A CALL FOR REMOVING VARIABILITY

Mathieu Acher[1,2], Luc Lesoil[1], Georges Aaron Randrianaina[1],
Xhevahire Tërnava[1], Olivier Zendra[1]

[1]Université de Rennes 1, CNRS, Inria, IRISA
[2]Institut Universitaire de France (IUF)
Rennes, France

VaMoS'23, January 25-27, 2023

Why removing variability?

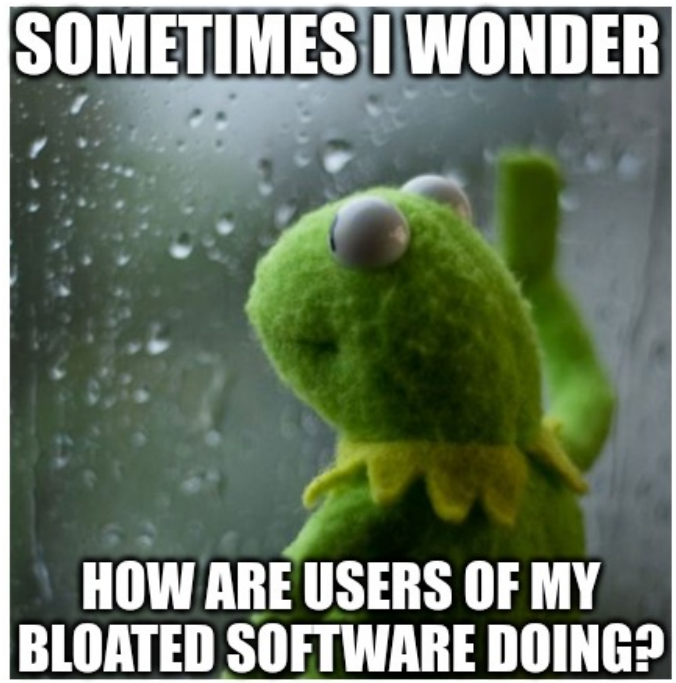
- Today's software systems include a multitude of features and tend to expand them over time
  - e.g., Linux: ≈20K options; $\approx 2^{20,000}$ configuration space (?)

Why removing variability?

- Today's software systems include a multitude of features and tend to expand them over time
  - ⊛ e.g., Linux: ≈20K options; $\approx 2^{20,000}$ configuration space (?)
- Development approaches strive to add variability

Why removing variability?

- Today's software systems include a multitude of features and tend to expand them over time
  - ⊛ e.g., Linux: ≈20K options; ≈ $2^{20,000}$ configuration space (?)
- Development approaches strive to add variability
- The removal of superfluous or underutilized variability has received far less to no attention from our community



HOW REDUCING VARIABILITY CAN BE A GOOD IDEA?

imgflip.com

How reducing variability can be a good idea?

- The ultra-high amount of variability in software systems is exceeding human and even machine limits to deal with it [1]
- Users are often overwhelmed by too many choices available[2] and lack the expertise and time to customize the system
- Up to 54.1% of options are rarely set by any user [3]
- Up to 75% of feature toggles become unused after 49 weeks [4]
- Up to 75.1% of software libraries are unneeded [5]

1: Hugo Martin, Mathieu Acher, Juliana Alves Pereira, Luc Lesoil, Jean-Marc Jézéquel, and Djamel Eddine Khelladi. 2021. Transfer learning across variants and versions: The case of Linux kernel size. TSE 2021

2: Linus Torvalds. Fragmentation is why Linux hasn't succeeded on Desktop. 2020

3: Tianyin Xu, Long Jin, Xuepeng Fan, Yuanyuan Zhou, Shankar Pasupathy, and Rukma Talwadker. Hey, you have given me too many knobs!: Understanding and dealing with over-designed configuration in system software. In the 10th Joint Meeting on Foundations of SE. 2015

4: Murali Krishna Ramanathan, Lazaro Clapp, Rajkishore Barik, and Manu Sridharan. Piranha: Reducing feature flag debt at Uber. ICSE-SEIP 2020

5: César Soto-Valero, Nicolas Harrand, Martin Monperrus, and Benoit Baudry. A comprehensive study of bloated dependencies in the maven ecosystem. EMSE 2021

Software systems are bloated [6,7]

utilities . . . . . Chromium

6: Gerard J. Holzmann. Code Inflation. Jet Propulsion Laboratory, California Institute of Technology. 2015

7: Chenxiong Qian, Hyungjoon Koo, ChangSeok Oh, Taesoo Kim, and Wenke Lee. 2020. Slimium: Debloating the Chromium Browser with Feature Subsetting. In SIGSAC CCS20. ACM, NY, 461–476

WHY SHOULD SOMEONE
BOTHER AND REMOVE VARIABILITY?

imgflip.com

Why should someone bother and remove variability?

- Reliability
  - Missconfigurations are often the source of software failures
  - May introduce technical debts
  - May change the primary purpose of the system (e.g., cat)
- Performance
  - Poor configuration choices
  - Systems tend to become encrusted with dubious features
- Security
  - May contain vulnerabilities
  - Corr: binary size, attack surface
  - Have the power to bankrupt the company (e.g., Knight Capital)
- Code complexity, testing burden, energy consumption, code hygiene, productivity of the devs, ...

Within 45 mins it lost $460M

"DO NOT EVER TOUCH THIS BUTTON"[1]

[1] https://www.youtube.com/watch?v=28ZAoStv-Xw

Why is removing variability not yet a major trend?

- Removing code is not a rewarding activity for Devs and PMs
- Lack of automated or integrated technologies
- Removing code is a complex socio-technical task
  - i.e., limited studies, understanding, and expertise on removing variability
  - Only 3 papers in 10 years in VaMoS and SPLC have "reducing" in the title
- Removing variability is different from: disabling it, removing dead (an unreachable) code, technical debt, or software bloat

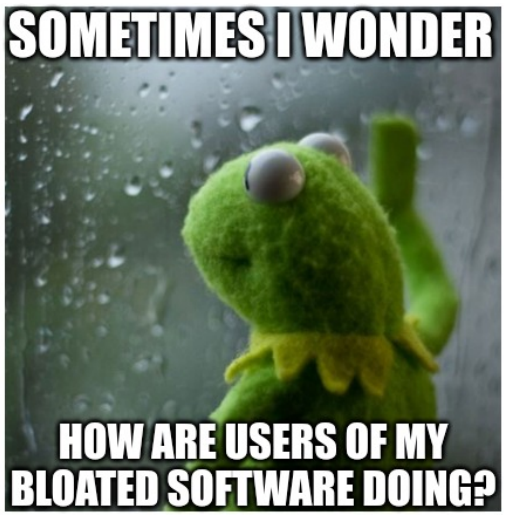
How to Remove Variability?

Some of the possible research directions are:

- Debloating variability (removing code is not completely new)
  - ⊛ What are the variability units subject to removal: features, options, feature flags, settings,...?; How to trace them?; How to remove them without breaking the remained functionalities? Who should remove them?
- Reverse engineering techniques should be revisited in light of a new objective: removing the unneded variability
- Designing software systems as "variability removal friendly"
- Developers' workflow for removing variability
- Removing variability: application or domain engineering?

A CALL FOR REMOVING VARIABILITY

Mathieu Acher, Luc Lesoil, Georges Aaron Randrianaina,
Xhevahire Tërnava, and Olivier Zendra



Related: Xhevahire Tërnava, Mathieu Acher, and Benoit Combemale. **Specialization of Run-time Configuration Space at Compile-time: An Exploratory Study.** The 38th ACM/SIGAPP Symposium on Applied Computing, Tallinn, Estonia, 2023 (SAC 2023). https://hal.science/hal-03916459

References and borrowed images

1. Image in slide 2: An adapted image from Kermit the Frog
2. Image in slide 3: Adapted the image from https://comicphrase.wordpress.com/reading/reading/
3. Image in sides 5 and 10: An adapted and borrowed images from Calvin and Hobbes