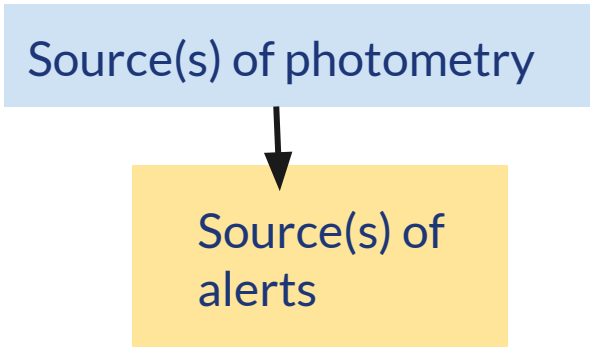
AMPEL

Alert Management, Photometry and Evaluation of Lightcurves

AMPEL is a tool allowing methods/tools/algorithms developed for individual transients to be consistently and reproducibly applied to large samples of objects.

Jakob Nordin, HU Berlin

The AMPEL project



The AMPEL project

Source(s) of photometry

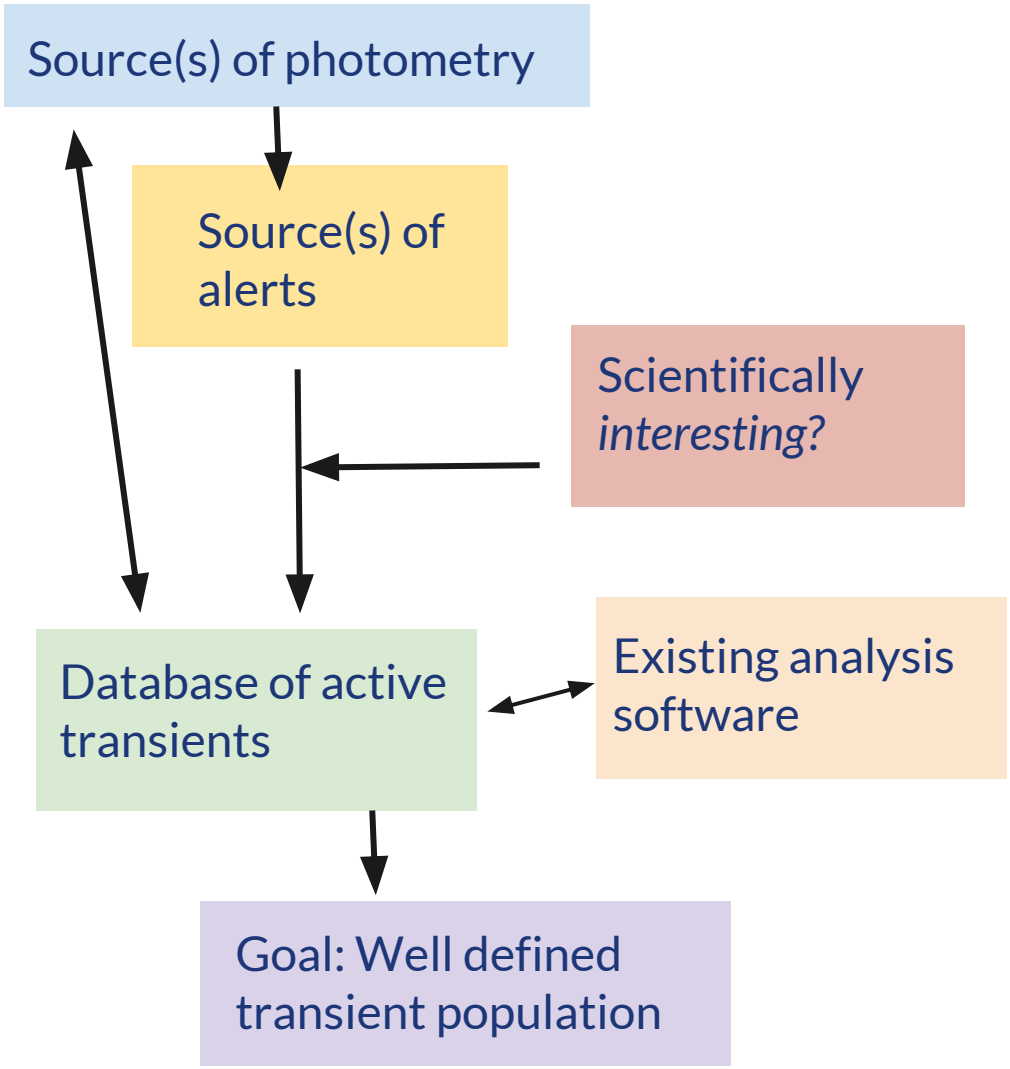
Source(s) of alerts

Scientifically *interesting?*

Existing analysis software

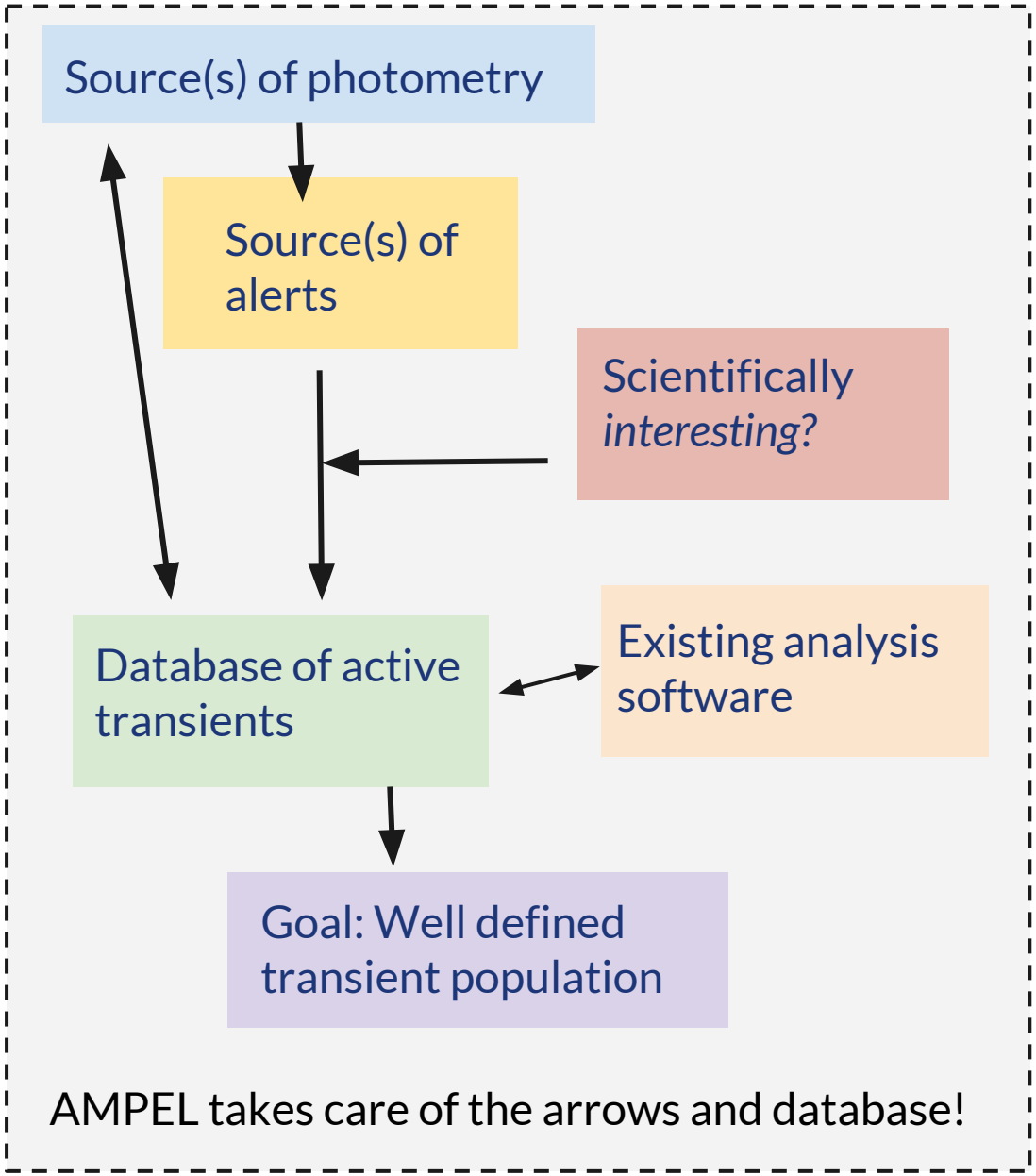
Goal: Well defined transient population

The AMPEL project



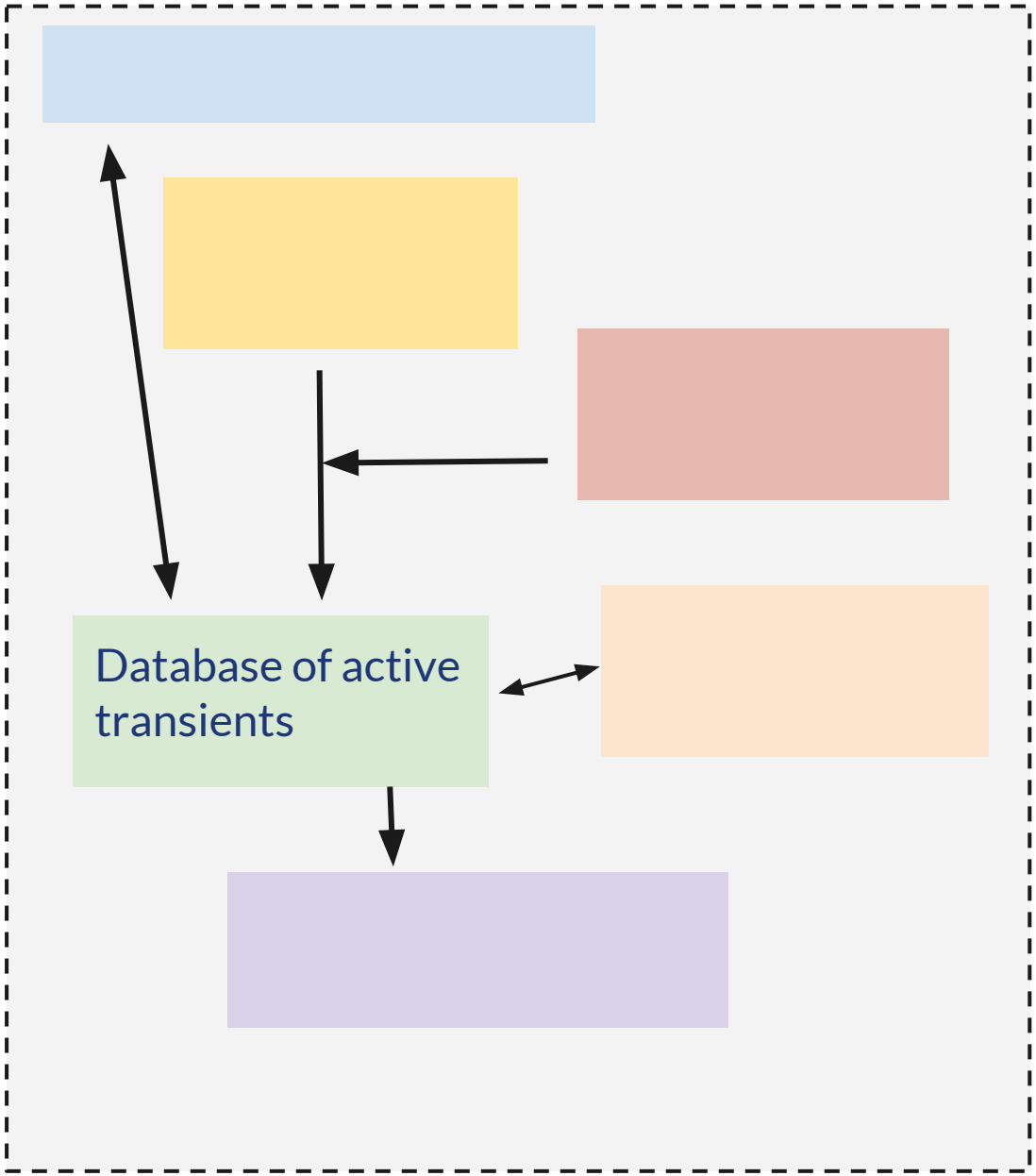
The AMPEL project

Should I consider this transient ?
(Ampel = Berlin traffic light)

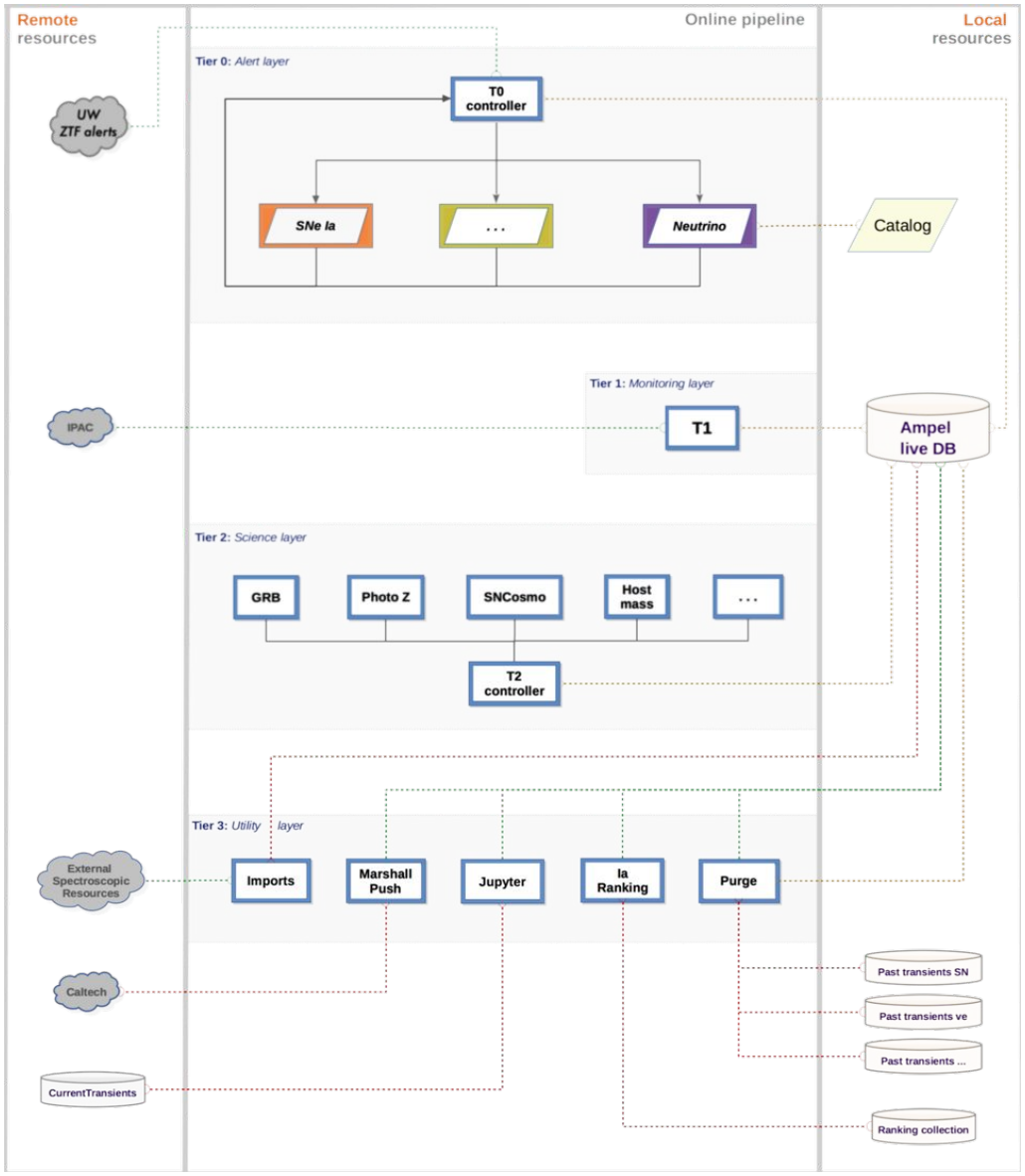


The AMPEL project

The AMPEL team focus is on the framework!



Schematic overview



Alerts are matched to the list of active channels

Accepted transients are monitored in the live DB

Requested analysis modules run as new data are added

Users get updates:

- Slack messages
- Marshal push
- Telescope trigger
- (Database copy)

Ampel 101

- Users access Ampel through a *channel*. This specifies both what kind of alerts could be interesting and what should be done to them.

Ampel 101

- Users access Ampel through a *channel*. This specifies both what kind of alerts could be interesting and what should be done to them.

- Ampel is internally divided into four layers or *tiers*:
  - Tier 0 *filters* what alerts are sufficiently interesting to be ingested into the DB,
  - Tier 1 *monitors* the state of known transients.
  - Tier 2 performs *analysis* calculations on a single transient state.
  - Tier 3 executes *actions* based on combined T0, T1, T2 information, possibly from multiple transients

Ampel 101

- Users access Ampel through a *channel*. This specifies both what kind of alerts could be interesting and what should be done to them.

- Ampel is internally divided into four layers or *tiers*:
  - Tier 0 *filters* what alerts are sufficiently interesting to be ingested into the DB,
  - Tier 1 *monitors* the state of known transients.
  - Tier 2 performs *analysis* calculations on a single transient state.
  - Tier 3 executes *actions* based on combined T0, T1, T2 information, possibly from multiple transients

- Running Ampel requires things to be done at the right level:
  - Tier 0 operations can be exposed to all incoming alerts (limited complexity).
  - Tier 2 modules can be more complex, but have lower priority.
  - All transfer of information from Ampel is done at the T3 level.

Ampel 101

- Users access Ampel through a *channel*. This specifies both what kind of alerts could be interesting and what should be done to them.

- Ampel is internally divided into four layers or *tiers*:
  - Tier 0 *filters* what alerts are sufficiently interesting to be ingested into the DB,
  - Tier 1 *monitors* the state of known transients.
  - Tier 2 performs *analysis* calculations on a single transient state.
  - Tier 3 executes *actions* based on combined T0, T1, T2 information, possibly from multiple transients

- Running Ampel requires things to be done at the right level:
  - Tier 0 operations can be exposed to all incoming alerts (limited complexity).
  - Tier 2 modules can be more complex, but have lower priority.
  - All transfer of information from Ampel is done at the T3 level.

- There is no “frontend” to Ampel, nor is it (currently) possible to “log in” to the Ampel DB. T3 functions exist to export information.

Example: AMPEL in the Cosm. SWG

Transient selection (T0): RB & SG limits; at least three detections

T2: Host galaxy (once / candidate)

- Host galaxy shape analysis
- Spec-z search
- Photometric redshift

T2: Template lightcurve fit (after each photopoint / non-detection)

- Determine fit qualities and estimated peak dates

SN classification (T3, run each day)

- Load candidates without type, with decent SNIa match, (photo)-z <0.1, predicted peak day in >-3 days, mag <~ 19.5
- Push ranked list to marshal/slack/telescope list (see Ulis talk)

SN pull (T3, run each day)

- Read classification information from external sources

After survey, rerun full stream while varying parameters. Compare ranks & final types..

How do I use Ampel?

- AMPEL will be a public service, keeping track of data access rights. There is an initial practical limit to the number of channels we host.
- Get general information at:
  - https://ampelproject.github.io/Ampel
  - http://noir.caltech.edu/twiki_ptf/bin/view/ZTF/CosmoAMPEL
  - Contact: ampel-info@desy.de
- Define a channel for a science goal:
  - Clone Ampel T0-ref repository
  - Edit the ***configuration file***
  - Implement specific T0 filter python class, if needed
  - Implement a T3 information push
  - Probably want a testing / visualization plan
  - Contact the Ampel channel manager: ampel-channels@desy.de
- Provide T2 analysis modules for use by you and/or others
- Provide catalogs/ToO maps (see later slide)
- Eventually, each channel will be provided with their final Mongo DB containing all transient history

Sample channel configuration file

```
[{      "_id": "HU_EARLY_SN",
        "version": 1.0,
        "active": true,
        "input": [
                {
                        "instrument": "ZTF",
                        "alerts": "IPAC",
                        "parameters": {
                                "ZTFPartner": true,
                                "autoComplete": true,
                                "updatedHUZP": false
                        } } ],
"t0Filter": {   "id": "BasicFilter",
                "runConfig": {
                        "criteria": [
                                {
                                        "rb": 0.8,
                                        "operator": ">"
                                }, ],
                        "len": 3,
                        "operator": ">"}    },
```

```
"t2Compute": [
        {
                "t2Unit": "SNCOSMO",
                "runConfig": "SALT2"
        }, ],
"t3React": [
        {
                "t3Unit": "SLACK",
                "runConfig": {
                        "criteria": [
                        {
                          "T3SNCOSMO": [
                                fitchi": 0.1,
                                 "operator": ">"
                        }]
                        "slack": [
                        {
                                "slackid": “ZTF Berlin”,,
                                "slackchannel”:”sandbox”,
                                “botid”:”24035704358”
                        }]
```

Full power through python implementation

```
[{    "_id": "HU_EARLY_SN",
      "version": 1.0,
      "active": true,
      "input": [
            {
                  "instrument": "ZTF",
                  "alerts": "IPAC",
                  "parameters": {
                        "ZTFPartner": true,
                        "autoComplete": true,
                        "updatedHUZP": false
                  } } ],
"t0Filter": {    "id": "MyFilterClass",
            "runConfig": {
                  "criteria": “Model”
},}
```

```
"t2Compute": [
      {
            "t2Unit": "SjoertsGaiaComparator",
            "runConfig": {
                  “Reject” : true,
      }, ],
```

A python package following the AMPEL IO rules, available with setup.py through eg github.

An implementation of the T0 ref filter class. Manipulation of all alert info + restricted catalog matching

Catalog matching

Ampel is not only a tool for catalog matching! That said, MG has eated a specific code for organizing and quering catalogs.

https://github.com/MatteoGiomi/extcats

Once the catalog is imported into AMPEL, it is easy to do catalog matching in T0/2 implementations.

Rule of thumb:

- “small” (<100M sources) catalogs can be done at the T0 level.
- “large” catalogs (e.g. PS1) has to be queried as a T2 module (or a ‘late’ T0).

Catalogs are DBs: they can be modified on the fly. A util module can then replay part of the alert stream for the affected filters. P*rovides mechanism for matching alerts to external detections (eg GW).*

More catalog matching talks to follow !

Modules currently developed by Cosmology, Neutrino, TDE groups

- Catalog matching
  - Wise
  - Pan Starrs
  - The Million Quasar Catalog
  - Gaia
- T2
  - Polynomial lightcurve fit
  - SNcosmo template fitting, Gaussian Process lightcurve, parametric rise/decline
  - Host galaxy identification and photometric redshift
- T3
  - TNS submission
  - Slack notification
  - GROWTH marshal save and annotation
  - “Dropbox” file transfer
  - UH88 / LCO telescope ranking & trigger

Relation to the brokers and the GROWTH marshal?

- A broker is a service that can add contextual information and then divert the alert flow (e.g. there is no notion of the set of currently live transients). AMPEL can, in principle, do this but it would not be very efficient and not use core AMPEL features.
- The GROWTH marshal is designed primarily for visualization and commenting of data of various kinds of individual transients. Information can be pushed from AMPEL to the marshal (annotating), including saving transients and information (comments) can be pulled back to AMPEL from the GROWTH DB.
    - Keeping track will be easier if one AMPEL channel corresponds one-to-one to a GROWTH science program (not technically necessary). Keep filtering consistent.
    - Use a machine readable way when making critical comments.

Ampel 222 - optional features

- A primary design goal is to allow for channels where the results are fully reproducible. This can be achieved as the full Ampel system is stored as a container that can be mounted later and users *can* define fully automatic channels. **I strongly recommend channels to be as automatic as possible and restrict human interaction to one "layer".**
  - Requires original alert stream to be available

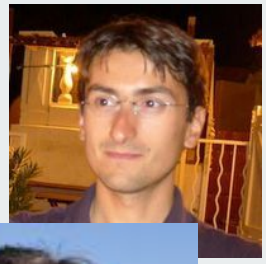
Allows to investigate the effects of sample selection through changing some of the channel configuration parameters and rerunning a set of alerts. Sample case: investigate when observations of an object would have been initiated, and how many false observations would have been carried out.

- A further consequence is the **delayed T0** - if you are made aware of an event that happened a few days ago you can create a new filter for this and reprocess a certain set of events.
- Straightforward to parallelize. No DB racing conditions.

Ampel 888 - technical notes for offline reading

- Ampel makes use of Mongo DB. Provides flexibility, but also limitations in terms of which fields can be quickly searched.
- Each channel needs a purge strategy for when objects leave the DB. At regular intervals each channel retrieves a DB cutout containing all the events it selected and their processing history.
- Ampel makes sharing software easy. After wrapping it as a T2 module you can let other users run it on their selection of alerts (or not).
- We could incorporate and combine different kind of alerts.
- Ampel will run on two dedicated machines in DESY (+ farm).
    - A central computer cluster provides the power to do real-time alert responses, and multiplex advantage in that many channels can benefit from the same operations.
    - The AMPEL code is frozen into docker files. We will make sure old alert streams are available, which guarantees that you can recreate the channel progress, or rerun previous data with updated search tools or tests.
    - An intermediate step will be to have a copy of the current DB available at a different machine where users can connect.

Ampel team and current state



Intense development phase including:.

- Valery Brinnel (DB and core software design)
- Jakob von Santen (Hardware, network, storage)
- Matteo Giomi, Robert Stein (alert filtering and catalog matching)
- Ludwig Rauch, Mickael Rigault (analysis modules like photometric redshifts)
- Anna Franckowiak, Marek Kowalski

Core AMPEL (framework) software running. Focus now on modules and interface. Production software versioned and stored as containers - the candidate processing can be exactly rerun.

Main servers installed at the DESY computer centre in Zeuthen, but network enclosure to be verified for active deployment. Wishlist include an external server for interactive database access.