Elettra Sincrotrone Trieste

Elettra
Sincrotrone
Trieste

CERTIFIED
MANAGEMENT SYSTEMS
CERTIQUALITY
UNI EN ISO 9001:2015
UNI CEI EN ISO/IEC 27001:2022
UNI EN ISO 45001:2023
UNI CEI EN ISO 50001:2018

Tango Multiclasses as a Vehicle for Composition over Inheritance

Q&A at slido.com #3700254

Elettra Sincrotrone Trieste

Where we left off

- ✓ Last year (Giulianova): three ways to build a complex device server.
  - Monolithic — one giant class per instrument. Drowns in duplicated code.
  - Split into many — glued by DeviceProxy strings. Drowns in configuration.
  - Multiclass — several classes, one executable, one Jive subtree; each still a separate, inspectable device.
- ✓ Multiclass won (for me at least). That was the setup.

Elettra Sincrotrone Trieste

This year's question

- ✓ Once several Tango classes share a process,
- ✓ what should the relationship between them look like?

- ✓ **Composition, not inheritance.**

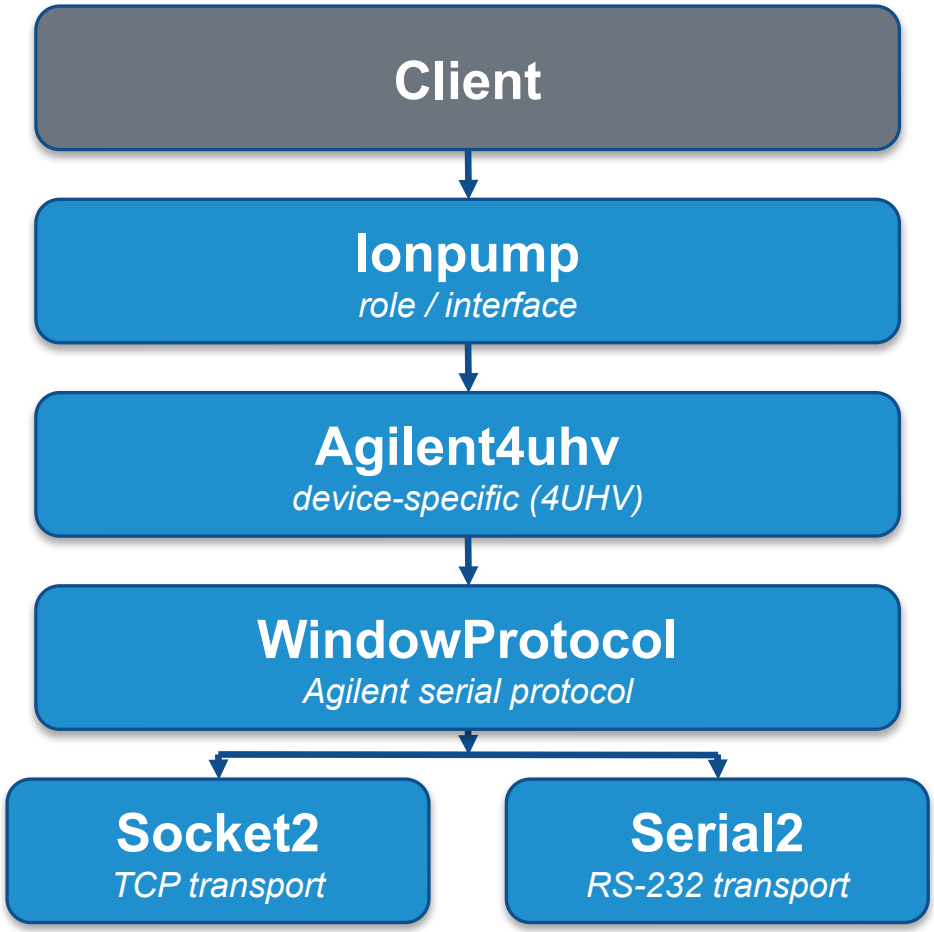
Elettra Sincrotrone Trieste

Concrete example — 4uhv-srv

- ✓ One server we run on the vacuum side. Five Tango classes, side by side:
  - Socket2 — TCP transport.
  - Serial2 — RS-232 transport.
    [USB2 – USB transport.]
  - WindowProtocol — Agilent Window Protocol:
    STX/ADDR/WIN/COM/DATA/ETX/CRC, faithful to the manual.
  - Agilent4uhv — device-specific: windows get meaning (812 = pressure, 011–014 = HV).
    [IPCMini]
  - Ionpump — canonical role: state, pressure, voltage, current.

Elettra Sincrotrone Trieste

The stack



Each layer does one thing.

Pattern 1 — Transport / Protocol / Device / Interface

- ✓ Transports know the wire, not the pump.
- ✓ Window Protocol: windows by number (812, 011…) + CRC — faithful to the manual.
- ✓ Agilent4uhv: gives windows meaning — 812 = pressure, 011–014 = HV.
- ✓ Role (Ionpump) exposes state, pressure, voltage, current.
- ✓ Each layer talks only to the one below.
- ✓ Pick boxes off a shelf; Window Protocol reused across Agilent gear.
- ✓ Each layer is its own device: read raw windows in Jive.
- ✓ A few extra lines bind an instance only to its in-server siblings.

Window Protocol — faithful to the manual

- ✓ A message is a frame: STX · ADDR · WIN · COM · DATA · ETX · CRC
- ✓ COM = 0x30 read · 0x31 write DATA = Logic / Numeric / Alphanumeric
- ✓ Read pressure, channel 1 → window 812:

```
PC→Ctrl    STX 02 | ADDR 80 | WIN 812 | RD | ETX 03 | CRC
Ctrl→PC    STX 02 | ADDR 80 | WIN 812 | WR | DATA 1.5E-8 | ETX 03 | CRC
```

- ✓ WindowProtocol speaks exactly this; Agilent4uhv knows 812 = pressure.

Elettra Sincrotrone Trieste

Pattern 2 — One role, many bodies

- ✓ Ionpump is just an interface.
- ✓ Sibling classes implement it for different vendors.
- ✓ New vendor = one new protocol class behind the same role.
- ✓ Clients, role, Jive views, and alarms untouched.
- ✓ A Tango class is the interface; a sibling class is the implementation.
- ✓ Pure Tango — no extra framework, language, or plugin system.

Composition vs inheritance

✓ Inheritance

- AgilentIonPump : IonPumpController : SerialDevice
- Works — until a vendor needs Ethernet instead of serial.
- Or until someone wants the protocol without the role.
- Deep, fragile, hard to reshuffle.

✓ Composition

- Siblings you line up and mix.
- Swap a transport without touching the role.
- Swap a vendor without touching the clients.
- Fits Tango: a distributed bus of peer objects.
- Classic Gang-of-Four advice, in pure Tango primitives.

The honest cost — and the enabler

- ✓ Sibling A → sibling B today:
  - DeviceProxy → CORBA → marshal / unmarshal / ORB round-trip…
  - …for two objects in the same address space.
- ✓ Branch add-deviceproxy-shortcut (cppTango):
  - DeviceProxy recognises in-process targets,
  - dispatches them as a direct function call.
  - Client code unchanged — same DeviceProxy, just faster when local.

Status & roadmap

- ✓ Done: commands, attribute read / write, state, status, ping.
- ✓ Roadmap: pipes, async, in-process event delivery.
- ✓ Branch is in review.
- ✓ When it lands: sibling-call cost approaches function-call cost.
- ✓ Composition becomes the default way to build Tango device servers.

Thank you!