Open Source Software

Developing and Maintaining Sustainably

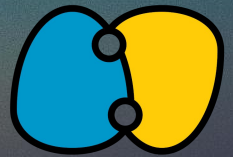
Everett Hildenbrandt

CEO, Runtime Verification

runtime verification

Software We Maintain

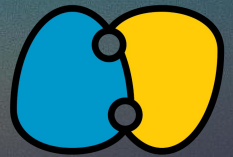
- 170 Public Repositories, 12-37 active
- 15+ different programming languages (C, C++, LLVM, Scala, Java, Rust, Python, JavaScript, Bash, Solidity, K, Haskell, MIR, WebAssembly, RISCV, ...)
- Dependency chains up to 6 links deep
- Teams from size 1 to size 10 (at different times)
- Active CI/CD automation



runtime verification

Challenge 1: Technical

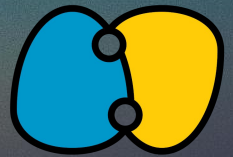
- Automation is key:
  - GitHub (or others) for enforcement
  - CI for testing
  - Automated releases/updates
- Software lifecycles: when to move from “rapid development” to “steady evolution”?



runtime verification

Challenge 2: Personnel

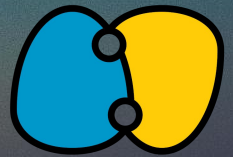
- “Learning to program is easy, learning to program in a group is hard.”
  - Must be respectful for others time.
  - Remote collaboration makes harder (esp. across timezones).
- Process is key:
  - Set expectations and stick to them.
  - Enforce when possible! (GitHub can do lots!)
  - Call people out (kindly) for unproductive/innefficient behavior.
  - Small delays add up!!
  - Ask for help.
- Onboarding is really hard: dedicate time to it.



runtime verification

Challenge 3: Money

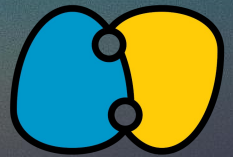
- ???????



runtime verification

Challenge 3: Money

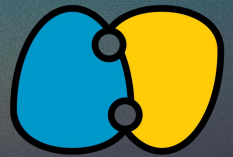
- Solution: Begging



runtime verification

Challenge 3: Money

- Solution: Begging
- Everyone is begging, then you need to introduce begging process.
- *Structured* Begging:
  - Grants
  - Retro Funding
  - Side Quests
- Becomes a *game*
  - Big players are better at games
  - Small player go back to unstructured begging



runtime verification