Nine mistakes I've done while building network automation

Allan Eising

For DKNOG13

Telia Company

Why this talk?

”Learn from the mistakes of others. You can’t live long enough to make them all yourself”
- Eleanor Roosevelt

About Me

— Worked in the service provider industry for 17 years
— Worked full time with network automation for three years now
— Lead developer and architect, automation team at Telia Company
— Primarily responsible for common architecture, and Norway as a country
— Co-founder of DKNOG
— Semi-dead blog: https://automate.network
— Twitter: @allaneising
— Mastodon: @eising@noc.social

— Made a lot of mistakes

Are mistakes bad?

— Maybe…

— Nothing is impossible to fix, but not all mistakes are possible to fix for me personally.

— It is mostly temporary solutions that end up being permanent. Not mistakes.

9 mistakes and some possible solutions

Telia Company

Mistake #1

Not testing my automation on real hardware

Mistake #1
Not testing my automation on real hardware

“I tested using **virtual** routers”

“I tested using **simulated** routers”

“My code broke when it hit production”

Mistake #2
Thinking I understand all the details

“I’m a **network engineer**. I know how this works”

”My code **assumed** something that was untrue
and now we have a **hard to find** bug”

Possible solutions

— Get a **lab**

— Insist on getting a **lab**

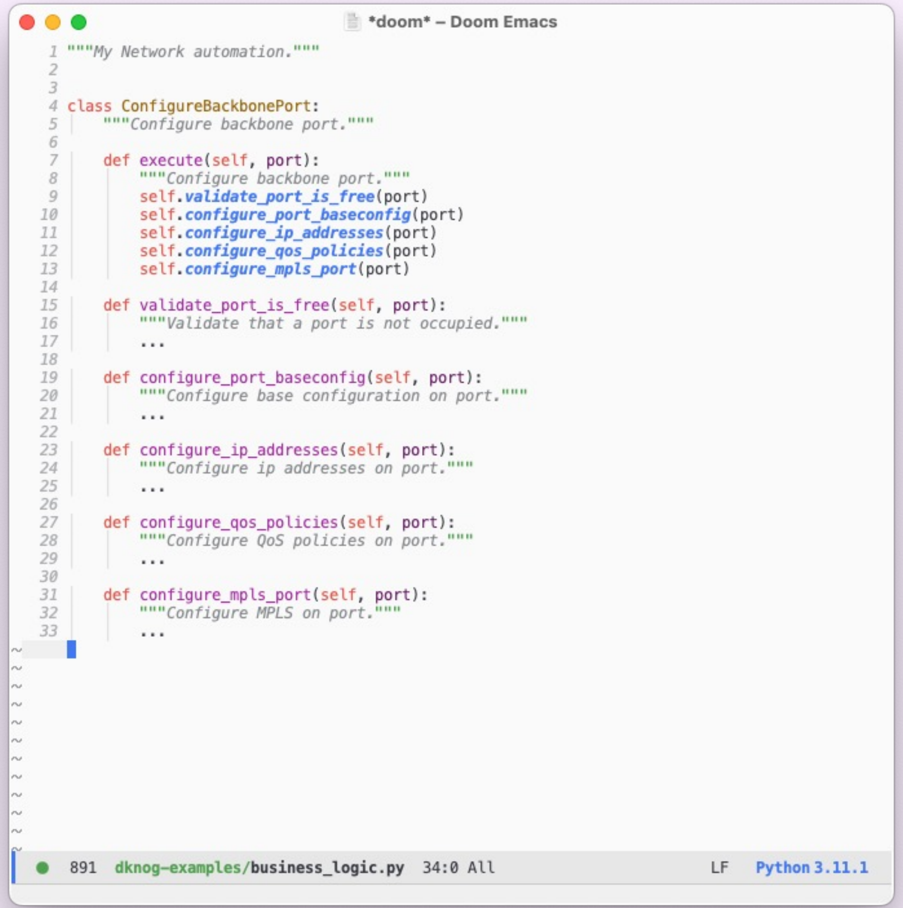
— (Don't) test in **production**

— Involve your **experts** early

— Give your experts time to **test** your code (preferrably in a lab)

— Write the **business logic** into your code

— Be humble



```python
"""My Network automation."""


class ConfigureBackbonePort:
    """Configure backbone port."""

    def execute(self, port):
        """Configure backbone port."""
        self.validate_port_is_free(port)
        self.configure_port_baseconfig(port)
        self.configure_ip_addresses(port)
        self.configure_qos_policies(port)
        self.configure_mpls_port(port)

    def validate_port_is_free(self, port):
        """Validate that a port is not occupied."""
        ...

    def configure_port_baseconfig(self, port):
        """Configure base configuration on port."""
        ...

    def configure_ip_addresses(self, port):
        """Configure ip addresses on port."""
        ...

    def configure_qos_policies(self, port):
        """Configure QoS policies on port."""
        ...

    def configure_mpls_port(self, port):
        """Configure MPLS on port."""
        ...
```

```
891  dknog-examples/business_logic.py  34:0 All        LF  Python 3.11.1
```

Mistake #3

The golden hammer

Mistake #3
The golden hammer

“We’re using *[framework]* for **everything**”

”Even though i**t doesn’t support it**, we’ve made [application] do it anyway”

”We pay a lot of money to [**vendor**], so we must use their [thing]!”

“Our solution has become way too **complex**!”

Mistake #4
Building too much complexity into my code

“Network engineering is **complicated**, and so is the automation”

“I’m executing complicated **workflows** from inside one application”

“This **corner case** is taking up the majority of my code”

“My colleagues are **avoiding me** at lunch”

Possible solutions

- **Who** did this to you?
- **Explore** new ways of doing things.
- Build your applications with **flexibility** in mind
- Talk to your industry peers and **learn new approaches**
- **Refactor** your code early and often, or **start over** fresh
- Identify if your logic is better solved with **smaller components** and an central **orchestrator**
- Have a **test suite**

Mistake #5

The lone cowboy

Mistake #5
The Lone Cowboy

“I made **something cool** and now it’s part of production”

“I have a different role, so I made it in **my spare time**”

“The thing **broke** and nobody are able to **help** me”

Mistake #6
Poor code quality

”My stack has **grown too big** and it is **failing randomly**”

“**Unknown exceptions** happen that make no sense to users”

”My code is **hard to read**, my colleagues can’t help me”

“My code is **hard to understand**, so I won’t let anybody help me”

Possible solutions

- Try not to write business critical software alone
- Don’t give your spare time to work for nothing.
- Try to sell it off as a project with proper time allocations.
- Make your manager your ally early
- Use code linting, unit tests, auto formaters, etc.
- Look up a code styleguides.
- Hold code reviews regularly

Python links

Google Python Style Guide: http://google.github.io/styleguide/pyguide.html

Black autoformater: https://github.com/psf/black

Python Language Server: https://github.com/python-lsp/python-lsp-server

Mistake #7
Automatic tests that don't test realistic scenarios

"I made up my **test data** myself"

"I test my services **one at a time**"

"I just **mock** the APIs of the systems I integrate with"

"My services didn't work with **real-world data**"

"The systems didn't respond like I thought they would"

Possible solutions

— Create an integration test environment

— Try to replicate what the users are doing in your tests

— Get real-life data from your users or experts and use that in your tests

— Test services in combination, not just create, but also delete and update

Mistake #8

Not listening to my users

Mistake #8
Not listening to my users

“I built the system, so **I know** best how to use it”

“Why can’t the users **read my mind?**”

“Why don’t they **read the documentation**?”

“They broke everything because they didn’t **read the warnings**!”

Possible solutions

— Treasure the user who doesn’t read the documentation and breaks things

— Make sure to log all interactions and provide ways to revert breaking changes

— Talk to your users often and understand how they are using your tools

— Consider doing small videos complementing your text documentation

— Consider classroom training

Mistake #9

Not making mistakes

Mistake #9
Not making mistakes

“I don’t **take risks**, but I don’t achieve very much”

“I **worry** too much about **failing** to even start”

”I **overthink** my solutions and will never be done”

“I only read the mistake slides in this presentation. Were there solutions too?”

Possible solutions

— Start small, define an MVP

— Think about how to test your solutions

— Make yourself accountable

— Allow yourself to fail and learn from the experience

— Find networkers at NOG events and listen to their war stories

— Tell about your mistakes at NOG events

Thank you!