ALMA Common Software

Introduction and Status

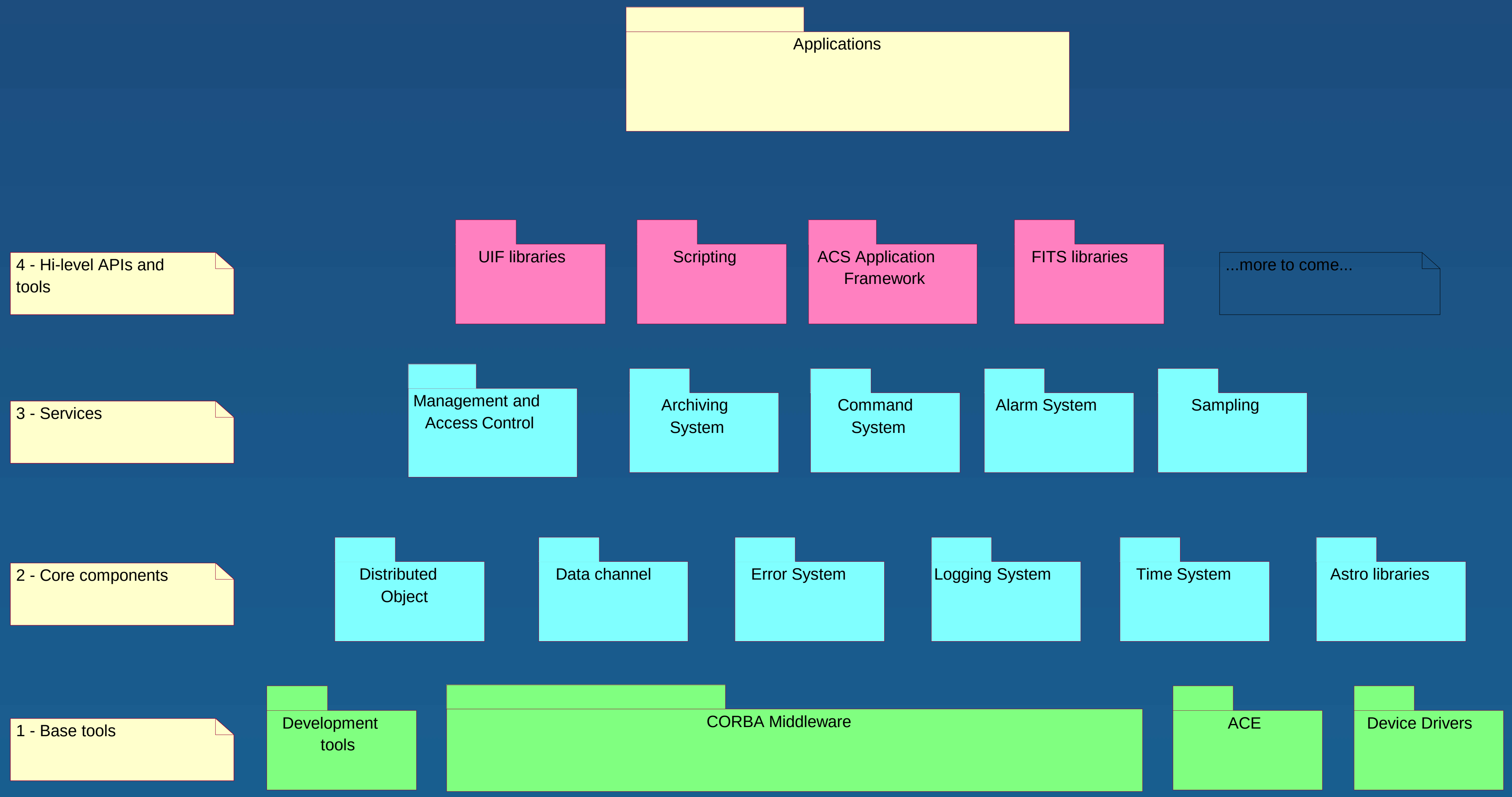
G.Chiozzi - ESO

Contents

- What is ACS?
- Why adopting a Technical Infrastruture Framework?
- Platforms
- ACS main characteristics
- The ACS Community
- Conclusion and lessons learned
- Questions and discussion

What is ACS?

- ACS is a SW Technical Infrastructure for Control Systems.
- ACS provides the basic services for OO distributed computing.
- ACS is based on a Component/Container model
- Development started in 1999 for ALMA, as an open source project.
- ACS is still actively developed.
- ACS is used in a number of projects outside ALMA
- There is a community of users contributing to the development.



Why a Technical Infrastructure?

An observatory is a *distributed system*.

Servers and clients are distributed on different machines:

✧ Possibly in different locations

✧ With different purpose and functionality

✧ With different requirements on performance and reliability

Why a Technical Infrastructure? (2)

An observatory is a *heterogeneous* distributed system.

Servers and clients may use different:

✧ Hardware

✧ System software

✧ Programming languages

Even development is distributed

Why a Technical Infrastructure? (3)

Transparent heterogeneous distribution is desirable:

- ✧ Application developers should be unaware of the underlying server architecture & vice-versa
- ✧ It should be possible to change the architecture of a server transparently to the client
- ✧ Application developers should not even need to know whether a server is local or remote.

Functional and Technical Architecture

Separation of *functional* from *technical* concerns is a strategy for

- ✧ enabling the application developer to focus on the specific aspects of the observatory
- ✧ minimizing the technical effort

Functional Architecture

A Functional Software Architecture (FSA) is a model that identifies enterprise functions, interactions and corresponding information technology needs.

- ✧ Software components/subsystems
  - ✧ Responsibilities
  - ✧ Interfaces
  - ✧ Primary relationships and interactions
- ✧ Physics and algorithms

It is developed by architect and subsystem leaders based on user requirements

Technical Architecture

The functional architecture must be supported by a *technical architecture* that describes (and implements) the technical aspects of the software, like:

- ✧ Programming model
- ✧ Communication mechanisms and networking
- ✧ Access to remote resources
- ✧ Store and retrieve data (Database technology)
- ✧ Manage security
- ✧ Software deployment and life cycle

It is provided by the technical team

typically based on derived requirements

Infrastructure Framework

The key to the separation between

Functional and Technical Architecture

Purpose of a framework is to:

- ✧ provide a programming model
  - ✧ ensure that the same thing is done in the same way in all the development locations
- ✧ provide common paradigm abstractions
- ✧ mask heterogeneity
- ✧ satisfy performance, reliability and security requirements

Which framework to chose?

All big projects have adopted an infrastructure framework

ACS in just one among several options, like

- ACS
- EPICS
- TANGO
- ESO VLT CCS
- ESO ELT CII
- ………

They are all rooted on the same basic principles described above.

They make specific technical choices and have an own history and a rationale for adopting any of them in a project, or to create a new one.

The ALMA Common Software (ACS) Framework

- ACS provides the basic services needed for OO distributed computing. Among these:
  - Transparent remote object invocation
  - Object deployment and location based on container/component model
  - Distributed error and alarm handling
  - Distributed logging
  - Distributed events / publisher-subscriber
  - Configuration database
- The ACS framework is based on CORBA and built on top of free CORBA implementations and services.
- Model driven development with code generation

Supported Platforms

- Operating system:
  - RH Enterprise / Scientific Linux
  - CentOS
  - Other linux versions supported by external projects
  - Windows added also by external initiatives
- Real-time:
  - VxWorks supported by and for APEX
- Languages: C++, Java, Python
- CORBA middleware: TAO (C++), JacORB (Java), Omniorb (Python), CORBA services.
- Embedded ACS Container (Experimental)

LGPL and open source software

The strategy to provide common features to users is:

- ✧ Integrate as much as possible open-source tools, instead of implementing things.
  - ✧ Do not reinvent the wheel
  - ✧ Reuse experience of other projects
  - ✧ Do not pay for licenses
  - ✧ Support from user community
- ✧ Identify the best way to perform a task among the possibilities
- ✧ Wrap with convenience and unifying APIs

ACS is distributed under the LGPL license

Open source software may have drawbacks:

- ✧ Fast lifecycle and support only of the newest
- ✧ Free/commercial support
- ✧ Documentation not as good as commercial products

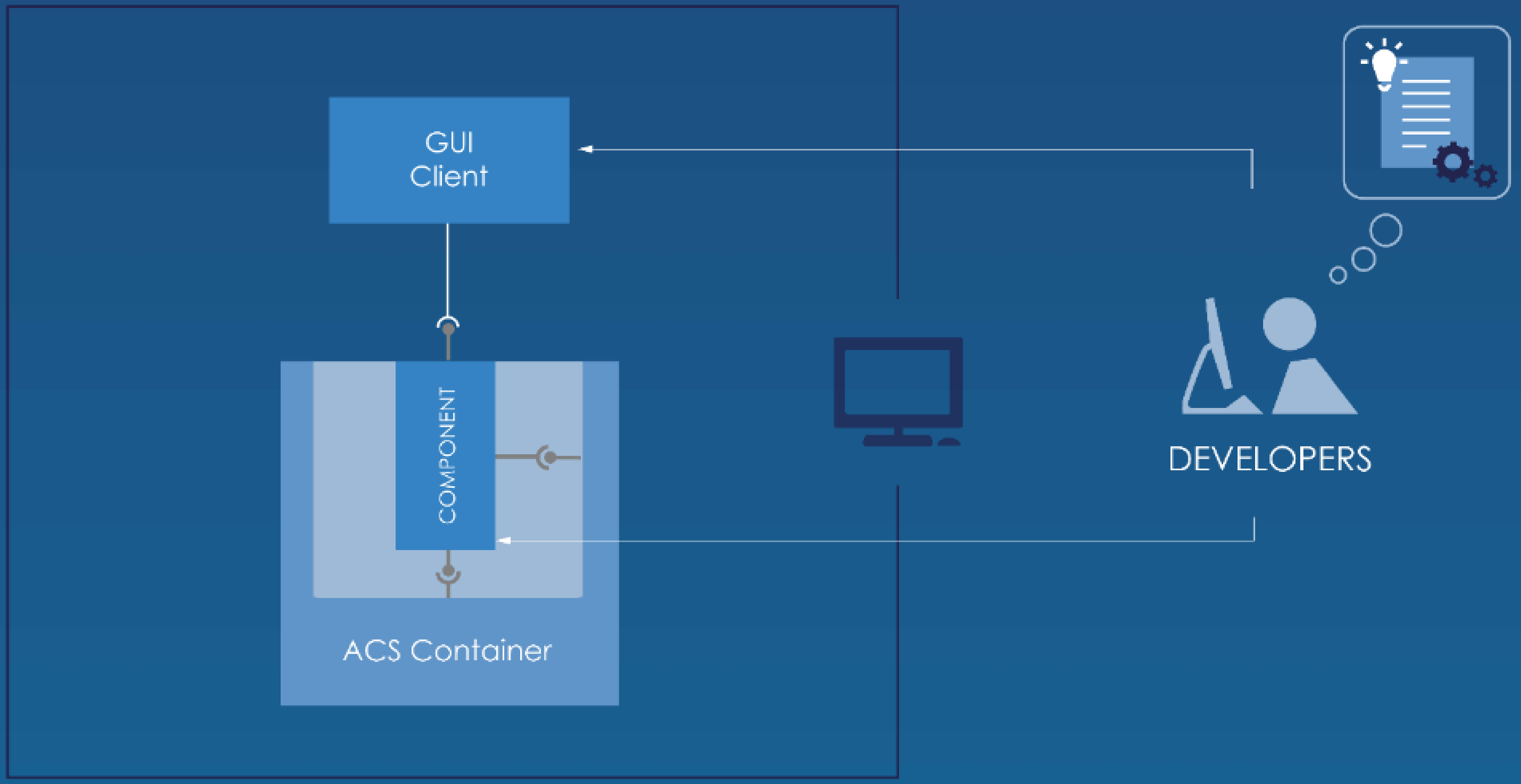
Separation of roles

ACS keeps separate 3 roles/phases:

- ✧ Development by software developers
- ✧ Deployment by operations engineers
- ✧ Runtime by system operators (clients)

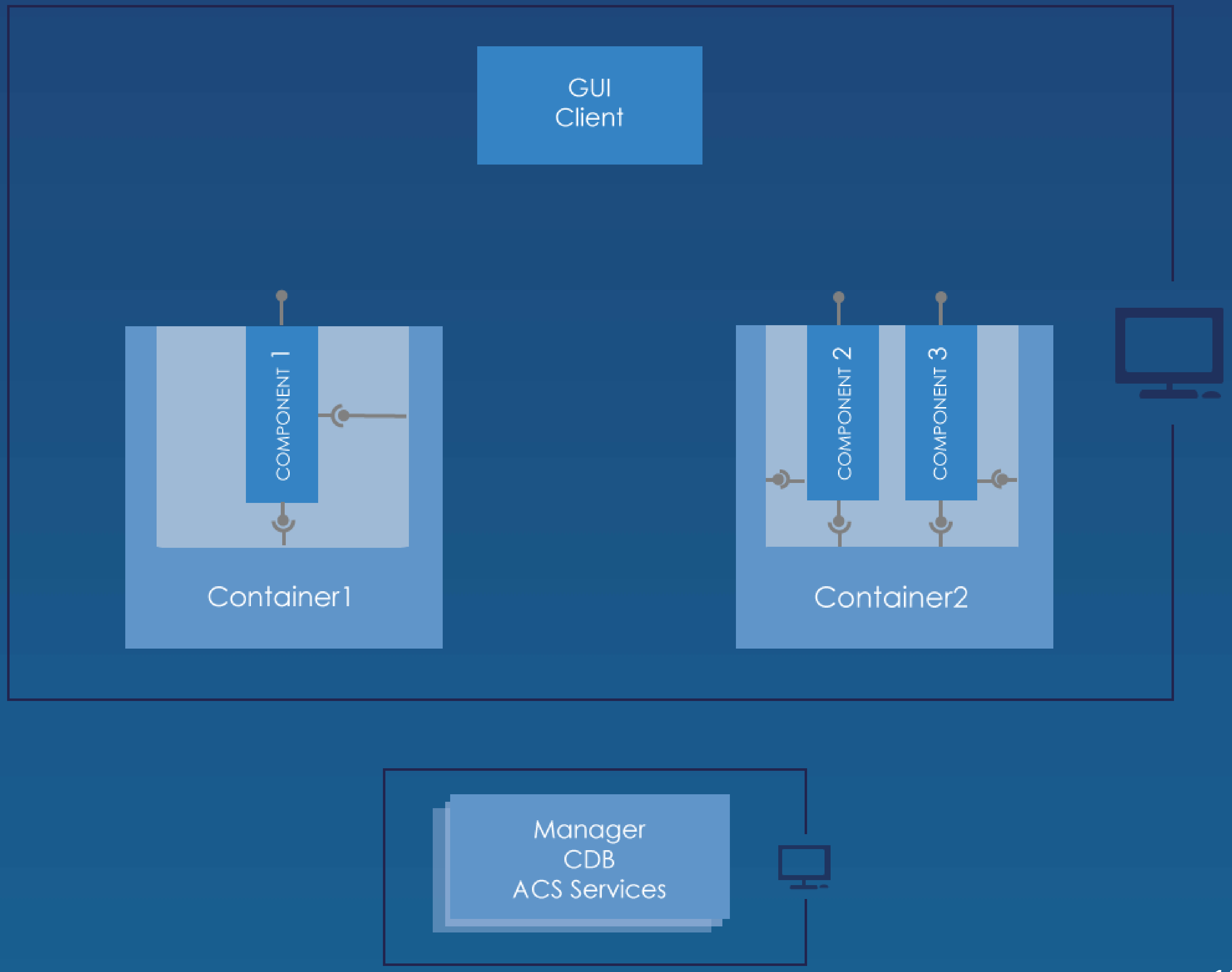
Development

- ✧ Developers write components and graphical user interfaces clients in C++, Java, or Python.
- ✧ ACS provides an integrated build environment based on application code modules.
- ✧ Communication from an application to a component, and among components, uses ACS as middleware.
- ✧ No thinking about starting and stopping components, or on which machine they should run later.



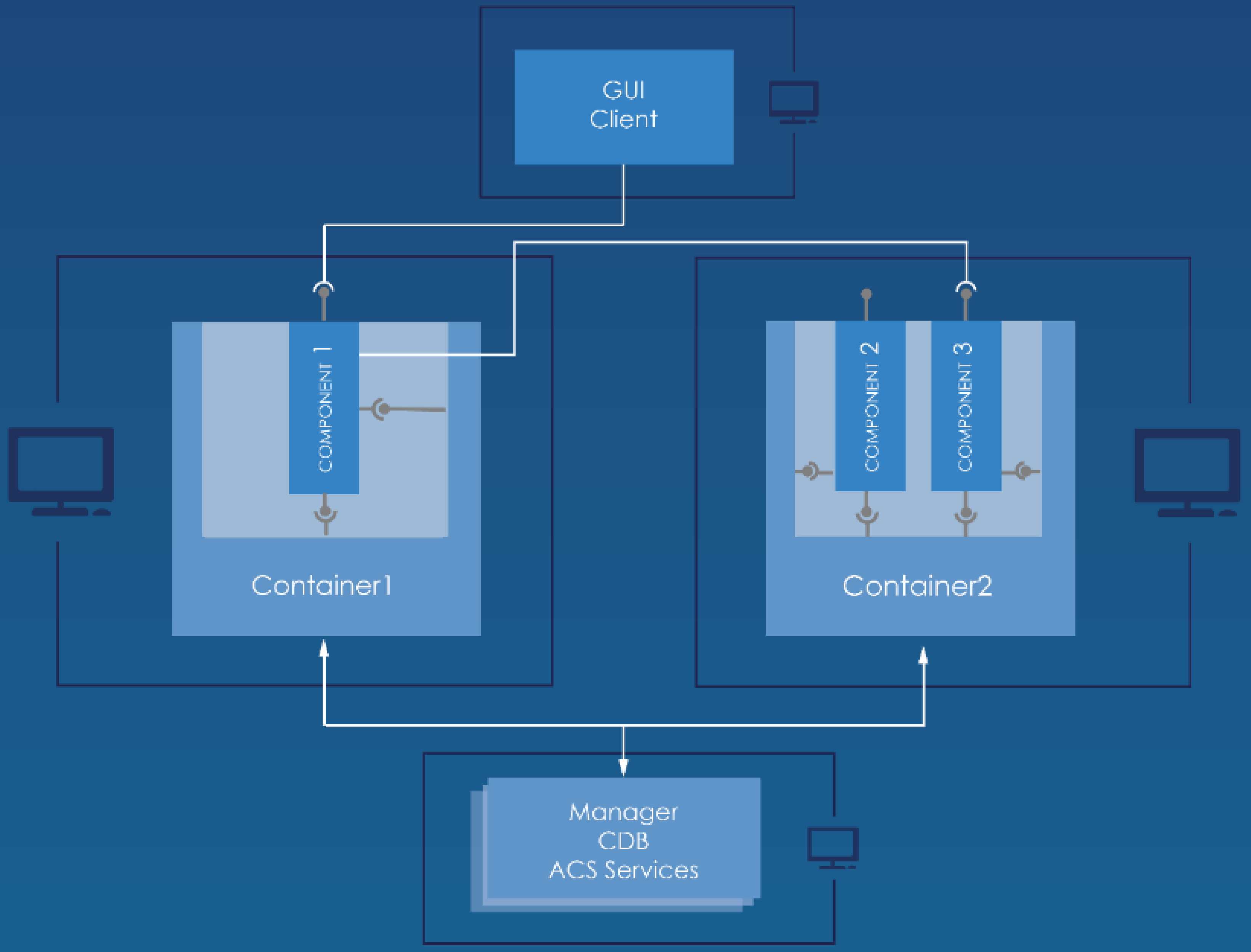
Deployment

- One or more containers get assigned to each computer.
- Components get assigned to containers.
- This location information is stored centrally in the Configuration Database (CDB).
- Other configuration data for containers and components are also stored in the CDB.
- There can be different deployments for unit tests, system tests, and various stages of the production system.



Runtime

- ✧ ACS containers start and stop components (lifecycle management) as needed.
- ✧ Containers provide components and clients with references to other components.
- ✧ The Manager is the central intelligence point that keeps the system together. Components never see it directly.

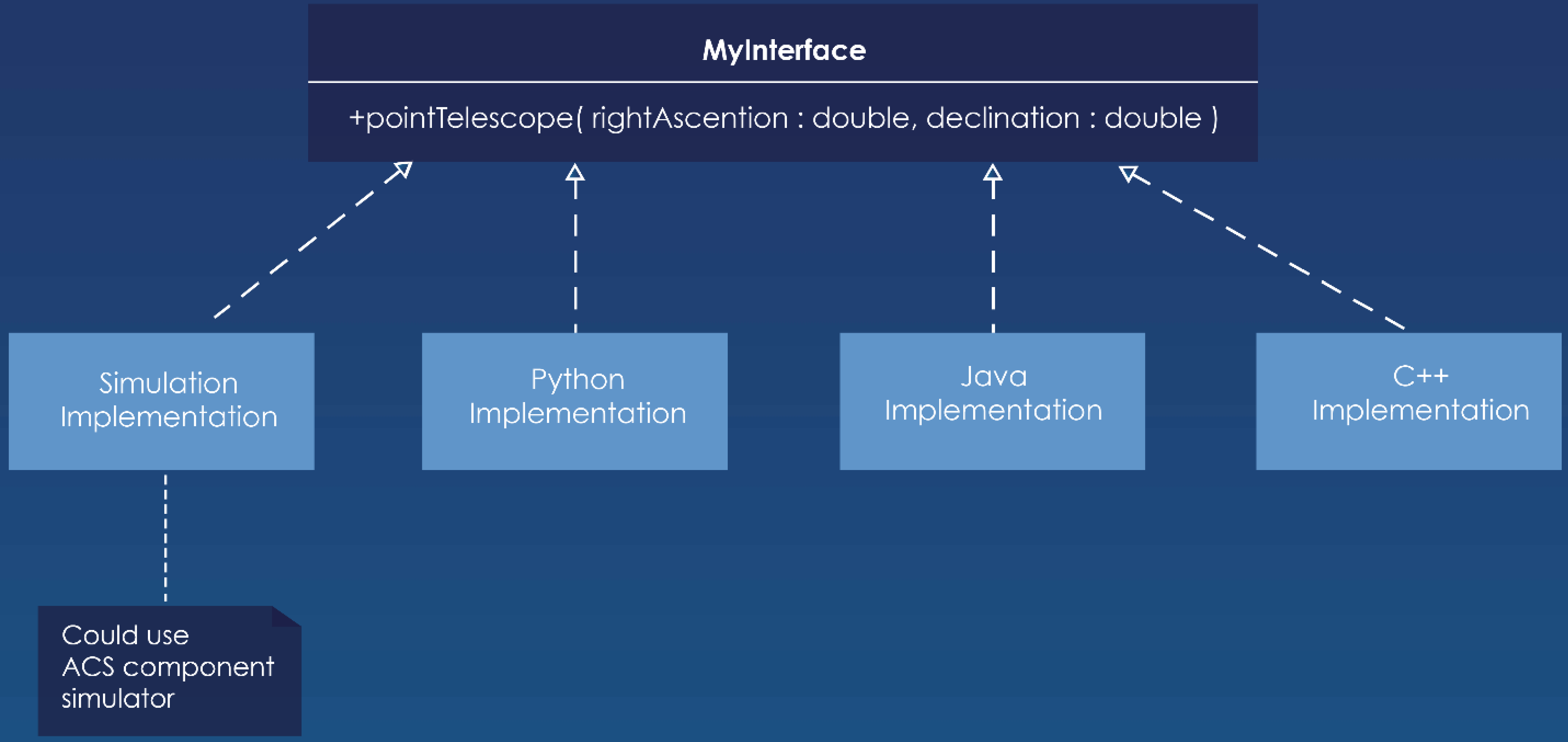


Interfaces versus Implementations

The contract between components is specified by defining interfaces.

- First step: Identify objects
  - Mount
  - Camera
  - Telescope
  - Observation
  - Exposure
- Second step: Define interfaces
  - Implementation comes later and is independent of interface
  - Deployment is also independent of interface definitions
  - Interfaces shall be kept as stable as possible, but it must be possible to have them evolve when needed.
  - A formal interface definition language is needed

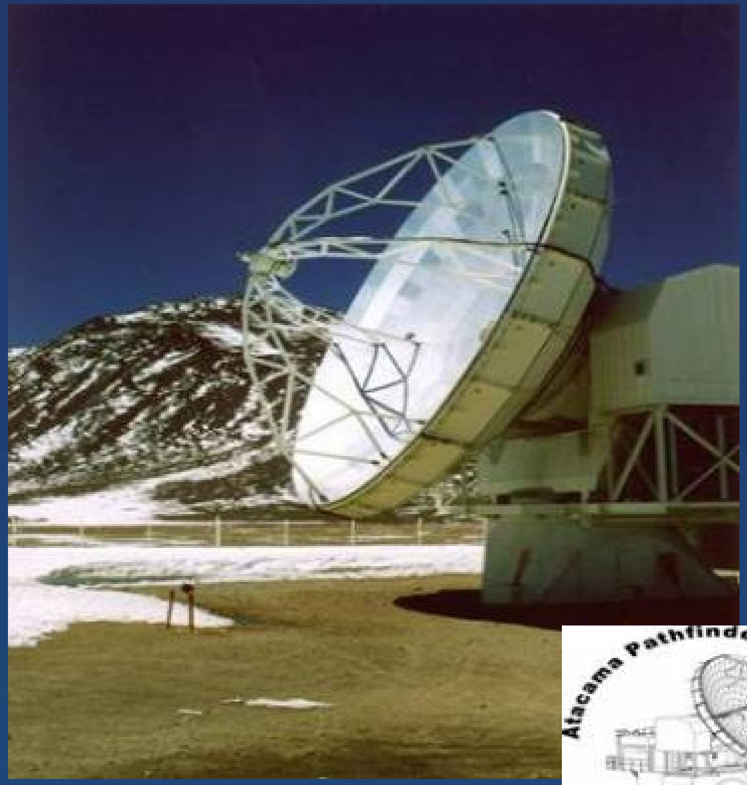
One Interface, many implementations


MyInterface
+pointTelescope(rightAscention : double, declination : double)
Simulation Implementation
Python Implementation
Java Implementation
C++ Implementation
Could use ACS component simulator

The ACS Community

- ALMA
- APEX
- CTA / ASTRI
- SRT / DISCOS
- LLAMA (Argentina)
- Yebes Observatory RT40m (Spain)
- HESS
- …. Some other smaller or perspective projects ….

- Strong expertise in Italy

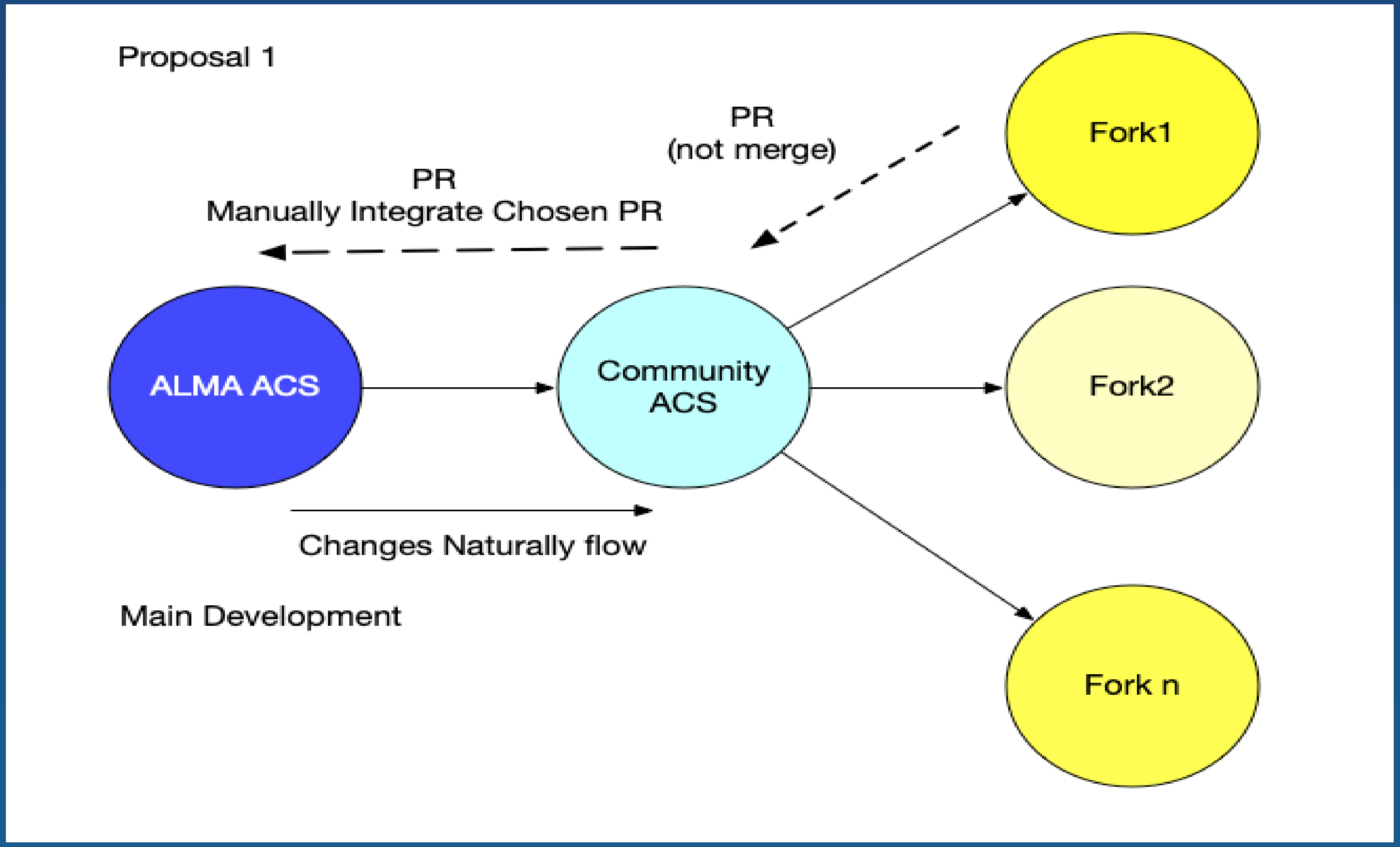

APEX (Chile)

ASTRI (Italy)

Yebes RT40m (Spain)

How does the community work today?

- ALMA is leading ACS Maintenance (2 FTEs) and Development (Best Effort)
  - Focused on ALMA’s priorities
- Preparing releases and making them available to the community
- Receiving questions, requests and suggestions from community
- Receiving patches and integrating them in ACS
- Creating tickets, following up and resolving them
- Organization of community meetings and workshops
  - Last workshop: July 2020 About 80 participants



- Web Confluence page: https://confluence.alma.cl/display/ICTACS/ACS+Community

ACS Community Objectives

- Increase Community Collaboration
  - Identify Current Community
  - Releases planning
  - Issue Tracking
  - Building / Packaging / Distributing
- Increase Community Engagement
  - More frequent community meetings
  - Better means of communication (Slack, Issue Tracking, etc.)
- Improve ACS Visibility
  - Website + Confluence
  - Improve Documentation
  - DockerHub Official Docker Image + Dev Images
  - ACS Community Slack Page
- Modernize the Framework
  - Replacement of technologies
  - New developments
  - Improve performance

Conclusions and lessons learned

- By now an “old” product
  - >20 years since inception
  - Is CORBA obsolete?
- Very stable and reliable: many years of continuous operation
- Actively supported by ALMA
- It is very difficult to engage the community in contributing
- Adoption pays off in relatively big projects
- What brakes adoption?
  - Steep initial learning curve.
    Higher level tools and more code generation would help.
  - Good documentation is critical
  - Not modular. Splitting in multiple independent packages would help but where to get resources with a relatively small community?
- ACS is getting new energy with projects like CTA and ASTRI
- There is wide expertise in Italy: it might be useful for new projects
- How to choose between the available alternative options?

Questions?

Acknowledgements
Most slides for this presentation are taken from the presentations given at the ACS Workshop 2020. ACS presentations were originally developed by the ACS team, used in many training courses since 2004. Main contributors are (listed in alphabetical order): Jorge Avarias, Alessandro Caproni, Gianluca Chiozzi, Jorge Ibsen, Bogdan Jeram, Thomas Jürgens, Matias Mora, Joseph Schwarz, Heiko Sommer plus many others

The Atacama Large Millimeter/submillimeter Array (ALMA), an international astronomy facility, is a partnership of Europe, North America and East Asia in cooperation with the Republic of Chile. ALMA is funded in Europe by the European Organization for Astronomical Research in the Southern Hemisphere (ESO), in North America by the U.S. National Science Foundation (NSF) in cooperation with the National Research Council of Canada (NRC) and the National Science Council of Taiwan (NSC) and in East Asia by the National Institutes of Natural Sciences (NINS) of Japan in cooperation with the Academia Sinica (AS) in Taiwan. ALMA construction and operations are led on behalf of Europe by ESO, on behalf of North America by the National Radio Astronomy Observatory (NRAO), which is managed by Associated Universities, Inc. (AUI) and on behalf of East Asia by the National Astronomical Observatory of Japan (NAOJ). The Joint ALMA Observatory (JAO) provides the unified leadership and management of the construction, commissioning and operation of ALMA.