

# Towards Quality Assurance of SPLs with Adversarial Configurations

Paul TEMPLE<sup>1</sup>   Mathieu ACHER<sup>2</sup>   Gilles PERROUIN<sup>1</sup>  
Battista BIGGIO<sup>3</sup>   Jean-Marc JEZEQUEL<sup>2</sup>   Fabio ROLI<sup>3</sup>

<sup>1</sup>PreCISE, NaDI, Université de Namur ; [@pleupi22](#) [@GPerrouin](#)






<sup>2</sup>Univ Rennes, CNRS, Inria, IRISA ; [@acherm](#) [@jmjezequel](#)

<sup>3</sup>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

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



**Sébastien Mosser** @petitroll · 25 févr.

...

"the number of atoms in the visible universe is  $10^{80}$ . There are  $2^{15000}$  different versions of the Linux kernel. So astrophysicists works with things way simpler than software engineers". @jmjezequel



JHipster: **50** options

- Generate web app
- Micro-services

---

Halin et al., <https://doi.org/10.1007/s10664-018-9635-4>

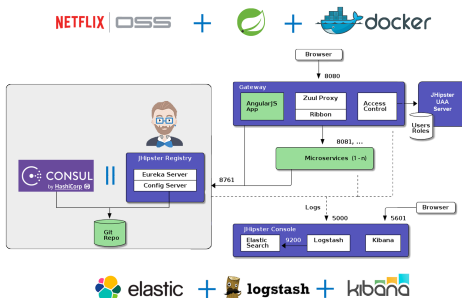
<https://www.jhipster.tech/microservices-architecture>

# JHipster



JHipster: **50** options

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



Halin et al., <https://doi.org/10.1007/s10664-018-9635-4>

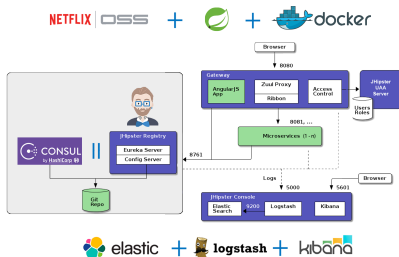
<https://www.jhipster.tech/microservices-architecture>

# JHipster



JHipster: **50** options

- Generate web app
- 26,000+ configurations



## 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 → **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

→ **Filter** to retrieve the interesting ones

# Reducing configuration space with ML

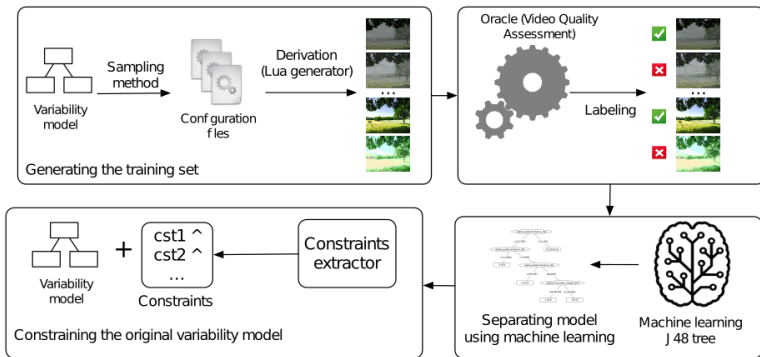
## Scope the configuration space

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

# 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 → errors

- Over-constraining
- Under-constraining

# Improving the classification of software configurations

## Impacts

Machine Learning is based on statistics → errors

- Over-constraining
- Under-constraining

## Over-constraining

- Forbid more configurations than necessary
- Lack of flexibility

# Improving the classification of software configurations

## Impacts

Machine Learning is based on statistics → errors

- Over-constraining
- Under-constraining

## Over-constraining

- Forbid more configurations than necessary
- Lack of flexibility

## 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



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

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

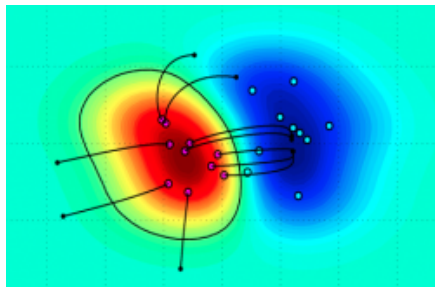
Sharif *et al.*, <https://dl.acm.org/doi/pdf/10.1145/2976749.2978392>

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



## Adv ML allows for:

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

## Adv ML allows for:

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

## 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 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 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](#)