USC Viterbi
School of Engineering
Information Sciences Institute

Modeling cognitive workload in open-source communities via simulation

Alexey Tregubov, Jeremy Abramson, Christophe Hauser, Alefiya Hussain, and Jim Blythe

Motivation

- Protecting open-source software (OSS)
- Understanding underlying socio-technical process
- Improve work experience of people engaged in the OSS community (e.g. Linux Kernel)

LKML

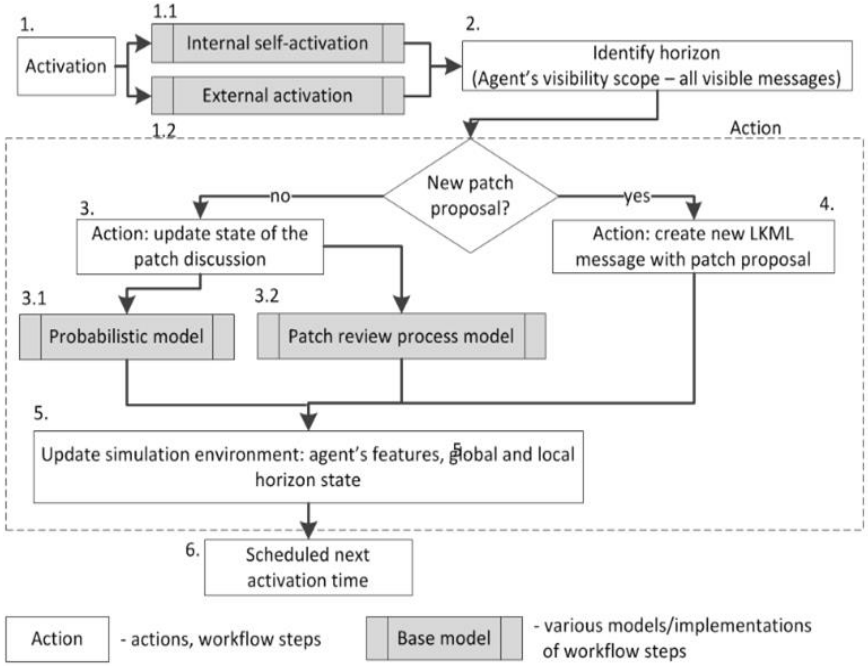
- Linux Kernel Mailing List (LKML) - OSS development artifacts: *code & communication*
- Developers, maintainers, reviewers discuss updates (patches) of the Linux Kernel.
- LKML messages capture a patch life cycle.

Goals

- Develop an agent-based model of OSS development
- Explore benefits of explicit process models as part of multi-agent systems (LKML patch review process)
- Compare predictions of buggy/reverted LKML patches by multi-agent simulation and by ML models (ABS vs. ML predictions)
- Explore relationships *the number of new patch proposals* and *the percent of patches that are later reverted*

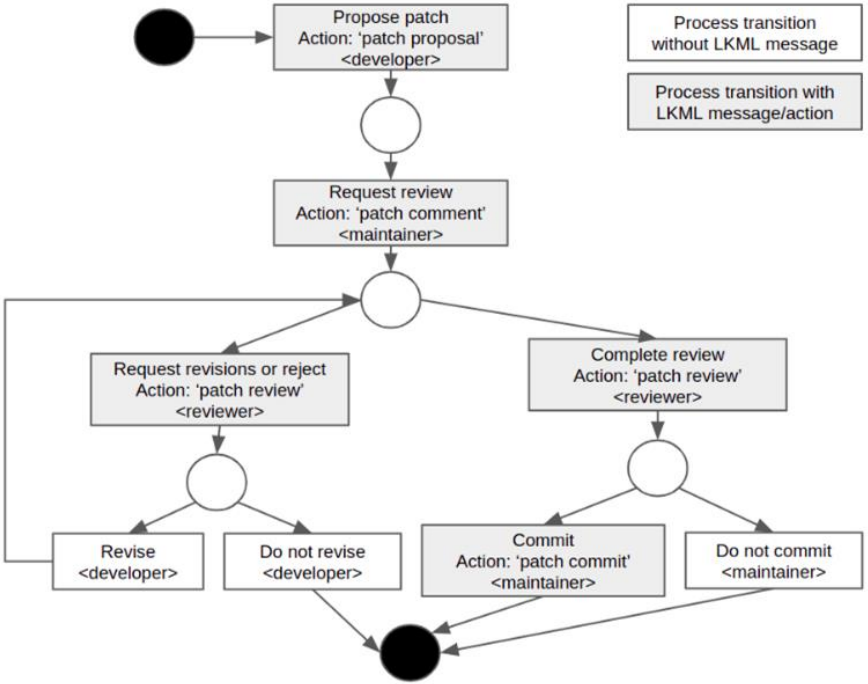
Agent-based Simulation Model of LKML

- DASH framework - a Python-based DES framework for modeling cognitive agents
- Patch discussions (message threads about patches) modeled explicitly, their status is updated by agents.
- Two simulation models:
  - probabilistic model
  - patch review process model



Patch review process

- Patch review process is modeled explicitly in our DASH model.
- State transitions are performed by agents/users with appropriate roles.



Simulation of Patch Reversal

In simulation models, the probability of patch reversal is dependent on likelihood of the developer to propose a patch that would be reverted (based on training data statistics) and of the reviewer to miss bugs in the review process.

We estimate the probability of patch being reversed as follows:

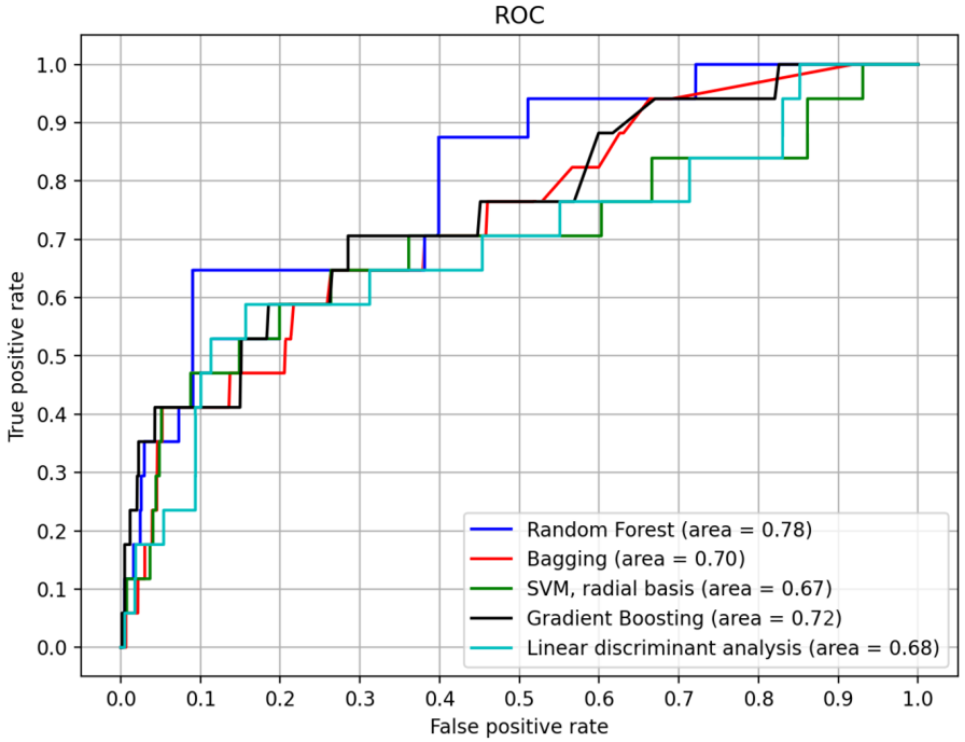
$$P = M_{load} \times P_{dev}$$

$$P_{dev} = \frac{n\_patches\_reverted}{n\_patches\_proposed}$$

M_{load} here is a reviewer workload/multitasking factor, which is a heuristic function that associates length of the work queue with increase of probability to miss a bug in the review process.

ML models for patch reversal prediction

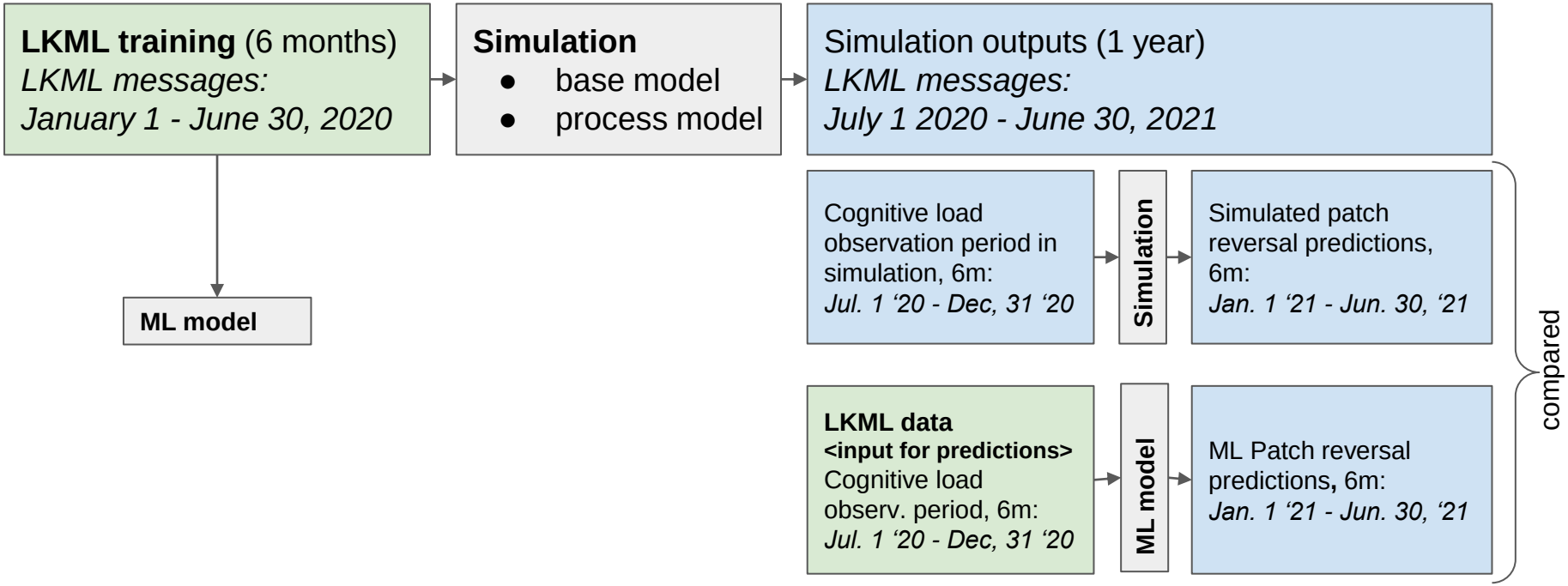
- Patch reversal prediction is a binary classification problem for a given feature vector of a LKML patch discussion
- Two groups of features used: code features and social features
- Several classifiers were compared, *Random Forest* classifier has highest AUC (0.78)



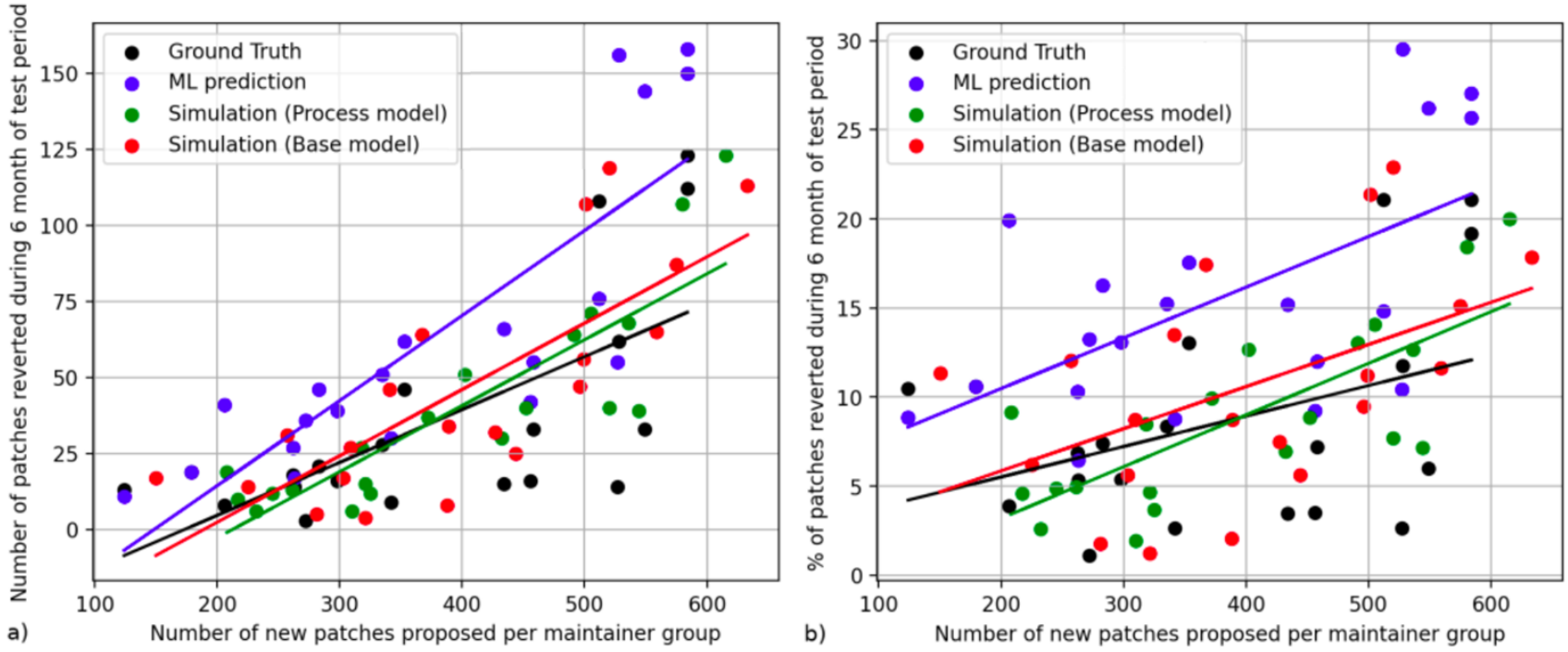
Experiment Setup

- Training: 212k LKML messages, 5K authors, January 1 - June 30 of 2020.
- Simulation: July 1 2020 - June 30 2021.
- The two simulation models: base model and process model
- Predictions on patch reversal were compared with ML model predictions.

Experiment Setup (cont-d)



Results: relation between number of new patches and patches reverted

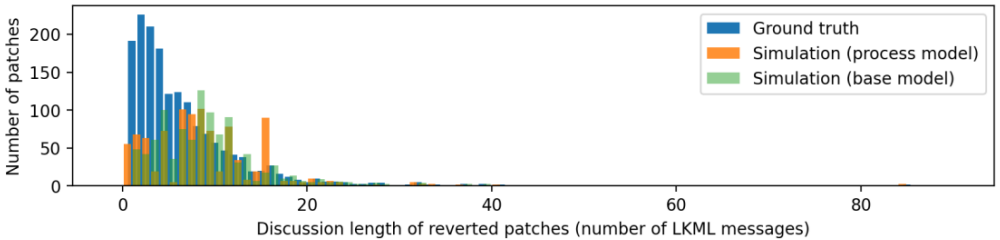


Results: goodness of fit to ground truth

- K-S test shows goodness of fit simulation models' predictions to the ground truth.

	D statistic	p-value
ML model	0.02107	0.7554
Simulation (Process model)	0.01618	0.9524
Simulation (Base model)	0.02157	0.7298

- Patch discussions produced by the simulation have different distributions of discussion lengths.



Contributions Summary

- Agent-based model of OSS development, including code review process
- Process model that captures repeated, structured multi-agent activities
- Multi-agent simulation improves the prediction of buggy (reverted) code in Linux Kernel over to statistical ML models
- Demonstrated that a simple model of cognitive workload shows the positive association between *the number of new patch proposals* and *the percent of patches that are later reverted**

*This association is captured by both the simulation model and ML models as well as observed in the ground truth.

Discussion, Q&A