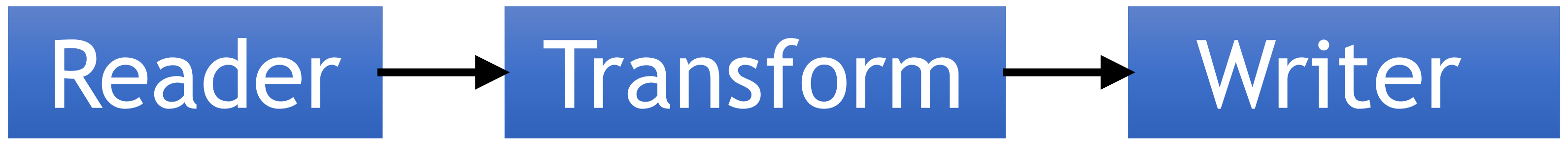
OpenRVDAS

An Open Source Framework for
Building Data Acquisition Systems



David Pablo Cohn
`openrvdas.org`

Why?

- Many ships each running homebrew derivatives of legacy systems (*dsLog*, *LDS* and others)

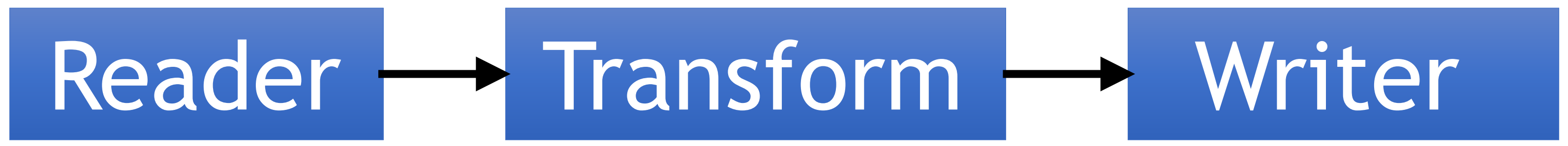
- Massive duplication of effort to support

- Common, open source codebase would allow pooling expertise and best practices

- MIT License allows unrestricted use/copying/modification/distribution/sublicensing for commercial/non-commercial purposes

An **framework**, not a **system**

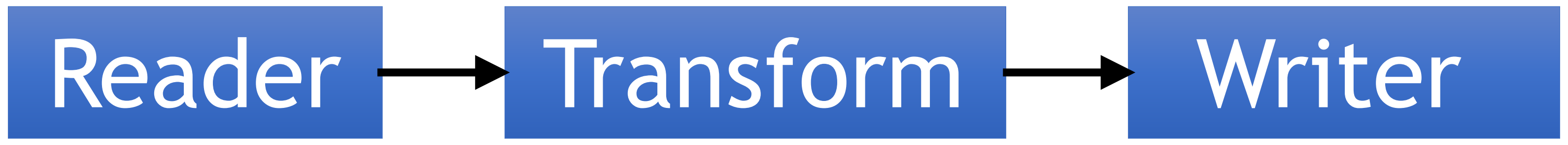
(Systems change as requirements change; a good framework lets you easily put together whatever system meets current requirements)



Read from serial port, prefix with timestamp and instrument id, write to file

Everyone's needs are different **now**
Everyone's needs will be different in 5 years

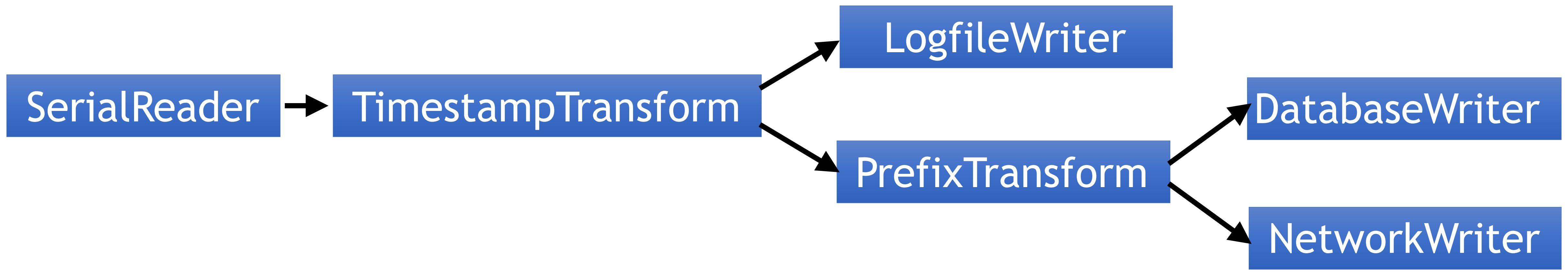
Solution: small set of Lego-like components that can be easily "snapped together" to create what you need



Read from serial port, prefix with timestamp and instrument id, write to file

Everyone’s needs are different **now**
Everyone’s needs will be different in 5 years

Solution: small set of Lego-like components that can be easily “snapped together” to create what you need



Readers, Transforms and Writers

Readers

- SerialReader
- NetworkReader
- FileReader
- DatabaseReader
- TimeoutReader

Transforms

- PrefixTransform
- TimestampTransform
- SliceTransform
- ParseNMEATransform
- FilterQCTransform

Writers

- LogfileWriter
- NetworkWriter
- DatabaseWriter
- AlertWriter

Simple API makes it easy to create your own as needs arise

Three easy ways to combine

In Code

```
reader = SerialReader(port='/dev/ttys1')
transform = TimestampTransform()
writer = LogfileWriter(filebase='/var/logs/knud')
while True:
  in_record = reader.read()
  out_record = transform.transform(in_record)
  writer.write(out_record)
```

Command line

```
 listener.py --serial port=/dev/ttys1 \
     --transform_timestamp \
     --transform_prefix knud \
     --write_logfile /var/logs/knud \
     --write_network :6221 \
     --write_database rvdas@openrvdas:test
```

Config file

```
{"knud->net": {
   "name": "knud->net",
    "readers": {
      "class": "SerialReader",
        "kwargs": {"port": "/dev/ttys1",
                   "baudrate": 9600
                  }
    ...
```

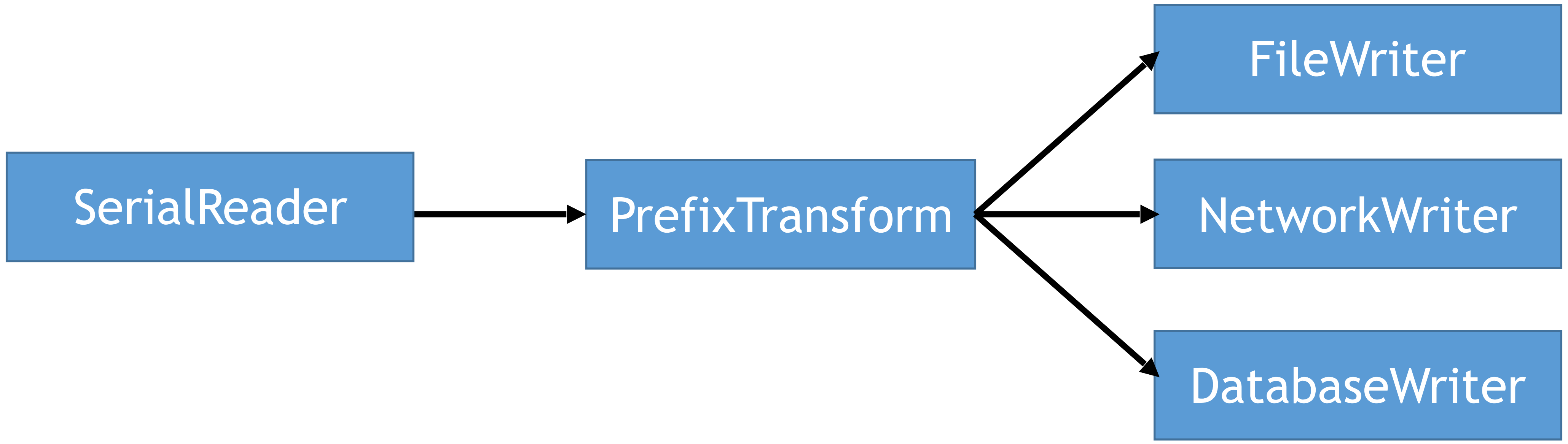
Python code API

```
reader = SerialReader(port='/dev/ttys1')
transform = TimestampTransform()
writer = LogfileWriter(filebase='/var/logs/knud')
while True:
  in_record = reader.read()
  out_record = transform.transform(in_record)
  writer.write(out_record)
```

API makes it easy to combine existing Readers/ Writers/Transforms (or create your own as needed)

Listeners - a command line interface

```
listener.py --serial port=/dev/ttys1 \
    --transform_timestamp \
    --transform_prefix knud \
    --write_logfile /var/log/knud \
    --write_network :6221 \
    --write_database rvdas@openrvdas:test
```



Configuration files

Single logger

```
listener.py --config test/config/sample_logger.json
```

A set of loggers

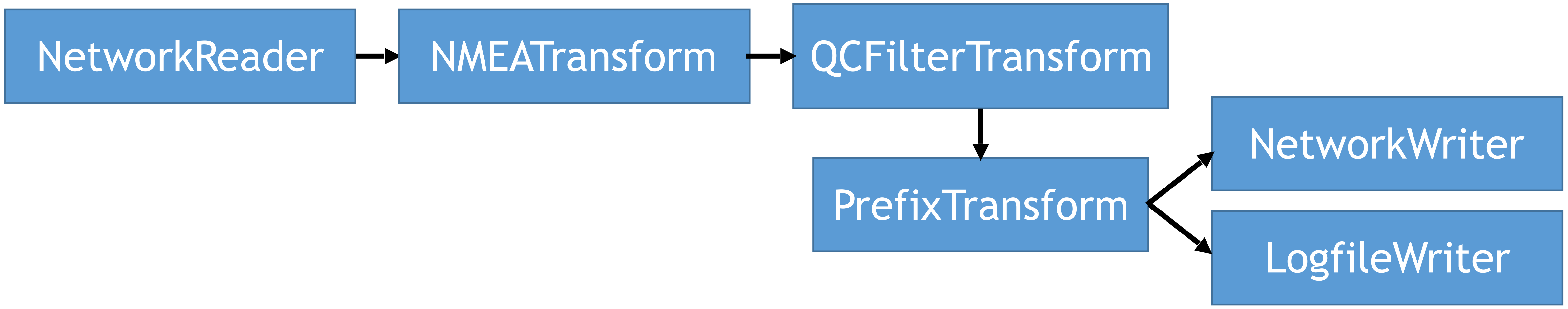
```
logger_runner.py --config test/config/sample_config.json
```

Full cruise control

```
logger_manager.py \
    --config test/config/sample_cruise.json \
    --mode underway \
    --database django \
    --websocket :8765
```

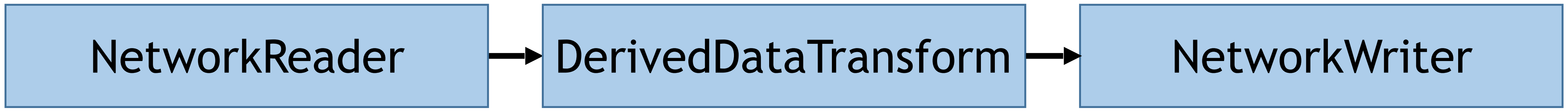
Example: realtime QC

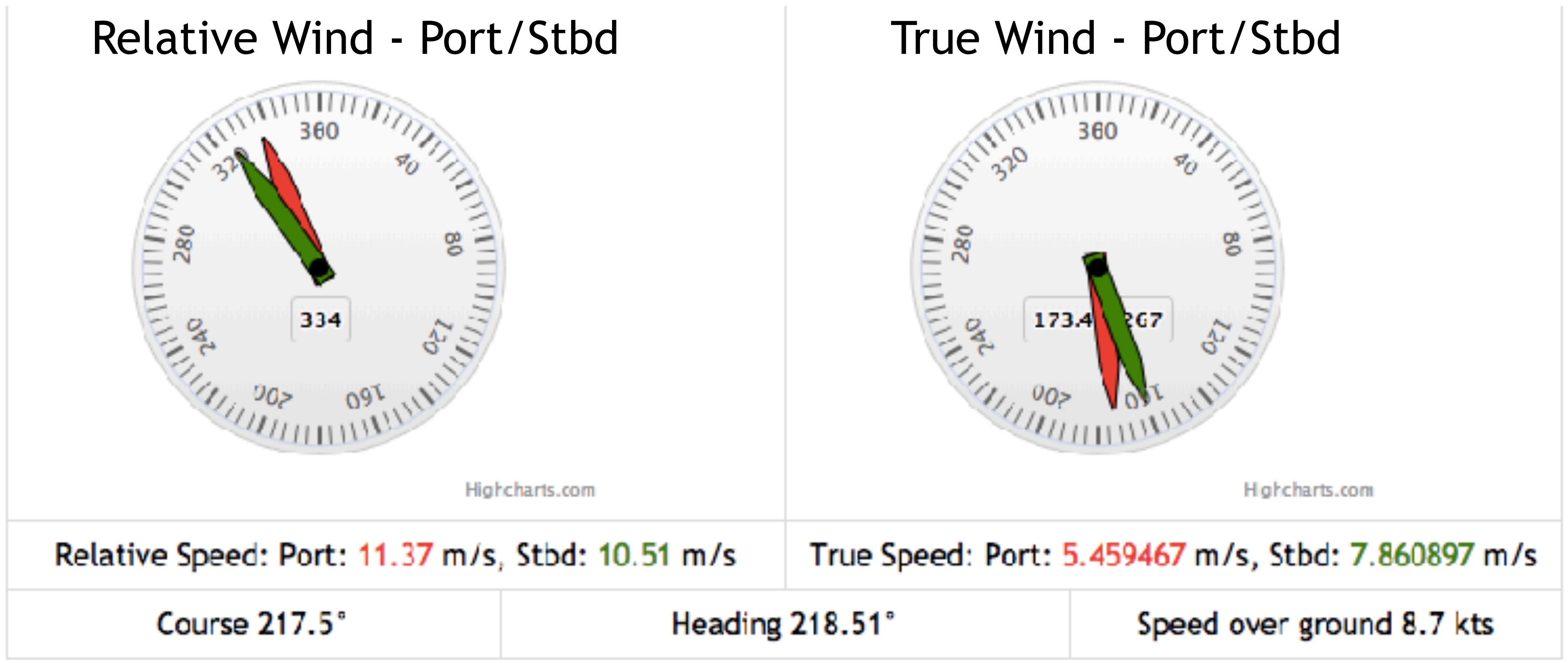
```
listener.py --network :6224 \
  --transform_parse_nmea \
  --transform_qc_filter \
    TWNCTension:-150:10000,TWNCPayout:-60:175000 \
  --transform_prefix 'twnc_error' \
  --write_network :6221 \
  --write_logfile /var/log/errors
```



Example: derived data values

Read inputs from file/database/network,
compute values, inject outputs back into stream





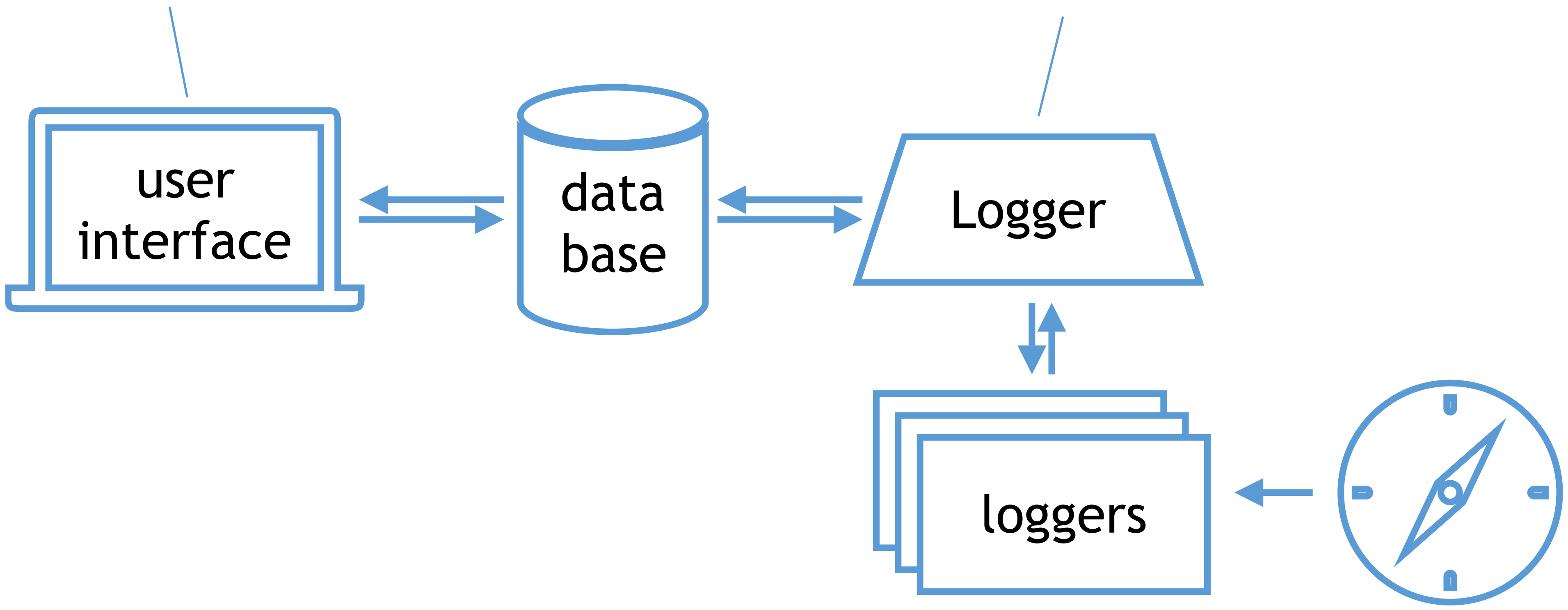
Relative Wind - Port/Stbd	True Wind - Port/Stbd
Relative Speed: Port: 11.37 m/s, Stbd: 10.51 m/s	True Speed: Port: 5.459467 m/s, Stbd: 7.860897 m/s

Course 217.5°	Heading 218.51°	Speed over ground 8.7 kts

Control architecture – a database approach

- writes desired state to database
- retrieves latest observed status from database

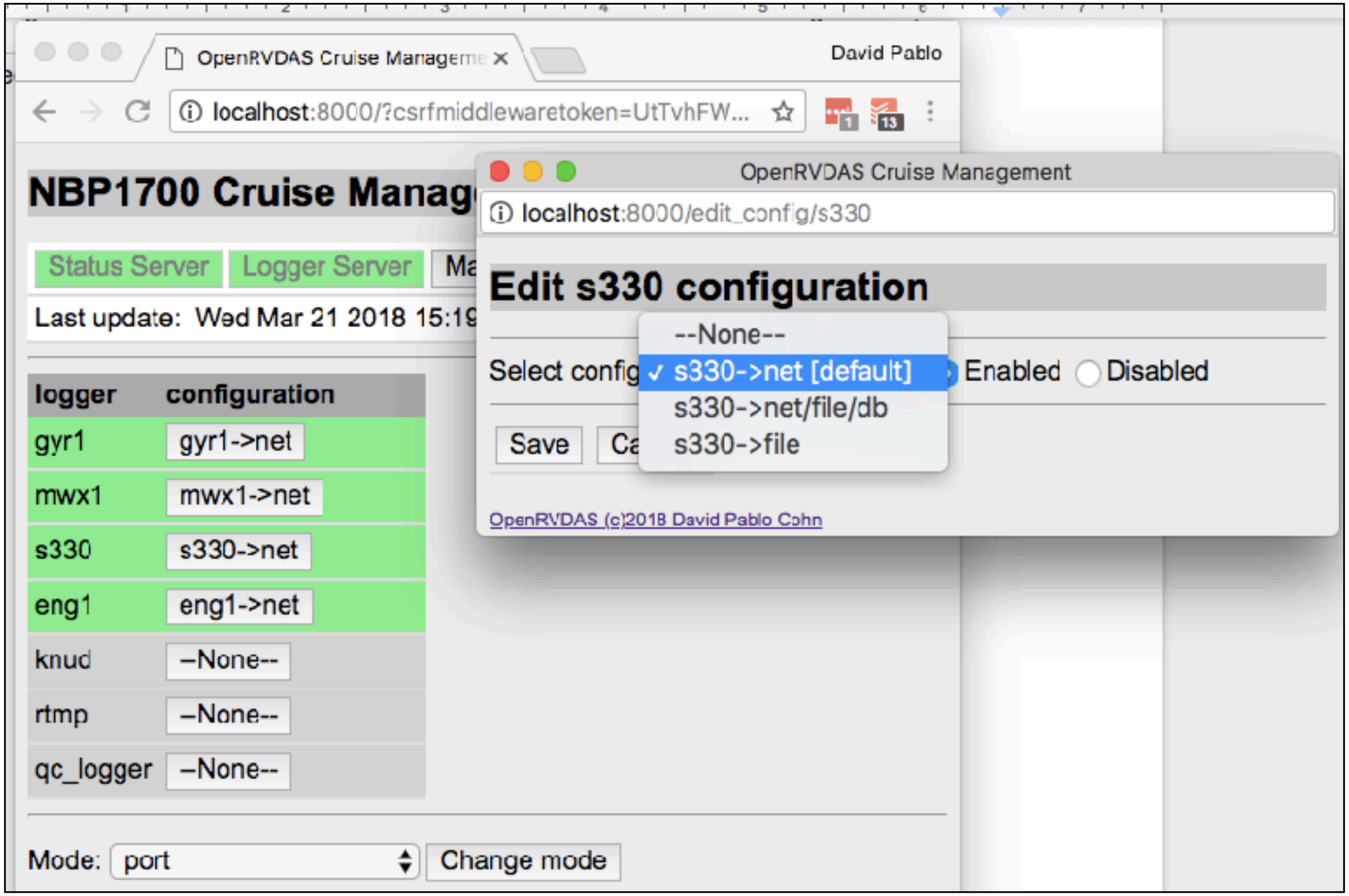
- reads desired state from database
- checks observed state from system
- starts/stops processes to reconcile



Command line interface

```
command? load_cruise test/configs/sample_cruise.json
command? cruises
     Loaded cruises: NBP1700

command? modes NBP1700
     Modes for NBP1700: off, port, underway

command? set_mode NBP1700 port
command? set_mode NBP1700 underway

command? logger_configs NBP1700 s330
     Configs for NBP1700:s330: s330->off, s330->net, ...
command? set_logger_config_name NBP1700 s330 s330->net

command? set_mode NBP1700 off
```

Web-based graphic interface


OpenRVDAS Cruise Manageme

David Pablo

localhost:8000/?csrfmiddlewaretoken=UtTvhFW...

NBP1700 Cruise Manag

Status Server | Logger Server | Ma

Last update: Wed Mar 21 2018 15:19

logger	configuration
gyr1	gyr1->net
mwx1	mwx1->net
s330	s330->net
eng1	eng1->net
knud	--None--
rtmp	--None--
qc_logger	--None--

Mode: port | Change mode

OpenRVDAS Cruise Management

localhost:8000/edit_config/s330

Edit s330 configuration

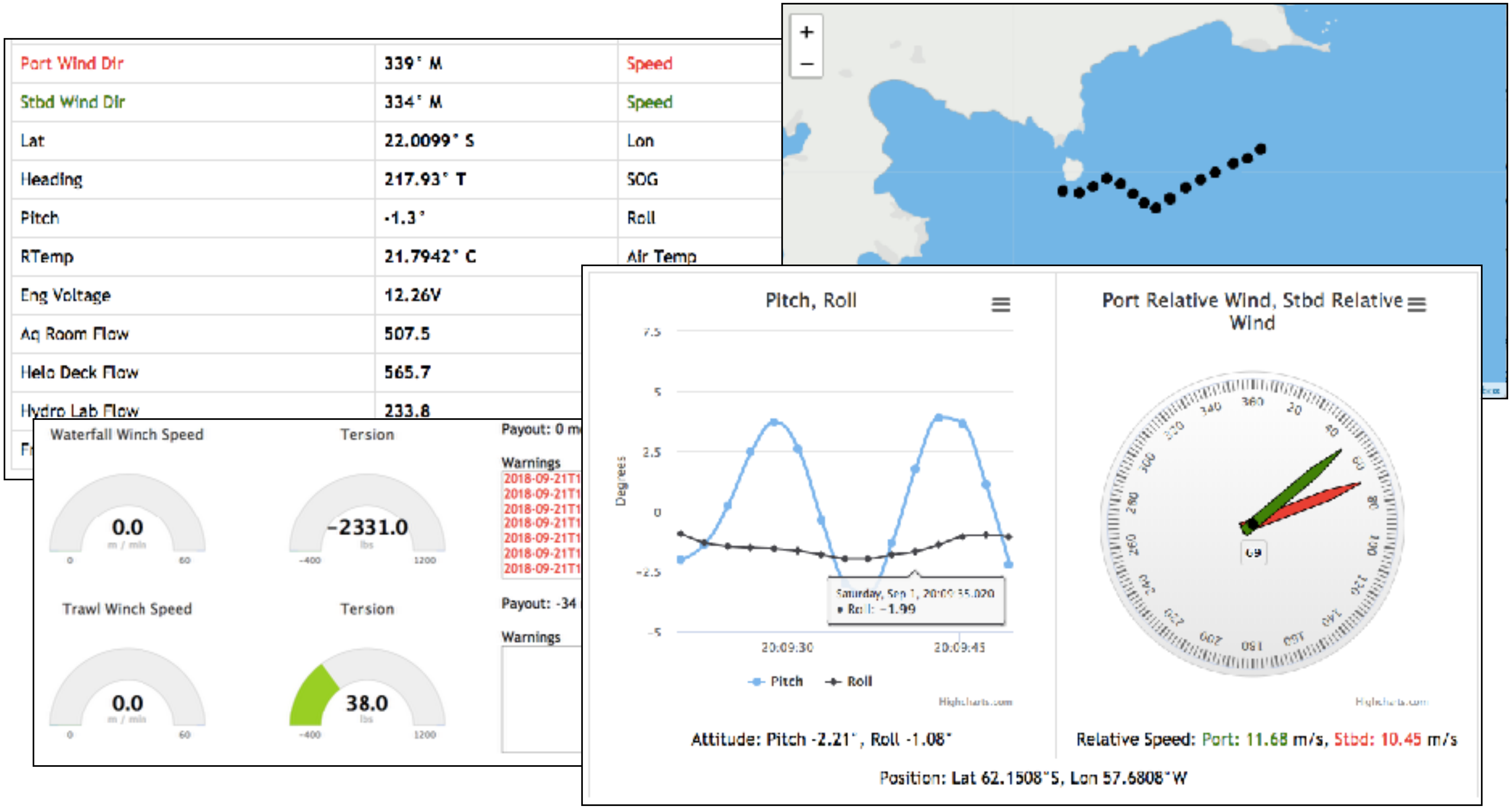
Select config: --None-- | ✓ s330->net [default] | s330->net/file/db | s330->file

[✓ Enabled] Disabled

Save | Ca

OpenRVDAS (c)2018 David Pablo Cohn

Javascript-based display widgets



Port Wind Dir	339° M	Speed
Stbd Wind Dir	334° M	Speed
Lat	22.0099° S	Lon
Heading	217.93° T	SOG
Pitch	-1.3°	Roll
RTemp	21.7942° C	Air Temp
Eng Voltage	12.26V	
Aq Room Flow	507.5	
Helo Deck Flow	565.7	
Hydro Lab Flow	233.8	
F		

Payout: 0 m

Warnings

2018-09-21T1
2018-09-21T1
2018-09-21T1
2018-09-21T1
2018-09-21T1
2018-09-21T1
2018-09-21T1

Payout: -34

Warnings

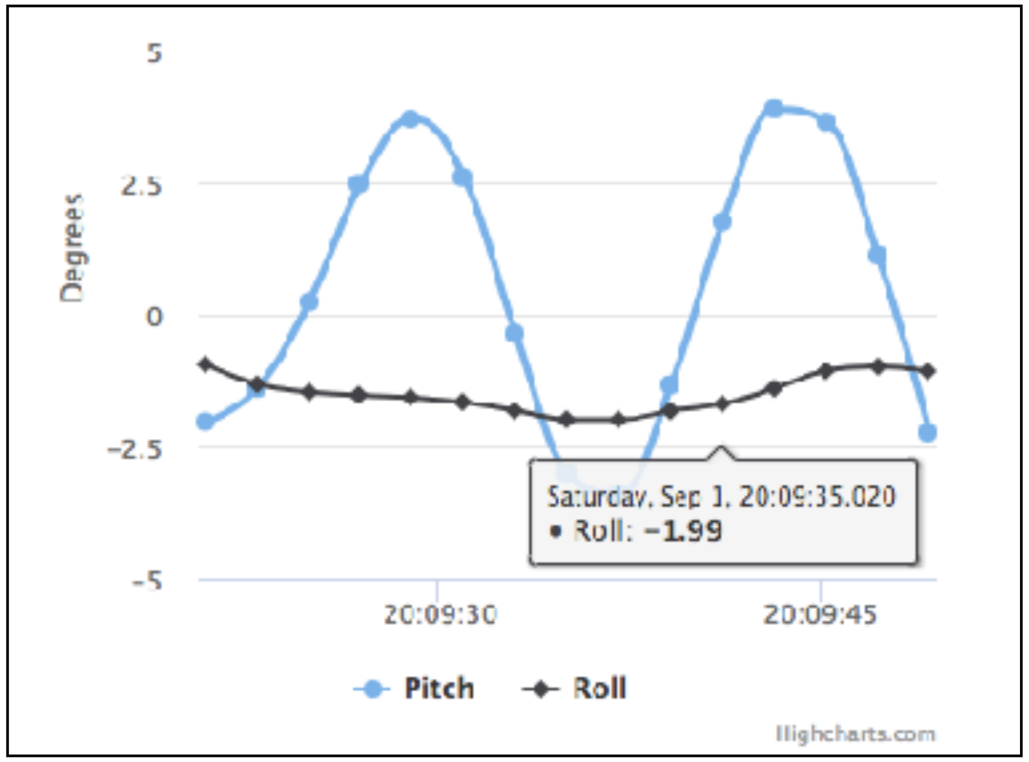
Attitude: Pitch -2.21°, Roll -1.08°

Relative Speed: Port: 11.68 m/s, Stbd: 10.45 m/s

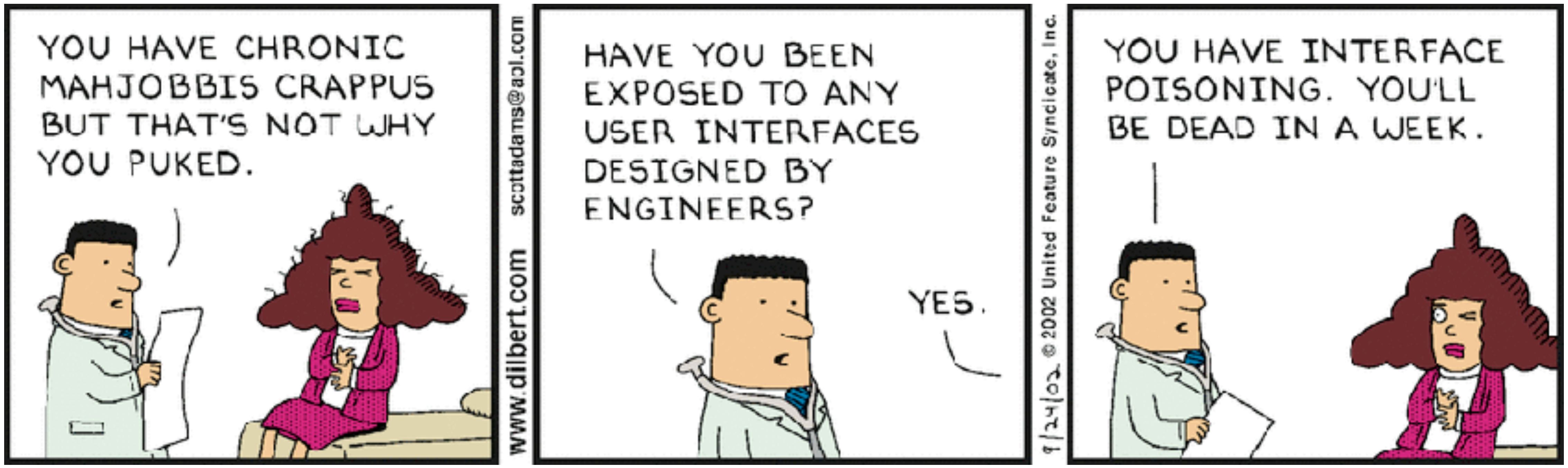
Position: Lat 62.1508°S, Lon 57.6808°W

Javascript-based display widgets

```
<div id="line-container" style="height:400px;min-width:310px"></div>

<script type="text/javascript">

  var line_fields = {
    S330Pitch: {
      name: "Pitch",
      seconds: 30
    },
    S330Roll: {
      name: "Roll",
      seconds: 30
    }
  };

  var widget_list = [new TimelineWidget('line-container',
                                        line_fields, 'Degrees'))];
  var widget_server = new WidgetServer(widget_list, WEBSOCKET_SERVER);
  widget_server.serve();
</script>
```



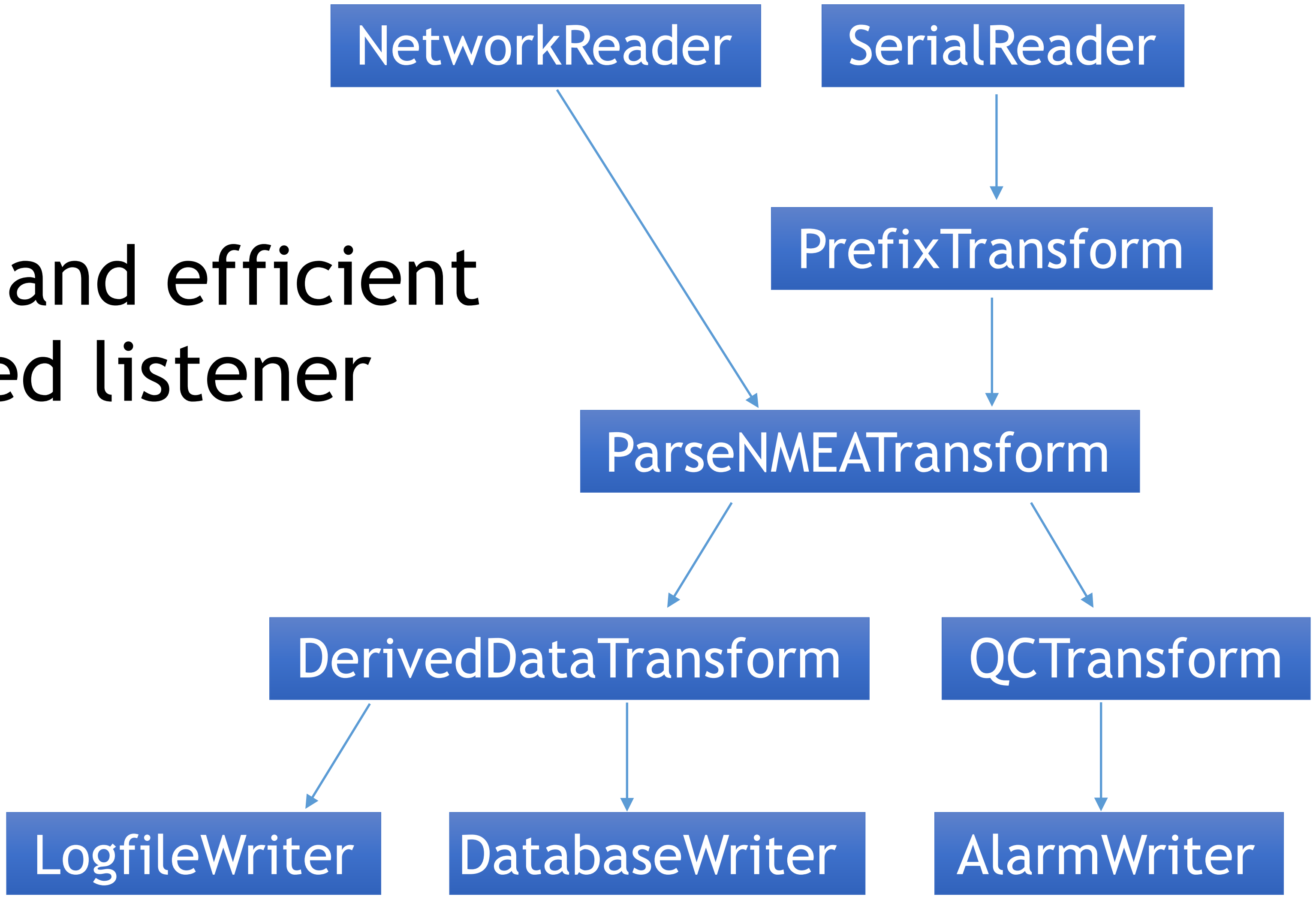
What’s next?


YOU HAVE CHRONIC MAHJOBBIS CRAPPUS BUT THAT'S NOT WHY YOU PUKED.
scottadams@aol.com
www.dilbert.com
HAVE YOU BEEN EXPOSED TO ANY USER INTERFACES DESIGNED BY ENGINEERS?
YES.
9/24/02 © 2002 United Feature Syndicate, Inc.
YOU HAVE INTERFACE POISONING. YOU'LL BE DEAD IN A WEEK.

Better Control UX

What's next?

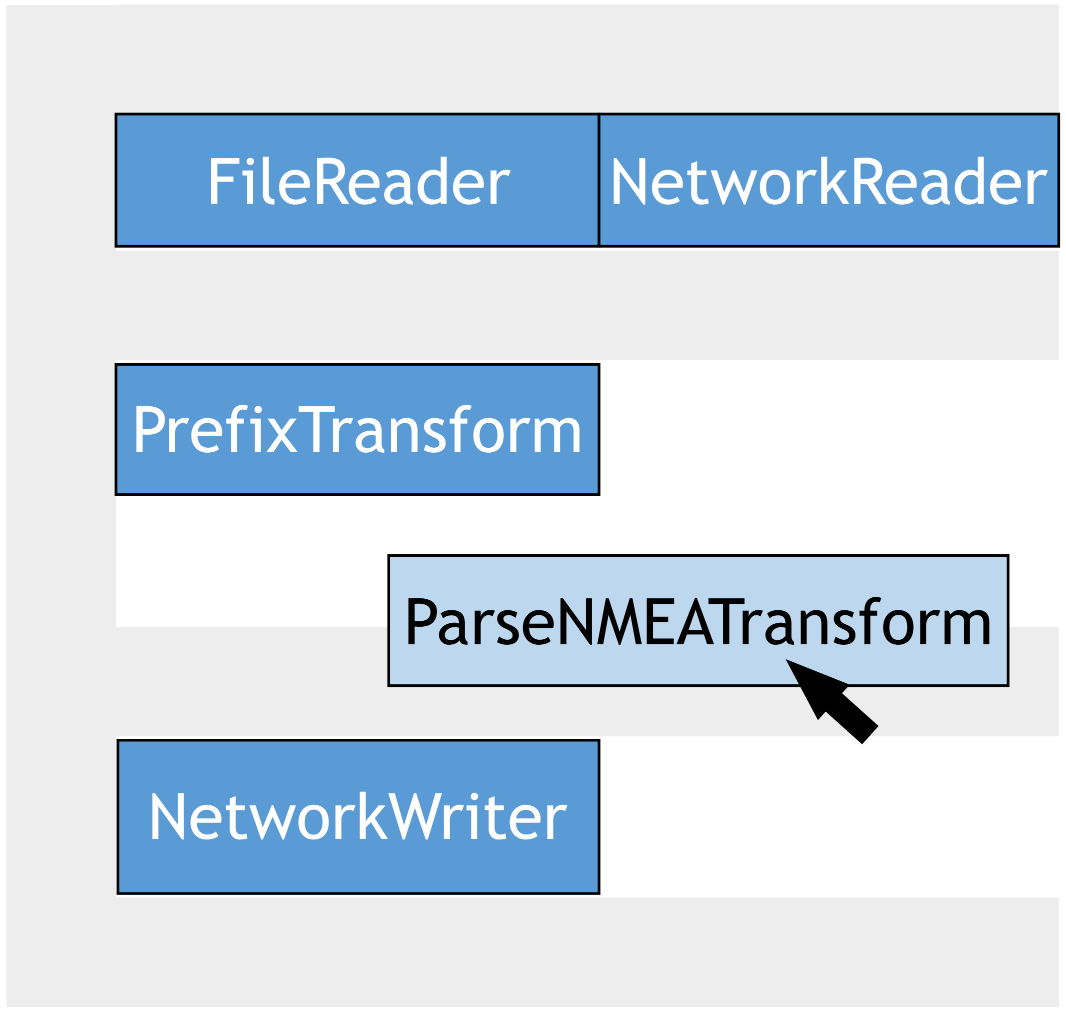
More flexible and efficient dataflow-based listener



What’s next?

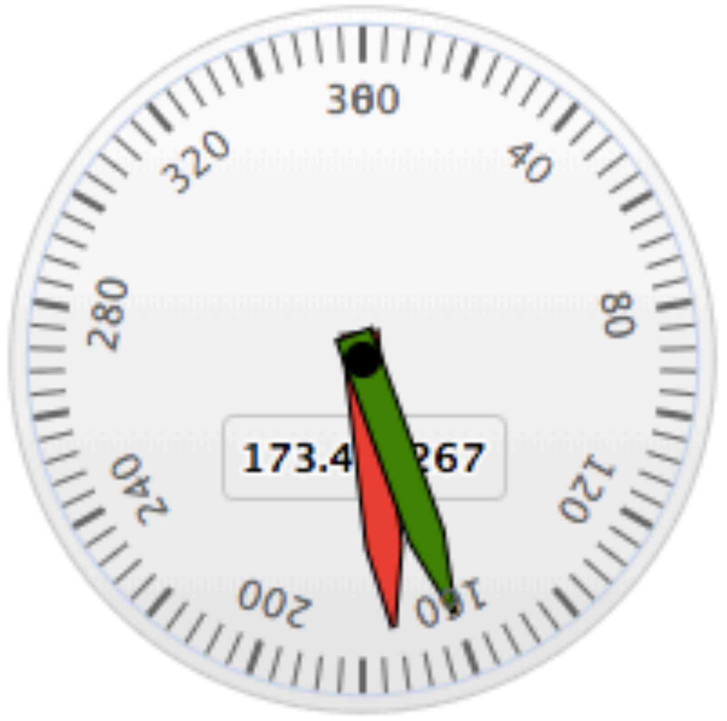
Better ways to compose Readers/Transforms/Writers

Maybe Scratch-like visual interface?



What’s next?

GUI-based creation and customization of entire displays



What’s next?

Sea trials!

For more information

http://openrvdas.org
http://github.com/davidpablocohn/openrvdas
david.cohn@openrvdas.org