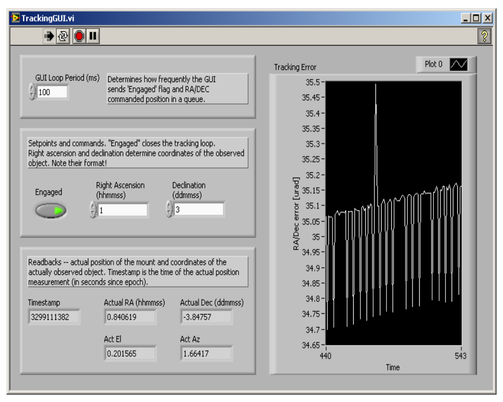
Integration of ACS and LabVIEW

WEP013

Klemen Žagar, Anže Žagar; Cosylab, Ljubljana, Slovenia
Bertrand Bauvir, Gianluca Chiozzi, Philippe Duhoux; European Southern Observatory, Garching, Germany

Motivation

- ☑ Allow engineers to develop user interfaces and control loops with LabVIEW, while retaining system-level manageability and control of ACS.



Example application: telescope tracking

ACS

- ☑ *Atacama Large Millimeter Array (ALMA) Common Software*
- ☑ CORBA-based infrastructure for control system development
- ☑ Component/container model, distributed management, bulk data transfer, logging, archiving, alarms, configuration database, …

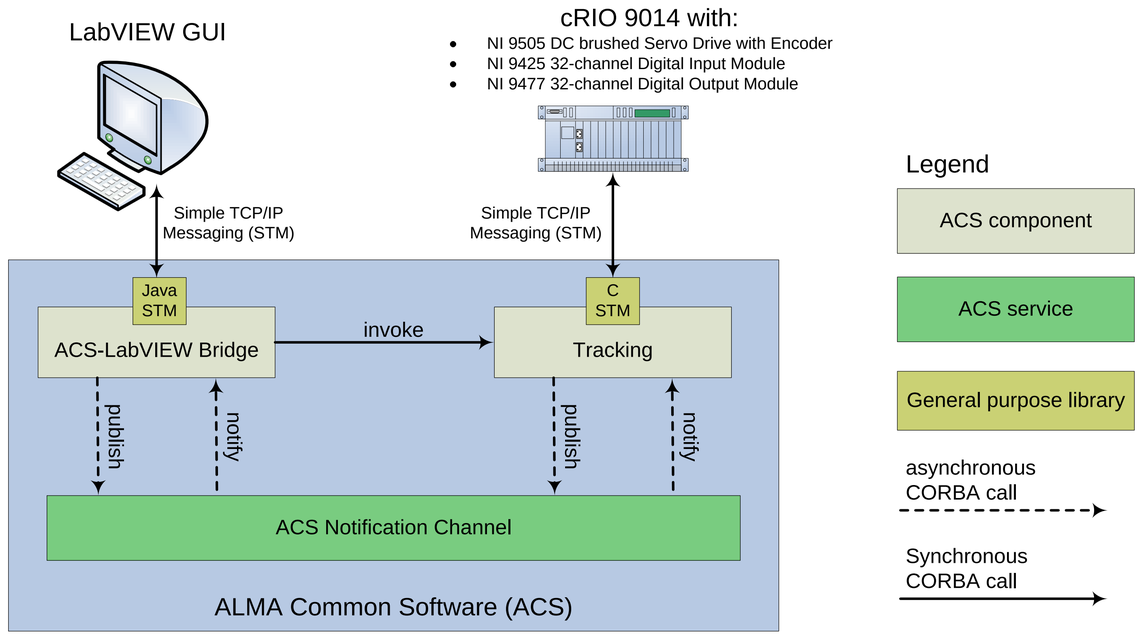
National Instruments LabVIEW

- ☑ A convenient, easy to learn environment for control system development.
- ☑ Graphical user interface and control loops (on various platforms, from non-real-time to FPGA)
- ☑ Wide array of supported instrumentation

Approaches

- ☑ Call Library Node LabVIEW mechanism:
  - ✓ LabVIEW runtime calls a function implemented in C
  - ✓ The C function wraps ACS
  - ✓ Advantage: efficiency
  - ✓ Disadvantages: possible destabilization of LabVIEW run-time, requires ACS libraries on all LabVIEW nodes, …
- ☑ TCP/IP communication (NI Simple TCP/IP Messaging)
  - ✓ LabVIEW sends a message over TCP/IP to an ACS process, and vice-versa
  - ✓ Advantages: ease of deployment, loose coupling
  - ✓ Disadvantage: reduced efficiency for some deployments
- ☑ LabVIEW shared variables and data sockets:
  - ✓ Make use of LabVIEW-provided shared variable or data socket communication mechanisms
  - ✓ Advantages: in principle, easy to set up
  - ✓ Disadvantage: not portable, lack of control over on-the-wire protocol, not reliable, unpredictable scalability

Architectural Overview



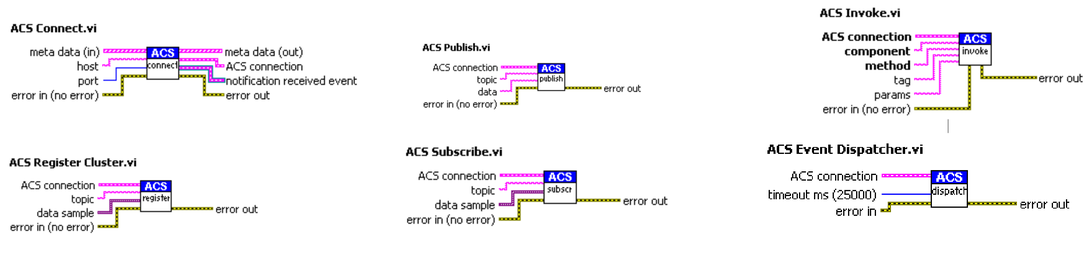
C and Java API

```
#include "lvstm.h"

// Connect to a LabVIEW process
lvstm_conn_t* c = lvstm_connect("host", port, read_callback, NULL, NULL);

// Prepare a message to send.
lvstm_message_t* msg = lvstm_create_message("Engaged");
lvstm_write_bool(msg, engaged);
// Send the message.
lvstm_send_message(c, msg);
lvstm_close_message(msg);

// Disconnect from LabVIEW
lvstm_disconnect(c);

// Called whenever LabVIEW sends a message
void read_callback(
  void* tag,
  lvstm_conn_t* conn,
  lvstm_message_t* msg
) {
   lvstm_read_double(msg, &act_az);
   lvstm_read_double(msg, &act_el);
   // ...
}
```

```
// Initialization
lvstm = new Lvstm("ACS-LabView Bridge", new MyListener(), THREADS_NUM, true);
lvstm.start(true, socket);

// Termination
lvstm.stop();

/* Implementation of listener of LabVIEW-related events. */
public class MyListener implements ILvstmListener {
    /* LabVIEW has sent a message. */
    public void messageReceived(LvstmConnection connection, LvstmMessage message) {
        if (message.getMetaId() == subscribeMeta) {
            String topic = message.getStringIntLen();

            // Send a message back to LabVIEW
            connection.sendMessage(/* ... */);
        }
        // ...
    }

    /* LabVIEW has connected. */
    public void connected(LvstmConnection connection) {
        // ...
    }
}
```

The APIs are ACS independent – can be used for integration into EPICS, TANGO and other control systems!

LabVIEW "API" for ACS



Conclusion

- ☑ We have shown that:
  - ✓ LabVIEW GUIs can be used to control ACS components
  - ✓ LabVIEW control loops monitored and controlled via ACS
  - ✓ Remote procedure call and message-oriented mechanisms supported
  - ✓ Support for most data types (including structures)
- ☑ Identified weaknesses and opportunities for improvement:
  - ✓ Reliability improvements (auto-reconnect, …)
  - ✓ Performance improvements (e.g., sending only changes)

For more info, contact: klemen.zagar@cosylab.com