1 Choice

In a world:

- where generative AI automates numerous low-level SE tasks
- where Starlnk and and ChinaSat and SatNet and Project Kuiper and ViaSat abd HughesNet and OneWeb and Eutelsat and Telesat etc connect every programmer on earth
  - even those willing to work for 5% of American saleries
- then the salaries for programmers are about to crash, big time.

In that world, do not want to be a programmer:

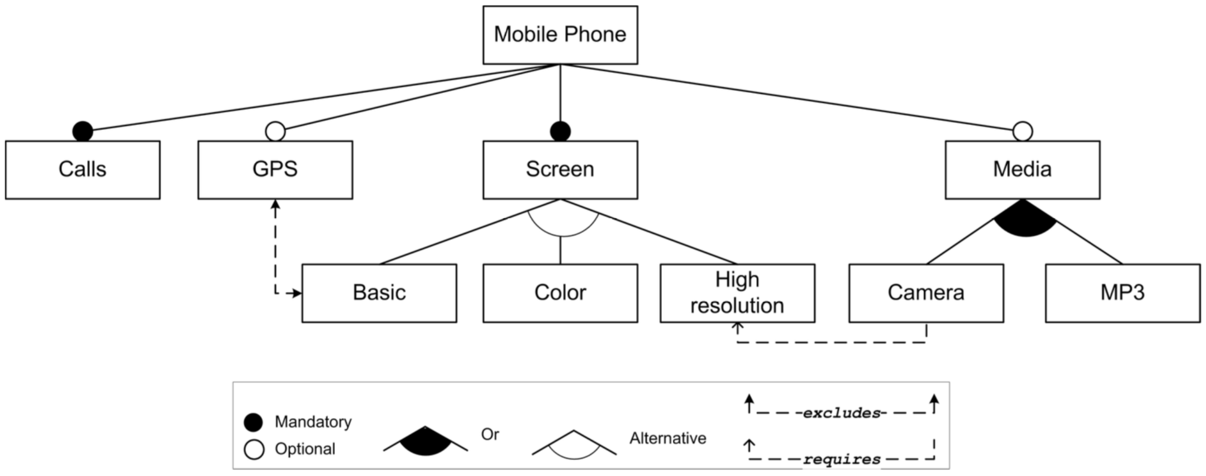
- You want to be the planning person deciding what programmers do
- Or you want to be the tester taking the code from the programmer, then assess it.

So this lecture argues that “choice”, not “coding” is a primary concern in SE

- which motivates the “choice-oriented” nature of the homewords and projects in this subject.

2 Motivating Example: Design of a phone

Here is a high-level design of a phone (written in Czarnecki's "feature model" notation [1]). What are some of the different kinds of phones condonned here? Given two such designs, when might you choose one, rather than the other?



Note how long to discuss just the choices in this tiny model. Real software is more more complex with many more choices. The LINUX kernel is the technological backbone of our global information society (it is used in the servers that power the internet, in data centers, in Android phones, just to name a few). That kernel has a grow number of thousands of features [2], each of which can be selected, or ignored in a particular system.

[1] Czarnecki, Krzysztof, Kasper Osterbye, and Markus Volter. "Generative programming." In European Conference on Object-Oriented Programming, pp. 15-29. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002. (for notation, see chapter 5)

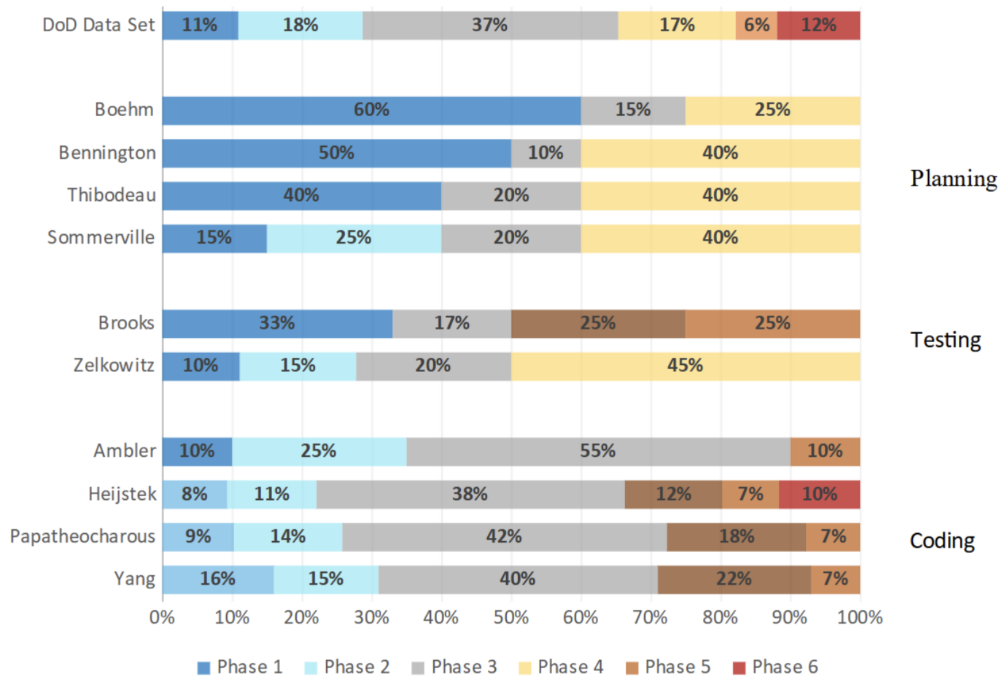
[2] L. Passos, et al.,"A Study of Feature Scattering in the Linux Kernel" in IEEE Transactions on Software Engineering, vol. 47, no. 01, pp. 146-164, 2021. doi: 10.1109/TSE.2018.2884911

3 SE is not about coding (usually)

Since there are so many of them, much of software engineering is not about coding. Rather it about the exporation and assessment of choice.

- Long et al. [3] ask the question "how much of SE is coding and how much is something else?".
- To answer that question, they look at 20 years of their own empirical data and 50 years of the SE literature.
- They break up SE development into six phases (listed below) and ask, in each, do software engineers (a) plan what to do; (b) do some coding; or (c) check what you got.
  1. Requirements Analysis
  2. Architecture & Design
  3. Coding
  4. System Integration
  5. Qualification Testing
  6. Development Test & Evaluation

Long et al. argue that, on balance, engineers spend a third of the time in planning, coding, and testing. And here are their numbers:



[3] D. Long, S. Drylie, J. Ritschel and C. Koschnick, 2023, "An Assessment of Rules of Thumb for Software Phase Management, and the Relationship between Phase Effort and Schedule Success," in IEEE Transactions on Software Engineering, doi: 10.1109/TSE.2023.3339383.

4 SE is about choice (often)

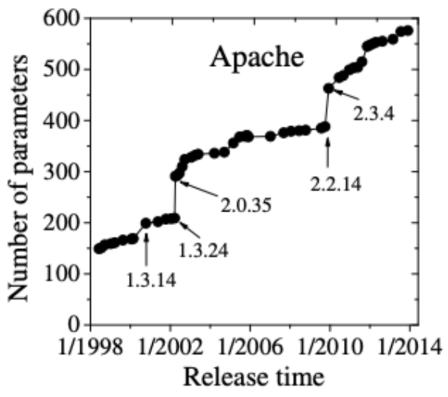
One thing that falls across all the above phases and activities is *choice*.

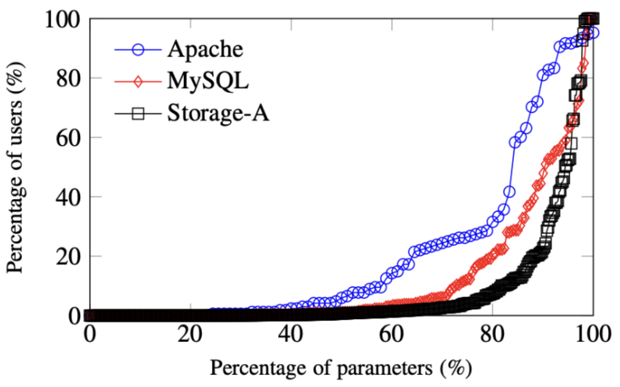
- requirements means helping stakeholders trade-off between choices;
- design means exploring and decising implementation choices;
- testing (all kinds) means choosing to test for this and not for that (since we cannot test for everthing).
- evalaution means choosing that A,B,C is more important than D,E,F,. . . .

4.1 So Many Choicesy

Choice is a **big issue** since there are so many choices and we usually do them so badly:

- Textbooks describing agile processes mentions over 128 options for managing agile projects. A model of these options has over a sextillion choices (2128=1038). Some of these options are inherently discriminatory against smaller organizations or organizations that include newcomers to the profession of programming.
- Theorem provers are AI tools for solving logical equations. State-of-the-art theorem provers use a wide range of optional tricks to solve different kinds of problems. For example, the cvc5 smt-lib solver comes with 400 configuration choices (of which 268 are labeled "experts only", see https://cvc5.github.io/docs/cvc5-1.0.8/options.html). Just think about that. Theorem provers are used to check safety critical systems– and the conclusions they generate are a quirk of what config settings we use.
- The number of control parameters of a software package grows linearly with time. Meanwhile, human understanding of those choices only ever grows sub-linearly [4].





And we really do not handle those choices very well.

- 30% of all cloud computing errors come from misconfigurations of cloud software [5].
- Even more alarming, 59% of the most severe performance bugs are caused by poor configuration– making bad choices one of the most dangerous threats to software quality [6].

[4] Tianyin Xu, Long Jin, Xuepeng Fan, Yuanyuan Zhou, Shankar Pasupathy, and Rukma Talwadker. 2015. Hey, you have given me too many knobs!: understanding and dealing with over-designed configuration in system software. In (ESEC/FSE 2015). 307-319. https://doi.org/10.1145/2786805.2786852

[5] Yuanyuan Zhou, Keynote address, IEEE Automated Software Engineering conference, San Diego, California, USA, 2019. https://2019.ase-conferences.org/info/keynotes#yuanyuan-yy-zhou-the-human-dimension-of-cloud-computing

[6] Xue Han and Tingting Yu. 2016. An Empirical Study on Performance Bugs for Highly Configurable Software Systems. ESEM 2016, 23:1-23:10. https://doi.org/10.1145/2961111.2962602

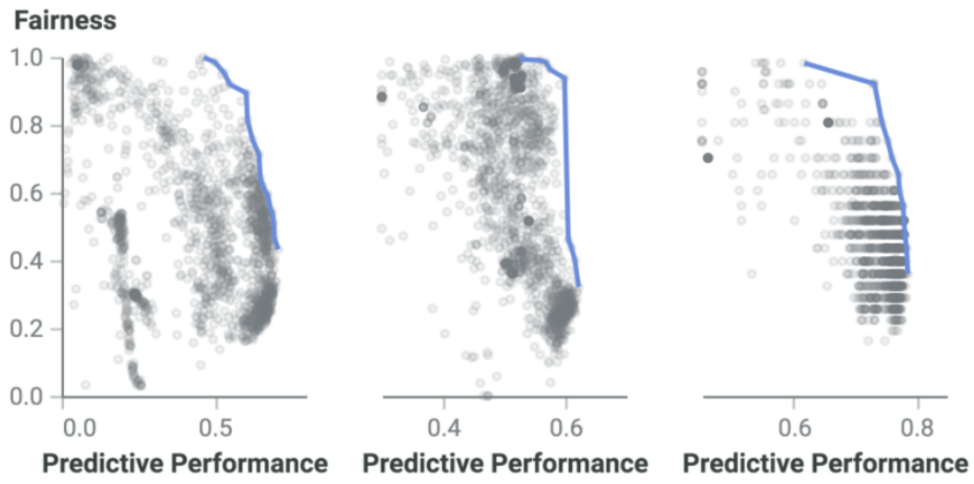
5 Unfair to make poor Choices

Poor choices can have a dramatic effect on (e.g.) fairness and predictive performance.

Here are 10,000 configuration choices for data miners about

- left: who gets a bank account,
- Middle: where to send grant money or
- Right: whether or not to give defendants bail or jail time [7].

- X-axis show accuracy and y-axis shows the ratio of false alarms between (left) men and women; (middle) rich schools and poor schools; whites and African American males. Note that the choices have a very wide range of effect.

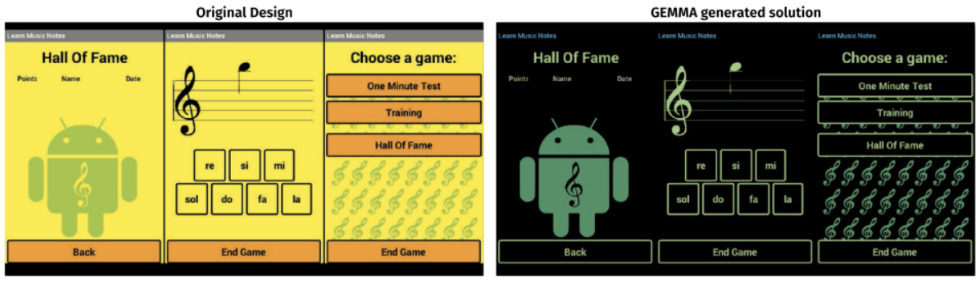


(Aside: the choices inside a learner are called the hyperparameters. The hyperparameters of (e.g.) Random Forests, learners include (a) how many T trees to build (e.g., $T \in \{10, 20, 40, 80, 160\}$); (b) how many features F to use in each tree (e.g., $F \in \{2, 4, 10, 20, sqrt, log2, all\}$); (c) how to poll the whole forest (e.g., majority or weighted majority); (d) what impurity measures (e.g., gini or entropy or log.loss); (e) what is the minimum examples needed to branch a sub-tree (e.g., $min \in \{2, 5, 10, 20, 50, 100\}$; (f) should branches be binary or n-arty. In all, this gives us $5 * 7 * 2 * 3 * 6 * 2 > 2,500$ different ways, just to configure a learner in the above figure.

[7] F.Cruz, P. Saleiro, C. Belem, C. Soares and P. Bizarro, “Promoting Fairness through Hyperparameter Optimization,” IEEE ICDM, 2021, pp. 1036-1041, doi: 10.1109/ICDM51629.2021.00119.

6 And now the good news

- It turns out that automatically making decisions about choices in software is a great unsung success story. AI tools are very successful at predicting how choices affect software [8].
- Those same AI tools, with small modifications can also tell us how to change those choices in order to improve runtimes, energy usage , fairness and performance [9].
- For example, automatic configuration tools can find better choices for the design of cell phone apps that uses far less energy [10]



- Outside of SE, it is standard to see automatic choose tools to self-configure their systems:
  - High performance computing environments come with their own tuning tools [11].
  - Also, in the 1990s, it was standard practice in the database community to offer tuning tools alongside of databases systems [12].

[8]Siegmund, N., Dorn, J., Weber, M., Kaltenecker, C., & Apel, S. (2022). Green configuration: Can artificial intelligence help reduce energy consumption of configurable software systems?. Computer, 55(3), 74-81.

[9]Jianmei Guo, Krzysztof Czarnecki, Sven Apel, Norbert Siegmund, Andrzej Wasowski: Variability-aware performance prediction: A statistical learning approach. ASE 2013: 301-311

[10]Mario Linares-Vasquez, Gabriele Bavota, Carlos Bernal-Cardenas, Massimiliano Di Penta, Rocco Oliveto, and Denys Poshyvanyk. 2018. Multi-Objective Optimization of Energy Consumption of GUIs in Android Apps. ACM Trans. Softw. Eng. Methodol. 27, 3, Article 14 (July 2018), 47 pages. https://doi.org/10.1145/3241742

[11]Golovin, D., Solnik, B., Moitra, S., Kochanski, G., Karro, J., & Sculley, D. (2017, August). Google vizier: A service for black-box optimization. KDD-17 (pp. 1487-1495).

[12]Surajit Chaudhuri and Vivek Narasayya. 2007. Self-tuning database systems: a decade of progress. In Proceedings of the 33rd international conference on Very large data bases (VLDB '07). VLDB Endowment, 3-14.

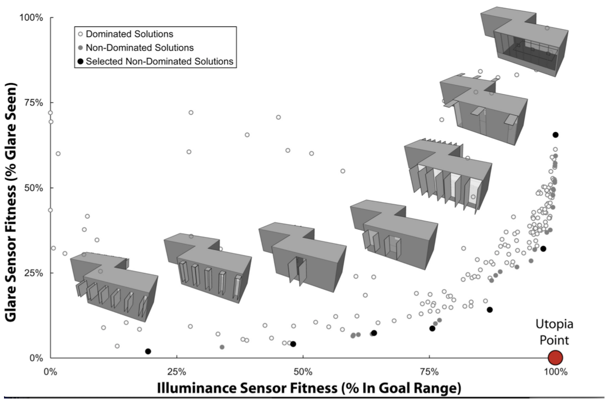
7 Challenges with Choice

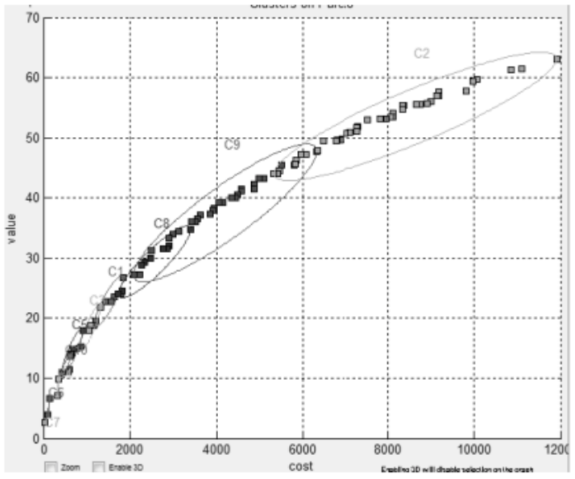
There is a need to integrate methods for automating choice into legislative frameworks.

- To date, regulation frameworks and certification standards ignore the fact that software is configurable.
- Certifying one configuration of a software (by making a set of choices) does not necessarily mean that other configurations (with other choices) shall also be certified or even applicable across other contexts.
- On the other hand, we cannot certify all configurations, as there are easily billions.
- More generally, this **certification problem** is an example of the **generalization** problem.

Another open challenge is the integration of human and artificial intelligence while making choices.

- There are different kinds of stakeholders so our tools must be tailored to their different needs.
- When choices conflict, subjective stakeholder opinion becomes important since those opinions let us make trade-offs across the decision space. e.g. of all the houses we could build, which do you prefer:





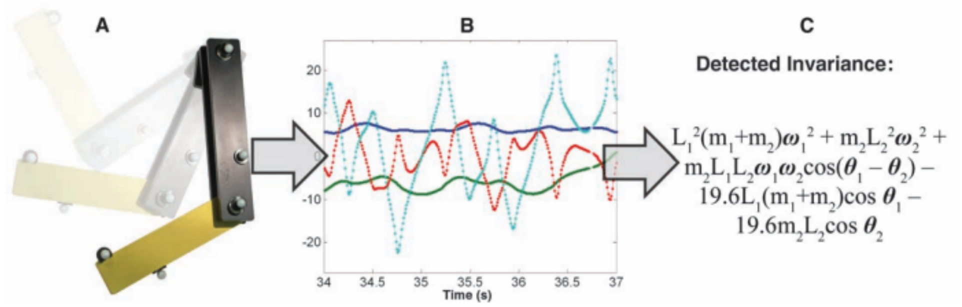
- Once an algorithm proposes some choices, it may still be up to stakeholders to decide which choices are inappropriate. For example, researchers exploring the requirements for London ambulances, found that the feasible solutions formed a couple of different clusters [13].

This tells us that there is not one solution, but several, and humans are needed to decide which solution they prefer. Many researchers work towards supporting this kind of human-in-the-loop decision making .

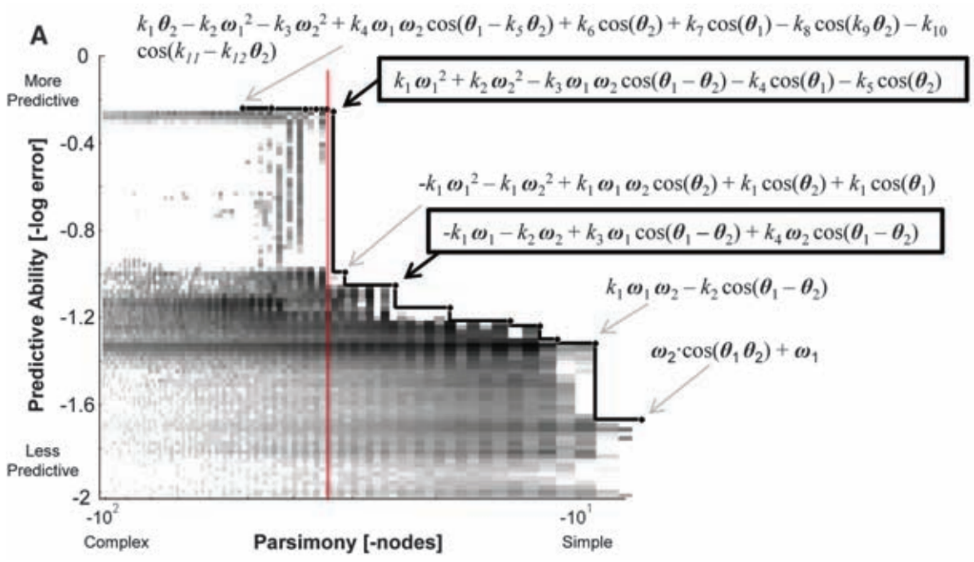
[13]V. Veerappa and E. Letier, “Understanding clusters of optimal solutions in multi-objective decision problems,” 2011 IEEE 19th International Requirements Engineering Conference, Trento, Italy, 2011, pp. 89-98, doi: 10.1109/RE.2011.6051654..

8 Too Many Choices?

But here's the biggest challenge: too many choices. Suppose you are trying to guess the equations descriping a double pendulum:


A
B
C
Detected Invariance:
$L_1^2(m_1+m_2)\omega_1^2 + m_2L_2^2\omega_2^2 + m_2L_1L_2\omega_1\omega_2\cos(\theta_1-\theta_2) - 19.6L_1(m_1+m_2)\cos\theta_1 - 19.6m_2L_2\cos\theta_2$
20
10
-10
-20
34
34.5
35
35.5
36
36.5
37
Time (s)

Here's a system called Eureqa [14] making those guesses. Eurequa makes guesses, prunes the worsts ones, then experimetns with the better ones. As shown here, there are very few best and so mnay worst.



Problems:

- **Verification** If a human wanted to check that Eureka dahn't missied anyrhing, do they have to check all these solutions?
- **Cogntive fatique** If Eureqa wants to check that its solutions are human-acceptabe, does a human ahve to check all these dots whenever they are generated?

By the time this subject is over, you'll ahve working answers to both these problems. You will be Kings and Queens of choice.

[14] Schmidt, Michael D. and Hod Lipson. "Distilling Free-Form Natural Laws from Experimental Data." Science 324 (2009): 81 - 85. https://faculty.washington.edu/morgansn/pmwiki/uploads/Site/schmidt-science2009.pdf