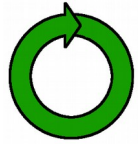
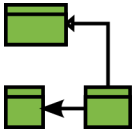
From Jupyter to Software



Lifecycle



project
structure

tests

docs

Dr. Kristian Rother

Jupyter

Copy-paste

Import everything

Improvise

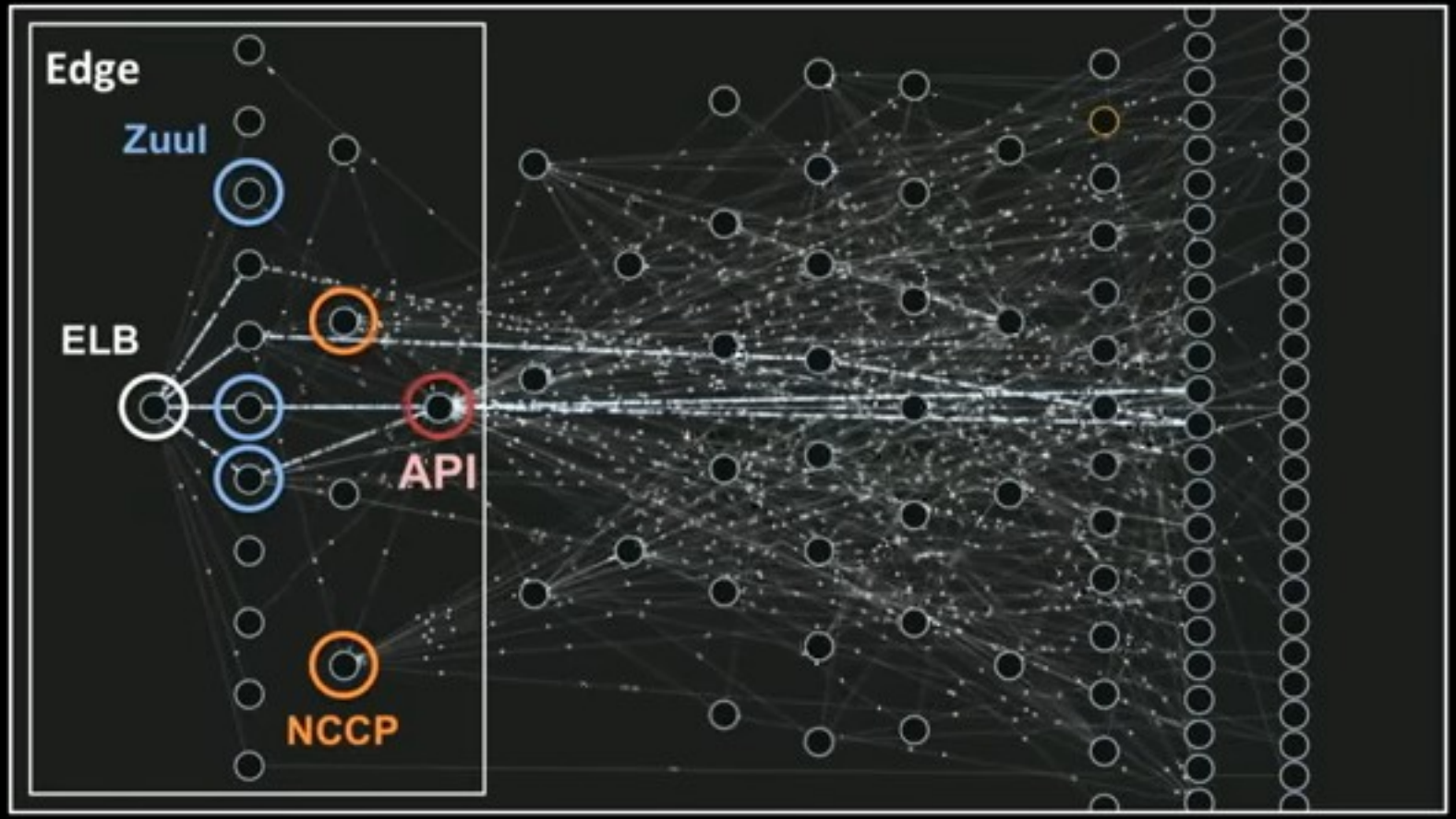
Just write it

Stackoverflow

Throwaway code

photo by zhang kaiyv on Unsplash

From: Mastering Chaos - A Netflix Guide to Microservice – Josh Evans



Big Programs

Architecture

Data Infrastructure

Design Patterns

Microservices

Monolith

Scrum

Software Engineering

UML

photo by Eduard Militaru on Unsplash

“Today I deleted

300 lines of code”

(Allegra Via)

photo by Erin Wilson on Unsplash

Lehmanns Laws

1) Software becomes less useful over time

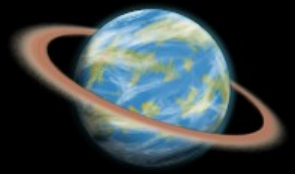
2) The complexity of software increases over time

..

7) Software deteriorates unless maintained rigorously

Conways Law

The architecture of a software system resembles the communication structure of its creators.



New Haven:

A earth-like yet uninhabited world.

Available commands:

[1] warp to Rorke
[2] warp to Octygon
[3] warp to Kucharsky
[4] collect food
[5] collect spices

[Esc] Exit

cargo bay:

artifacts:

github.com/krother/space

Rule #1: Know your league

Lines of Code	Structure
< 100	Jupyter Notebook / simple script
< 1000	Functions
< 10000	Classes
< 100000	Packages
< 1 million	*do you really want to do this?*

Count lines of code: `cat *.py | wc -l`

Rule #2: Structure Your Data

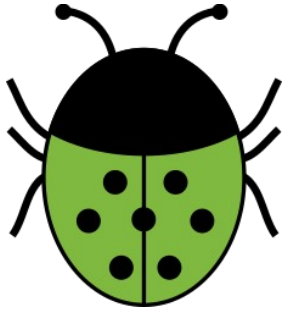
```
{"name": "Vega",
  "description": "A fertile world with rich aquatic life forms.",
  "image": "005",
  "connections": ["Sirius", "Centauri", "Octygon", "Aquatic City"]
},
```

One JSON file is better than 10 classes

Rule #3:

git

Rule #4: Learn ipdb



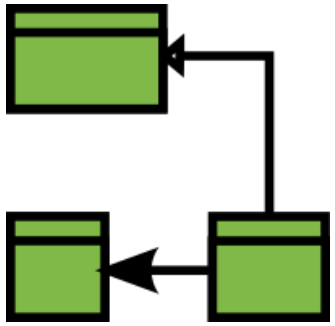
Ipdb:
interactive Python Debugger

```
(base) kristian@kristian-spiced:~/projects/space$ python -m ipdb space.py
> /home/kristian/projects/space/space.py(3)<module>()
      2 Space Traveller - main app
----> 3 '''
      4

ipdb> b 50
Breakpoint 1 at /home/kristian/projects/space/space.py:50
ipdb> c
> /home/kristian/projects/space/space.py(50)__init__()
     49         self.ship = Spaceship()
1--> 50         self.ship.location = start_location
     51         self.commands = self.ship.get_commands()

ipdb> start_location.name
'Terra'
```

Rule #5: Add essential files



- README.md
- .gitignore
- requirements.txt
- setup.py
- LICENSE (MIT good default)

- Tools: *cookiecutter, fabric*

Rule #6: pylint

```
(base) kristian@kristian-spiced:~/projects/space$ pylint planets.py
************* Module planets
planets.py:12:18: C0326: Exactly one space required after assignment
        self.name =kwargs['name']
                  ^ (bad-whitespace)
planets.py:9:0: R0902: Too many instance attributes (15/7) (too-many-instance-attribut
es)

--------------------------------------------------------------------
Your code has been rated at 9.62/10 (previous run: 9.61/10, +0.01)
```

Enforces the PEP8 style guide – makes your code more readable

Still Rule #6: isort

```
import time
from ships import Spaceship
from planets import create_galaxy
from views import outro, print_message
import views
import re
import random
import arcade
from arcade.key import ESCAPE, SPACE
from arcade import key as akeys
from lang import EN as TEXT
```

before

```
import random
import re
import time

import arcade
from arcade import key as akeys
from arcade.key import ESCAPE, SPACE

import views
from lang import EN as TEXT
from planets import create_galaxy
from ships import Spaceship
from views import outro, print_message
```

after

Rule #7: pytest

```
class TestSpace:

    def test_pickup(self, space):
        space.move(3)
        assert space.ship.cargo == 'food'

    def test_warp(self, space):
        space.move(1)
        assert space.ship.location.name == 'Centauri'

    def test_triple_warp(self, space):
        travel(space, [1, 1, 1])
        assert space.ship.location.name == 'New Haven'
```

Rule #7: pytest

```
_____________________________ TestSpace.test_triple_warp ______________________

self = <test_space.TestSpace object at 0x7fd504bd90b8>
space = SpaceGame(width=1500, height=1000)

    def test_triple_warp(self, space):
        travel(space, TRIPLE_JUMP)
>       assert space.ship.location.name == 'New Haven'
E       AssertionError: assert 'Terra' == 'New Haven'
E         - Terra
E         + New Haven

test_space.py:37: AssertionError
```

Rule #7: pytest

```
(base) kristian@kristian-spiced:~/projects/space$ pytest
================================= test session starts ==================================
platform linux -- Python 3.6.5, pytest-3.8.2, py-1.8.0, pluggy-0.9.0
rootdir: /home/kristian/projects/space, inifile:
plugins: remotedata-0.2.1, openfiles-0.3.0, doctestplus-0.1.3, cov-2.6.0, arraydiff-0.
2
collected 6 items

test_space.py ......                                                            [100%]

============================== 6 passed in 1.90 seconds ================================
```

Rule #8: Travis CI

```
language: python
before_script:
  - sleep 3
python:
  - "3.6"
# command to install dependencies
install:
  - pip install -r requirements.txt
  - pip install -r requirements_dev.txt
# command to run tests
script:
- pylint *.py
- pytest test_space.py
```

Space

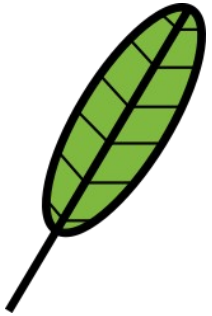
contact: krother@academis.eu

Swiss Cheese Model
(PD, Wikimedia Commons)

Rule #9:
Verification & Validation

- Code reviews
- Mypy (static type checker)
- Random testing (hypothesis)
- non-Python languages

Rule #10: Documentation



- README.md
- GitHub pages
- Sphinx

github.com/krother/space