

# Guided discussion

## *AI for software development*

Paul and Duncan (and Claude)

# What AI told me about my job

- (some of) our challenges
  - A ***lot*** of legacy code (yes, also in the case of ET which owes a ***lot*** to LIGO and Virgo)
    - Thick software stacks with complex dependencies (documentation anyone?)
    - Scientifically excellent, often with excellent performance
    - Rarely implemented to precise design requirements
    - Effort not always aligned to collaboration development priorities
  - Decades long lifetime, high profile, and publicly funded
    - ET will start around 2040 and run for 50 years (data and software needed beyond that)
    - European governance requires detailed auditing
    - Design choices need to be justified over time, while (human) knowledge will be lost
    - Budgets can be maintained at best (no expectation of budget growth)
  - International collaborations with challenging governance models

# What AI thinks we should do - thanks AI !

- The lifetime of ET in particular implies a focus on constraint-driven development
  - We like solving problems, we are less keen on defining the requirements of the problem precisely
  - Constraints become a first-class artefact, embodying institutional knowledge of design choices and changes over the lifetime of the experiment
    - Expensive (in terms of tokens) but repeated rarely

You don't want *'an AI coding assistant.'*

You want: **A governed knowledge system with AI as a consumer**

***I think that means we want a managed collection of artefacts (our knowledge), and feed that to AI to generate and maintain our software***

# Discussion topic 1 - the design process

- Constraint-led design paradigm
  - *Einstein Telescope has a decades long lifetime*
    - We need the opposite of “move fast and break things” culture
  - Using AI to make bullet-proof design requirements is valuable
    - My experience was humbling!
    - This goes hand-in-hand with bullet-proof validating/auditing procedures
      - We value correctness over time-to-production
    - We care about quality and (human) maintainability - see Chad’s talk
  - Cost and risk management is crucial, we’re publicly funded
    - Flat budgets over a decades-long lifetime is a strong constraint! Up-front heavy design costs could make sense, but we need to demonstrate this to funding agencies

# Discussion topic 2 - the design teams

- What will software development teams look like?
  - Software engineers
  - Domain scientists with software skills - translators in well-constructed teams, and the heroes of the hero-model we too often use in science
  - Managers - sources of many requirements, policies, etc. Defines “correct”
  - Operations teams
- Or are these roles all replaced by Agents?
- AI is a multiplier
  - My hope: we need similar teams to today, but with AI doing heavy lifting of tedium
  - I think (!) we can defend a small team of software and computing experts, working with AI

# Discussion topic 3 - attribution

- Where does the AI start and the human input end? And do we care?
  - Attributing credit to software developers has been a long arduous journey
  - Considered very important for careers if we want to keep some expertise in the field, and we think (!) we will still need some experts (maybe fewer than today)
- Who owns the software? Could software credit be an incentive to industry?
  - Seen at CERN and elsewhere, the positive view that society has of awe-inspiring science can be very attractive for industries helping to “accelerate science”

***Black holes colliding certainly grabs the imagination!***

# Discussion topic 4 - Who hosts?

- Democratisation of access - experiment-specific LLMs, hosted by the Research Infrastructure as a solution?
  - How do we 'keep up' with latest industry solutions and best practice?
  - Related to the cost model and managing risk
- Related to software credit, publicly available LLMs that can generate all of our scientific software negates licenses
  - A concern beyond scientific collaborations