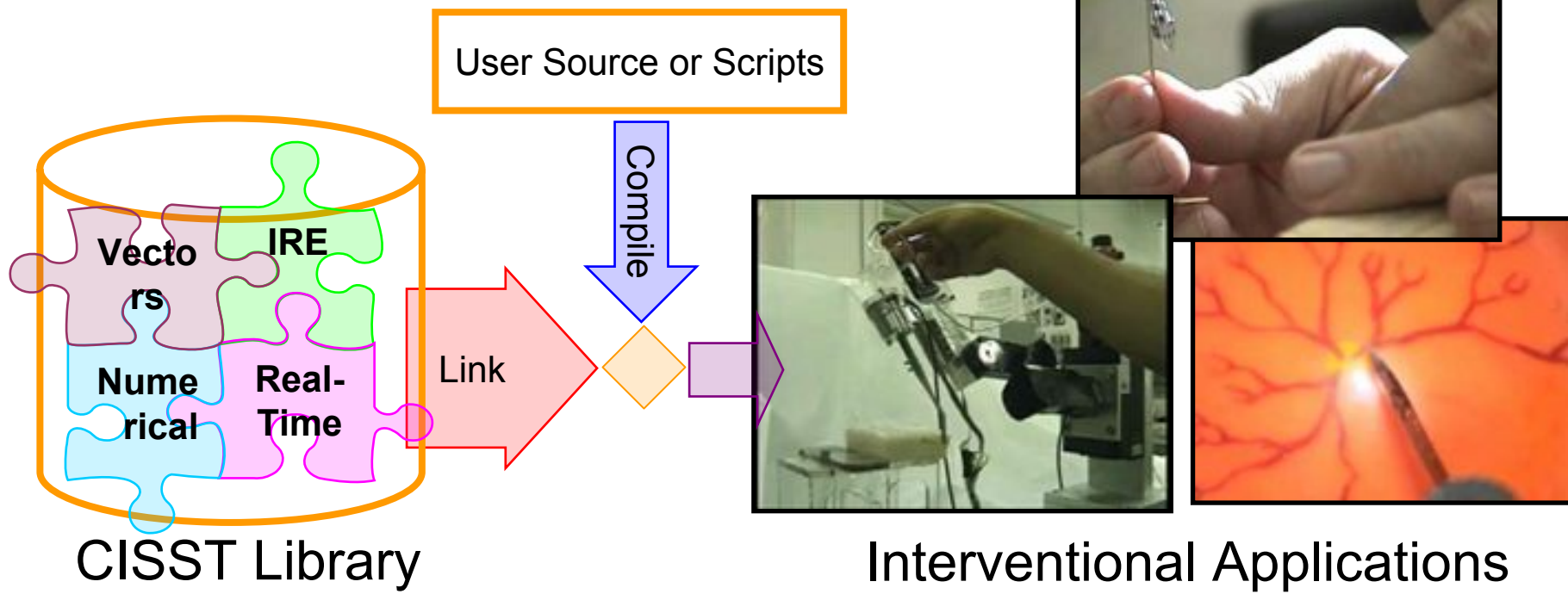


NDI Polaris



Motivation and Goals

- **Open Source Software for CIS use that can be certified for clinical evaluation**
- **Support devices other than robots, with different features**
- **Interchangeability of features provided in hardware (*devices*) and those in software (*tasks*)**



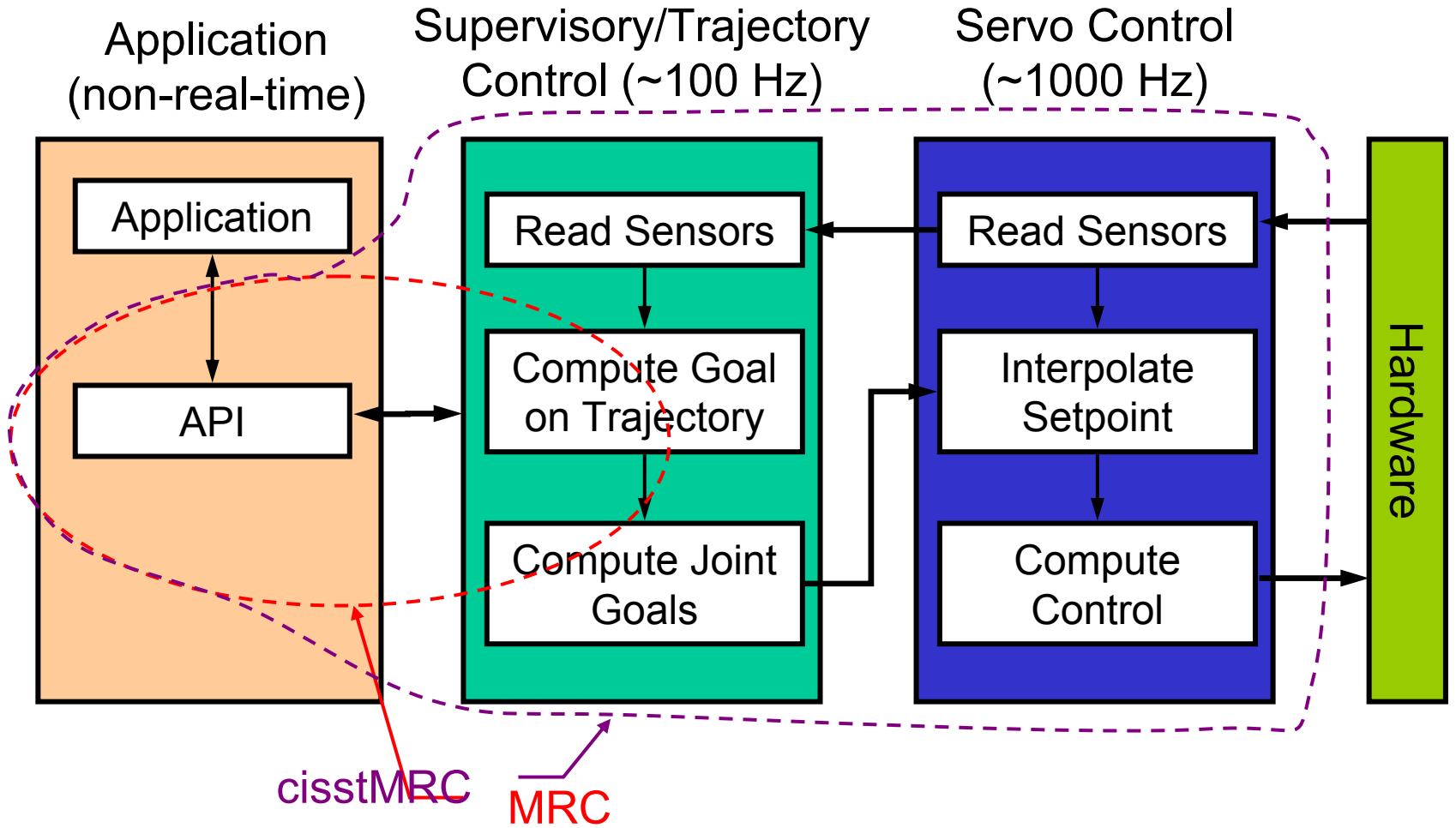


Related Works

- Related work in robot controller software
 - OROCOS
 - Not designed for COTS hardware/software and is consequently not supported on the Windows operating system.
 - ORCA & MCA2
 - framework of binary components designed for mobile robots
 - Peters et al, IROS 2005
- Previous work at ERC, Johns Hopkins University
 - *CIS Library*
 - Interface to [tracking systems](#)
 - Basic tools for CIS such as [registration](#), [numerical methods](#)
 - *MRC Library*
 - Common interface to different [robots](#) (Kinematics etc)
 - API to “wrapper” low-level control

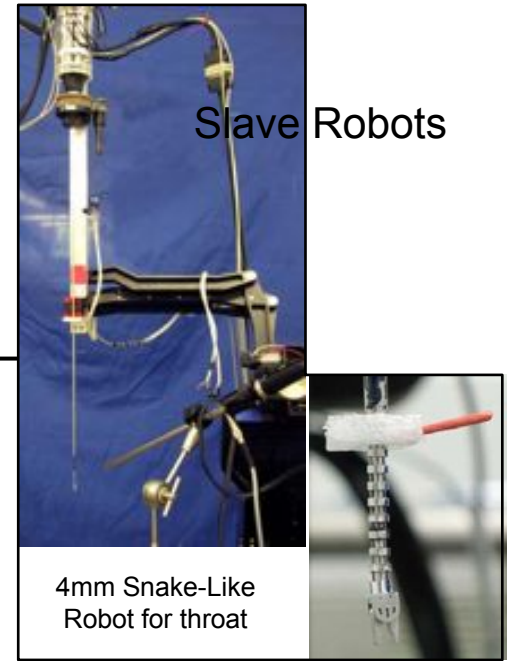
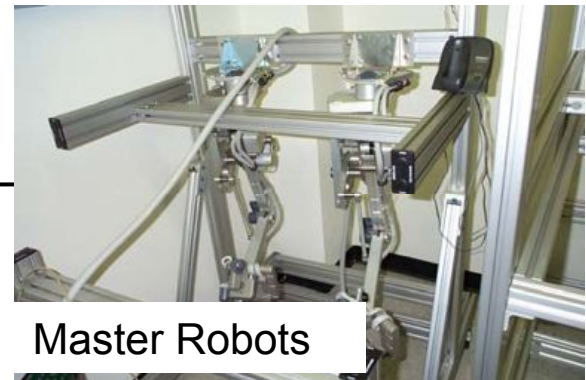
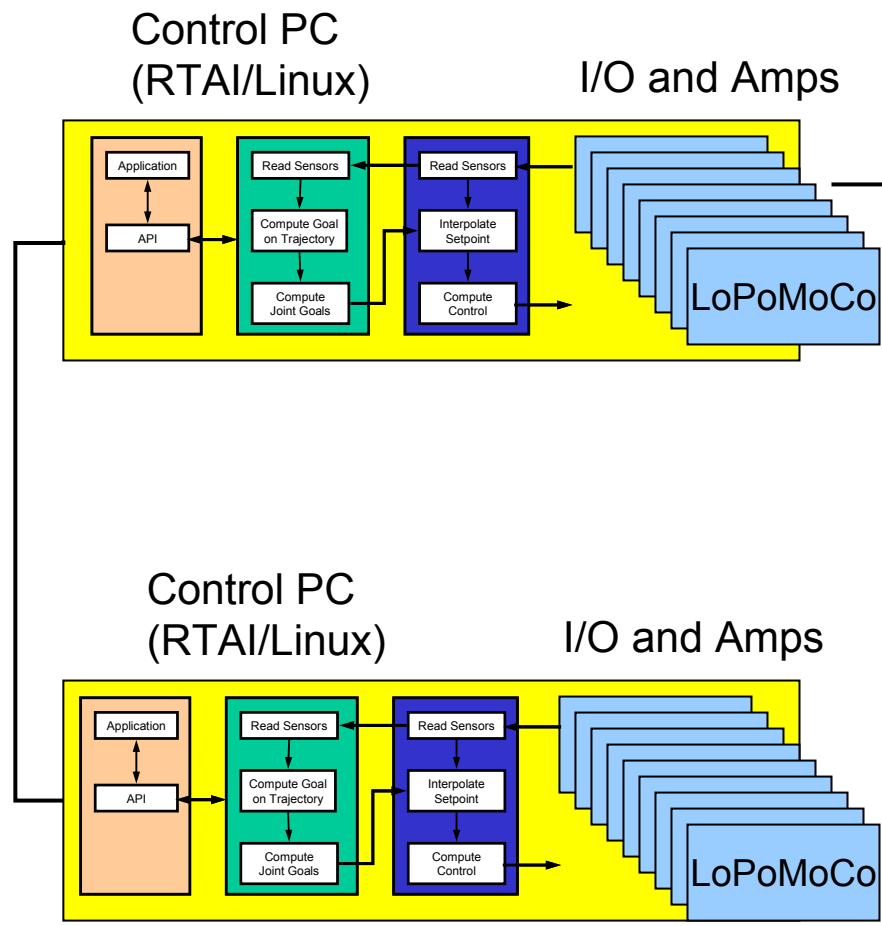


Robot Controller Architecture

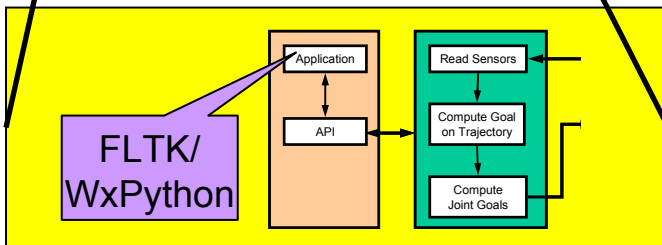
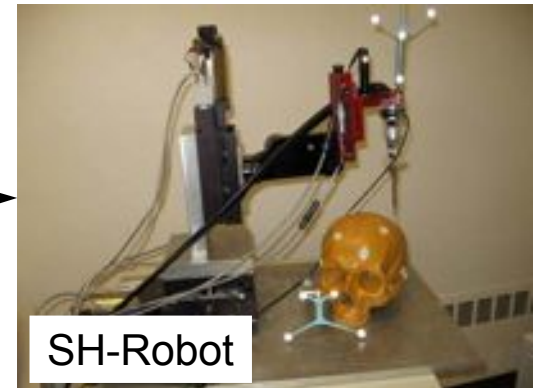
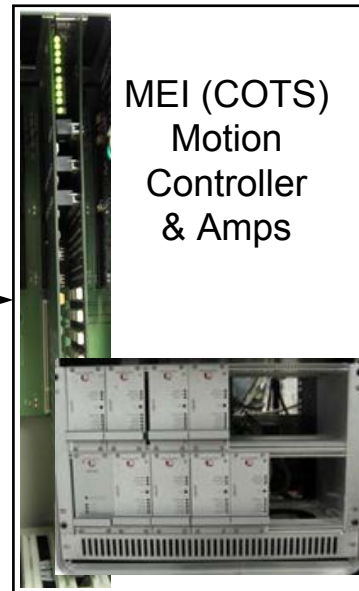




Example: Teleoperation of Snake Robot



Example: Neuro-Robot



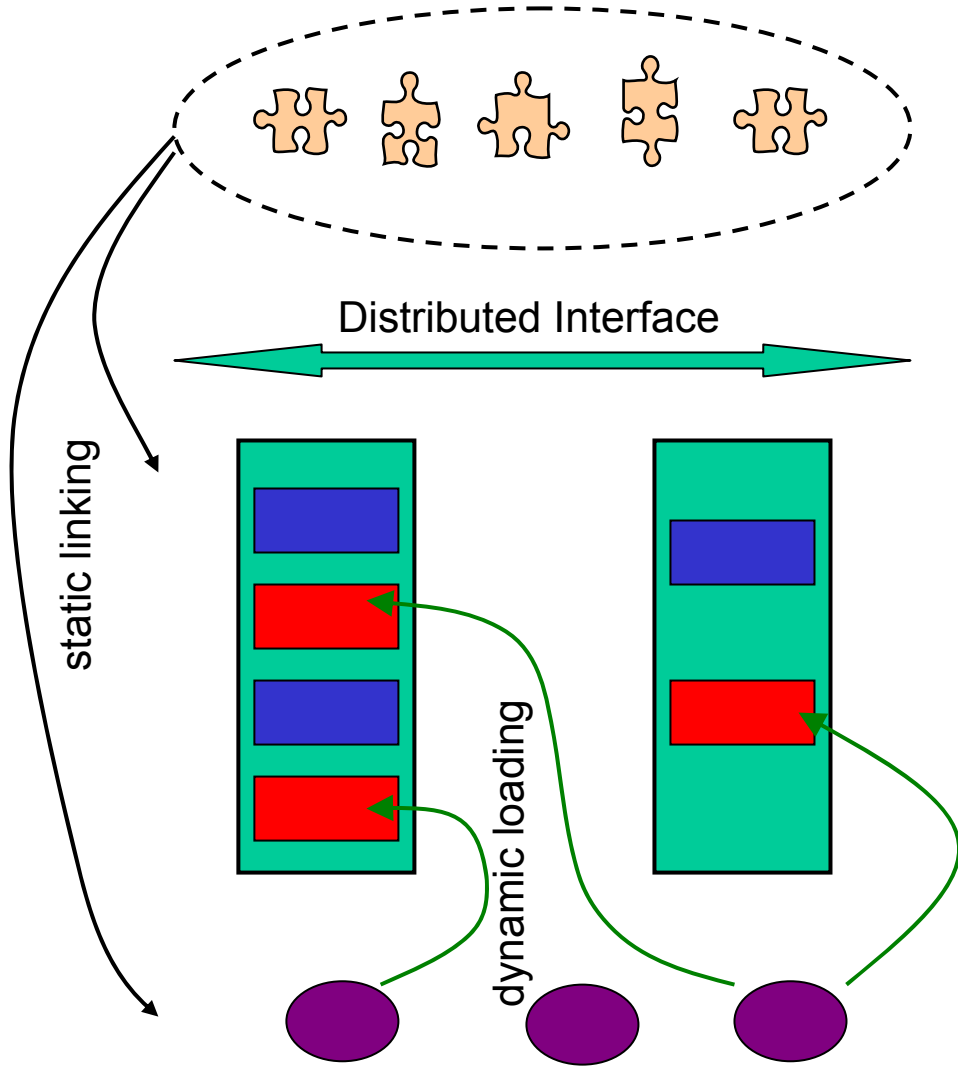


Requirements

- Support the integration of both commercial off-the-shelf (COTS) and custom hardware/software
- Provide a documented and validated “core” software that facilitates research and clinical use
- Serve as a component within a larger medical device
 - Consistent and interoperable interface with other components
- **No strict hierarchical relationship between devices**



Where are we going?





C++ Software Libraries

- cisst libraries
- other libraries

Frameworks

- Based on system complexity
- Component of larger system

 frozen spots
 hot spots

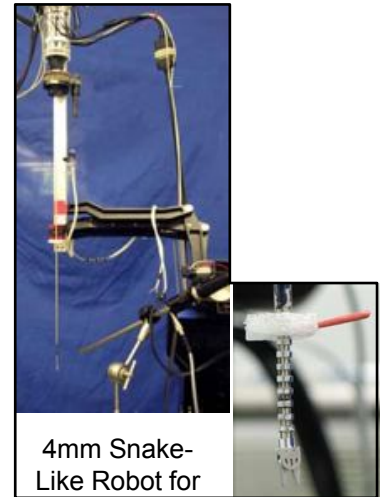
Binary components

- hardware interfaces
- research algorithms



Family of Application Frameworks

- Complete Software Control
 - Two low-level tasks, servo & supervisory
 - Maximum flexibility for researcher
 - Example: Snake Robot Controller
- Software Supervisory Control
 - One supervisory task
 - Uses COTS for low-level control
 - Example: Steady Hand Robot
- Software API only
 - API for complete control of robot based on COTS
 - Example: Rodent Therapy Robot



4mm Snake-Like Robot for throat



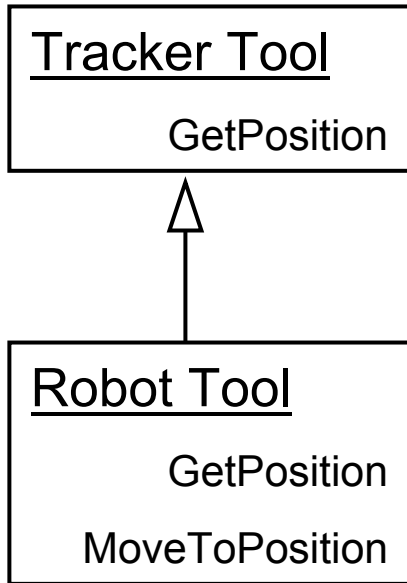
Steady Hand Robot



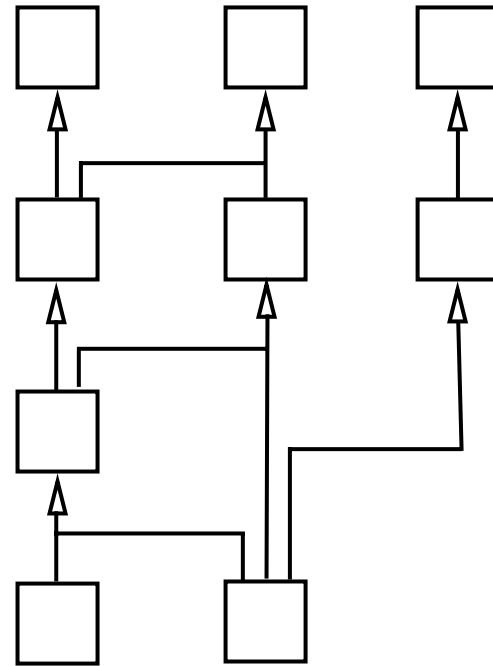
SARRP

Device Interface

- Device Hierarchies:



Can lead to complexity!

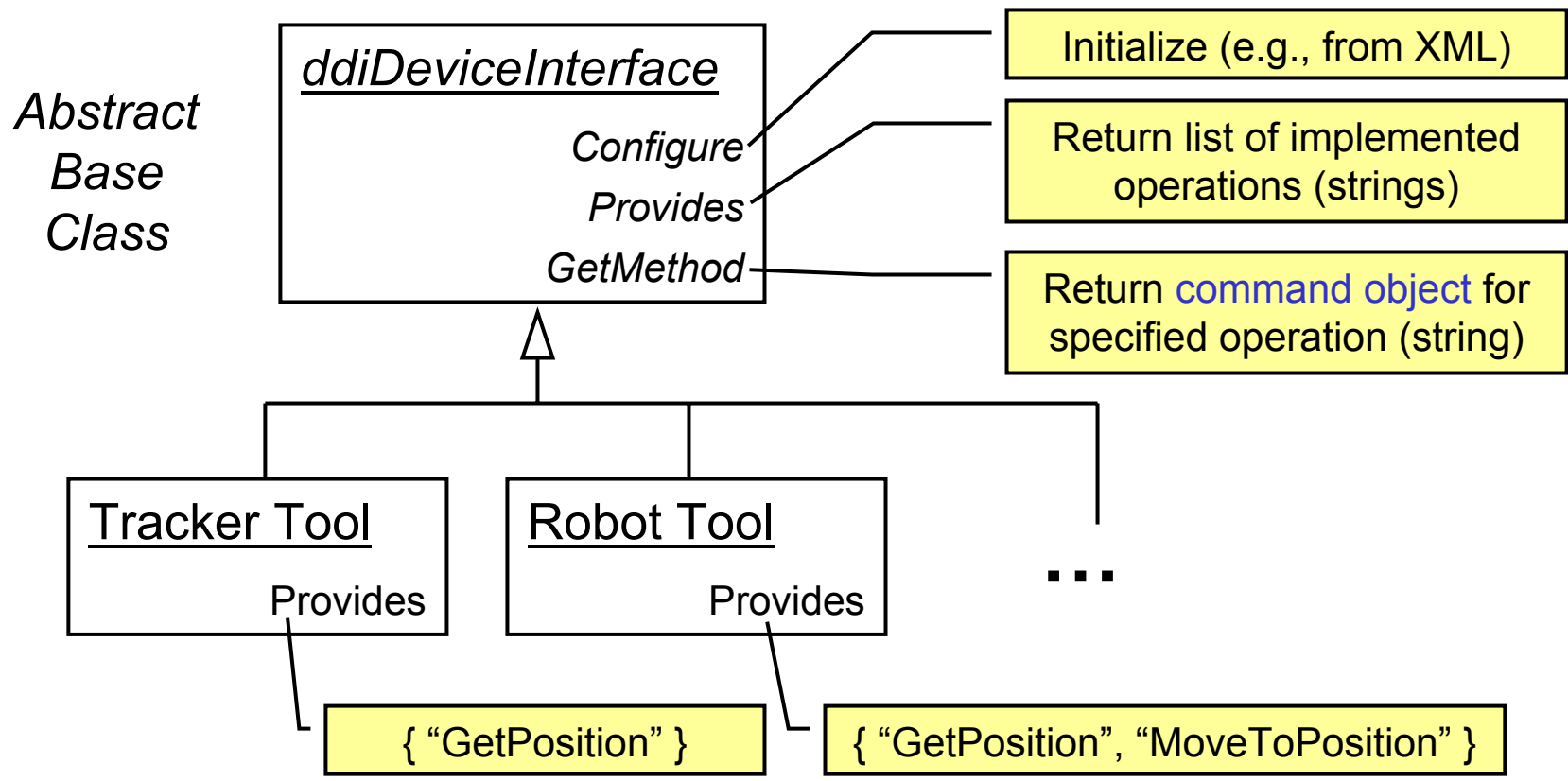


Should be able to use robot in place of a tracker



Device Interface

- Our solution: query with string to obtain command object (Command Pattern)





Device Interface

```
class myTask : public rtsTask {  
private:  
    ddiDeviceInterface* dev;  
    ddiCommand* cmd;
```

public:

```
void Startup();
```

```
...  
dev = ptr to device  
dev->Configure(...);  
cmd = dev->GetMethodByName("GetPosition");  
...
```

Non-real-time

```
void Run();
```

```
};
```

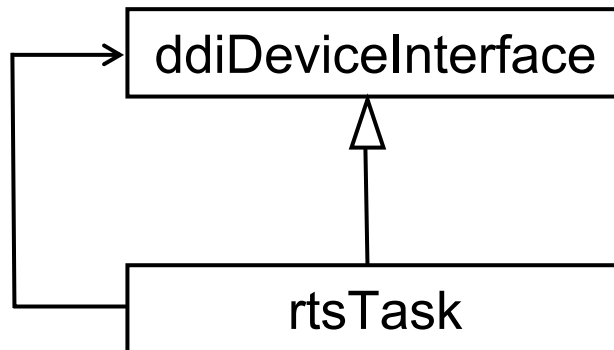
```
...  
cmd->Execute(data);  
...
```

Real-time



Real Time Support

1. Devices and Tasks should be interchangeable:
 - Example: Servo control via an intelligent device or via a software task
 - Solution: Task class derived from device class, but also includes a device





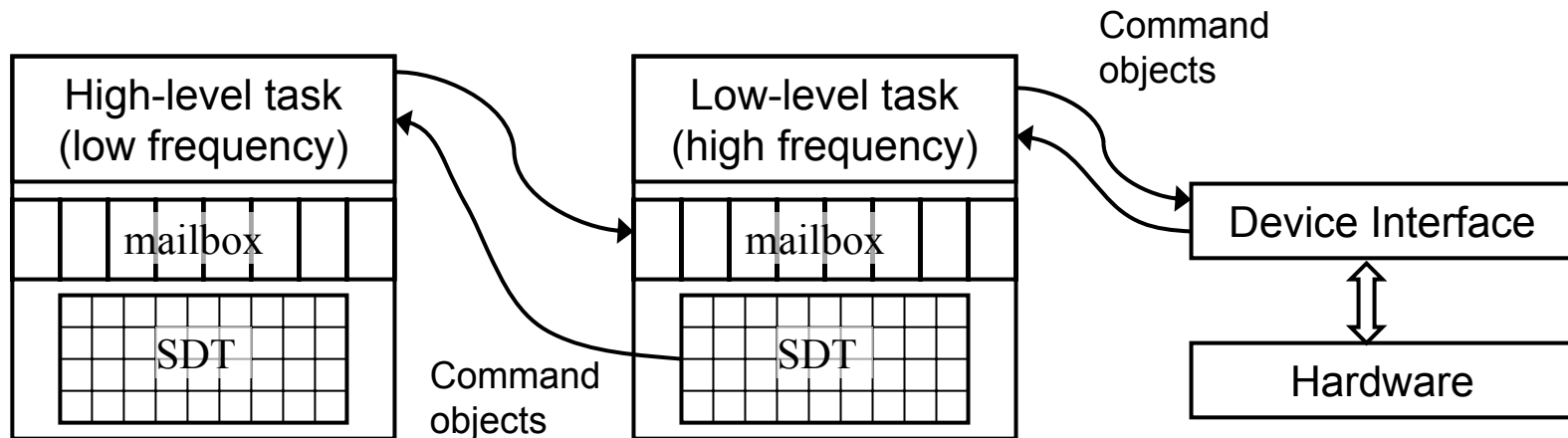
Real Time Support

2. Maintain time history of important state data

- State Data Table (SDT), indexed by time and data id

3. Task communication with Command Pattern:

- Read from Task SDT or Device
- Write to Task Mailbox or Device
- Command object can handle remote communications



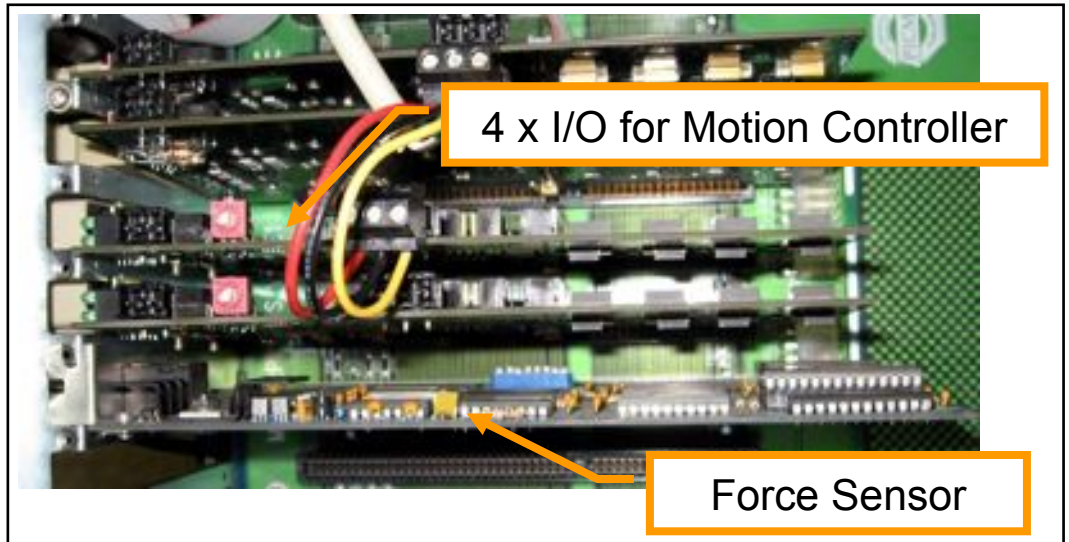
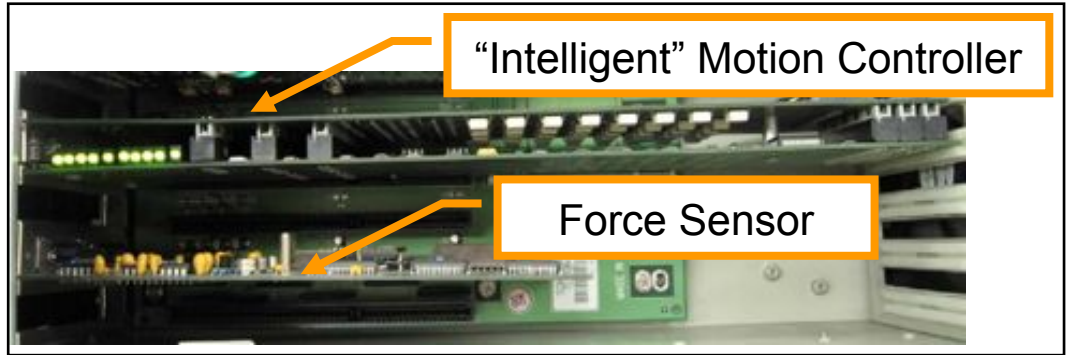
Composite Devices



Steady Hand Robot



4mm Snake-Like Robot for Throat



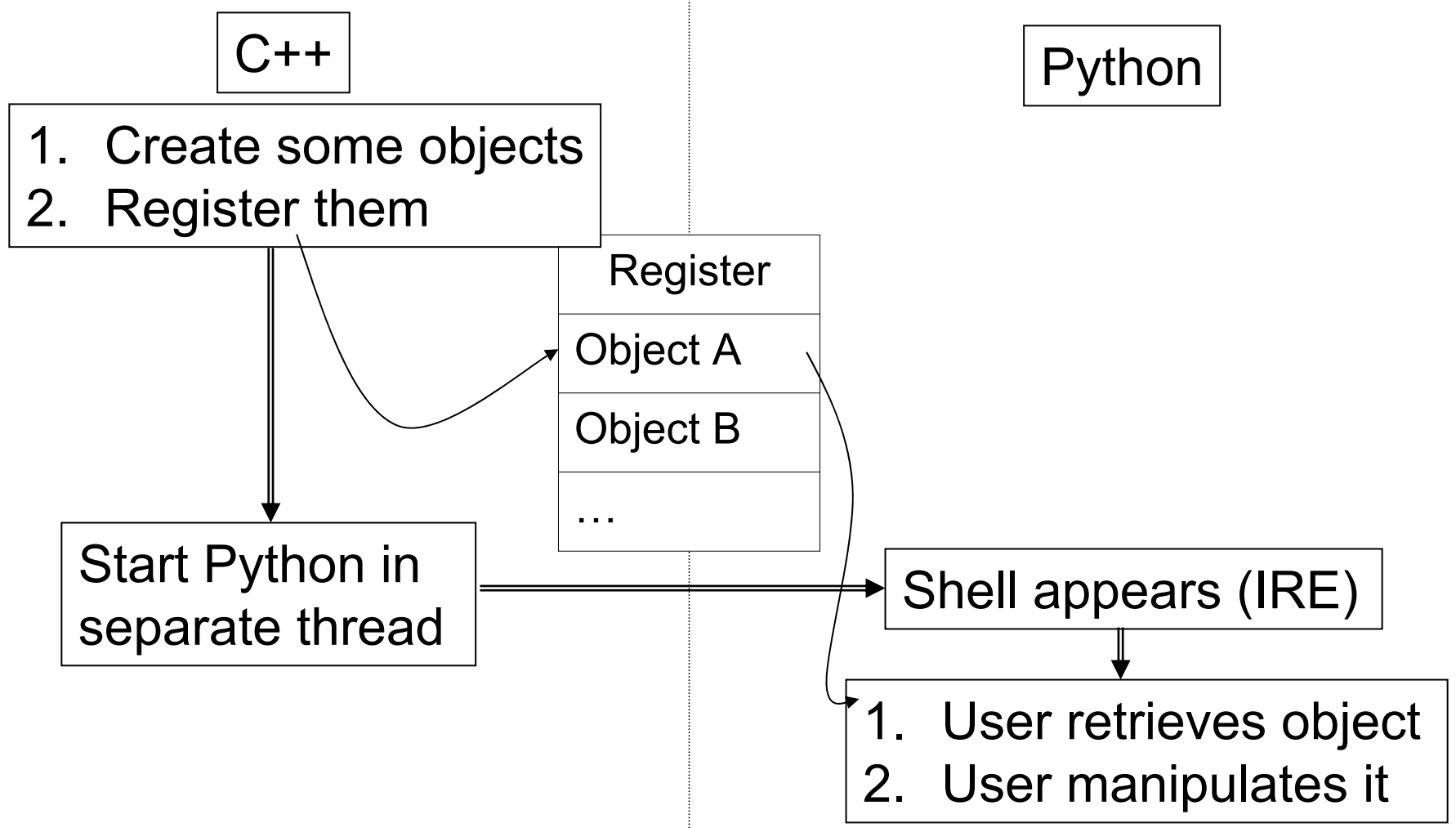


Python wrappers, motivation

- Wrapper for rapid prototyping
 - Run as you type (as for Matlab)
GUI features provided by `cisstInteractive`
 - Large collection of existing libraries with Python interface for networking, GUI, visualization (OpenGL, VTK), ITK ...
- Debugging tool
 - Use as a back door to a C++ application
 - Requires an “Object Register”
- Limited overhead
 - Use SWIG for automatic generation of wrappers



Python wrappers for debug





Python wrappers for debug

The screenshot shows the CISST Interactive Robotic Environment (IRE) shell. The window title is "CISST Interactive Robotic Environment". The menu bar includes "File", "Edit", "Tools", and "Help".

On the left side, there are two panels:

- Register Contents:** A table with two columns: "Identifier" and "Type". It contains one entry: "rob" with type "mskccRobot".
- Shell Variables:** A table with two columns: "Identifier" and "Type". It contains three entries: "CMN_LOG_DE..." with type "int", "rob" with type "mskccRobotPy...", and "shel" with type "instance".

The main area is a shell window titled "*Shell". It displays the following text:

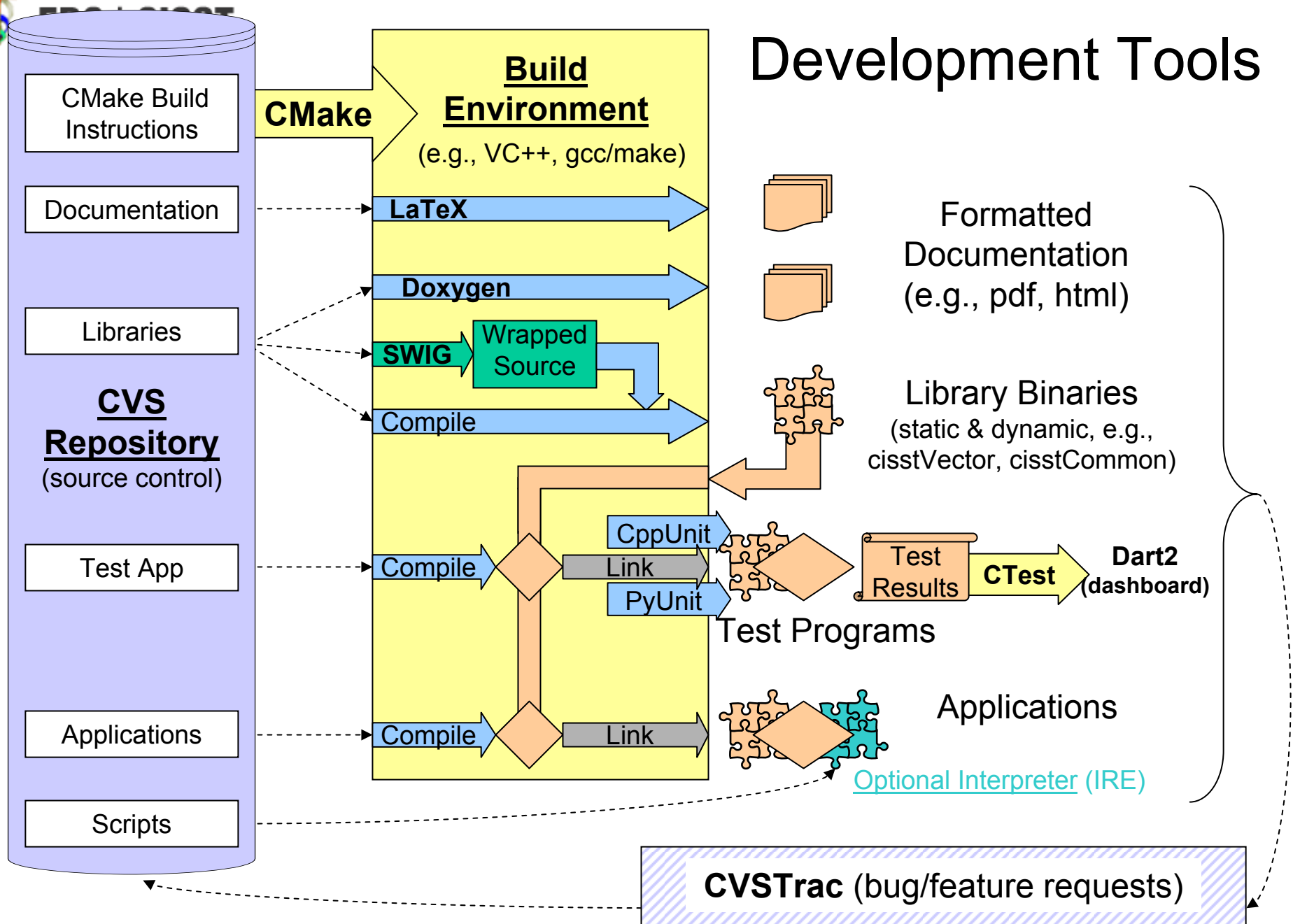
```
1 Welcome to the IRE Shell.
2 Ctrl-Up for history!
3 Python 2.4 (#60, Nov 30 2004, 11:49:19) [MSC v.1310 32 bit (Intel)] on win32
4 Type "help", "copyright", "credits" or "license" for more information.
5 >>>
6 >>> from cisstCommonPython import *
7 >>>
8 >>> from mskccRobotPython import *
9 >>>
10 >>> rob = smnObjectRegistrar.Get("rob")
11 >>> rob.Mov
```

A dropdown menu is open over the "rob.Mov" line, showing a list of attributes:

- JT_FWDLIM_MASK
- JT_HOME_MASK
- JT_MOTOR_MOVING_MASK
- JT_MOTOR_OFF_MASK
- JT_REVLIM_MASK
- MAX_AXES
- MOTOR_MOVING_MASK
- MOTOR_OFF_MASK
- MoveJoints



Development Tools



Conclusions

- Modular software environment for medical robotics
 - Object-oriented
 - Command objects with self-describing interfaces
 - Interactive scripting and debugging
 - Compatible with other toolkits and packages (e.g., ITk, VTK, LAPACK, BLAS)
- “Good practices” software engineering
- Availability
 - Routinely used in CISST ERC
 - Open source distribution
 - See www.cisst.org/software for timeline & details

Acknowledgements

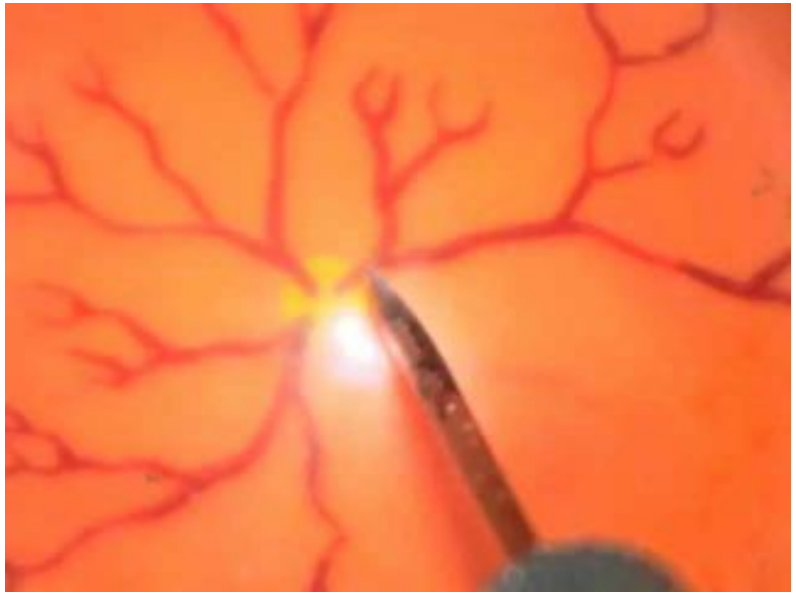
- Prof. Russell H. Taylor
- Ofri Sadowsky
- NSF grant EEC9731748



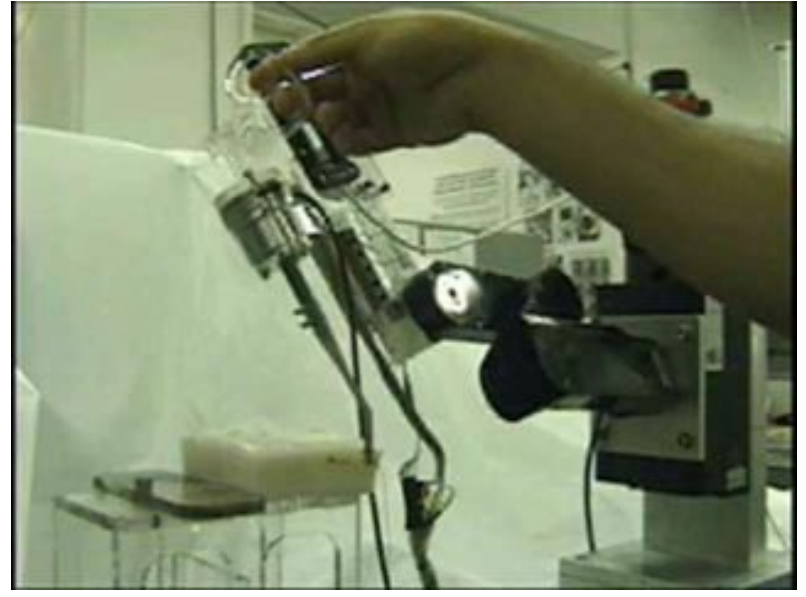
Systems using CISST Library



With R.Taylor, N. Simaan, K. Xu, W. Wei



With R.Taylor, G. Hager, I. Iordachita



With R.Taylor, M. Li