ISON Data Acquisition and Analysis Software

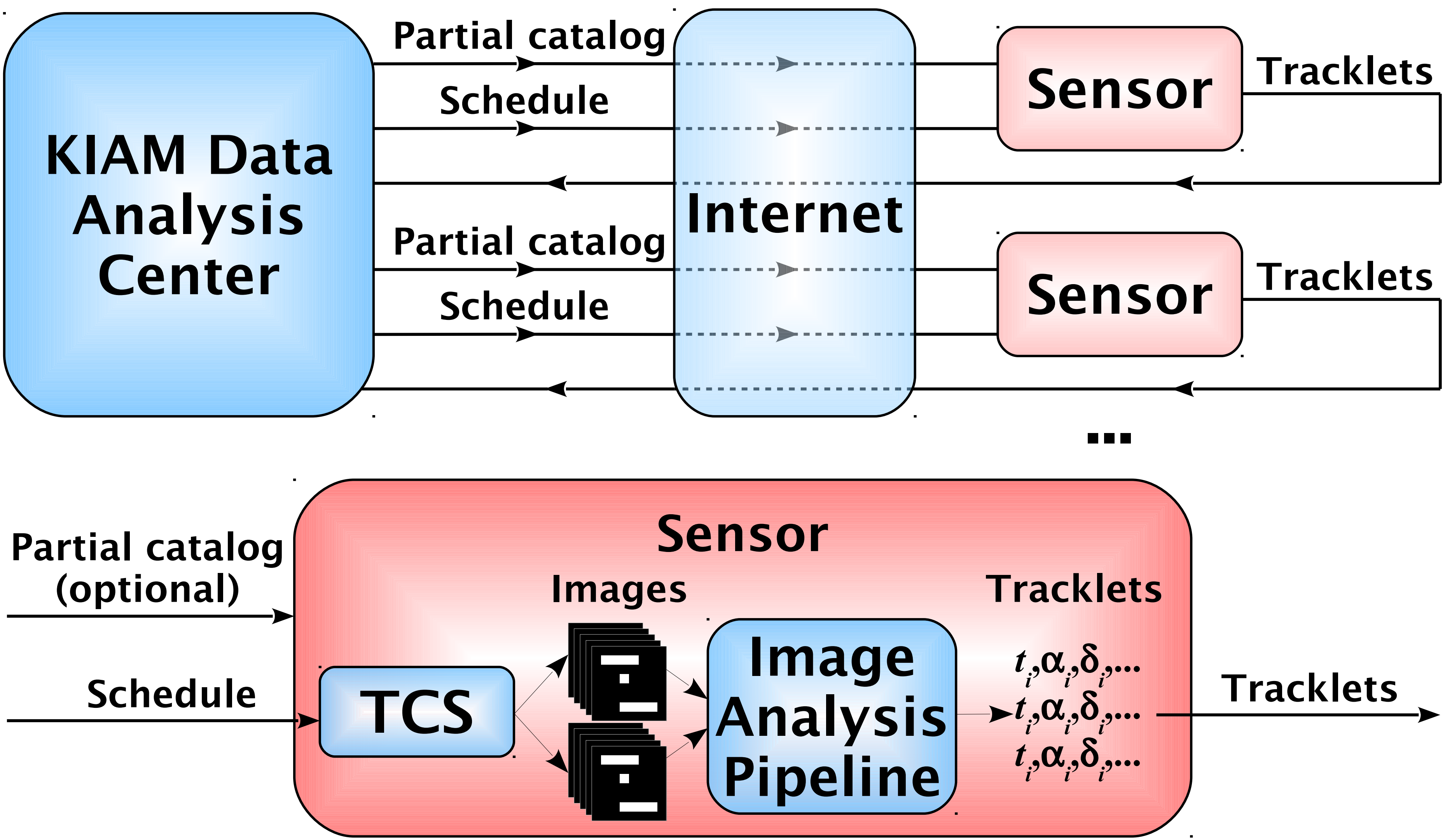
Vladimir Kouprianov

Central (Pulkovo) Observatory, Russian Academy of Sciences

ASC Project-Technics, JSC

V.K@BK.ru

Basic ISON Data Flow



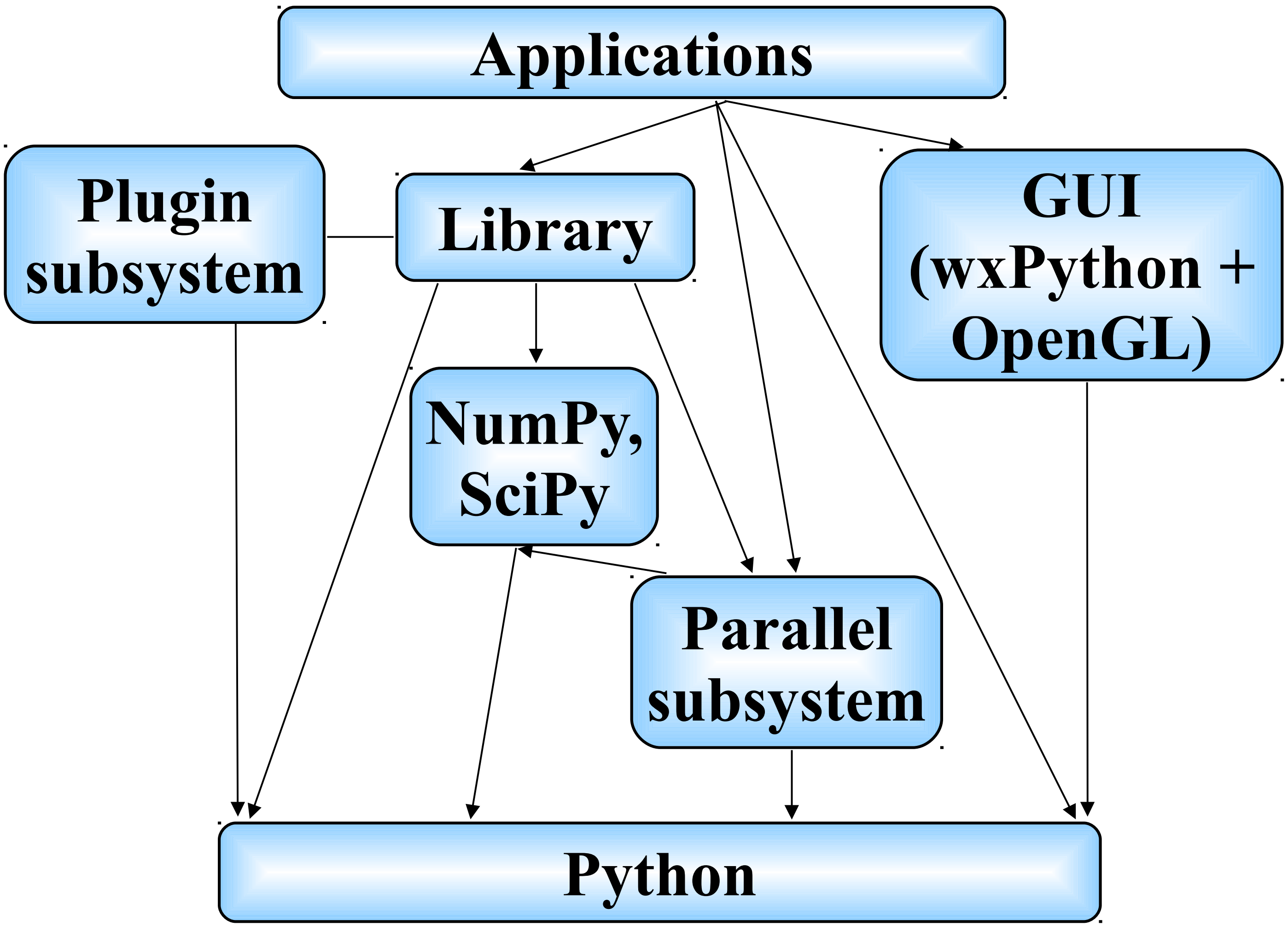
Early History (2001–2005)

- First experimental satellite observations with existing optical facilities and software
- **Fall 2004**: establishment of Pulkovo Cooperation of Optical Observers as a self-supporting initiative
- Miscellaneous software for telescope control and image acquisition (Pulkovo)
- **2004**: first version of **Apex II** (NEA research)
- **2005**: first version of **Apex II** extra package for Earth-orbiting objects

Apex II: Motivation

- No general-purpose packages suitable for high-precision **astrometry** were available, in particular – for working with **trailed sources**
- Demand for high **flexibility** due to the diversity of instruments and tasks in ISON
- Demand for high **accuracy** for fast wide-FOV sensors and undersampled images
- Fully **unattended** operation, scripting
- Existing packages (IRAF, MIDAS, IDL, MATLAB …) — deprecated software technologies/ hard to adapt/ closed source

APEX II: Outline



Apex II: Key Features for Space Surveillance

- Use of **parallel computing** for real-time image analysis without loss of sensitivity and flexibility
- Images with trailed sources → **binary morphological filtering** for fast detection without loss of sensitivity
- **PSF fitting** (incl. trailed sources) for accurate astrometry
- **kd-tree** based approach for linking detections into tracklets; similar to PanSTARRS (*Kubica et al.*, Icarus, 2007, **189**, 151)

Apex II: Parallel Computing

- Old **Apex II** parallel subsystem: relies on OS processes → utilizes multiple CPUs → can process many images in parallel
- New parallel core: relies on **OpenCL** → utilizes CPUs and GPGPUs (currently work in progress jointly with TFRM team, Univ. Barcelona) → accelerating pixel operations, object detection and measurement
- Works on the KIAM cluster: detection of faint space debris beyond the sensitivity limit (*Yanagisawa et al.*, Proc. 4th European Conf. Space Debris, Darmstadt, 2005)

Apex II: Mathematical Morphology

Traditional approaches to detection of fast-moving objects:

- Difference images → noise, false detections.
- Compare coordinates of all detections → bad performance.
- Both are unsuitable for space surveillance.

Our approach: binary morphological filtering — distinguish Earth-orbiting objects from field stars by shape (*Kouprianov*, Adv. Space Res., 2008, **41**(7), 1029-1038):

- Can detect objects in a single frame.
- Fast (cf. *Lévesque*, Proc. AMOS-Tech 2011, E66)

TCS Software Requirements

Requirements specific to space surveillance:

- Accurate timing (10ms to <1ms)
- Fast variable-rate tracking
- Simultaneous control of multiple mounts and optical channels
- Feedback from the image analysis pipeline
- Scheduling, taking into account sky conditions, for maximum performance

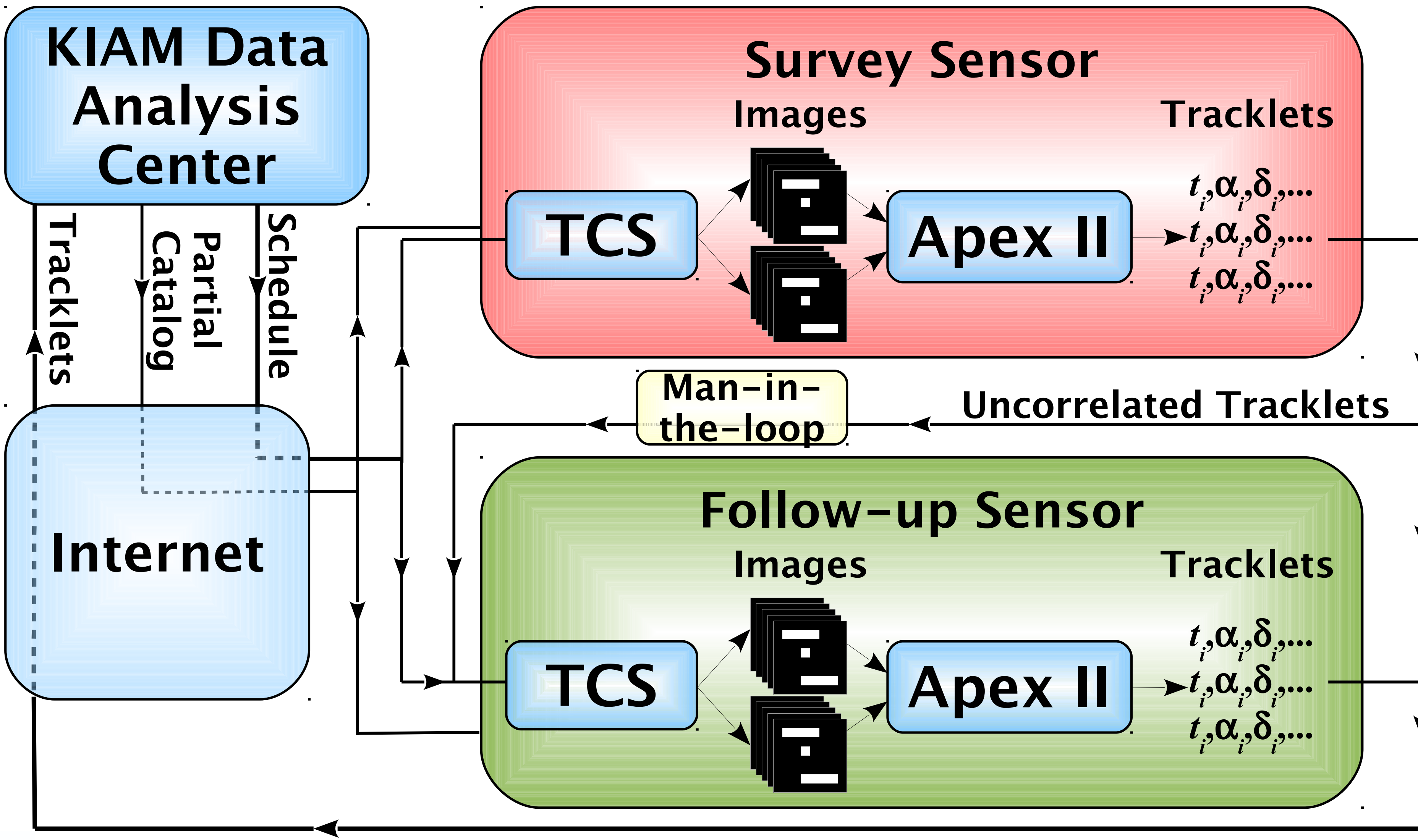
TCS Software Requirements

General requirements:

- Distributed architecture
- Local and remote access via web interface
- Autonomous operation with the possibility of manual intervention
- Well-designed hardware driver API
- Datalogging

Closest approach known — INDI www.indilib.org

On-site Follow-up



ISON Data Acquisition Software: First Generation

- Modular design; accurate timing; oriented towards supervised automated observations
- **CHAOS**: basic scheduling, mount control (LX200, SynScan, SiTech, EQMOD, ASCOM, …), dome, focusing
- **CameraControl**: imaging (FLI, SBIG, Apogee, … CCDs), filter wheel, image examination and storage
- **Datalogger**
- Hardware-disciplined **timing subsystem**

Pros: well-tested, suits most basic ISON needs

Cons: deprecated design, no integration with image analysis, lack of flexibility for advanced observation strategies

FORTE

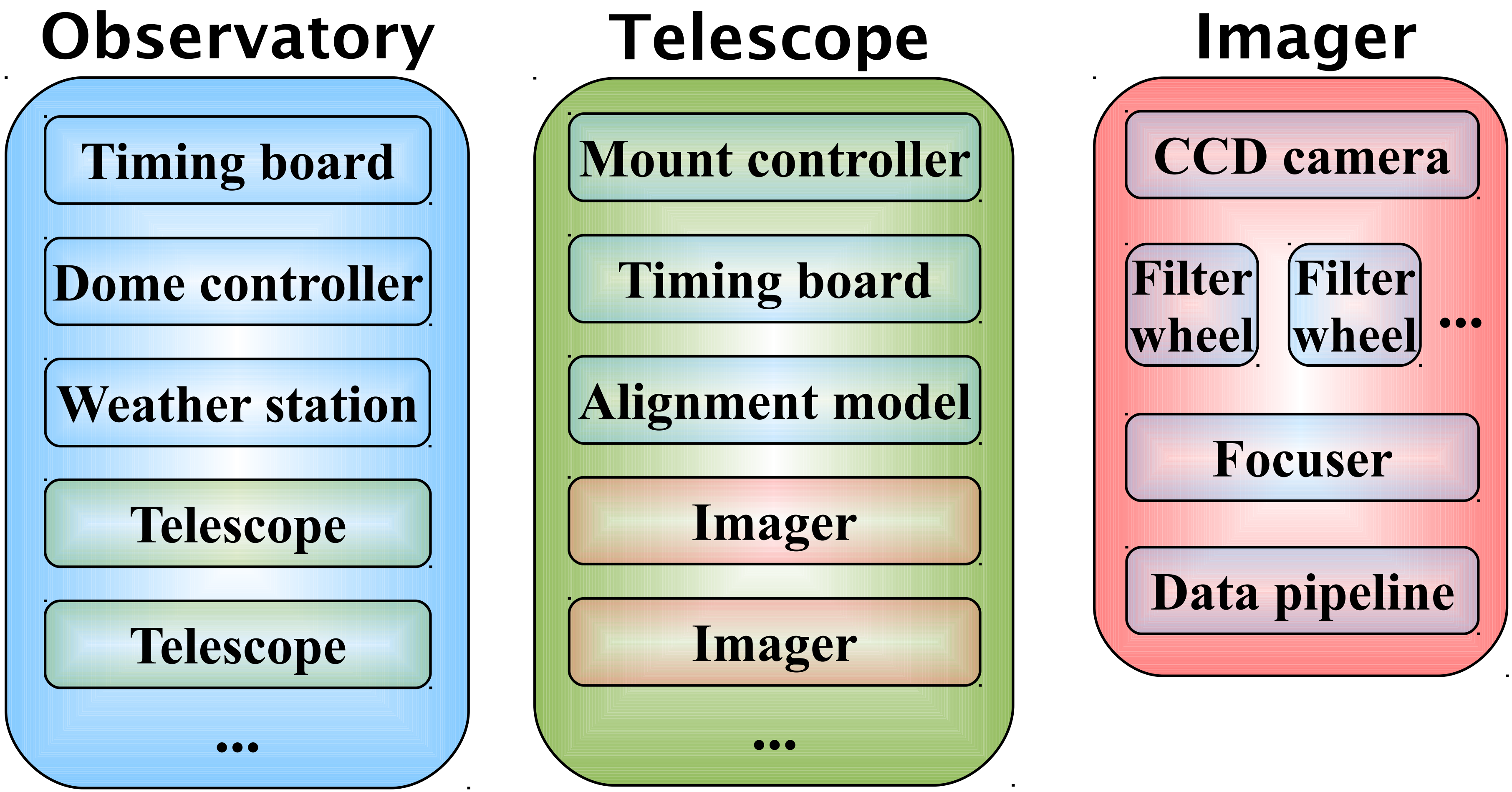
Facility for **O**perating

Robotic **T**elescope **E**quipment

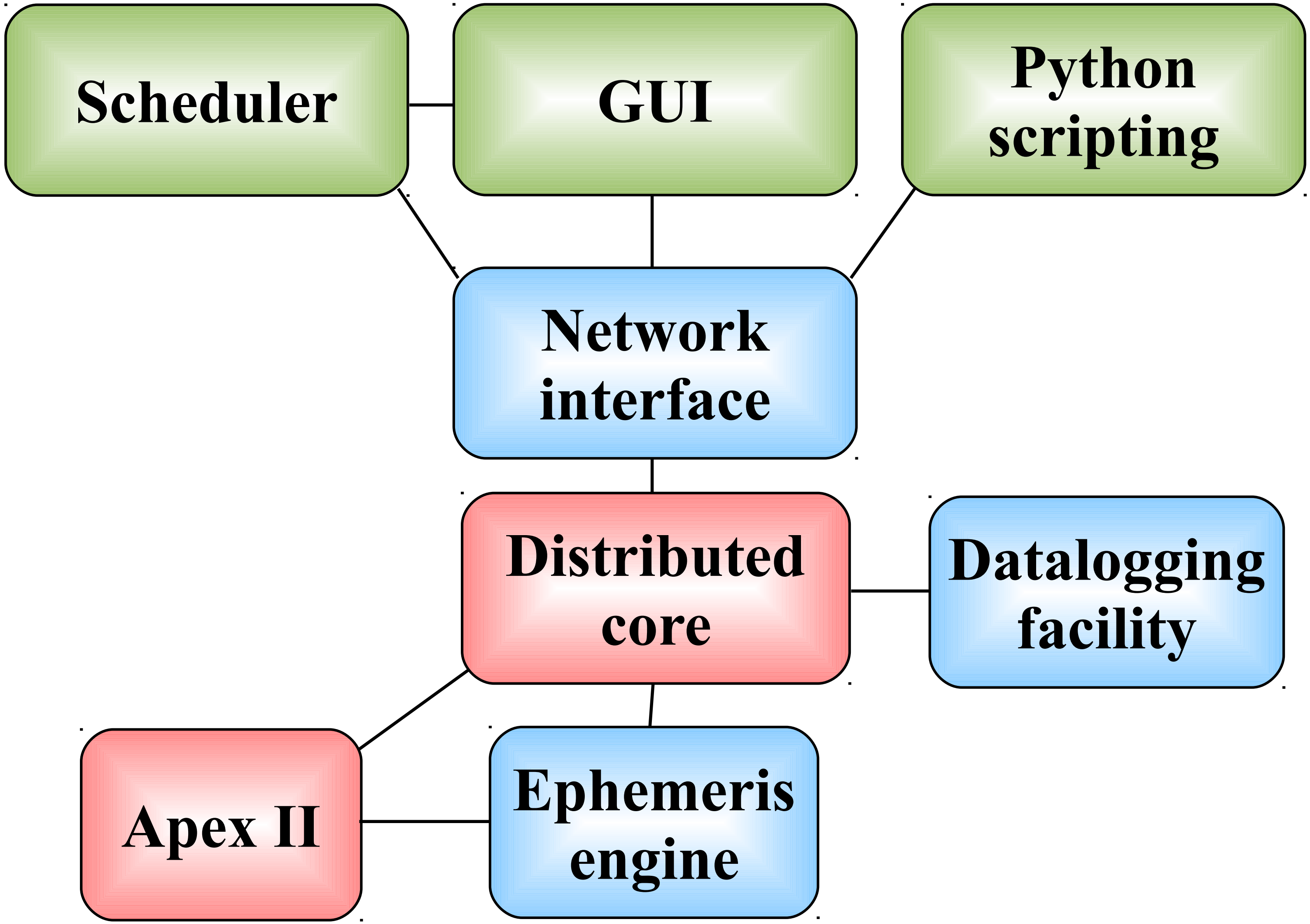
- Written in Python
- Tight integration with **Apex II**
- Distributed
- Flexible
- Scalable

f

FORTE: Outline



FORTE: Basic Structure



Hardware states

- **offline** – ready for poweroff (TE cooling disabled, scope in safe position, ...)
- **suspend** – long delay in operation, e.g. due to weather conditions (only hardware sensors working, roof closed, ...)
- **standby** – ready for normal operation (TE cooling stabilized, roof open, ...)
- **online** – system is fully operational

Remote Procedure Call

- **Internal** – communication between devices working on the same or on separate TCS workstations; transport based on Python serialization over TCP/IP
- **External** – high-level TCS control via the Observatory interface; transport based on XML packets over TCP/IP

FORTE RPC supports transparent remote actions on any Python data structures, including IPC synchronization primitives.

Image Pipelines

- Run in parallel, sequentially, or in any combinations
- Fully customizable by the user
- Can be dynamically overridden for each exposure
- Run asynchronously just after image acquisition
- Examples: data storage, image examination, image analysis

Events

- Generated by all TCS components at different important moments (state transitions, change of conditions, end of exposure, ...)
- Customizable event parameters
- Customizable actions assigned to each event
- Examples: stand by if too cloudy; suspend if humidity above 95%; re-focus if ambient temperature changes by 10º

Datalogging

- Uses built-in Python logging facility
- Backends: disk files (incl. auto-rotation), syslog daemon, NT event log, sockets
- Customizable logging destinations and formats separately for every logging channel
- Collect hardware statistics (shutter cycles, motor revolutions, voltages, …) for scheduling maintenance

Other features

- Automatic focusing and scope alignment
- Sophisticated soft limits with auto-recovery
- Support for various hardware timing modes
- High-level web interface with modules for automatic scheduling of different kinds of observations, incl. GEO/HEO survey modes
- Interoperability with **Apex II** via web interface: new images are placed on the processing queue; all detections, incl. uncorrelated tracks, are displayed immediately

New ISON Measurement Format


```
<meas>
  <sensor>12345</sensor> <id>12001002</id>
  <filename>/.../25.20120101T001122345.fit</filename>
  <utc>2012-01-01T00:11:22.345678</utc>
  <ra_j2000>1.2345678</ra_j2000>
  <dec_j2000>-2.345678</dec_j2000>
  <ra_j2000_error>0.123</ra_j2000_error>
  <dec_j2000_error>0.234</dec_j2000_error>
  <mag>15.678</mag> <mag_error>0.05</mag_error>
  <snr>5.678</snr> <x>123.456</x> <y>789.012</y>
  <x_error>0.0234</x_error> <y_error>0.0345</y_error>
  <vel_ha>-0.123</vel_ha> <vel_dec>1.234</vel_dec>
  <length>39.7</length> <width>2.5</width> <rot>43</rot>
      ...
</meas>
<meas>
   ...
</meas>
```

Conclusions

- Among other factors, performance of ISON sensors was formerly limited by non-realtime image analysis and its weak integration with hardware control loop.
- During the years 2010–2013, **Apex II** parallel subsystem, extensive use of mathematical morphology for object detection, and the kd-tree tracklet linking algorithm together lead to much higher performance of initial data reduction.
- A new observatory control system, **FORTE**, is tightly integrated with the data reduction pipeline which significantly improves the ISON space debris discovery rate.