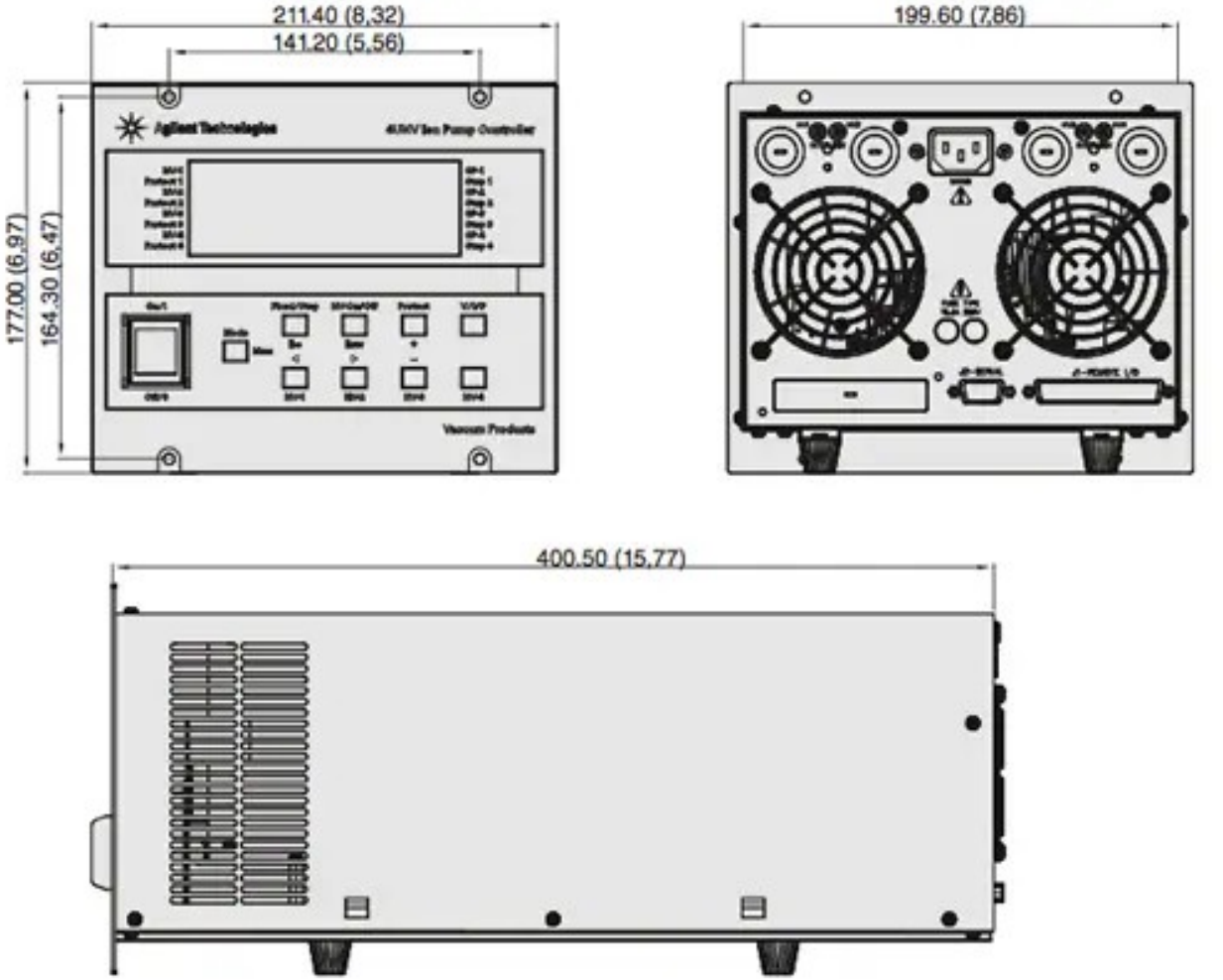
Elettra Sincrotrone Trieste

Elettra
Sincrotrone
Trieste

Tango's Multiclasses: How to use them to simplify development and deployment and get the configuration right (an attempt to suggest a small and humble guideline for device server development)

It is a very mundane topic… You are warned!

How would you integrate this device?



Dimensions: millimeters (inches)

The simplest approach

The monolithic one:

One instrument == one Tango class == one Tango Device Server

The DSs made this way are:

- Quite complex
- They contain a lot of duplicate code
- Difficult to maintain consistent quality across devices (bug fixes)
- Prone to errors

A better approach

The divide et impera one:

Due to the growing number and complexity of the instrumentation devices, **we started to split our DS into smaller DSs connected using DeviceProxy** (weak dependency through a string in the Device Property). In this way:

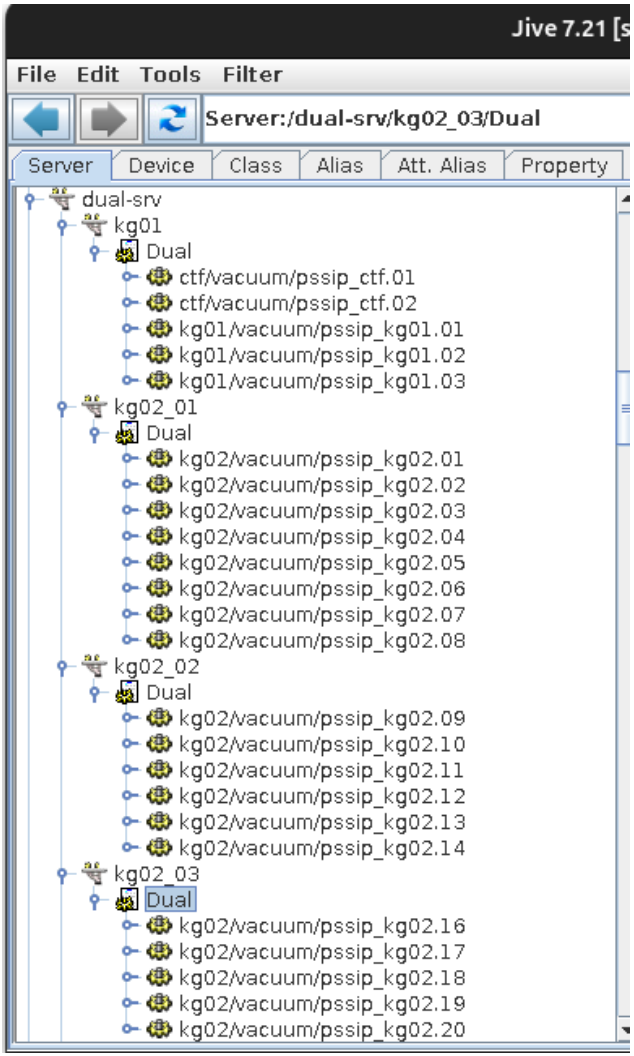
- DSs became more numerous but simpler
- We gained the chance to reuse the more abstract ones
- We could use a few of them as 'interfaces'

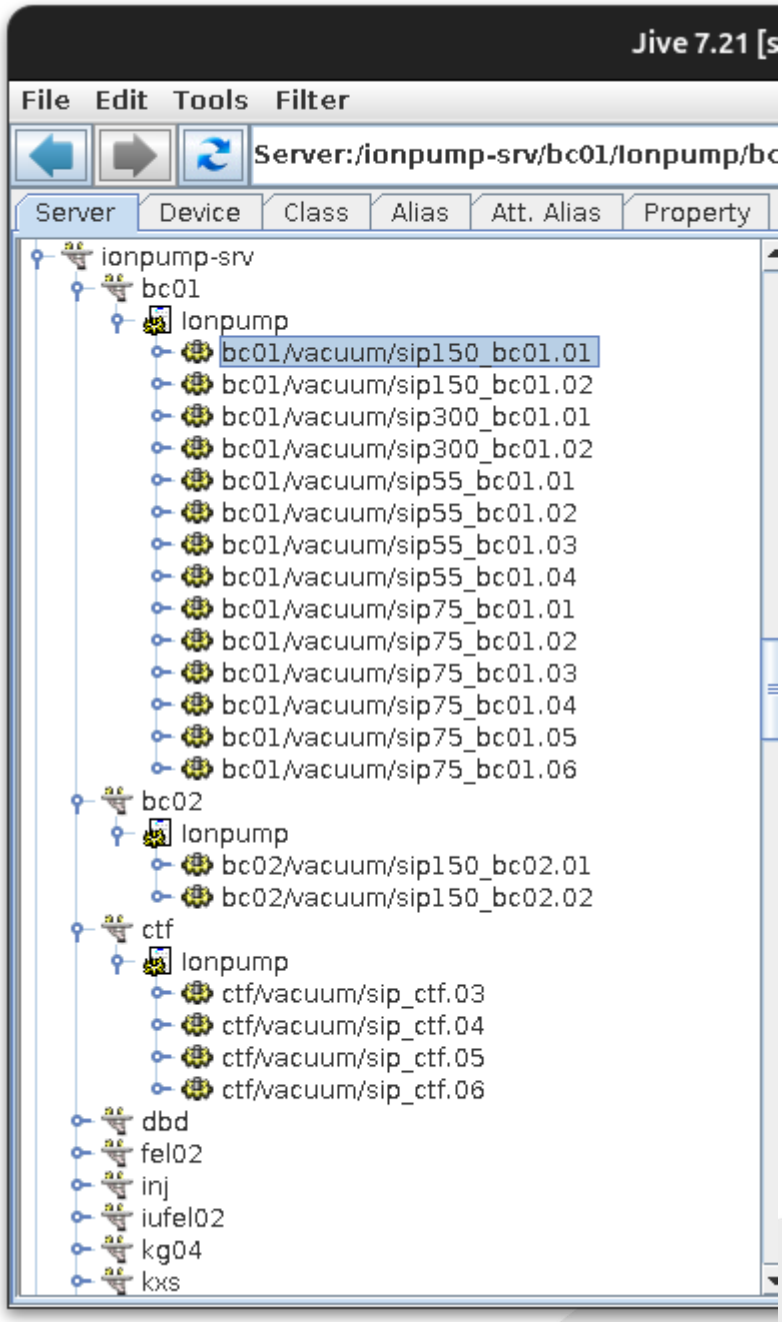
Unfortunately this approach also has its downsides:

- We faced server installation and configuration proliferation
- We experienced difficulty in visualizing devices in the tree (Jive)
- We encountered a high rate of human configuration errors
- We had to handle version management complexity

Elettra Sincrotrone Trieste

A better approach





More sophisticated approach

The Multiclass one:

Try to mitigate the latest problems without adding new ones or reusing the oldest approach:

- Combining multiple Tango classes into a single DS
- Keeping functionally related classes 'close' (also in Jive)
- Simplifying installation, development, and configuration
- Aligning DS classes naturally (in the same executable)
- Updating incrementally without risk to previous installations because all necessary classes are included ("self-contained”)

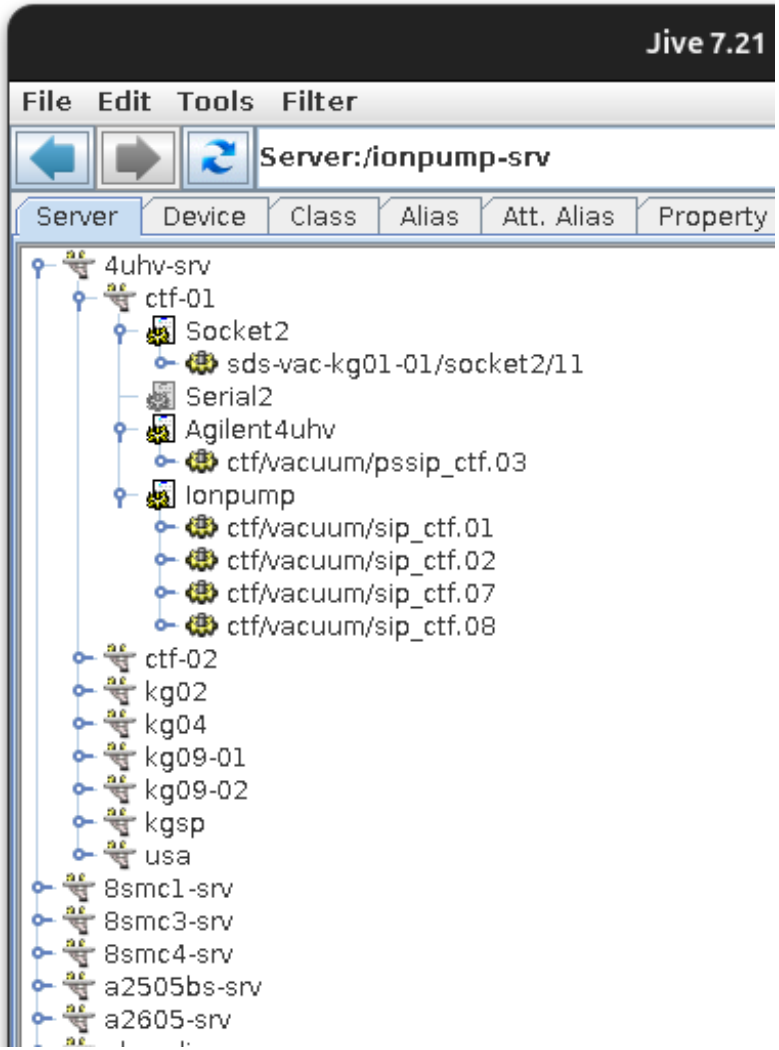
More sophisticated approach

We implement this approach at Elettra by combining:

- git repositories (either managed as single classes or combined in a multiclass as git submodules)
- Simple path convention for multiclass case (deps/repositoryname)
- Use of wildcard (*) in our Makefile

In this way we keep sources (and bug fixes) in one place and avoid git repository proliferation. Additionaly we prevent configuration errors, using custom code, by blocking DS startup if duplicate resources exist or DeviceProxy points outside of the Multiclass.

More sophisticated approach



Future Goals

- **Skip the network stack (CORBA) for internal communications when DeviceProxy recognizes that the origin and destination are within the same Multiclass. This allows it to resolve invocations with little more than an internal function call.**
- **B**y moving beyond the "one instrument == one Tango class == one Tango Device Server" approach, we can use a Tango class as a more generic computational unit for very dense DSs that maintain inspectability with traditional tools (Jive)
- Promote programming by composition

Elettra
Sincrotrone
Trieste

Thank you!

www.elettra.eu