The Babylonization of Control Systems

P. Duval, Z. Kakucs - DESY
M. Kadunc, I. Kriznar, M. Plesko, A. Pucelj, G. Tkacik – JSI and Cosylab
Available Control Systems

- Complete systems
  - ACS
  - COACK
  - DOOCS
  - EPICS
  - TANGO
  - TINE
  - ...

- APIs and GUIs
  - Abeans and CosyBeans
  - ACOP
  - CDEV

- Machine physics packages
  - Databush
  - XAL
The Layers of Control System Packages

- EPICS
- chan. acc.
- MEDM
- TINE
- ACOP
- DOOCS
- DDD
- COACK
- TANGO
- ACS
- CDEV
- JDM
- Abean
- CosyBeans

driver data
SERVER

commu-
nication

API visual support
CLIENT
## CS Comparison: Basic Features

<table>
<thead>
<tr>
<th></th>
<th>EPICS/CA</th>
<th>TINE</th>
<th>DOOCS</th>
<th>COACK</th>
<th>ACS</th>
<th>TANGO</th>
<th>TACO</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>monitors</strong></td>
<td>on-change</td>
<td>on-change, on-timer, peer-to-peer, peer-to-network</td>
<td>As per TINE or CA</td>
<td>On-change On-timer interval</td>
<td>on-change, on-timer, threshold and interval settable for each client</td>
<td>on change, on alarm, periodic</td>
<td>server specific</td>
</tr>
<tr>
<td><strong>groups of signals</strong></td>
<td>Partially (buffers requests)</td>
<td>YES</td>
<td>YES via TINE</td>
<td>YES (*command)</td>
<td>NO</td>
<td>future</td>
<td>yes via data collector</td>
</tr>
<tr>
<td><strong>built-in simulation</strong></td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES (Virtual-machine)</td>
<td>external</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td><strong>Dynamic addition of changes</strong></td>
<td>Changes dynamic, Add/Remove needs reboot</td>
<td>Minimal. Security changes (server), address changes (client) do not require reboot</td>
<td>Device Address changes do not require reboot</td>
<td>Comonent base addition does not require reboot</td>
<td>add/remove device</td>
<td>YES via database</td>
<td>YES via database</td>
</tr>
</tbody>
</table>
### CS Comparison: I/O Integration

<table>
<thead>
<tr>
<th></th>
<th>EPICS/CA</th>
<th>TINE</th>
<th>DOOCS</th>
<th>COACK</th>
<th>ACS</th>
<th>TANGO</th>
<th>TACO</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config Database</strong></td>
<td>text-based .db file, templates + substitutions</td>
<td>Distributed (unspecified)</td>
<td>Distributed (DOOCS)</td>
<td>XML based</td>
<td>XML-based, hierarchical data supported</td>
<td>MySQL</td>
<td>ndbm, Oracle or MySQL</td>
</tr>
<tr>
<td><strong>Explicitly supported I/O platforms (e.g. VME, PCI, CAN-bus,...)</strong></td>
<td>VME, cPCI (in test), CAN, GPIB, VXI, Allen Bradley, Serial, PLCs, BitBus, and others...</td>
<td>SEDAC</td>
<td>SEDAC, PROFlbus, CANbus, Shared MEM</td>
<td>CAN-bus, LonWorks</td>
<td>OPC</td>
<td>serial line, gpib, modbus</td>
<td></td>
</tr>
<tr>
<td><strong>Explicitly Supported I/O cards</strong></td>
<td>see: <a href="http://www.aps.anl.gov/epics/modules/bus.php">www.aps.anl.gov/epics/modules/bus.php</a></td>
<td>n.a.</td>
<td>All with standard UNIX driver</td>
<td>Anyone with Win32 drivers</td>
<td>n.a.</td>
<td>growing list</td>
<td>large list</td>
</tr>
<tr>
<td><strong>support for external timing</strong></td>
<td>YES</td>
<td>NO (do-it-yourself)</td>
<td>YES</td>
<td>NO (do-it-yourself)</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
</tbody>
</table>
# CS Comparison: Services and Tools

<table>
<thead>
<tr>
<th></th>
<th>EPICS/CA</th>
<th>TINE</th>
<th>DOOCS</th>
<th>COACK</th>
<th>ACS</th>
<th>TANGO</th>
<th>TACO</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Event service</strong></td>
<td>via CA</td>
<td>YES (NETMEX)</td>
<td>Via TINE</td>
<td>YES</td>
<td>YES</td>
<td>work in progress</td>
<td>yes</td>
</tr>
<tr>
<td>(data channel)</td>
<td>gateways</td>
<td></td>
<td>NETMEX</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Alarm services</strong></td>
<td>Distributed</td>
<td>Distributed + Central</td>
<td>Distributed + Central</td>
<td>YES</td>
<td>Distributed</td>
<td>log service</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>archive services</strong></td>
<td>Central + Distributed</td>
<td>Distributed + Central</td>
<td>Distributed (in Database)</td>
<td>YES</td>
<td>central, through notify service</td>
<td>via logging</td>
<td>YES (uses Oracle)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>postmortem services</strong></td>
<td>NONE</td>
<td>YES, Event driven</td>
<td>YES</td>
<td>NO</td>
<td>black box</td>
<td>NO</td>
<td></td>
</tr>
</tbody>
</table>

| **Alarm Viewer** | YES | YES | YES | YES | from Abeans | |
| **Log Viewer**   | CMlog | YES | YES | YES | Chainsaw | |
| **Sequencer**    | FSQT, SNL | NOT Standard, but in place for HERA | YES | NO | Python | Python, SPEC or Tcl |
## CS Comparison: Application Development

<table>
<thead>
<tr>
<th></th>
<th>EPICS/CA</th>
<th>TINE</th>
<th>DOOCS</th>
<th>COACK</th>
<th>ACS</th>
<th>TANGO</th>
<th>TACO</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>API</strong></td>
<td>Active-X, Python, Perl, Fortran, IDL, Java, tcl, C, C++</td>
<td>ActiveX, VB, C, C++, Java, MatLab, FORTRAN, Java, MatLab, LabView</td>
<td>C, C++, FORTRAN, Java, MatLab, LabView</td>
<td>COACK components for VB/LabView /HP-VEE</td>
<td>device based through CORBA IDL (BACI model)</td>
<td>C++, Java, Matlab, Labview and Python</td>
<td>C, C++, Python, SPEC, Tcl, Matlab, Labview, IDL</td>
</tr>
<tr>
<td><strong>GUI builders</strong></td>
<td>DM2K, MEDM, EDM</td>
<td>VB, JAVA IDEs + TINE client wizard for VB and Java, DDD, LabView, HPVEE</td>
<td>DDD, LabView, MatLab</td>
<td>VB LabView HP-VEE</td>
<td>uses Abeans</td>
<td>Forte, QT-designer, Matlab, Labview</td>
<td>BuilderAccessory, QT-designer, Labview, Matlab</td>
</tr>
<tr>
<td><strong>GUI libraries</strong></td>
<td>from EDM</td>
<td>ACOP, Cosybeans, VB, MatLab, ActiveX, LabView</td>
<td>LabView, MatLab</td>
<td>VB LabView HP-VEE</td>
<td>CosyBeans</td>
<td>ATK</td>
<td>NO</td>
</tr>
<tr>
<td><strong>scripting tools</strong></td>
<td>tcl/tk, Perl, Python, UNIX</td>
<td>Command line calls can be used in standard UNIX scripts</td>
<td>Standard UNIX</td>
<td>Python</td>
<td>Python</td>
<td>Python, SPEC, Tcl</td>
<td></td>
</tr>
</tbody>
</table>
The Really Difficult Questions

- Applications developers:
  - How easy is it to learn the system?
  - How easy is it to create an application?
  - What features are missing?
  - What features are good?
  - What are the limitations?
  - Are the applications primitive (connecting data to display tools only)?
  - Can the applications be programmed?

- Operators:
  - How good are the applications in general?
  - What's missing?
  - What could be better?
  - What is the information density on a typical application?
  - How quickly can I find and isolate problems?
  - How often do failures in the control system hinder operations?
Other Difficult Questions

• Engineers
  - How easy is it to remotely test my equipment?
  - How easy is it to find problems?
  - How good is the alarm system notification of pending hardware failures?

• Inventors of the CS
  - what’s your problem? Everything is perfect and there is nothing better!

• Machine physicists
  - How good are the on-line analysis tools?
  - How easy is it to get the live data (archived data) into my favorite analysis tool?

• Directors of institutes
  - Could machine operations be better/smoothier/less expensive with another system?
  - mostly care about PR
Mixing Orange Clients with Apple Servers

1. ‘Apple-to-orange’ gateway
   a separate process utilizing the client/server APIs of both systems

2. Client-side ‘apple-plugs’
   client program developers think they are talking to oranges, but they are really speaking native ‘apple’

3. Server-side ‘orange-translators’
   server IOCs think they are being addressed by apples but are really speaking native orange.
Why Phil Thinks Translators are Better

- Server-side systematics are guaranteed to be there.
  - local alarm server, local history server, queries, etc.
- The data are as close to the source as possible
- can use the best applications for each problem
- Gateways bring a host of intermediate problems
  - e.g. connectivity problems are difficult to locate
  - another link in the chain
Why Mark Agrees with Him …
…but Is Still Happy with Abeans

• Client-side plugs usually provide only
  – common functionality over several CS
    • if the server-side features are not available, you come up empty
  – generic API
    • requires knowledge of CS and is therefore practically useless

• Abeans have a solution though
  – allow different CS data models (Channels, Devices, etc.)
  – interfaces are facades for services (plug-ins) with always at least one default implementation
  – we work hard to provide each single feature in a generic way – if TINE has ArchiveReader, also EPICS gets it!
EPICS, TINE and DOOCS Translator

- run EPICS2TINE directly on the IOC
  - don't speak channel access at all and access EPICS database directly
  - we have also elegantly solved the 16 Kbyte/message barrier (i.e. 4000 floats) of the old EPICS release

- DOOCS servers and clients offer the traditional SUN RPC interface as well as a TINE interface
  - EPICS IOCs are immediately available to DOOCS DDD clients. Using TINE2EPICS, the DOOCS IOCs are likewise available to EPICS MEDM clients.

- Likewise, running Abeans with a TINE plug will see all IOCs as TINE servers irrespective of their parentage.
Abeans plugs for TINE and EPICS

• DESY: Abeans-TINE plug
  – run TINE client applications on non-Windows machines
  – keep access to the full TINE API and services
  – Abeans provide this, plus add a rich client framework

• SNS: Abeans-EPICS plug
  – already has XAL: API to EPICS + machine physics package
  – now wants Abeans as a layer between XAL and EPICS, because of its many advanced capabilities

• Design guidelines:
  – full encapsulation, while keeping all functionality
  – generic solutions preferred
  – use standard Java APIs where possible
  – code decoupling
A Future Scenario?

- Take the “best of all possible worlds”
  - EPICS and DOOCS to integrate VME I/O cards (because they have the drivers)
  - TINE as the access protocol (for multicast capability)
  - DOOCS DDD or COACK (for developing synoptic GUI panel)
  - ABeans/CosyBeans (for complex applications and multiplatform capabilities)
- Still use all existing conventional applications
- Further possibilities include advanced features
  - TINE archiver
  - ACS logging
  - ABeans resource loading, etc.
Conclusions

- With translators and plugs we can mix any set of apples and oranges
- Appeal to CS package developers
  - don’t reinvent the CS core over and over
  - think more about services that could be used in a generic way by other control systems
  - provide good general purpose applications and tools
  - good documentation and tutorials won’t hurt either
  - this is where we are the weakest
- The Tower of Babel fell because of bad monocultural applications!