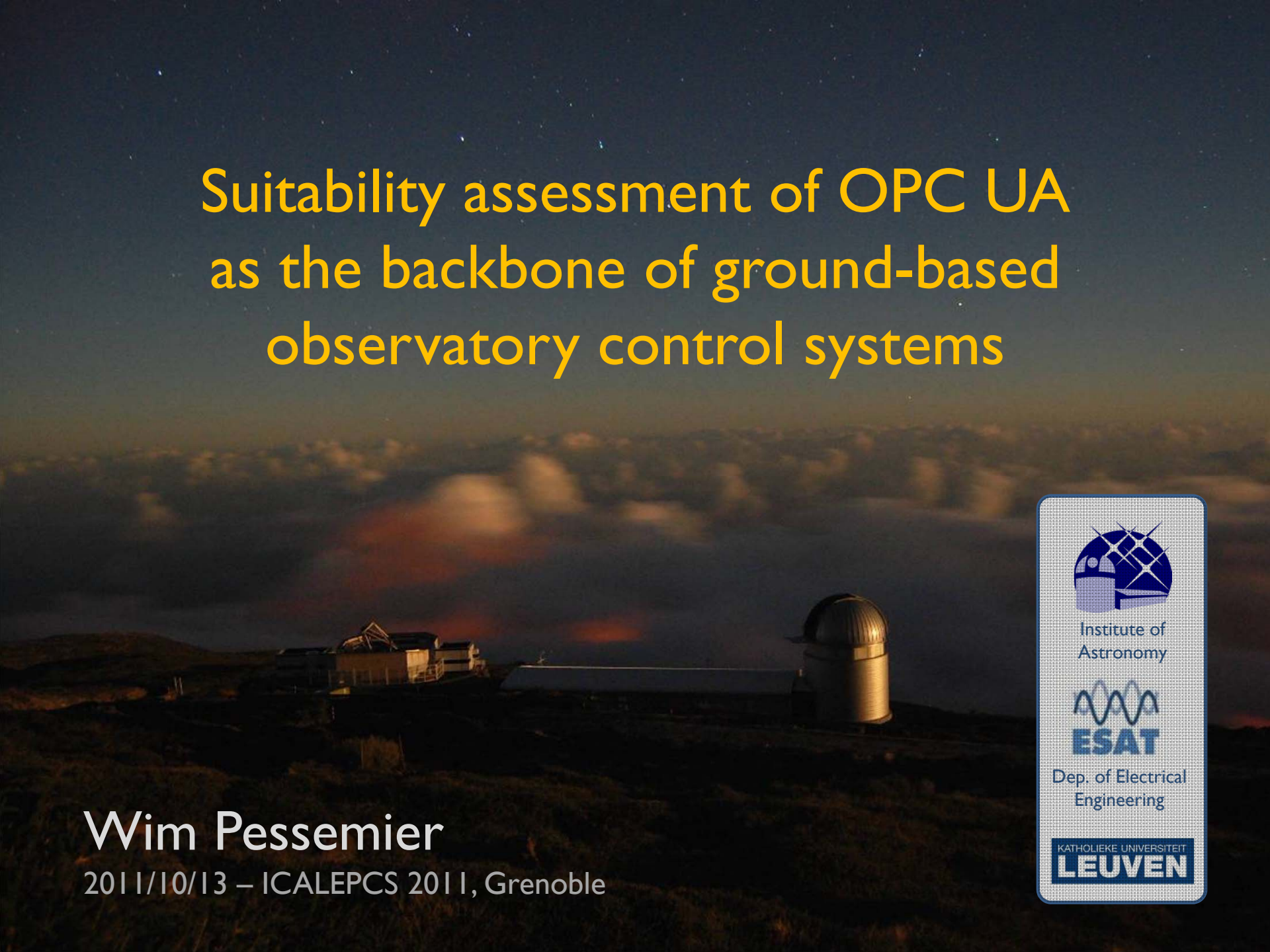
# Suitability assessment of OPC UA as the backbone of ground-based observatory control systems




Wim Pessemier

Institute of Astronomy

ESAT

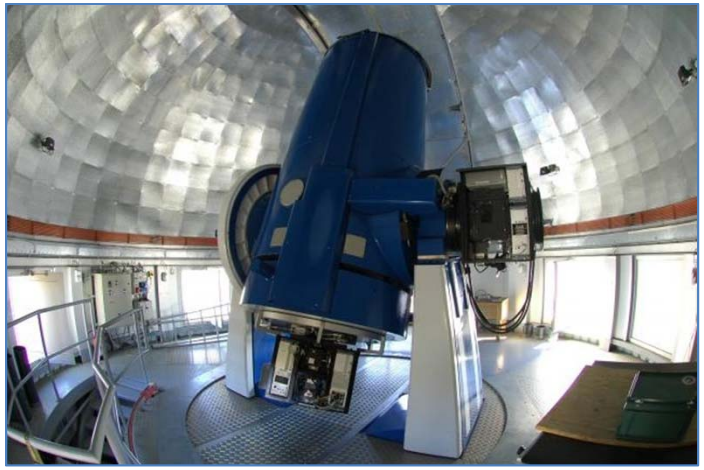
Dep. of Electrical Engineering

KATHOLIEKE UNIVERSITEIT LEUVEN


2011/10/13 – ICALEPCS 2011, Grenoble

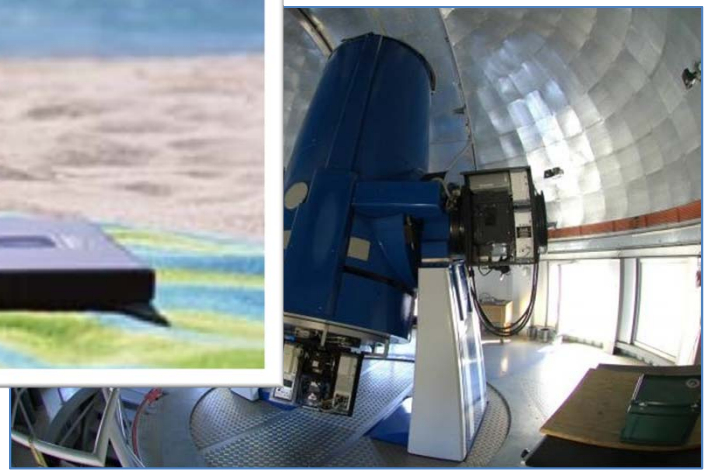
# Mercator Telescope

- 1.2 m optical telescope with
  - Fiber-fed echelle spectrograph (R~85000)
  - 2K*6K FT CCD camera
  - 3-channel 2K*6K FT CCD camera (‘12)
- La Palma (Canary Islands, Spain)
- Transputer based legacy TCS is gradually being replaced by a **Python-based framework** providing:
  - common services
  - communication drivers



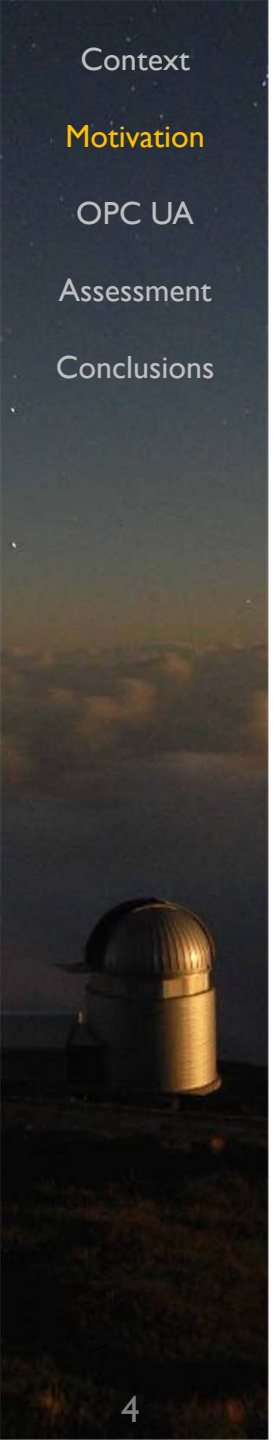
# Mercator Telescope

- 1.2 m optical telescope with
  - Fiber-fed echelle spectrograph (R~85000)
  - 2K*6K FT C
  - 3-channel 2K
- La Palma (Ca
- Transputer ba
  gradually beir
  **Python-bas**
  providing:
  - common ser
  - communicati



# However ...

- Even in our own (little) *Mercator* project:



# However ...

- Even in our own (little) *Mercator* project:

```
while stage.status() == STS_MOVING:
 try:
   if stage.sensors.home.isActive():
     pos = stage.drive.position()
```

We need an MSc in software engineering and Linux system administration to control (or troubleshoot) our instrument/dome/...

# However ...

- Even in our own (little) *Mercator* project:

```
while stage.status() == STS_MOVING:
 try:
   if stage.sensors.home.i
     pos = stage
```

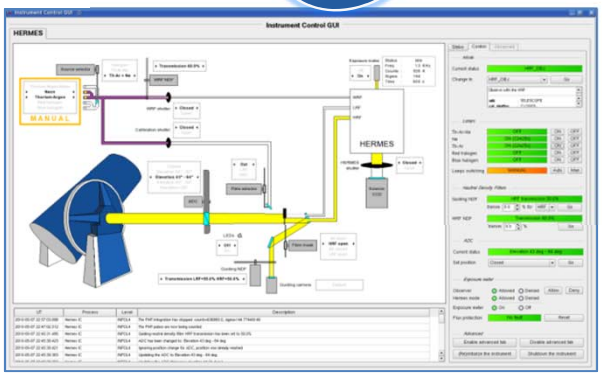
We nee an S software engineering and Linux system administration to control (or troubleshoot) our instrument/dome/...

OPC UA

# However ...

- Even in our own (little) *Mercator* project:

```
while stage.status() == STS_MOVING:
 try:
   if stage.sensors.home.i
     pos = stage
```




OPC UA

We nee an S ftware engineering and Linux system administration to control (or troubleshoot) our instrument/dome/...

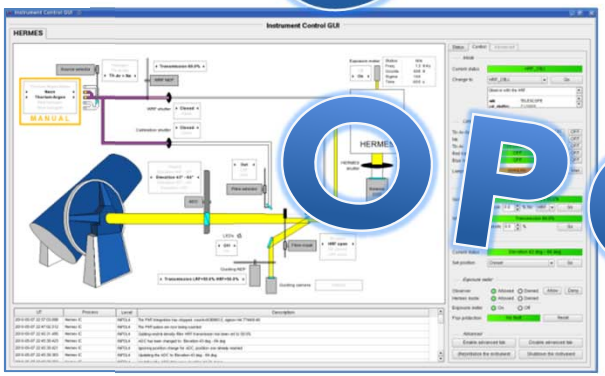
We spend too much time on developing SCADA software...

# However ...

- Even in our own (little) *Mercator* project:

```
while stage.status() == STS_MOVING:
 try:
   if stage.sensors.home.i
     pos = stage
```

We nee an S ftware engineering and Linux system administration to control (or troubleshoot) our instrument/dome/...




We spend too much time on developing SCADA softwre.

OPC UA

OPC UA

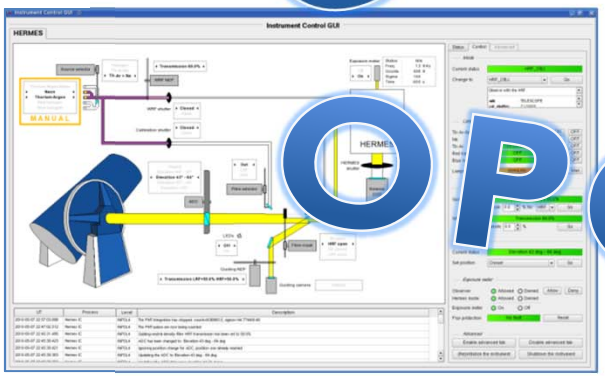
# However ...

- Even in our own (little) *Mercator* project:

```
while stage.status() == STS_MOVING:
 try:
   if stage.sensors.home.i
     pos = stage
```

We nee an S ftware engineering and Linux system administration to control (or troubleshoot) our instrument/dome/...

OPC UA




We spend too much time on developing SCADA softw re.

OPC UA

We get complaints of other developer teams that want to use the framework services, but feel too constrained by its implementation details

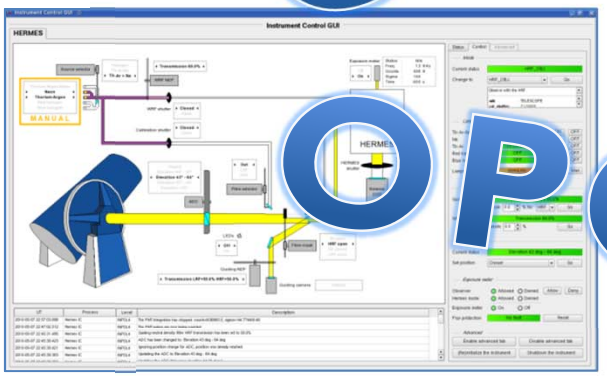
# However ...

- Even in our own (little) *Mercator* project:

```
while stage.status() == STS_MOVING:
 try:
   if stage.sensors.home.i
     pos = stage
```

OPC UA

We nee an S ftware engineering and Linux system administration to control (or troubleshoot) our instrument/dome/...




OPC UA

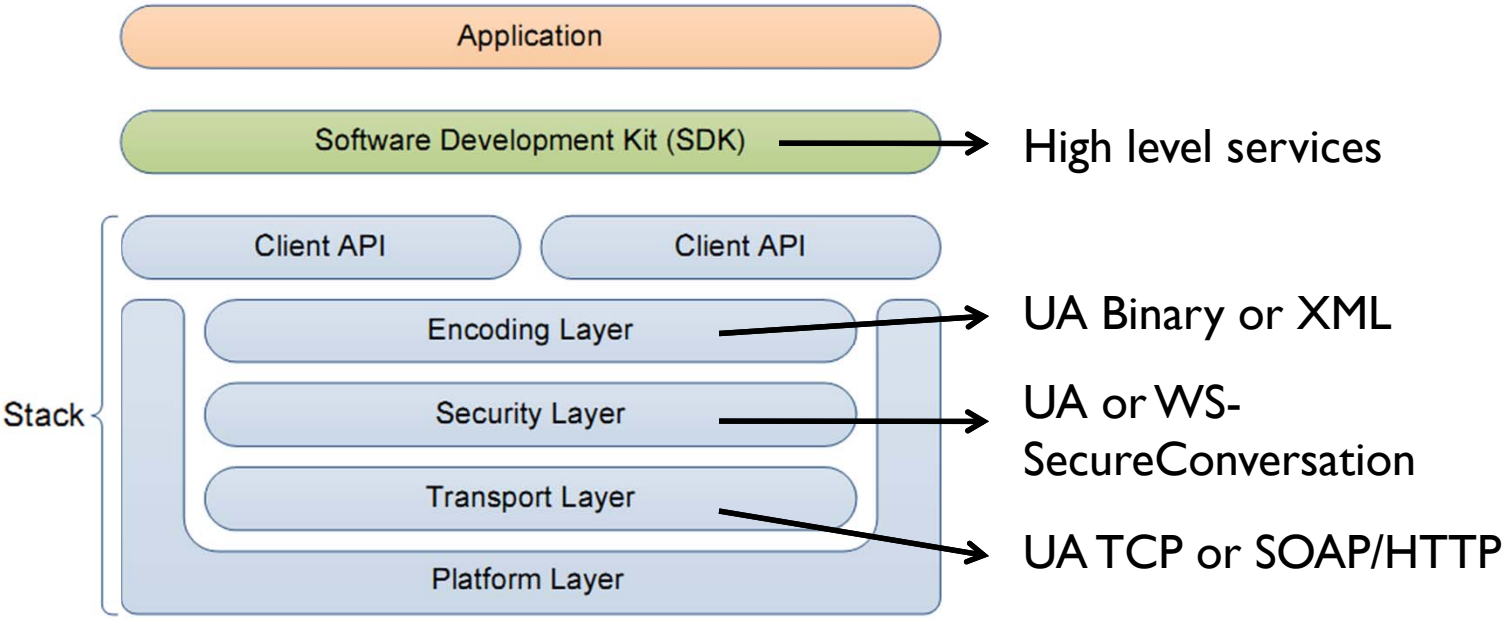
We spend too much time on developing SCADA softwre.

OPC UA

We get complaints of other developer teams that want to use the framework services, but feel too constrained by its implementation details

# OPC Unified Architecture

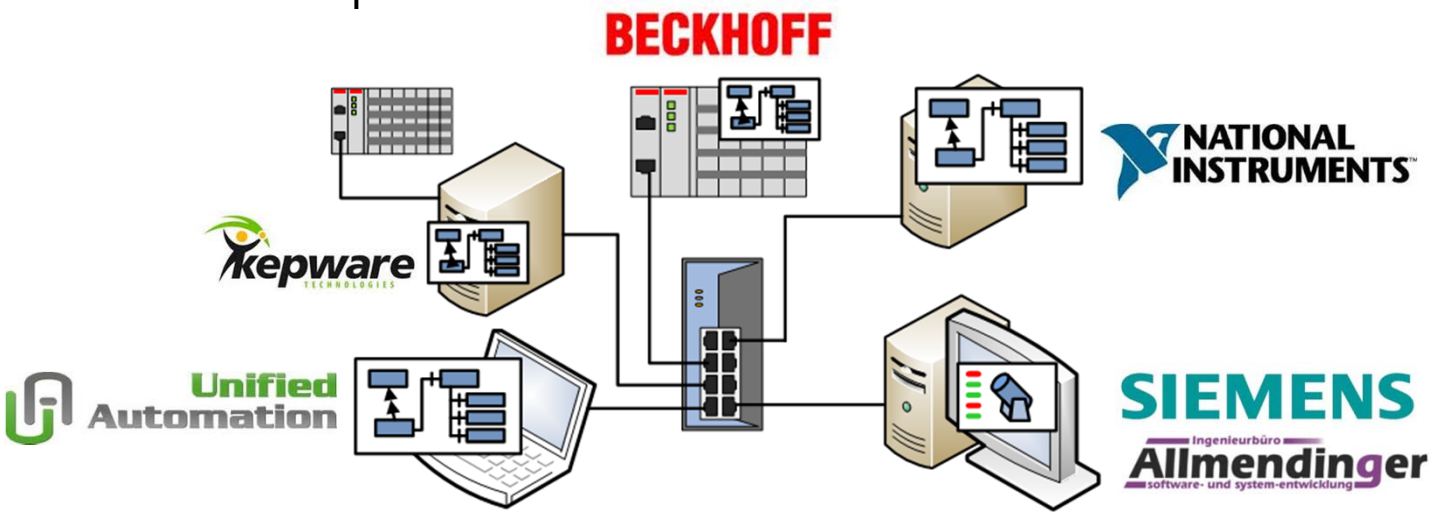
- "Classic" OPC functionality
  New concepts and functionality
  New technology
  } OPC UA
- Service Oriented Architecture
- Specification documents + code deliverables



# Assessment

- Scope:
  - Requirements analysis
  - Feasibility analysis
- Approach:
  - Analyze OPC UA specification
  - Analyze COTS implementations: completeness, maturity

➔ Test set-up:



# Requirements

- Platform independence
- Scalability
- Reusability
- Communication paradigms
- Complex data
- Alarms
- Logging
- Location transparency
- Historical archive
- Dependability
- Lifecycle management
- Performance

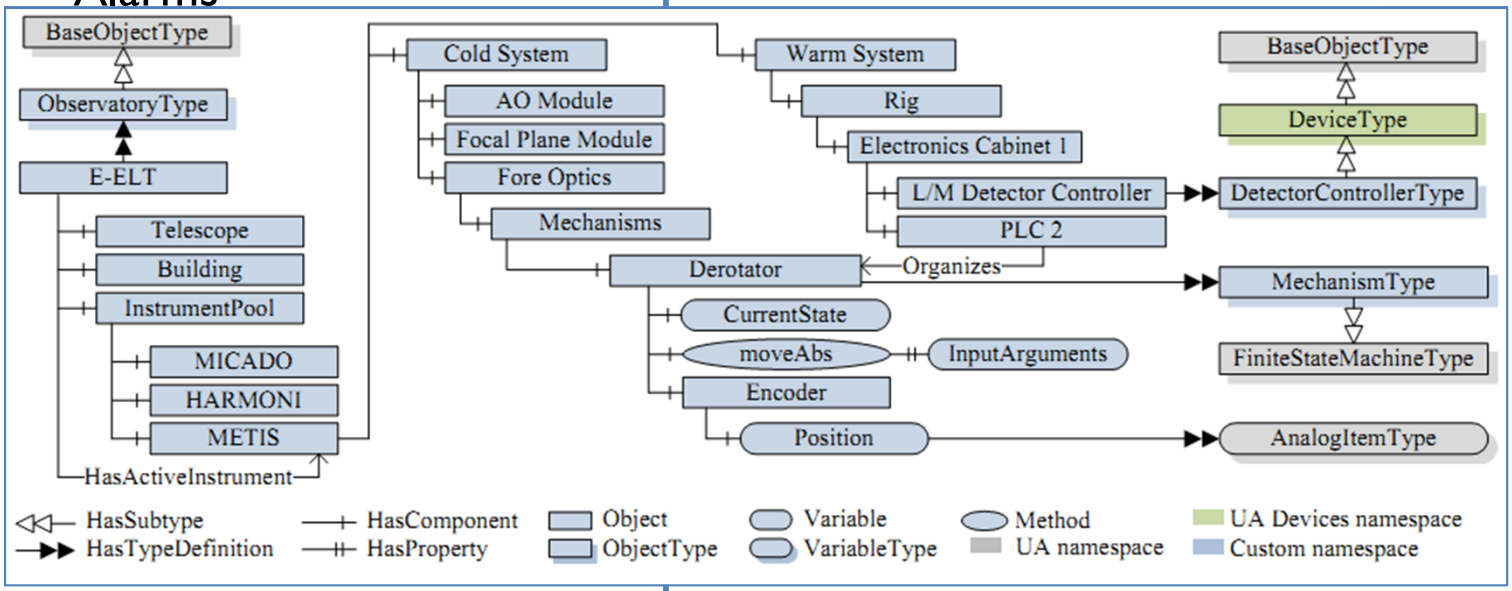
# Requirements

- Platform independence
- Scalability
- Reusability
- Communication paradigms
- Complex data
- Alarms
- Logging
- Location transparency
- Historical archive
- Dependability
- Lifecycle management
- Performance

# Requirements

- Platform independence
- Scalability
- Reusability
- Communication paradigms
- Complex data
- Alarms

- Vertical reusability of industrial products becomes easier
  - SCADA
  - BAS
  - Safety controllers
  - ...
- Horizontal reusability via standardized type definitions → companion standards

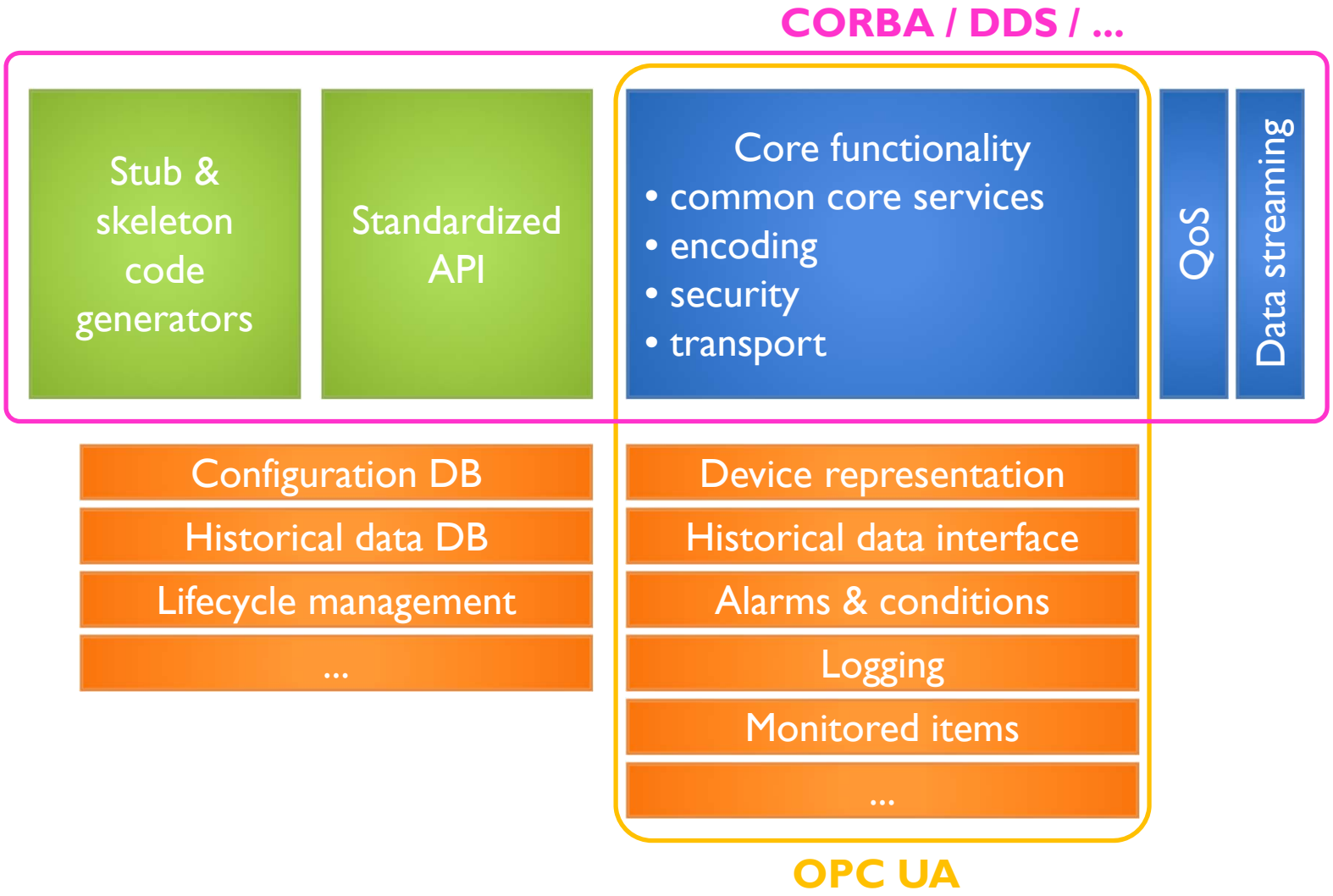


# Requirements

- Platform independence
- Scalability
- Reusability
- Communication paradigms
- Complex data
- Alarms
- Logging
- Location transparency
- Historical archive
- Dependability
- Lifecycle management
- Performance

- Reliability and availability
  - Heartbeat in both directions
  - Lost connection ~~=>~~ lost data
  - Redundancy for clients and servers
  - ...
- Security
  - Signing
  - Encryption
  - Certificates
- Maintainability
  - Extensible (encodings, transport)
  - No standardized API
    - ➔ dependent on particular SDK

# More than middleware



# Commercial support

- SDKs: all required functionality is supported, but binary only (↔ official C# stack)
- The rest: still very limited
  - Data access is well supported (read, write, monitored items)
  - Alarms and Conditions are poorly supported (Beckhoff PLCs, UaExpert)
  - Historical Data Access is poorly supported (Beckhoff PLCs, UaExpert)
  - Methods are very poorly supported (UaExpert)

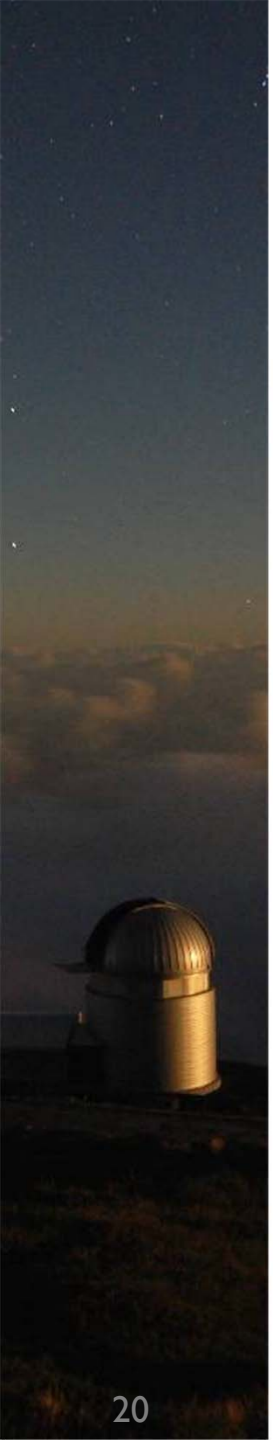
  BUT:
  - it's already more functionality out-of-the-box than what can be achieved with simple communication drivers
  - support will grow in the future
    → new products in the pipeline
    → new technologies in the pipeline
    e.g. 3rd edition of IEC61131-3 → OOP (*CLASS*, *METHOD*, *INTERFACE*, …)

# Conclusions and outlook

- OPC UA specification: interesting technology (to say the least)
  - Very detailed, tailored for **dependable heterogeneous control systems**
- OPC UA stacks and SDKs:
  - Useable right now!
  - It will take some effort to create a framework layer on top of an SDK to allow rapid and comfortable application development
    - Server: skeleton code generation (C++, SWIG), address space management, …
    - Client: stub code generation (C++, SWIG), session/subscription/monitored item handling, …
    - Infrastructure: managers (~container/component), configuration DB, historical DB, A&E GUI, …

    ➔ We already developed a lot of "experimental" code

    ➔ Hope to have basic but functional framework by mid-2012
- OPC UA COTS servers and clients (PLC/SCADA/LabVIEW/…):
  - Only most basic functionality supported now (read, write, monitored items, historical data access, alarms and events, methods)
    ➔ Additional efforts needed (~ aggregating)
  - Backed by a huge organization ➔ support will grow, big potential



# Thanks for your attention!

(and please let's talk to share ideas, experiences, code, ...)

wim.pessemier@ster.kuleuven.be