CONTROL SYSTEM INTEROPERABILITY
AN EXTREME CASE:

Merging DOOCS and TINE

P. Duval, A. Aghababayan, O. Hensler, K. Rehlich, DESY, Hamburg, Germany
Interoperability?

• With commercial packages …

• Wrapping/Binding non-native interfaces?
  o e.g. How to interface EPICS with LabView?
    • or MatLab, Python, Perl, Root, etc.
    • Just ask COSYLAB to do it?
  o Use the interoperability tools that come with the package.
    • Java + JNI
    • .NET + System.Runtime.InteropServices
    • MatLab + MEX interface
    • LabView + external library support
    • etc.

• Go native?
  o e.g. STARS has a native Perl interface.
  o Most have a native java interface.
  o Not always practical!
Interoperability?

• With other **control system frameworks** ...
  o “We love our EPICS and they love their TANGO ...”
  o What to do?

• **Gateways or translation layers** that ‘trade data’.
  o epics2tango, tango2epics ...

• Assume as given:
  o control system frameworks manage interoperability to commercial software.

• Concentrate on:
  o interoperability **between** the **control system frameworks**!

• Why worry about this?
Motivation

• **Warring factions** within a single institute or project!
  - And those at the top who want results …
  - (But this never happens, right?)

• Experiments, Test Equipment *from other institutes*
  - Sudden deadlines to get something new into the system …

  - Recent example at DESY:
    - **OLYMPUS** experiment brought some detector software from MIT with **EPICS** interface.
    - *Immediate integration* with the rest of the control system via **epics2tine**.
Motivation

• Include *useful features* of another system
  o e.g. **STARS**
    • excellent system for beamline control
    • use **STARS-TINE** bridge for multicasting BEAM parameters to the 70 end stations at Photon Factory in **KEK**.

• **Major Release upgrades**
  o e.g. **TACO** and **TANGO**

• And if you’re *not 100% EPICS*
  o you’ll probably have to interoperate with it.
Both are mature control systems
- Primarily in use at DESY
- But used at other institutes and industry as well
- (No, the entire rest of the control-system world does NOT use either EPICS or TANGO).

**TINE:**
- An ISOLDE spin-off (CERN ~1991)
- Transport is socket based

**DOOCS:**
- Early collaboration with TACO (~1995)
- Transport is SunRPC based

**ALL** accelerator control at DESY is either TINE or DOOCS.
- Additional motivation to have seamless interoperation.
- (Yes, you will find EPICS in cryogenics and infrastructure and TANGO at the HASYLAB beamlines).

Strategies for interoperability ...
How best to ‘trade data’?

• System A (oranges) and System B (apples).
• Have to deal with apples and oranges one way or another!
  o Translate System A request into System B language:
  o Translate System B response back to System A language:

  o 1. Apple to Orange ‘gateways’
    • lives as an external process
  o 2. Apple to Orange ‘plugs’
    • live on ‘Orange’ clients
  o 3. Apple to Orange ‘translators’
    • live on ‘Apple’ servers
Translation Layer

- **Method 1** (*gateway*)
  - Requires setting up an extra process for each target server.
  - Connectivity problems harder to trace?
  - Least invasive

- **Method 2** (*client-side plugs*)
  - Popular: JCOP, cdev, abeans, CSS, jddd, ACOP, ...
  - Available features depend on target server!
    - *e.g.* asking a server to multicast data would only work on a TINE server!

- **Method 3** (*server-side translator*)
  - Most invasive
    - New software (new risks) on critical server components.
  - Best method for merging **ALL** control system features.
Frameworks Models

Brief Review:

1: Database Model
   - **EPICS, VISTA** (i.e. VSystem not the OS)

2: Device Server Model
   - **TANGO, DOOCS, ACS, STARS, TINE**
   - Server offers methods to a collection of ‘devices’ at some location.

3: Property Server Model
   - **STARS, TINE**
   - Server is a service with properties, which can have keywords.

How to map e.g. model 1 to model 2 and vice versa?
DOOCS/TINE Merger

- Uses server-side translation! (Method 3)
  - All TINE features available to a DOOCS server!

- DOOCS device servers maps perfectly into TINE device servers and vice versa!

- TINE property servers present a browsing issue with some DOOCS utilities.
  - Straightforward to deal with!
Server-side Translation

• **DOOCS** *DAQ protocol*
  • independent issue
  • does not impact the DOOCS/TINE merger
• **DOOCS** transport based on *SunRPC*
  o Synchronous polling and scalability problems?
  • Aside:
    o TACO later *(post-DOOCS collaboration)* introduced inverted SunRPC client-servers to accommodate asynchronous transfer.
    o TACO + *SunRPC* gave way to **TANGO** + **CORBA**
  • **DOOCS**: Make use of **TINE** from the merger

**TINE**:
  o Asynchronous transfer
  o QoS steering *(UDP, TCP, Multicast)*
  o *Contract coercion*
Contract Coercion

• “Joe the Programmer” is driving the data flow
  o The ability to do things efficiently (e.g. asynchronous updates on event) does NOT mean application programmers will do it this way!
  o Synchronous calls are easy to understand and program!

• Panel builders (jddd, MEDM, …) designed to be simple.
  o Optimized and efficient transport is NOT simple!
  o How is “Joe the Programmer” using the panel builder?

• What does your MatLab interface look like?
  o Note: Yes, you can do callbacks in MatLab!

• e.g.
  o Try synchronously polling all 300 BPMs individually at 10 Hz within some client application and then run the application on 10 different stations!
A Server takes control of its Clients

**TINE: Contract Coercion in the transport protocol**

Example: doing 1 thing for 1 effective client instead of 600 things for 10

---

**Client**
- Give me property “Pressure” for pump “OL146.2”
- Ok then, monitor “Pressure” for pump “OL146.2”
- Ok then, monitor “Pressure” for all pumps
- Ok then, I’ll listen for the multicast

**Server**
- No! You’ll have to monitor this!
- No! You’ll have to monitor the entire MCA (look for index 17)
- Ok, but I’m going to multicast it!
Let the Merger Begin ...

- **Step 1: request-response mapping**
  - Data type mapping
    - **primitives** exist in both frameworks
    - **compound data types** must ALL map!
      - e.g. NAME-FLOAT-INT32 as an atomic data type
    - **TINE** offers user-defined data types (**structures**)
      - DOOCS doesn’t
      - DOOCS -> TINE not a problem
      - TINE -> DOOCS?
        - structure fields are accessible!
  - Dispatch mapping
    - client is calling property P, is sending type T1, wants type T2, access = A
    - can now use **TINE scheduling in DOOCS**!
  - Error/Status code mapping
    - Status = 0 is always ‘success’
    - But: can send data with status (e.g. here’s the data, but it’s not calibrated)
DOOCS/TINE Merger

- **Step 2: transport mapping**
  - **client Side API**
    - should support *asynchronous communication*
    - can disguise asynchronous listeners with a *synchronous façade*.
    - asynchronous API should be *rich* enough to support **ALL features**
      - e.g. How to launch an asynchronous data link but WAIT for the initial callback?

- **Is that it? Are we done?**
  - With a gateway you might even be done at Step 1!
DOOCS/TINE Merger

- We’re done when a random TINE or DOOCS server passes the Duck Test using a random TINE or DOOCS control system utility.

“If it looks like a duck, swims like a duck, and quacks like a duck, then it probably is a duck”

- i.e. A kind of ‘Turing Test’.
DOOCS/TINE Merger

• **Step 3**: *central Service and Utilities*

  - Naming
  - Archiving
  - Alarm
  - Security
  - Remote Management
  - etc.
DOOCS/TINE Merger

- **DOOCS Naming**
  - `/facility/server/location/property`
  - Equipment Name Server (ENS) provides facility and server(s)
  - **No** separate subsystem identification.
    - Subsystem is usually applied to the facility.
    - Server “Modulator” in “FLASH.RF” instead of “FLASH”
  - **Strict** OO Device Server model
    - locations have properties
  - **Meta Properties** not principally distinguished from Properties
    - “P” is on the same footing as “P.EGU” or “P.HIST”

- **TINE Naming**
  - `/context/server/device/property`
  - Equipment Name Server (ENS) provides context and server(s)
  - **Separate** subsystem identification! (not part of name space)
    - OO Device Server model
      - devices have properties
    - Or Property Server model
      - properties have keywords
    - **Meta Properties** are distinguished from Properties
      - “P” is NOT on the same footing as “P.EGU” or “P.HIST”
DOOCS/TINE Naming

DOOCS ‘RPCtest’ Utility:

TINE ‘Instant Client’:

Analogous to TANGO + jive EPICS + ?
DOOCS/TINE Archiving

**DOOCS**
- Independent **DAQ** system
  - can also tag ‘events’.
- Local archiving of specific (configured) properties
- Record = single channel
- Accessible via “<P>.HIST” meta-property
- No sampling raster
- ** Thumbnails** available
  - fast access of general information over long time intervals

**TINE**
- Central **Event Archive**
- Local archiving of specific (configured) properties
- Record = single channel or **multi-channel arrays**!
- Accessible via “<P>.HIST” meta-property
- Sampling raster configurable
- Automatic raster for optical zooming
  - “points of interest” insert peaks and valleys.
- **Central archiving** of specific properties
  - Many possible filters
DOOCS/TINE Archiving

DOOCS local histories example:
DOOCS/TINE Archiving

TINE Archive Viewer

Optical or Digital Zooming, Trends and Snapshots, Movies, Correlations, FFT, Fits, etc.
Drag-and-drop between the two: No mean feat!
**DOOCS/TINE Alarms**

- **DOOCS Alarms**
  - Devices have alarms
  - 5 severity levels
  - **Push** system
  - Alarms have a status/error string
  - Also set TINE Alarm on the server side
    - error string -> alarm data

- **TINE Alarms**
  - Devices have alarms
  - 15 severity levels
    - 4 principal categories
  - **Pull** system
  - Alarms have **data** (up to 64 bytes)
  - Also push DOOCS Alarm at the Central Alarm Server.
DOOCS/TINE Alarms

DOOCS Alarm / Info Display:
DOOCS/TINE Alarms

TINE Alarm Viewer:
**DOOCS/TINE Security**

**DOOCS Security**
- Open READ
- WRITE (set) calls must pass security!
- Server Level or Property Level
- UNIX style
  - `gid`, `uid` of the caller determines access rights
- Non-UNIX systems
  - Locate caller ‘user name’ in a nis (ldap) database to ascertain gid, uid.

**TINE Security**
- Open READ (default)
  - Can configure ‘exclusive’ READ
- WRITE (set) calls must pass security!
- Server level, property level or device level.
- Compare `user name` and `address` of caller to the configured ACL tables.
- Can acquire an Access Lock
DOOCS/TINE
Remote Management

**DOOCS**
- Process watchdog
  - Unix-like or windows
- Special DOOCS server
- Monitors process statistics
- (re)starts missing server processes
- Allows remote stop and start

**TINE**
- Process watchdog
  - Different solutions for
    - Unix-like
    - Windows
    - VxWorks
    - DOS
- Monitors process statistics
- (re)starts missing server processes.
- Allows remote stop and start.
DOOCS/TINE
Remote Management

DOOCS Watchdog Panel:
DOOCS/TINE Remote Management

TINE FEC Remote Panel:
DOOCS/TINE Remote Management

TINE: attachfec /REGAE/VAC.ION_PUMP (a native DOOCS server !)
Culture Shock

• **DOOCS**
  - Generally use **jddd panels**
    • Simple clients with display widgets
  - MatLab, etc.
  - **No** explicit multi-channel support
    • But heavy reliance on wildcards and filters.
  - **No** user-defined structures (hard to attach to a widget)
  - Names tend to be
    • ALL **uppercase** with underscores
    • e.g. “PROPERTY_ONE”

• **TINE**
  - Generally use **rich clients** (java, .NET)
    • RAD tools (**ACOP beans**)
  - MatLab, etc.
  - **Explicit** multi-channel support
    • Can also use wildcards and filters
  - User-defined structures are popular with some developers!
  - Names tend to be
    • **camel case**
    • e.g. “PropertyOne”
jddd Applications
DOOCS/TINE Merger

• In the field:
  o FLASH
    • DOOCS culture with notable TINE servers (e.g. magnets)
  o PETRA
    • TINE culture with notable DOOCS servers (e.g. vacuum)
  o REGAE
    • Many native DOOCS and TINE servers all speaking TINE
    • Many jddd panels and acopbeans rich clients.
    • Many MatLab applications
    • Generally smooth operations for the past half-year.
  o XFEL
    • To be: DOOCS centric with DOOCS and TINE servers all speaking TINE
      o Currently gaining experience via REGAE
    • Similar mix of jddd, rich clients, MatLab as in REGAE (?)
    • Heavy use of DOOCS DAQ
    • We’ll see how it goes …
DOOCS/TINE Merger

• **Status**
  - response-request translation: ~98% complete
  - services mapping: ~80% complete
  - culture shock:
    - Although most all ‘features’ are mapped, those in one ‘world’ often remain unknown and unused the other ‘world’.
    - Can sometimes battle different ‘mindsets’ with contract coercion.
      - can trap synchronous polling of individual channels, etc.
    - There are still sometimes ‘gateways’ that are created for no other purpose than to bridge cultural differences.
  - Components still carry a ‘brand name’ (and probably always will)
  - **Strive for a non-zero sum game (WIN-WIN!)**

http://doocs.desy.de
http://tine.desy.de