AMPEL: An analysis platform as broker

AMPEL is a *modular* and *scalable* platform with explicit *provenance* tracking, suited for systematically processing large - possibly complex and heterogeneous - datasets in real-time or not.

This includes selecting, analyzing, updating, combining, enriching and reacting to data.

Jakob Nordin (& the AMPEL team)
Low-latency alerts & Data analysis for Multi-messenger Astrophysics, Jan 13-14 2022

Standard Work-Flow

1. Retrieve data
2. Run analysis software
3. Evaluate result

Standard Work-Flow : Limited Scalability

1. Retrieve data
2. Run analysis software
3. Evaluate result

Poorly scalable due to:

I **Data volume**

\+ Data heterogeneity

II **Analysis complexity**

\+ Software development

Code-To-Data

Established solution in the "big data" era, but challenging to implement in astrophysics due to splintered research environment.

Code-To-Data

Established solution in the "big data" era, but challenging to implement in astrophysics due to splintered research environment.

1 Retrieve data

2 Run analysis software

3 Evaluate result

3 Define expected result

2 Develop and push software

1 Connect to data streams

AMPEL is a software framework which fully enables Code-To-Data in the regime of time-dependent Astronomy and Astrophysics.

AMPEL consists of:

- A core set of public libraries.
- Scientific analysis units contributed by user teams.
- A live instance hosted by DESY Zeuthen.

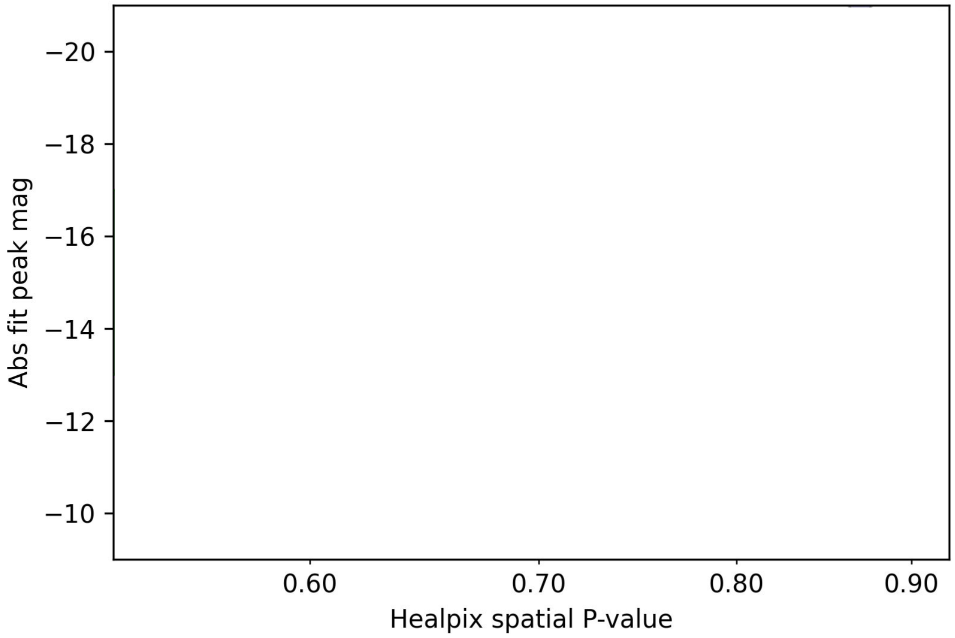
Example: POSSIS kilonova search in LIGO

③ **Define expected result** ② Software ① Connect

Goal: Count *all* ZTF kilonova candidates compatible with realistic models

Steps:

- Select all alerts within LIGO map
- Fit by POSSIS kilonova models
  https://github.com/mbulla/kilonova_models
- Select from intrinsic peak brightness



Example: POSSIS kilonova search in LIGO

③ Result ② **Develop and Upload Software** ① Connect

Implement analysis:

1. Python class for POSSIS fit

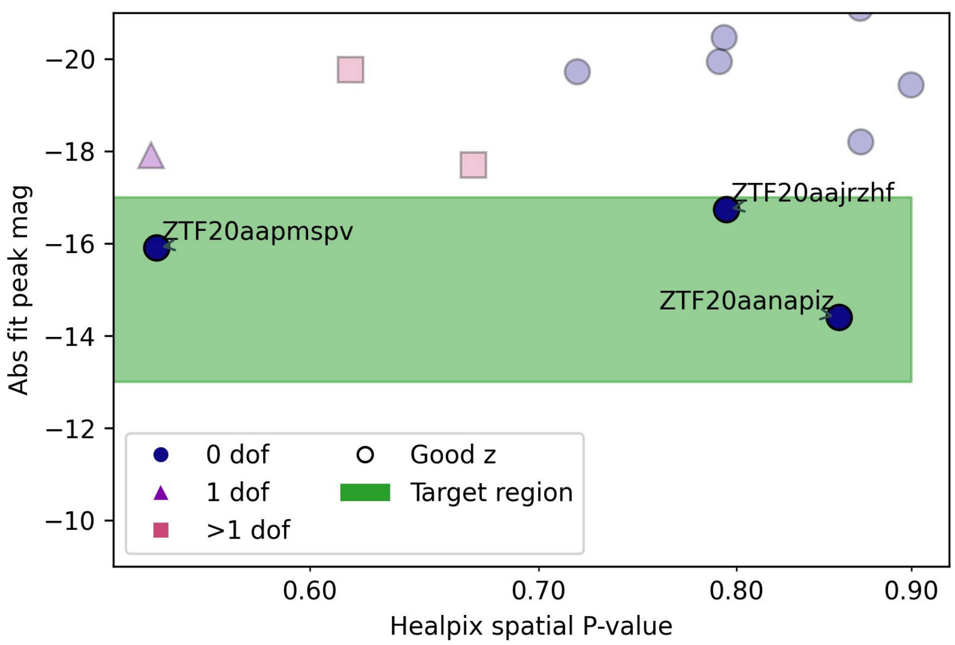
```python

class T2RunPossis(T2RunSncosmo):
    """

    Load one of the POSSIS models and create an sncosmo_model
    model for fit by T2RunSncosmo.

    """

    # Parameters determining which POSSIS model will be read

    # Currently references to sample model included in Ampel-HU-astro
    possis_dir: str = 'data/kilonova_models'
    model_gen: str = 'bns_m3_3comp'
    mej_dyn: float = 0.01
    mej_wind: float = 0.09
    phi: int = 45
    cos_theta: float = 0.3    # Typically 0., 0.1, ...1.0

    sncosmo_model_name: str = '_'.join(map(str, [model_gen, mej_dyn,
        mej_wind, phi, cos_theta]) )

    # Fix time to specific explosion timestamp
    # StockTriggerTime assumes the value is updated during runtime
    explosion_time_jd: None | float | Literal['StockTriggerTime']

```

Example: POSSIS kilonova search in LIGO

3 Result   2 Develop and Upload Software   1 Connect

Implement analysis:

1. Python class for POSSIS fit
2. Local test



Example: POSSIS kilonova search in LIGO

3 Result 2 Develop and Upload Software 1 Connect

Implement analysis:

1. Python class for POSSIS fit
2. Local test
3. Distribute

Example: POSSIS kilonova search in LIGO

3 Result 2 Develop and Upload Software 1 Connect

Implement analysis:

1. Python class for POSSIS fit
2. Local test
3. Distribute
4. Request incorporation with live instance

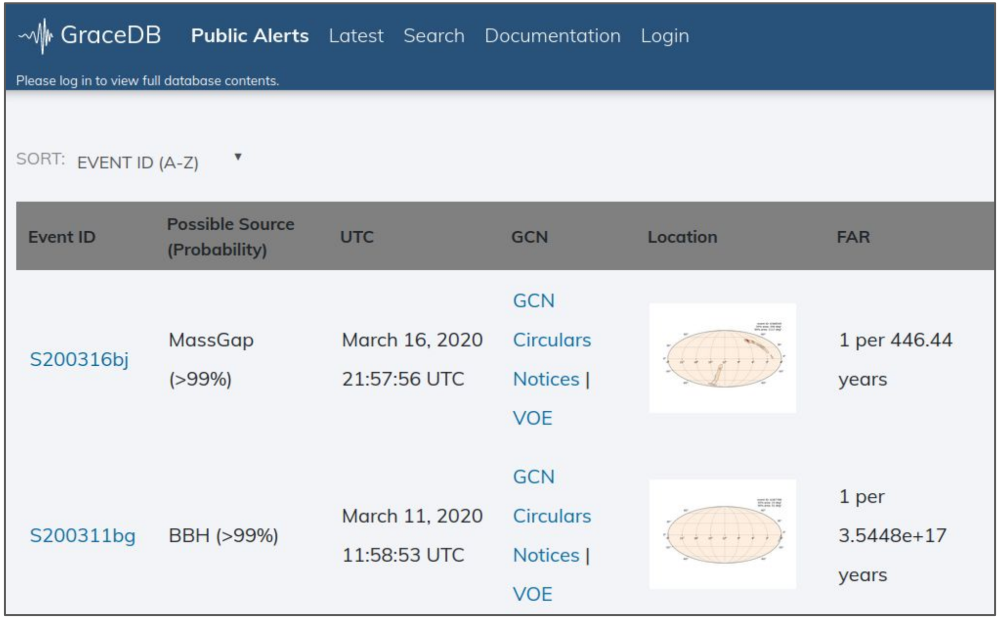
```
ampel-info [at] desy.de
```

Example: POSSIS kilonova search in LIGO

3 Result 2 Software 1 Connect to data streams

Trigger:

1. Find (new) skymap



GraceDB Public Alerts Latest Search Documentation Login

Please log in to view full database contents.

SORT: EVENT ID (A-Z)

Event ID	Possible Source (Probability)	UTC	GCN	Location	FAR
S200316bj	MassGap (>99%)	March 16, 2020 21:57:56 UTC	GCN Circulars Notices | VOE		1 per 446.44 years
S200311bg	BBH (>99%)	March 11, 2020 11:58:53 UTC	GCN Circulars Notices | VOE		1 per 3.5448e+17 years

Example: POSSIS kilonova search in LIGO

3 Result 2 Software 1 Connect to data streams

Trigger:

1. Find (new) skymap
2. Post path through API

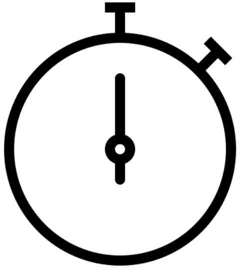
```
curl -X 'POST' \
  'https://ampel.zeuthen.desy.de/api/ztf/job/runpossis' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "map":
"https://gracedb.ligo.org/api/superevents/S200112r/files/LALInference.fits.gz",
    "channel": "my_program",
    "pvalue_limit": 0.9,
  }'
123456
```

Example: POSSIS kilonova search in LIGO

3 Result   2 Software   1 Connect to data streams

Trigger:

1. Find (new) skymap
2. Post path through API
3. [await processing]



```
curl -X 'GET' \
  'https://ampel.zeuthen.desy.de/api/live/jobstatus/123456 \
    -H 'accept: application/json'
completed
```

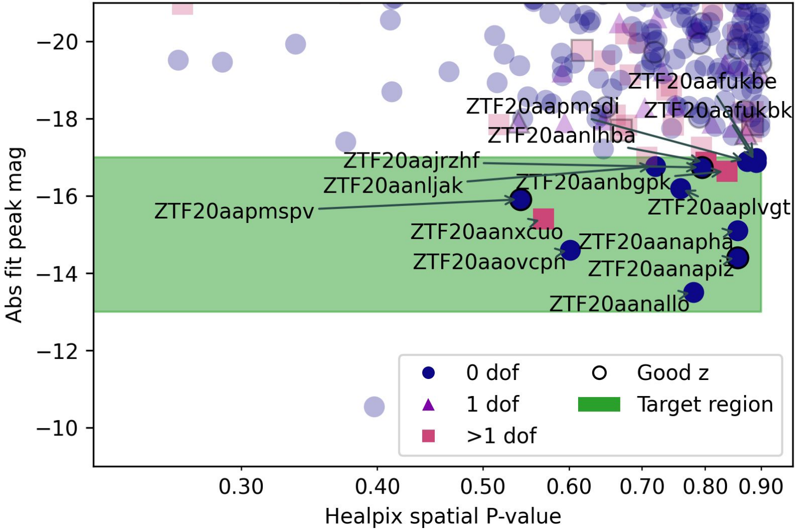
Example: POSSIS kilonova search in LIGO

3 Result 2 Software 1 **Connect to data streams**

```
curl -X 'GET' \
  'https://ampel.zeuthen.desy.de/api/live/job/123456/t3/HealpixCorrPlotter \
   -H 'accept: application/json'
```

Trigger:

1. Find (new) skymap
2. Post path through API
3. [await processing]
4. **Get result**



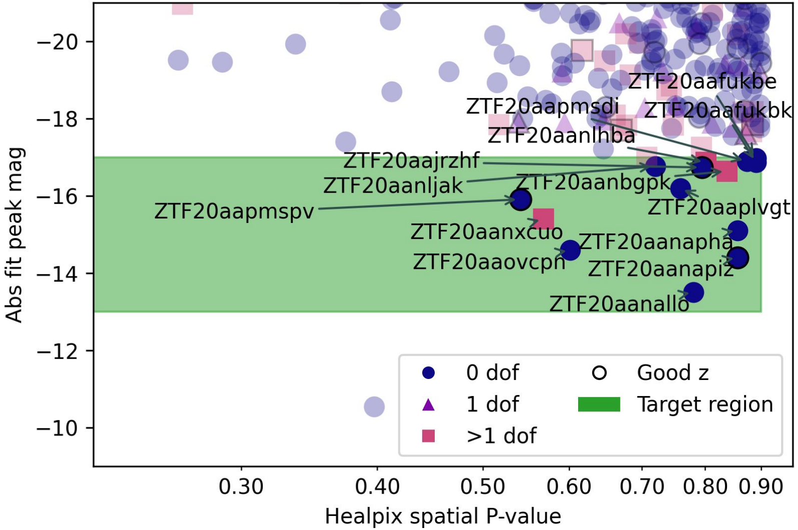
Example: POSSIS kilonova search in LIGO

3 Result 2 Software 1 Connect to data streams

```
curl -X 'GET' \
  'https://ampel.zeuthen.desy.de/api/live/job/123456/t3/HealpixCorrPlotter \
   -H 'accept: application/json'
```

Trigger:

1. Find (new) skymap
2. Post path through API
3. [await processing]
4. Get result

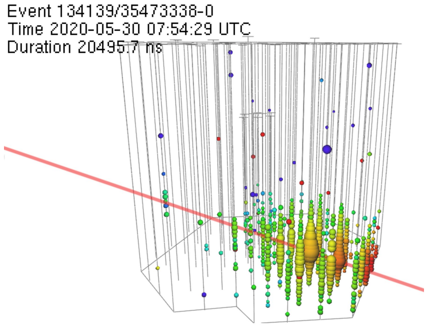


Full analysis schema:
https://github.com/AmpelProject/Ampel-HU-astro/blob/v0.8.2-dev/examples/ligo-kilonova.yml

Sample current projects

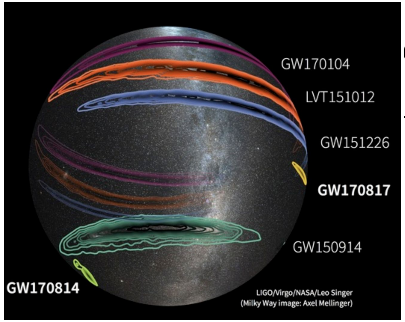
IceCube + ZTF

29% of alerts followed up

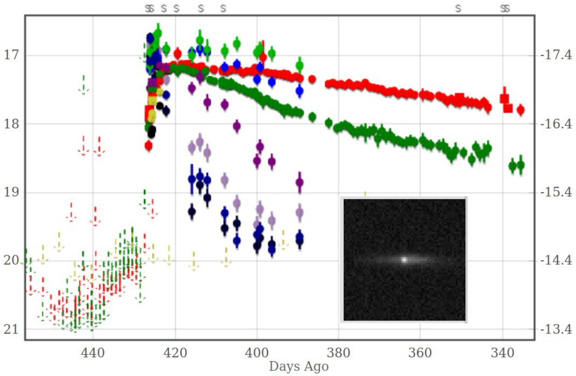


O3 LIGO/Virgo + ZTF

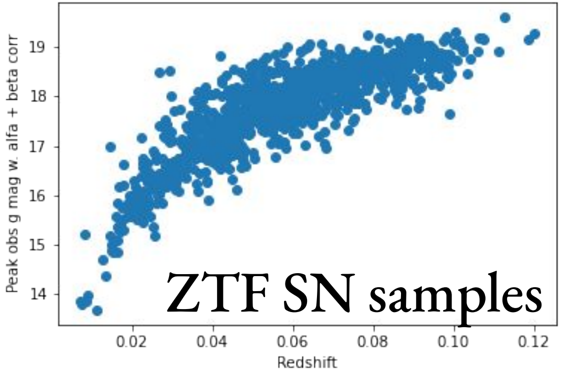
<39%> of enclosed probability



Robotic, autonomous follow-up trigger



ZTF SN samples



Further information

Up and running now.

Explore the code: https://ampelproject.github.io/

Try it out: https://github.com/AmpelProject/Ampel-contrib-sample

More about structure/motivation:

J. Nordin, V. Brinnel, J. van Santen, M. Bulla, U. Feindt et al. *Transient processing and analysis using AMPEL: alert management, photometry, and evaluation of light curves*. Astronomy and Astrophysics 631 (2019) A147. doi:10.1051/0004-6361/201935634

Contact us: ampel-info at desy.de, jnordin at physik.hu-berlin.de

Summary

AMPEL is a real-time data processing platform which introduces the Code-To-Data paradigm to astrophysics.

Science teams develop analysis pipelines locally, which are then pushed to an AMPEL live instance.

Pipelines here will be exposed to the full alert streams from e.g. ZTF, LSST, LIGO/VIRGO, Ultrasat, IceCube … (as well as user provided inputs), as well as data archives/releases.

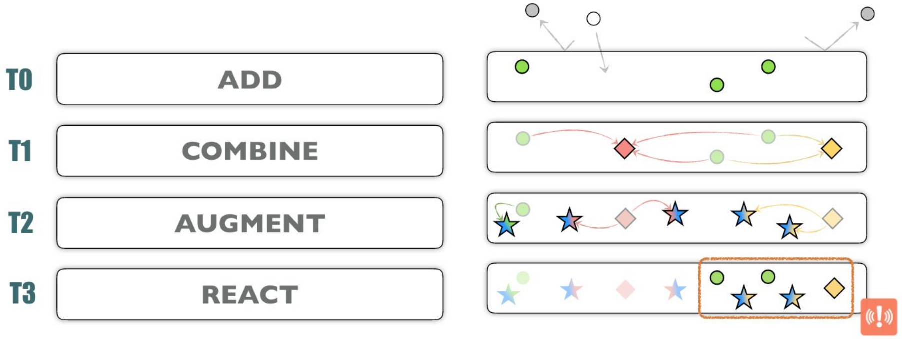
VERA C. RUBIN
OBSERVATORY

Backup

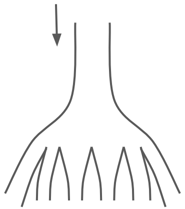
Heterogeneous data

Combining heterogeneous streams requires *standardization*. This is naturally encouraged in a multi-developer, Code-to-Data paradigm and practically enforced in AMPEL through internal processing tiers.

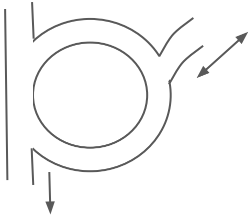
3. Define expected result
2. Develop and push software
1. Connect to data streams



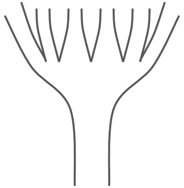
Multiple stream operations form a full analysis



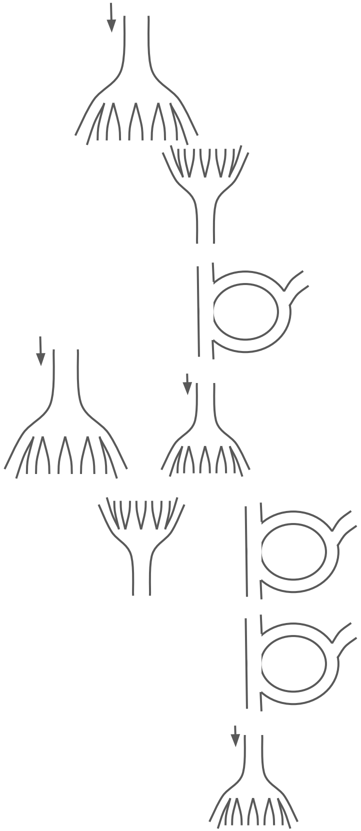
“Broker”
Divide an input stream
according to some criteria.



“Supplement”
Annotate stream, e.g.
based on a classifier.



“Join”
Combine different streams.

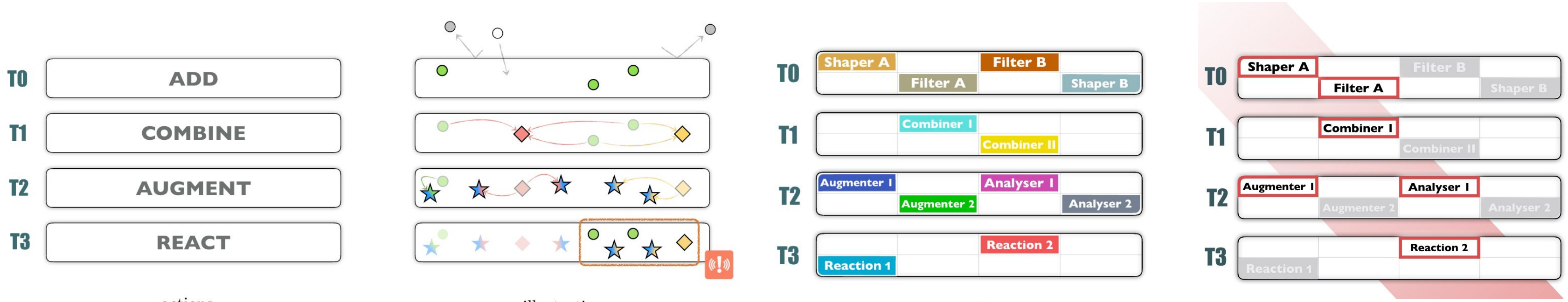


Key concepts for a Code-to-Data platform

Information based approach where an analysis program can be specified through four kinds of questions:

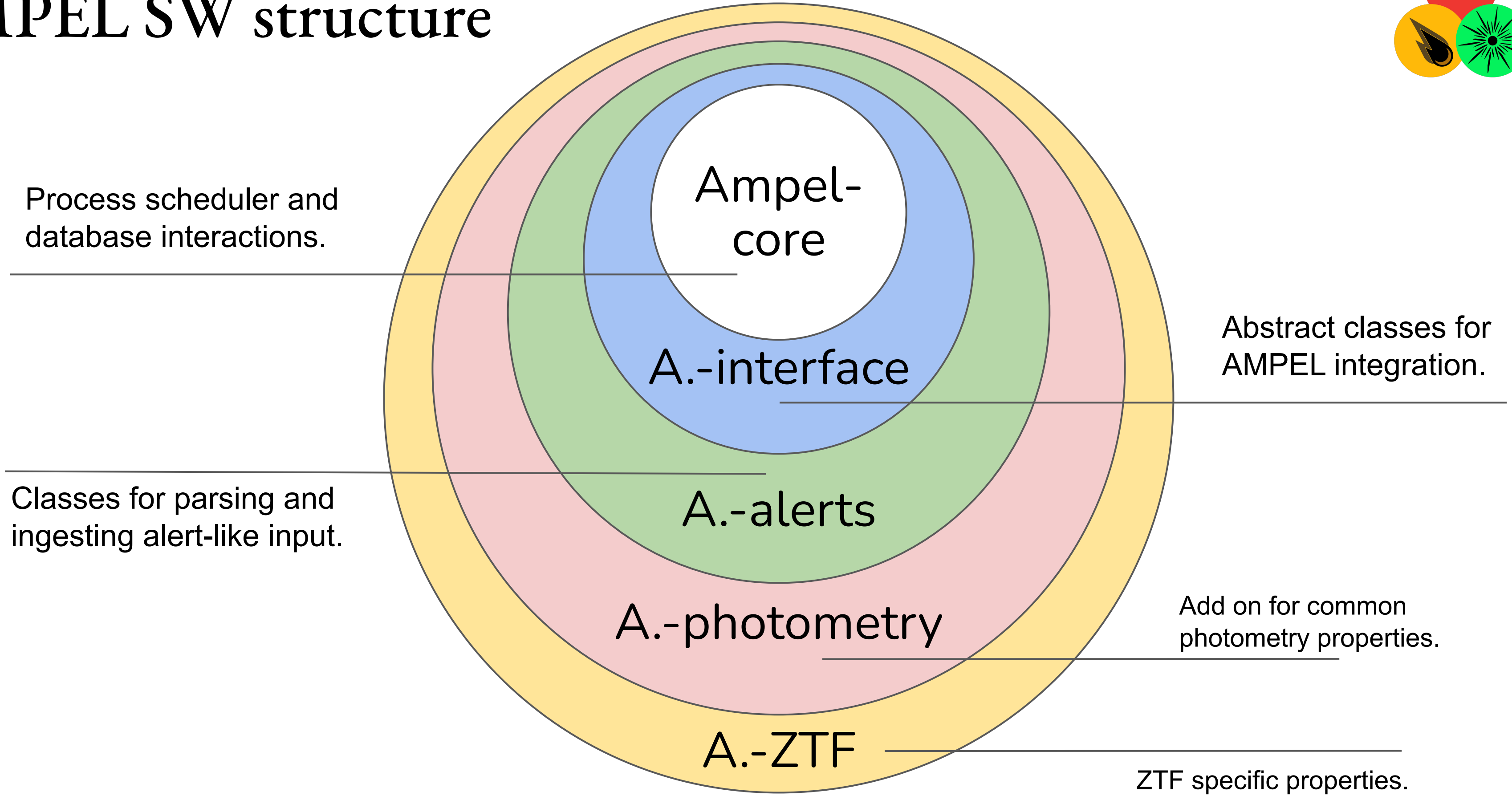
- What input data should be considered?
- How to combine data into compounds?
- What can you derive from compound data?
- Which reactions should be triggered under what conditions?

Each kind of question is addressed through implementation of a specific abstract class
Ampel-core connects, schedules and transfer information between these..

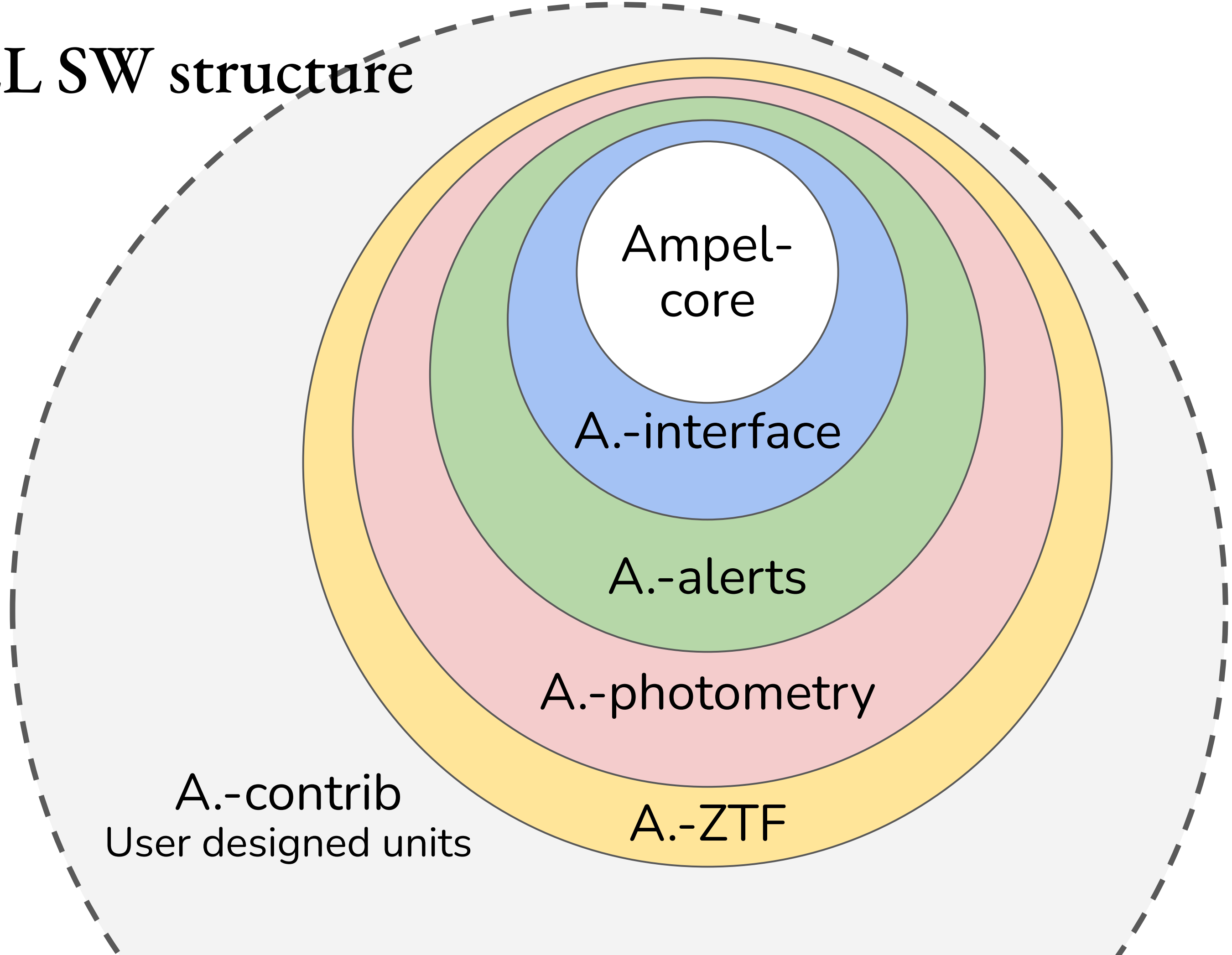


actions illustration units

AMPEL SW structure



AMPEL SW structure



AMPEL: analysis schema

Ampel-contrib-sample / conf / ampel-contrib-sample / channel / **SAMPLE_CHANNEL.yml**

50 lines (49 sloc) | 1.05 KB

```yaml
channel: SAMPLE_CHANNEL
contact: maintainer_contact@doe.com
active: false
auto_complete: live
template: ztf_uw_public
t0_filter:
  unit: SimpleDecentFilterCopy
  config:
    min_rb: 0.3
    min_ndet: 7
    min_tspan: 10
    max_tspan: 200
    min_gal_lat: 15
t2_compute:
  - unit: T2SNcosmoComp
    config:
      target_model_name: v19-2009ip
      base_model_name: salt2
      chi2dof_cut: 2.
  - unit: T2MultiMessMatch
    config:
      temporal_pull_scaling: 1
      spatial_pull_scaling: 3.
      energy_pull_scaling: 0.001
      match_where: 'latest'
t3_supervise:
    template: ztf_periodic_summary
    schedule: every(30).seconds
    load:
      - TRANSIENT
      - COMPOUND
      - DATAPOINT
      - T2RECORD
    filter:
      t2:
        all_of:
        - unit: T2SNcosmoComp
          match:
            target_match: true
        - unit: T2MultiMessMatch
          match:
            best_match:
              $lt: 1
    run:
      - unit: T3HelloWorld
        config:
          t2info_from:
            - T2SNcosmoComp
            - T2MultiMessMatch
```

An analysis schema fully specifies a science program through specifying which units are to be run with which parameters.

To be referenced, distributed, tested and/or developed while applied to real-time, archived or simulated data.