Towards Quality Assurance of SPLs with Adversarial Configurations

Paul TEMPLE [1] Mathieu ACHER [2] Gilles PERROUIN [1]
Battista BIGGIO [3] Jean-Marc JEZEQUEL [2] Fabio ROLI [3]

[1]PReCISE, NaDI, Université de Namur ; @pleupi22 @GPerrouin

[2]Univ Rennes, CNRS, Inria, IRISA ; @acherm @jmjezequel

[3]PRALab, University of Cagliari ; @biggiobattista

Published at EMSE'21

Journal : `https://doi.org/10.1007/s10664-020-09915-7`
Open Access : `https://hal.inria.fr/hal-03045797/document`

Configurable systems

Linux Kernel: **15,000** options

Configurable systems

Linux Kernel: **15,000** options

$2^{15,000} \approx 10^{3,250} >> 10^{1,000} >>$ estimated # of particules

JHipster

JHipster: **50** options

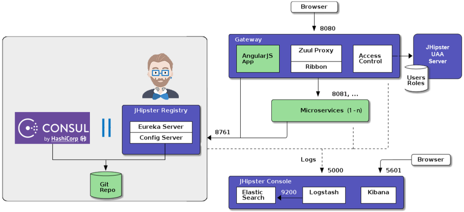
- Generate web app
- Micro-services

Halin *et al.*, https://doi.org/10.1007/s10664-018-9635-4
https://www.jhipster.tech/microservices-architecture

JHipster

JHipster: **50** options

- Generate web app
- Micro-services
- 26,000+ configurations

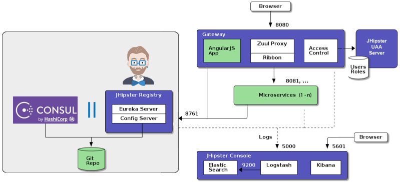


Halin *et al.*, `https://doi.org/10.1007/s10664-018-9635-4`
`https://www.jhipster.tech/microservices-architecture`

JHipster: **50** options

- Generate web app
- 26,000+ configurations

NETFLIX OSS + + docker



elastic + logstash + kibana

Problem:

- Which variants can be build under X seconds?
- Which variants can run with limited resources? (less than Y Watts)

Reducing configuration space with ML

JHipster → 26,000+ configurations

- **Exploring** all configurations is **impossible**
- **Measuring** performances is **costly**
- **Few** configurations are **acceptable** (meet performances)

→ How to find the few interesting one?

Reducing configuration space with ML

JHipster → **26,000+** configurations

- **Exploring** all configurations is **impossible**
- **Measuring** performances is **costly**
- **Few** configurations are **acceptable** (meet performances)

→ How to find the few interesting one?

Performance predictions

- Assumption: similar configuration → similar performances
- **Train** a model on **few** configurations
- **Predict** on **all** the others

Reducing configuration space with ML

JHipster $\rightarrow$ **26,000+** configurations

- **Exploring** all configurations is **impossible**
- **Measuring** performances is **costly**
- **Few** configurations are **acceptable** (meet performances)

$\rightarrow$ How to find the few interesting one?

Performance predictions

- Assumption: similar configuration $\rightarrow$ similar performances
- **Train** a model on **few** configurations
- **Predict** on **all** the others

$\rightarrow$ **Filter** to retrieve the interesting ones

Reducing configuration space with ML

Scope the configuration space

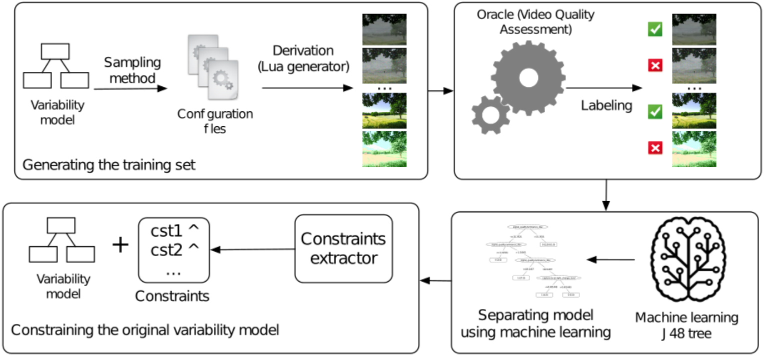
- User: "I want technology X and Y but the system should run under S seconds"
- Learn which configurations are ok with that

Temple *et al.*, Using machine learning to infer constraints for product lines, SPLC'16

Reducing configuration space with ML

Scope the configuration space

- User: "I want technology X and Y but the system should run under S seconds"
- Learn which configurations are ok with that



Temple *et al.*, Using machine learning to infer constraints for product lines, SPLC'16

Improving the classification of software configurations

Impacts

Machine Learning is based on statistics $\rightarrow$ errors

- Over-constraining
- Under-constraining

Improving the classification of software configurations

Impacts

Machine Learning is based on statistics $\rightarrow$ errors

- Over-constraining
- Under-constraining

Over-constraining

- Forbid more configurations than necessary
- Lack of flexibity

Improving the classification of software configurations

Impacts

Machine Learning is based on statistics $\rightarrow$ errors

- Over-constraining
- Under-constraining

Over-constraining

- Forbid more configurations than necessary
- Lack of flexibity

Under-constraining

- Allow more configurations than necessary
- Waste of resources

Improve the quality of the classifier

Robustifying the model

- Show new configurations
- Configurations with **high risk** of misclassification

Goodfellow *et al.*, https://arxiv.org/pdf/1412.6572.pdf

Elsayed *et al.*, https://proceedings.neurips.cc/paper/2018/file/8562ae5e286544710b2e7ebe9858833b-Paper.pdf

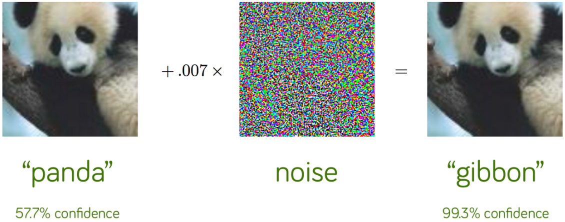
Sharif *et al.*, https://dl.acm.org/doi/pdf/10.1145/2976749.2978392

YT: https://www.youtube.com/watch?v=6Xh1vuwnVhU

Improve the quality of the classifier

Robustifying the model

- Show new configurations
- Configurations with **high risk** of misclassification



Improve the quality of the classifier

Robustifying the model

- Show new configurations
- Configurations with **high risk** of misclassification

Robustifying the model

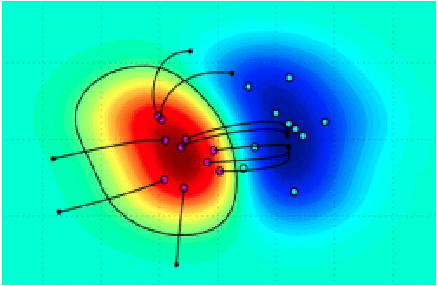
- Show new configurations
- Configurations with **high risk** of misclassification

Goodfellow *et al.*, https://arxiv.org/pdf/1412.6572.pdf
Elsayed *et al.*, https://proceedings.neurips.cc/paper/2018/file/8562ae5e286544710b2e7ebe9858833b-Paper.pdf
Sharif *et al.*, https://dl.acm.org/doi/pdf/10.1145/2976749.2978392
YT: https://www.youtube.com/watch?v=6Xh1vuwnVhU

Improve the quality of the classifier

Adv. configurations, how does it work?

- Configurations of a system = **vector of (Binary) options**
- Model is trained on few configurations
- Choose a (known) configuration
- Apply gradient descent → **modifications** on the values of options
- **Repeat** gradient descent until happy



Biggio *et al.*, `https://arxiv.org/pdf/1708.06131.pdf`

Adv. ML and software systems

Adv ML allows for:

- Automatic generation of new configurations → iterative modifications
- prediction error mitigation → adversarial retraining
- enhanced design space coverage → adversarial configurations

Adv. ML and software systems

Adv ML allows for:

- Automatic generation of new configurations $\rightarrow$ iterative modifications
- prediction error mitigation $\rightarrow$ adversarial retraining
- enhanced design space coverage $\rightarrow$ adversarial configurations

Future works

- Need for better support for domain constraints
- Integration with constraint solvers
- Can adv ML be used as a new sampling strategy?

Sum-up

- Introduce adv ML to variability
- ML models useful to deal with configurable design spaces
- Generate configurations with high risk of misclassification
- Open directions for new sampling strategies?
- Need for better constraint support

Sum-up

- Introduce adv ML to variability
- ML models useful to deal with configurable design spaces
- Generate configurations with high risk of misclassification
- Open directions for new sampling strategies?
- Need for better constraint support

Contact us:

- Journal : `https://doi.org/10.1007/s10664-020-09915-7`
- Open Access : `https://hal.inria.fr/hal-03045797/document`

Twitter: @pleupi22 / @acherm / @GPerrouin / @biggiobattista / @jmjezequel