ALMA Common Software

Basic Track

Characteristic components
BACI properties and DevIO classes

I. Oya, CTAO gGmbH

Component Model

- An ACS component is a piece of software that is executed within a container running on a given machine
  - Container spawn threads for component execution
- ACS implements a distributed object model
- Components are CORBA objects that are remotely accessible from other computers through the client-server paradigm
- A Component is the natural base class for physical and logical "devices"
- ACS components follow a standard component lifecycle

Components and characteristics components

- Abstraction of hardware devices
  - Actions
  - Control/monitor points
  - Characteristics
- A characteristic component aggregates Characteristics and BACI properties of different data types:
  - BACI: Basic Access Control Interface. based on the Component - Property – Characteristic, standard in control systems
  - Characteristics: static data store in the CDB
    - units, default values, monitor*, alarm*, archive*
- All telescope components such as mount, control units, power supplies are characteristic components
- Same structure of components/devices

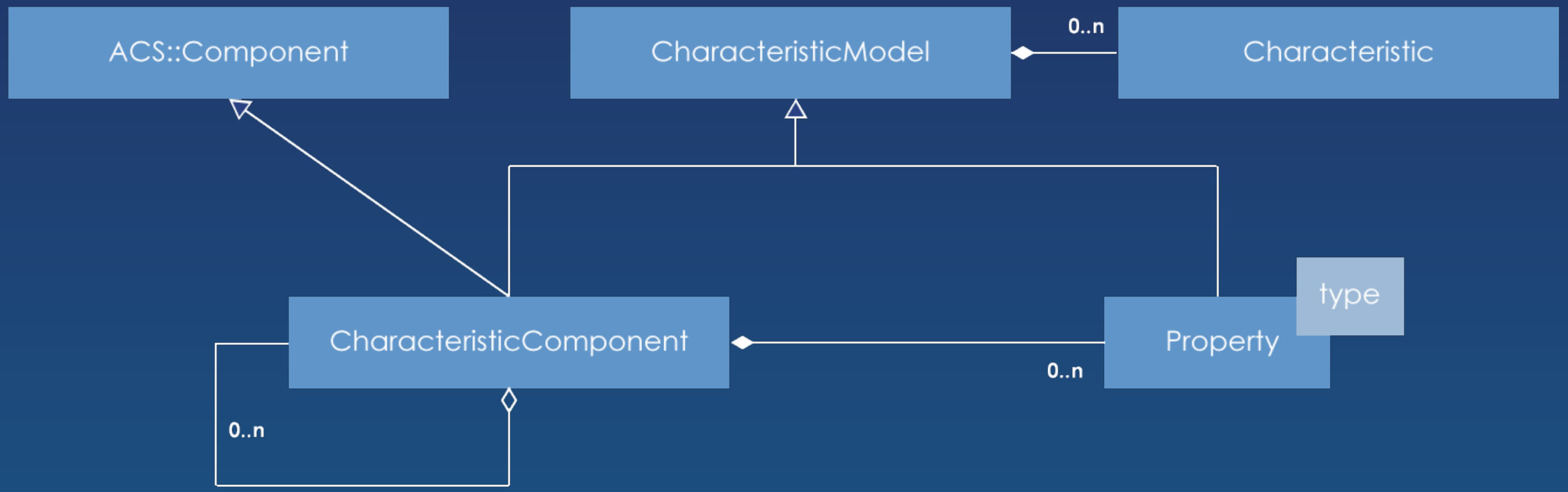
BACI property

- ✧ High-level representation of a monitoring or control point/entity
- ✧ It is a statically defined item
- ✧ It has a value and attributes
- ✧ The value is strongly typed
- ✧ Only basic types are available
  - ✧ double, long, string, pattern, enum, longSeq
  - ✧ limited unsigned support
- ✧ Read-only (RO) and read-write (RW) access
- ✧ Defines an interface, which is extended by developer
  - ✧ Developer implements functions read() and write() functions
- ✧ Combines value(s) with “attributes”
  - ✧ Description
  - ✧ Unit
  - ✧ Monitoring parameters
  - ✧ Alarms thresholds

BACI property (continued)

- ✧ All properties have the same attributes!
  - ✧ This cannot be modified
- ✧ Clients can get / set methods
  - ✧ Synchronous and asynchronous
- ✧ Clients can monitor property values (callback mechanism)
  - ✧ Interval
  - ✧ On change
  - ✧ Keeps history (last 10 values)
- ✧ Value archiving
  - ✧ Same as for monitoring
- ✧ Alarms build-in

Components and characteristics components



- **Component:** software representing a physical/logical device (e.g. temperature sensor, motor)
- Each Component can have **Properties** (e.g. status value, position - control/monitor points).
- **Characteristics** of Components and Properties (static configuration data, e.g. serial number, CAN-Bus-ID, default value)

Example of Characteristic component

```
interface PowerSupply : ACS::CharacteristicComponent
    {
    void on(…);
    void off(…);
    void reset(…);

    readonly attribute ACS::RWdouble current;
    readonly attribute ACS::ROdouble readback;
    readonly attribute ACS::ROpattern status;
    };
```

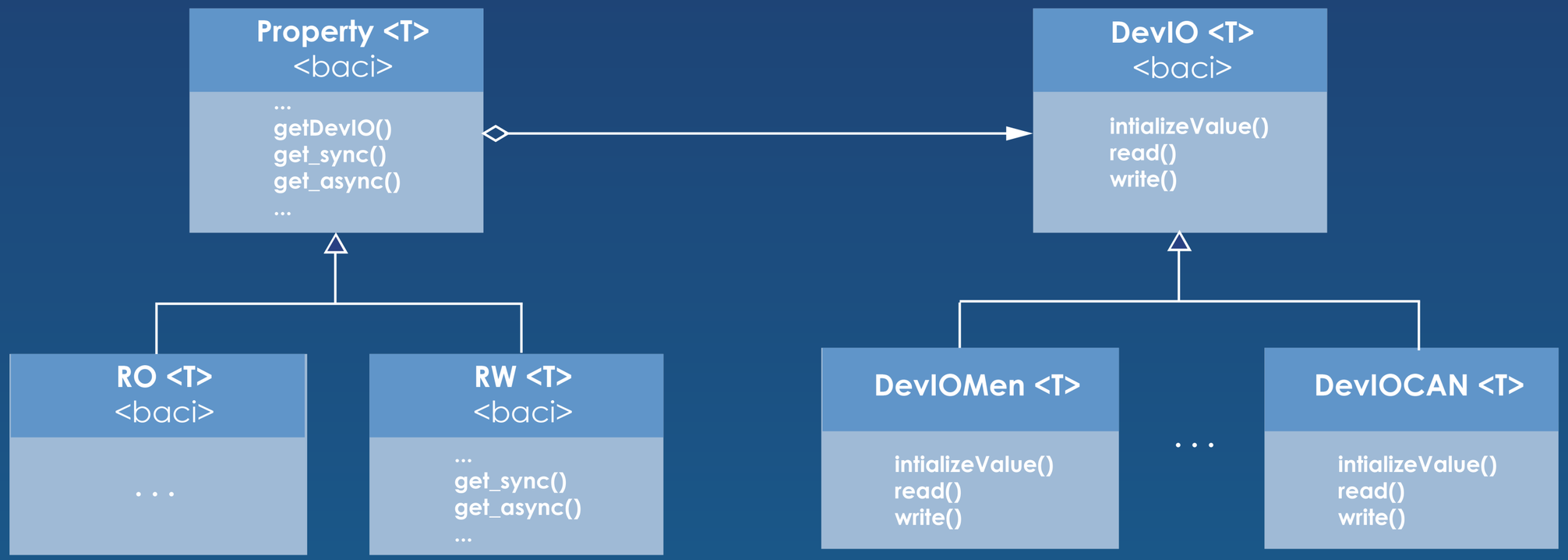
DevIO classes

- ✧ Provides the “value” part in BACI properties
- ✧ Bridge design pattern – access actual hardware
- ✧ Can be extended for real hardware
- ✧ Can be extended for simulation purposes (f.i. DevIOMem)
- ✧ Does not prevent race conditions
- ✧ Does not take care of device init, etc.
- ✧ Does not do error handling when hardware fails
- ✧ Decouple software and hardware implementing a bridge pattern
  - ✧ read() / write() / initializeValue() methods can be overloaded

- ✧ Examples existing:
  - ✧ Memory location (ACS default implementation)
  - ✧ CAN bus access (ALMA)
  - ✧ Socket generic interface (APEX)
  - ✧ …
  - ✧ Generic DevIO for OPU UA communication (D. Melkumyan, CTA)

DevIO example class: DevIOMem

- ✧ Inherits from DevIO
- ✧ Useful for simulation and testing
- ✧ Implements read(), write(), initializeValue() methods
- ✧ Very flexible



DevIO and device drivers

- Usual use case that several BACI properties of a component need to share state across DevIOs instances.
- For example, if the device uses a serial line, we do not want to open a connection per property.
- In that case the usual pattern is to create a device driver class that is a singleton and handles the access to the device information.
- It can manage data caching there as well.
- Many examples in Alma source code.

Questions?

Acknowledgements
ACS presentations were originally developed by the ALMA Common Software development team and has been used in many instances of training courses since 2004. Main contributors are (listed in alphabetical order): Jorge Avarias, Alessandro Caproni, Gianluca Chiozzi, Jorge Ibsen, Thomas Jürgens, Matias Mora, Joseph Schwarz, Heiko Sommer.

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.