Week 5 - Project Management

- “Measurement is difficult in Software Engineering” - in theory you have to monitor all these aspects at the same time!
- “Cyclomatic Complexity” - how complex is a particular implementation? Maybe have another look at this for calculating V(G). Also what does V(G) stand for? Cyclomatic number
- Problem with white box complexity metrics - need to have everything written down and then commit to your implementation before you make any judgement. Maybe get a list of pros/cons?
- However, in a broader and more practical context when applied to entire systems or multi-procedure programs, P usually stands for the number of distinct procedural components or routines in the program that are being analyzed. This interpretation aligns with the scenario where the program might consist of multiple distinct procedures (functions, methods, etc.), each representing a separate part of the graph.
  Thus, for most applications, particularly when analyzing a single procedure, P is typically set to 1. If the entire system with multiple procedures is being analyzed, P would be the count of those procedures, reflecting the complexity added by having multiple distinct entry and exit points in the overall control flow of the system. This helps in understanding the overall complexity from an integration or system-level testing perspective, where each procedure might need to be tested individually and in combination with others.
- Black box - starting before we have any code
- “Storey Points (Size Estimation) how many Storey points per user story. Storey points can vary as it is relative! It is used for comparing different user stories. For some groups a Storey point may be 10 hours of work, and for others 30 minutes!
- “Planning Poker” - is it always a fibonacci sequence? No it doesn’t, you can do what you want! Maybe get a list of pros/cons. Also, 10 points is usually quite big, so maybe it should be split into smaller user stories. Wait until the end to reveal how many Storey points each person put for each individual user story. They then discuss their opinions when revealed - vocalise issues! The cycle repeats after discussions until agreement (MAX 3 TIMES) - they don’t all have to agree completely though, they just find a way to sort it out - maybe by taking an average! The scales is completely relative within teams.

Pros

Promotes Team Collaboration: Planning poker involves the entire team, promoting active participation and collaboration. This helps in building team cohesion and ensuring that everyone's perspective is considered.

Improves Estimation Accuracy: Since each team member provides an independent estimate based on their understanding, the variety of perspectives can lead to more accurate estimations compared to individual or dictated estimates.

Facilitates Knowledge Sharing: The discussions that follow different team members' estimates help spread knowledge about the task's requirements and potential challenges, as well as about estimation techniques themselves.

Detects Misunderstandings Early: The process can highlight different understandings of a task's scope and complexity, allowing for clarification and alignment early in the project lifecycle.
Engages and Empowers Team Members: By involving team members in the estimation process, it empowers them and enhances their commitment to the project, as they have a say in the planning process.

Cons

Time-Consuming: The process can be lengthy, especially for teams new to the technique or for complex tasks that require extensive discussion. This can slow down the planning phase.
Potential for Groupthink: Despite the independent voting mechanism, team dynamics might still lead to conformity, where members adjust their estimates to align with what they perceive to be the majority view, especially after several rounds of voting.
Challenging for Remote Teams: While digital tools can facilitate remote planning poker sessions, nuances in communication and participation can be more challenging to manage remotely compared to in-person sessions.
Requires Experience to Be Effective: Less experienced team members might struggle to provide accurate estimates, or they might feel pressured by more experienced colleagues, potentially skewing the results.
Not Suitable for All Projects: For very small or straightforward tasks, the planning poker process might be an overkill, consuming resources disproportionately to the benefit gained.

- Sprint duration is also team and sprint specific!
- Weakness in planning = bias towards seniority!!! Variable task duration depending on individual strengths!
- Patents/copyright etc - constraints in order to be legal. Check differences between patents and copyright!
- Check definitions within “Copyright Theft?”. Especially releasing things under licences.
- Why black box vs white box

Week 7 - HCI Evaluation Part One

- Quantitative = numerical
- Get a qualitative definition!
- Numbers are good (statistical analysis) but qualitative often paints a richer picture
- Quantitative = what, qualitative = why/how
- Think Aloud = easy implementation, but analysis afterwards can be hard!
- Heuristic = skiing - never eat yellow snow
- Heuristic evaluation - particularly useful for interface - user interacting with it
- 10 principles can be tailored towards target group! Maybe save a more detailed description of them all! Can one thing solve/address/fail multiple principles
- Best ratio = 4 to 5 evaluators = best balance between cost/identifying problems

Visibility of System Status
Principle: The system should always keep users informed about what is going on, through appropriate feedback within a reasonable time.
Details: This heuristic ensures that the system communicates what actions have been taken, changes in state, or errors that have occurred. For example, a loading spinner indicates that the system is processing a user request.
Match Between System and the Real World
Principle: The system should speak the users' language, with words, phrases, and concepts familiar to the user, rather than system-oriented terms.
Details: This involves using real-world conventions and making information appear in a natural and logical order. Icons, metaphors, and terms should be easily recognizable and understandable by the target user base.
User Control and Freedom
Principle: Users often perform actions by mistake. They need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue.
Details: This heuristic emphasizes features like undo and redo, and the ability to easily navigate back and forth without experiencing consequences or having to commit to an irreversible action.
Consistency and Standards
Principle: Users should not have to wonder whether different words, situations, or actions mean the same thing.
Details: This principle demands that the interface be consistent within itself and with industry standards. For example, icons and actions should not perform differently in different parts of the application.
Error Prevention
Principle: A careful design which prevents a problem from occurring in the first place is better than good error messages.
Details: This involves designing the system in such a way that potential errors are minimized by either eliminating error-prone conditions or checking for them and presenting users with a confirmation option before they commit to the action.
Recognition Rather Than Recall
Principle: Minimize the user's memory load by making objects, actions, and options visible.

Details: The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

Flexibility and Efficiency of Use

Principle: Accelerators — unseen by the novice user — may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users.

Details: Allow users to tailor frequent actions, provide shortcuts, and adapt the system to the user's pace and skill level.

Aesthetic and Minimalist Design

Principle: Dialogues should not contain information which is irrelevant or rarely needed.

Details: Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

Help Users Recognize, Diagnose, and Recover from Errors

Principle: Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

Details: The system should provide users with clear, understandable explanations of issues and effective ways to overcome them.

Help and Documentation

Principle: Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation.

Details: Any such information should be easy to search, focused on the user's task, list concrete steps to be followed, and not be too large.

Week 8 - HCI Evaluation Part Two

- Quantitative techniques - what
- Can include attention checking questions in a questionaire
- NASA TLX - very hard to measure or sense as an observer - measures perceived workload
- NASA TLX - what is the weighting? Will one have weighting of 0?
- NASA TLX - if they mark a space not a tick, take the tick to the right!
- NASA TLX - mostly nowadays people don't do the relative importance stuff, they just use no weighting, they just do step 2!
- NASA TLX - looking at individual scores can still be valuable and insightful
- SUS = SYSTEM USABILITY SCALE - LIKERT SCALES
- SUS - what if they mark a line?
- SUS - even numbered questions, disagree is more usability, odd is opposite!
- SUS - do not look at individual responses to questions - not meaningful
- If W is exactly equal to the sig Alpha level, then there is NO SIGNIFICANT DIFFERENCE
- Mann-Whitney U test compares values by two different groups. Read here: https://www.statology.org/mann-whitney-u-test/

Week 9 - Software Quality & Testing

- Financial loss, loss of TRUST e.g. Bard, reputation
- Cost of Product & Maintenance:
- Software certification is sometimes required before operation in industries e.g. airlines
- Organisational certification: proper set of practices and quality checks which ensure the end products will be of a certain standard
- Moral/ethical codes of practice - e.g. using data from charging points and other scenarios
- Some quality issues are trade offs against each other
- “Steps Towards Software Quality” - define bullet points!
- Software Development Process - can change and improve over time!
- Testing coverage: code executed during testing
- Branch - ALL PATHS MUST BE COVERED - all links and conditionals
- Black box - no access to source code, or if the system is so big the whole thing can’t be white box, maybe only use white box for really complex stuff!
- You want to have the minimum amount of test to cover all testing requirements