Week 5

Overview

- About Measurement

- Measurement under White Box:
  - Lines of code
  - Cyclomatic Complexity

- Measurement under Black Box:
  - Planning Poker

- Software Laws:
  - Patents, Copyright, Contract, Privac

What is “Measurement”?

- **Attributing values to objects. (为对象赋予价值)**
  - The fuel efficiency of a car (gallons per mile)
  - The number of goals scored by a footballer
  - The cost of a house

- **Can use these values as basis for comparison (可以使用这些值作为比较的基础)**
  - What is the cheapest house?
  - Who is the best goal scorer?

- **Can use these measurements and comparisons to *__make better decisions.__***

  (可以利用这些测量和比较来做出更好的决策)
  - Which car should I buy (e.g., given five candidate cars)
  - Which striker should I put in my team?

Usual Metrics: Size and Complexity 常用指标：大小和复杂性

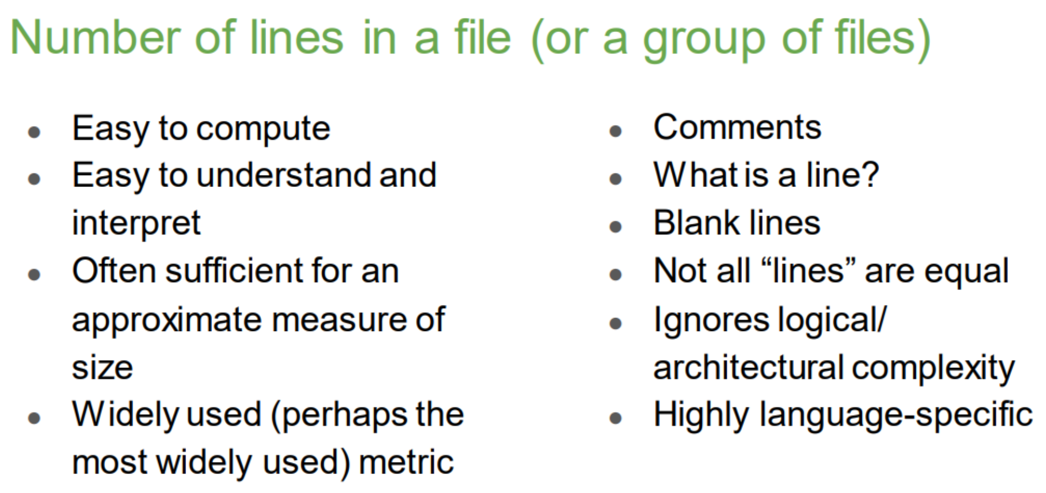
● **After development ...**

○ How much effort will it require for maintenance? ○ Where should we direct testing effort? ○ How much effort was required for development? ○ Metrics are based upon source code (“white box”)

● **Before development has started ...**

○ How much programming effort will module X require? ○ What will be the estimated cost of the final product? ○ Metrics are based upon requirements / specification (“black box”)

Number of lines in a file



Number of lines in a file (or a group of files)

- Easy to compute
- Easy to understand and interpret
- Often sufficient for an approximate measure of size
- Widely used (perhaps the most widely used) metric

- Comments
- What is a line?
- Blank lines
- Not all “lines” are equal
- Ignores logical/ architectural complexity
- Highly language-specific

Cyclomatic Complexity

● **Calculated from the control flow graph:**

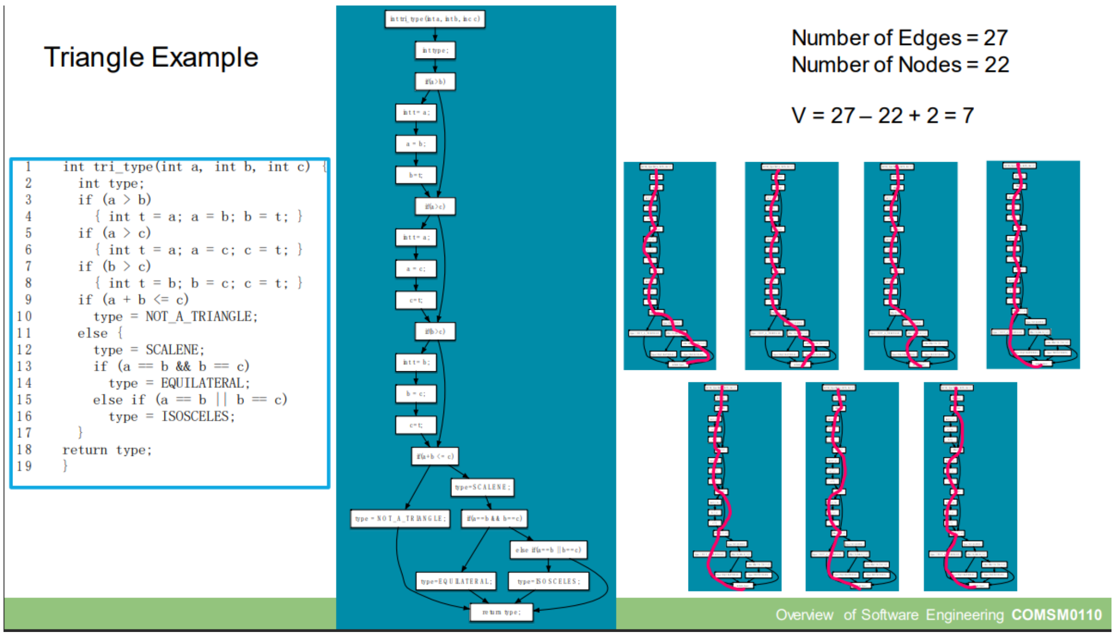
$$V(G) = E - N + 2P$$

Where E – number of edges; N – number of nodes; P – number of procedures (usually 1)

● Number of independent paths through the code

● Independent path – any path that introduces at least one new statement/condition

Example:



Triangle Example

```
int tri_type(int a, int b, int c) {
  int type;
  if (a > b)
    { int t = a; a = b; b = t; }
  if (a > c)
    { int t = a; a = c; c = t; }
  if (b > c)
    { int t = b; b = c; c = t; }
  if (a + b <= c)
    type = NOT_A_TRIANGLE;
  else {
    type = SCALENE;
    if (a == b && b == c)
      type = EQUILATERAL;
    else if (a == b || b == c)
      type = ISOSCELES;
  }
  return type;
}
```

Number of Edges = 27
Number of Nodes = 22

V = 27 – 22 + 2 = 7

Overview of Software Engineering **COMSM0110**

Black Box Complexity Merics

Storey Points (Size Estimation)

● **An informal, agile unit of “size measurement”**

○ Usually an estimate from 1-10

● **Derive an estimate from the whole team at sprint planning meetings**

● **Based on the idea of the “Wisdom of the Crowds”**

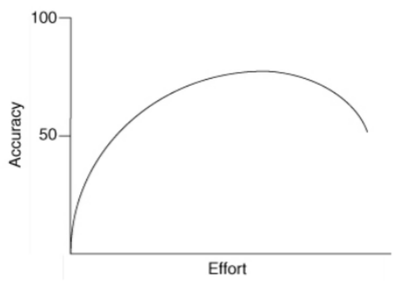
○ The collective estimate of groups (i.e., of effort required for a story) is better than the estimate of an individual

● 在冲刺计划会议上从整个团队中得出评估结果

● 基于“大众的智慧”的理念

○ 群体的集体估计（即一个故事所需的努力）优于个人的估计

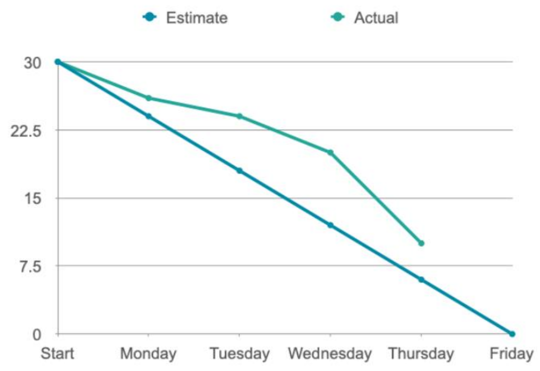
Accuracy vs Effort in Project Estimation



Team Velocity

- **Number of (estimated) story points implemented per sprint.**
- **Can be derived from previous sprints.**
  - e.g., Average points implemented from previous x sprints.
- **Can be used to estimate:**
  - Time required to complete project.
  - Target number of stories that can be completed in a sprint. (冲刺中可以完成的目标故事数量)

Burn Charts



Software Laws: Patents, Copyright, Contract, Privacy

Patent Law 专利

A government license giving a right for a set period, especially to exclude others from making, using, or selling an invention

- Granted by the government

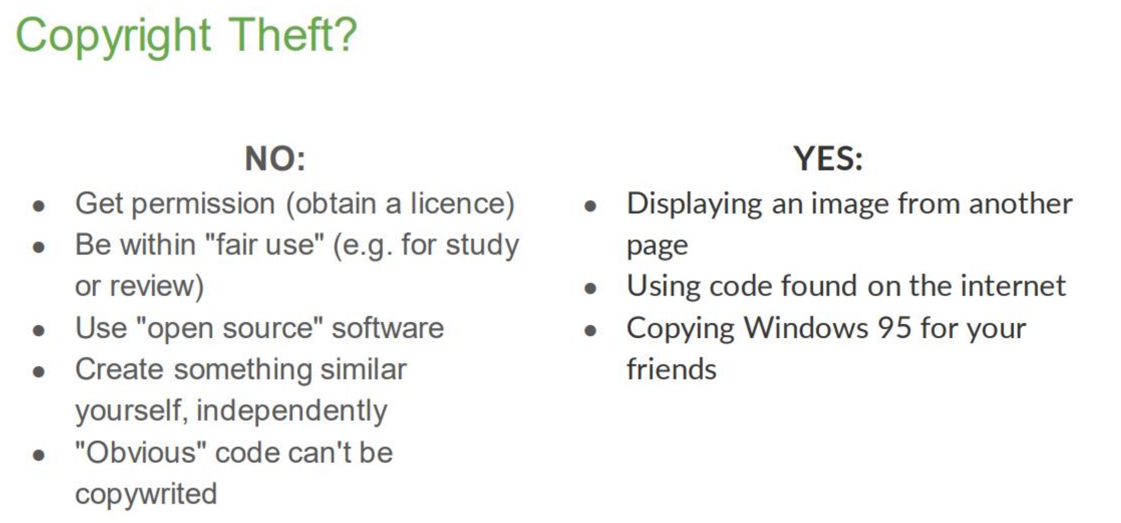
● to stop others exploiting <u>your</u> invention
● Lasts 20 Year

Inventions Must
● be new
● be an inventive step (not an obvious improvement)
● capable of industrial application

Copyright

● Creator has **exclusive rights** to perform, copy, adapt their work.
● Everyone else must get **Permission** (and possibly pay)
● "literary, dramatic, musical and artistic works" includes software
● Automatically owned (not granted)
● Lasts **70 years** after authors death (lots of exceptions)

This affects software in 2 different ways:
● Illegal Copies of Applications (Piracy) !
● Using someone else's code/UI design/etc. in your application (Not the "idea" but the actual "stuff" (code, design, documents) created by someone else)



Copyright Theft?

NO:
- Get permission (obtain a licence)
- Be within "fair use" (e.g. for study or review)
- Use "open source" software
- Create something similar yourself, independently
- "Obvious" code can't be copywrited

YES:
- Displaying an image from another page
- Using code found on the internet
- Copying Windows 95 for your friends

Contract Law 合同法

Employer contracts usually force an employee to:
● Not work for anyone else
● Hand over any ideas (Intellectual Property) 知识产权
● Not disclose company secrets (Non-disclosure-agreements) (even after you stop

working for them) 不泄露公司秘密（保密协议）（即使在您停止为他们工作之后）

Data Protection

UK : Data Protection Act
EU : Data Protection Directive
US : a "patchwork" of state and national laws

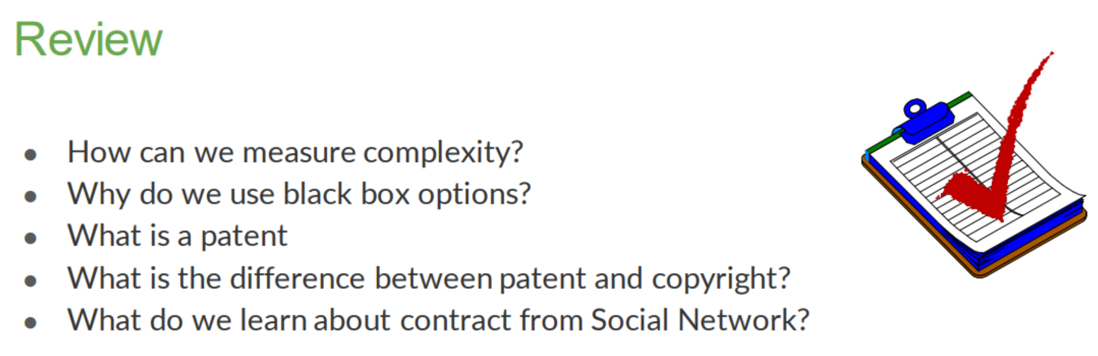
8 Principles of Data Protection:

Any company storing "personal data" must make sure it is:

- fairly and lawfully processed (consent, contractual and legal obligations, public interest, ...)
- processed for limited purposes;
- adequate, relevant and not excessive;
- accurate and, where necessary, kept up to date;
- not kept longer than necessary;
- processed in accordance with the data subject's rights;
- secure;

not transferred to countries without adequate protection

- 公平合法地处理（同意、合同和法律义务、公共利益……）
- 出于有限目的进行处理；
- 充分、相关且不过分；
- 准确，并在必要时保持最新；
- 保存时间不得超过必要的时间；
- 根据数据主体的权利进行处理；
- 安全；



Review

- How can we measure complexity?
- Why do we use black box options?
- What is a patent
- What is the difference between patent and copyright?
- What do we learn about contract from Social Network?

Measuring Complexity: Look at how much time and space a task needs as it gets bigger, or count the parts and connections in a system.
Black Box Options: We use them because they work without needing to understand everything inside, making it easier to predict outcomes.
Patent: It's a legal right that lets an inventor be the only one to use or sell their invention for a certain time.

Patent vs. Copyright: Patents are for inventions, copyrights are for creative works like books or music. Patents cover the function; copyrights cover the expression.
Contracts from 'The Social Network': The movie shows it's crucial to have clear written agreements in business to prevent misunderstandings.
衡量复杂性：查看任务变大时需要多少时间和空间，或者计算系统中的部件和连接。
黑匣子选项：我们使用它们是因为它们无需了解内部的所有内容即可工作，从而更容易预测结果。
专利：这是一项法律权利，允许发明人成为在一定时间内唯一使用或出售其发明的人。
专利与版权：专利适用于发明，版权适用于书籍或音乐等创造性作品。 专利覆盖该功能； 版权涵盖该表达。
《社交网络》中的合同：这部电影表明，在商业中签订明确的书面协议对于防止误解至关重要。

Week 7

Overview

● What is HCI evaluation?
● Why is it important
● The Think Aloud evaluation technique
● Heuristic evaluation

HCI Evaluation

● Evaluation is a crucial part of the user-centred development process – we want to ensure our software meets our users' requirements
● The focus of this lecture is on Think Aloud technique and Heuristic Evaluation, which are two of the most widely used evaluation methods in industry
● They are methods that we recommend you carry out on your game as part of your group project – you can write up the results in your report

Why is evaluation important?

"Iterative design, with its repeating cycle of design and testing, is the only validated methodology in existence that will consistently produce successful results. If you don't have user-testing as an integral part of your design process you are going to throw buckets of money down the drain."
"迭代设计及其重复的设计和测试周期是现有的唯一经过验证的、能够持续产生成功结果的方法。 如果你没有将用户测试作为设计过程中不可或缺的一部分，那么你就会浪费大量资金。"

The Think Aloud evaluation technique

● Users are asked to verbalise(言语表达) what they are thinking and doing as they perform a task using your software
● The Think Aloud technique provides insights into the user experience of using your software (深入了解软件的用户体验)
● It can identify issues with the software e.g. navigation problems or content that can be

improved
● It can be used as part of the software development process to iteratively improve software or used with a finished product

Benefits of Think Aloud

● Cheap
● Relatively easy
● It provides insight into people's experiences as they interact with your product
● It can be carried out with low numbers of participants
● Fits in with most software development processes

Drawbacks of Think Aloud (Disadvantage)

• it relies on people verbalising thoughts and impressions, rather than objective measures (它依赖于人们表达的想法和印象，而不是客观的衡量标准)
• Participants may say what they believe to be the right answer rather than what they really think (social desirability). This can distort your results and conclusions. (参与者可能会说出他们认为正确的答案，而不是他们真正的想法（社会期望）。 这可能会扭曲您的结果和结论)

Planning a Think Aloud evaluation

● Decide what questions you want your study to answer. For example, whether users can find particular content or what their understanding is of the information presented.
● Write down the tasks you want the user to complete while using your software
● Decide how many participants you want to recruit and how long you want the sessions to last (45 to 90 minutes works well)
● 决定你希望你的研究回答什么问题。 例如，用户是否可以找到特定内容或者他们对所呈现的信息的理解如何。
● 写下您希望用户在使用您的软件时完成的任务
● 决定您想要招募多少参与者以及您希望会议持续多长时间（45 到 90 分钟效果最好）

Carrying out a Think Aloud evaluation 1

● Have a facilitator（协调员） to run the evaluation and one or two observers to take notes on what the user says
● Explain to the participants how a think aloud works: they should tell you their thoughts, reactions and emotions as they occur while they are performing the task
● Explain that there is no right answer and it's fine to be critical
● Ask the participants to complete the tasks you have planned. This should be uninterrupted(不间断的) as far as possible, although the facilitator will probably need

to give some prompts.
● If the user goes silent then prompt them to verbalise their thoughts by saying “what are you thinking”
● 由一名协调员进行评估，并由一到两名观察员记录用户所说的内容
● 向参与者解释出声思考的运作方式：他们应该告诉你他们在执行任务时发生的想法、反应和情绪
● 解释说没有正确答案，批评也可以
● 要求参与者完成您计划的任务。尽管协调员可能需要给出一些提示，但这应该尽可能不间断。
● 如果用户沉默，则提示他们说出自己的想法，比如“你在想什么”

Analyzing a Think Aloud evaluation

● Put the written notes together from both observes in to one document
● Organize the notes into meaningful categories e.g. what features helped users; what features led to problems; any additional features that users wanted.
● You can make your own meaningful categories(有意义的类别)
● Count the number of times users comment about different categories to identify the biggest issues
● 将两次观察的书面笔记合并到一份文档中
● 将笔记组织成有意义的类别，例如：哪些功能对用户有帮助；哪些功能导致了问题；用户想要的任何附加功能。
● 你可以创建自己有意义的类别（有意义的类别）
● 统计用户对不同类别的评论次数，以确定最大的问题

Heuristic evaluation

What is a heuristic?

● A rule of thumb
● Experienced-based strategies
● E.g. if you’re doing some DIY then ‘measure twice, cut once’ is a useful heuristic

Heuristic evaluation

● An evaluation technique conducted **without** users
● Also known as **expert** evaluation as it’s sometimes carried out by external experts (sometimes by the development team) aka evaluators
● It’s a type of **analytical** evaluation, that is, based on a set of principles or a model…
● …rather than by observing users (which is known as **empirical** evaluation)

● It’s an **inspection** method – it involves inspecting a design to find usability problems
● This involves asking whether the design complies with **usability principles** (a set of heuristics)
- 无需用户进行的评估技术
- 也称为专家评估，因为它有时由外部专家（有时由开发团队）又称评估员进行
- 这是一种分析评估，即基于一组原则或模型……
- ……而不是通过观察用户（称为经验评估）
- 这是一种检查方法——它涉及检查设计以发现可用性问题
- 这涉及询问设计是否符合可用性原则（一组启发式）

Heuristic evaluation is widely used because…(advantage)

● It’s **cheap** (only needs a small number of evaluators and no specialist equipment or labs)
● Relatively **easy to carry out** (can do it after a few hours of training)
● **Instant gratification** – lists of problems are **available immediately** after the inspection (即时满足——检查后立即提供问题清单)
● It **fits in** with most software development processes used in industry
● It’s a very **cost effective**: benefit-cost ratio of 48: cost of $10,500; expected benefits $500,000 (Nielsen 1994).

Drawbacks of heuristic evaluation (disadvantage)

● Important issues may get missed
● Might identify false issues
● Many trivial issues are often identified, making it seem overly critical
● Experts have biases
- 重要问题可能会被遗漏
- 可能会发现错误的问题
- 经常发现许多琐碎的问题，使其显得过于重要
- 专家有偏见

Where are the users?

● Heuristic evaluation is based on HCI researchers’ extensive experience of designing and evaluating interfaces (启发式评估基于人机交互研究人员设计和评估界面的丰富经验)
● By focusing on users, HCI researchers learned what works and what doesn’t (通过关注用户，HCI 研究人员了解到什么有效、什么无效)
● Their experience is distilled into **usability principles** (a set of heuristics) (他们的经验

被提炼成可用性原则（一组启发法））
- The principles represent the findings from thousands of user studies
- They have been used for over 30 years

Nielsen's 10 principles of heuristic evaluation

- visibility of system status
- match between system and real world
- user control and freedom
- consistency and standards
- error prevention
- recognition rather than recall
- flexibility and efficiency of use
- aesthetic and minimalist design
- help users recognise, diagnose and recover from errors
- help and documentation

- 系统状态的可见性 • 系统与现实世界的匹配
- 用户控制和自由 • 一致性和标准
- 预防错误 • 识别而不是回忆
- 使用的灵活性和效率 • 美观和简约的设计
- 帮助用户识别、诊断错误并从错误中恢复
- 帮助和文档

Nielsen's 10 principles of heuristic evaluation (minimal information)

- feedback
- metaphor
- user control and freedom
- consistency
- error prevention
- recognition not recall
- flexible use
- minimal information
- error recognition and recovery
- help

- 反馈
- 隐喻
- 用户控制和自由
- 一致性
- 错误预防
- 识别而非回忆
- 灵活使用
- 最少的信息
- 错误识别和恢复
- 帮助

Visibility of system status – feedback （不能让用户干等，需要告知到是否成功）

- Inform the user about what's going on:
  - show appropriate feedback and progress
  - do not show blank screens
  - do not show static "load" or progress messages

Match between system and real world – metaphor （需要符合现实世界逻辑）

- There must be a match between the system's interface controls and the real world
- The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms

● Follow real-world conventions, making information appear in a natural and logical order

User control and freedom – navigation (需告知用户现在所在的层级)

● Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialog.
● Support undo and redo and a clear way to navigate.
● Provide breadcrumbs to clearly show where the user is
● 用户经常会错误地选择系统功能，并且需要一个明确标记的"紧急出口"来离开不需要的状态，而不必通过扩展的对话框。
● 支持撤消和重做以及清晰的导航方式。
● 提供面包屑以清楚地显示用户所在位置

Consistency and standards

● Users should not have to wonder whether different words, situations, or actions mean the same thing.
● Follow platform conventions.
● 用户不必怀疑不同的词语、情况或动作是否意味着同一件事。
● 遵循平台惯例。

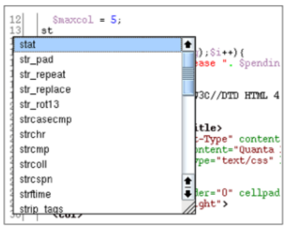
Error prevention

• Even better than good error messages is a careful design which prevents a problem from occurring in the first place.
• Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.
• 比良好的错误消息更好的是精心设计，可以从一开始就防止问题发生。
• 消除容易出错的情况，或者检查这些情况并向用户提供确认选项，然后再执行操作。

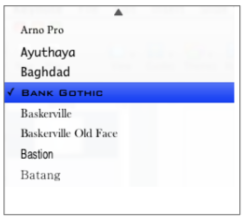
Recognition rather than recall

● Minimize the user's memory load.
● Make objects, actions, and options visible.
● 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.
● 最大限度地减少用户的记忆负担。
● 使对象、操作和选项可见。
● 用户不必记住从对话的一个部分到另一部分的信息。
● 系统的使用说明应该是可见的或在适当的时候易于检索。

Recognition: examples



Quanta IDE
Auto completion for coding in a development environment

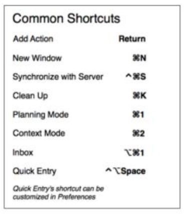


Keynote
Previews the fonts you can pick from, instead of just the font name

Flexibility and efficiency of use (如快捷键，自动补全，内置函数…)

● Accelerators — unseen by the novice user — may often speed up the interaction for the expert user so that the system can cater to both inexperienced and experienced users
● Allow users to tailor frequent actions

Flexibility and efficiency: examples

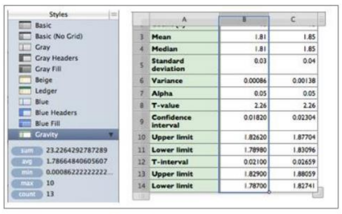


Common Shortcuts

Add Action	Return
New Window	⌘N
Synchronize with Server	^⌘S
Clean Up	⌘K
Planning Mode	⌘1
Context Mode	⌘2
Inbox	⌥⌘1
Quick Entry	^⌥Space

Quick Entry's shortcut can be customized in Preferences

OmniFocus
List of keyboard shortcuts and accelerators



sum	23.2264292787289
avg	1.78664840605607
min	0.00086222222222…
max	10
count	13

	A	B	C
3	Mean	1.81	1.85
4	Median	1.81	1.85
5	Standard deviation	0.03	0.04
6	Variance	0.00086	0.00138
7	Alpha	0.05	0.05
8	T-value	2.26	2.26
9	Confidence interval	0.01820	0.02304
10	Upper limit	1.82620	1.87704
11	Lower limit	1.78980	1.83096
12	T-interval	0.02100	0.02659
13	Upper limit	1.82900	1.88059
14	Lower limit	1.78700	1.82741

Numbers by Apple
Previews common function results on the left when a column is selected, more efficient than clicking on an action in the toolbar

Aesthetic and minimalist design

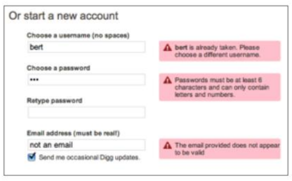
● Dialogues should not contain information which is irrelevant or rarely needed
● Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility
● Visual layout should respect the principles of contrast, repetition, alignment, and proximity.

● 对话不应包含不相关或很少需要的信息
● 对话中每一个额外的信息单元都会与相关信息单元竞争并降低它们的相对可见性
● 视觉布局应遵循对比、重复、对齐和接近的原则。

Help users recognize, diagnose and recover from errors (错误信息)

● Help users recognize, diagnose, and recover from errors.
● Error messages should be expressed in plain language (no jargon), precisely indicate the problem, and constructively suggest a solution.

Error recognition and recovery: examples



Or start a new account

Choose a username (no spaces): bert
⚠ bert is already taken. Please choose a different username.

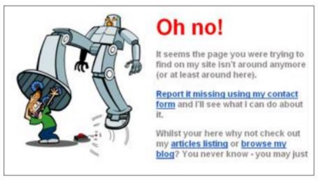
Choose a password: •••
⚠ Passwords must be at least 6 characters and can only contain letters and numbers.

Retype password

Email address (must be real!): not an email
⚠ The email provided does not appear to be valid

[✓] Send me occasional Digg updates.

Digg
Provides immediate feedback with specific instructions



Oh no!

It seems the page you were trying to find on my site isn't around anymore (or at least around here).

Report it missing using my contact form and I'll see what I can do about it.

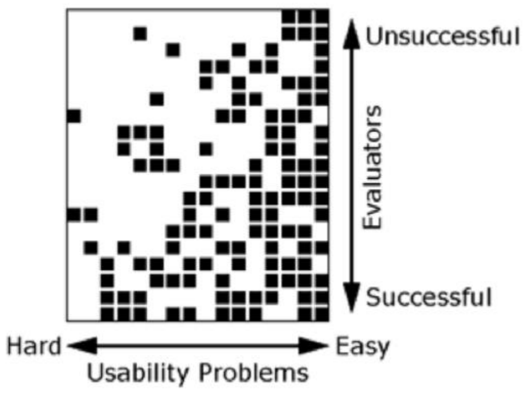
Whilst your here why not check out my articles listing or browse my blog? You never know - you may just

Humorous ' Page Not Found' Error
Uses a funny image and text, but provides viable alternatives (article listings and blog link) and a course of action (report it)

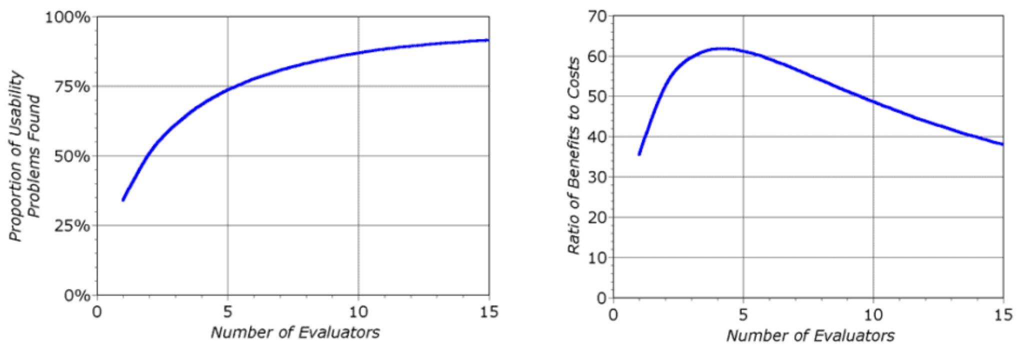
Help and documentation

● Even though it is better if software can be used without documentation, it may be necessary to provide help and documentation.
● Any such information should be contextual, easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

How many evaluators are needed for heuristic evaluation?



Practical considerations



Proportion of usability problems found

How to run a heuristic evaluation

● Each of the 3 – 5 evaluators does a heuristic evaluation of an interface alone
● Sometimes a facilitator can record the evaluator's comments, sometimes the evaluator does it
● A facilitator can answer evaluators' questions, in contrast to traditional user testing, particularly if it's not a walk up and use system
● Heuristic evaluation can be done on **paper prototypes**
● Heuristic evaluations typically last 1 – 2 hours, but it does depend on the complexity of the software
● The expert goes through the interface several times – first time to get a feel for the system, second time to focus on specific elements
● Evaluators can be given scenarios that describe typical usage scenarios (built from a task analysis of users)
● Evaluators produce a list of usability problems: the usability principle and the design feature that violated it
● 3 – 5 个评估者中的每一个都单独对一个界面进行启发式评估
● 有时协调员可以记录评估者的评论，有时评估者会这样做
● 与传统的用户测试相比，协调员可以回答评估者的问题，特别是当它不是直接使用系统时
● 可以在纸质原型上进行启发式评估
● 启发式评估通常持续 1 – 2 小时，但这取决于软件的复杂性
● 专家多次浏览界面 – 第一次是为了感受系统，第二次是为了关注特定元素
● 可以为评估者提供描述典型使用场景的场景（根据用户的任务分析构建）
● 评估人员列出可用性问题清单：可用性原则和违反该原则的设计特征

Week 8 HCI Evaluation part 2

Overview

● Questionnaires
● NASA Task Load Index (NASA TLX)
● System Usability Scale (SUS)
● Statistical tests to determine if the perceived workload or system usabiity score has changed significantly

Questionnaires - defined

● Questionnaires involve asking people to answer questions either on paper or digitally e.g. on a webpage or app
● They can be used at scale with low resource requirements
● They generate a collection of demographic data and user opinions
● They can be used to evaluate designs and for understanding user requirements
● 问卷调查要求人们以纸质或数字方式回答问题，例如：在网页或应用程序上
● 它们可以大规模使用，资源需求低
● 他们生成人口统计数据和用户意见的集合
● 它们可用于评估设计和了解用户需求

Two widely used questionnaires - NASA TLX

● **The NASA Task Load Index** (TLX) is a questionnaire that estimates a user's perceived workload when using a system.
● NASA 任务负载指数（TLX）是一项调查问卷，用于估计用户在使用系统时感知到的工作负载。

● Workload is a complex construct but essentially means the amount of effort people have to exert, both mentally and physically, to use a system.
● 工作负载是一个复杂的结构，但本质上是指人们为了使用系统而必须付出的精神和体力上的努力量。

● It was developed by Sandra Hart of NASA's human performance group and Lowell Staveland of San Jose University.

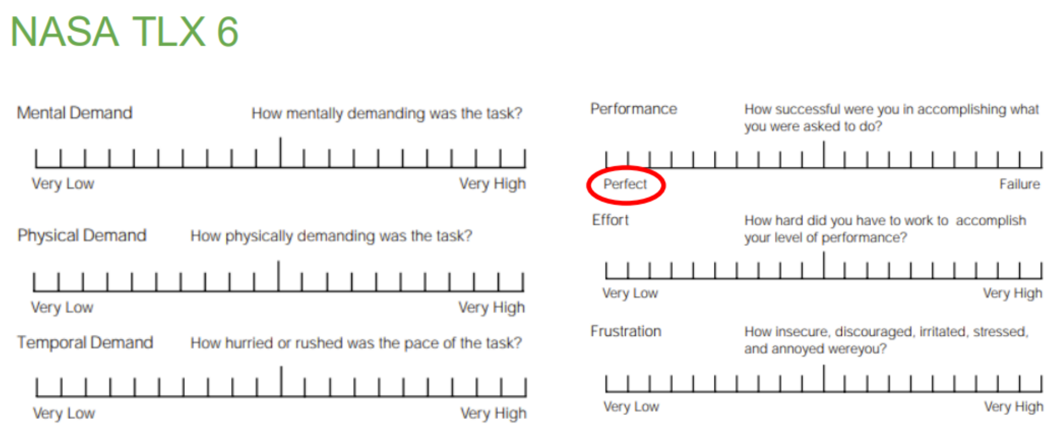
● The focus is on measuring the "immediate often unverbalized impressions that occur spontaneously" (Hart and Staveland, 1988). These are difficult or impossible to observe objectively.
● 重点是衡量"自发发生的直接的、通常是非语言的印象"（Hart 和 Staveland，1988）。 这些是很难或不可能客观观察的。

● Originally the NASA TLX questionnaire was developed for use in aviation but it's since been used in many different domains, including air traffic control, robotics, the automative industry, healthcare, website design and other technology fields.
● NASA TLX 问卷最初是为航空应用而开发的，但后来被用于许多不同的领域，包括空中交通管制、机器人、汽车工业、医疗保健、网站设计和其他技术领域。
● Since it was introduced in 1988, it has had over 8000 citations.
● It is viewed as the **gold standard** for measuring subjective workload.

● The NASA TLX uses a multi-dimensional rating procedure that derives an overall workload score based on a weighted average of ratings on six subscales:

- Mental Demand	- 精神需求
- Physical Demand	- 实物需求
- Temporal Demand	- 时间需求
- Performance	
- Effort	
- Frustration	- 沮丧

● Mental demand – how much mental and perceptual activity was required?
● Physical demand – how much physical activity was required?
● Temporal demand – how much time pressure did the user feel due to the rate at which tasks occurred?
● Frustration – how insecure, discouraged or irritated did the user feel in the task?
● Effort – how hard did the user have to work (mentally and physically) to accomplish their level of performance?
● Performance – how successfully did the user think they accomplished the task?
● 脑力需求——需要多少脑力和知觉活动?
● 体力需求——需要多少体力活动?
● 时间需求——由于任务发生的速度，用户感受到了多少时间压力?
● 挫败感——用户在任务中感到多么不安全、沮丧或恼怒?
● 努力程度——用户需要付出多大的努力（精神上和体力上）才能达到他们的绩效水平?
● 性能——用户认为他们完成任务的成功程度如何?


NASA TLX 6

Mental Demand — How mentally demanding was the task?
Very Low — Very High

Physical Demand — How physically demanding was the task?
Very Low — Very High

Temporal Demand — How hurried or rushed was the pace of the task?
Very Low — Very High

Performance — How successful were you in accomplishing what you were asked to do?
Perfect — Failure

Effort — How hard did you have to work to accomplish your level of performance?
Very Low — Very High

Frustration — How insecure, discouraged, irritated, stressed, and annoyed wereyou?
Very Low — Very High

NASA TLX Scoring 1

● Users answer the NASA TLX after they have completed a task. This is necessary as asking them to complete it during task is typically not possible. However, it may mean that users forget details of the perceived workload.

● The questionnaire is scored in a two step process:

1. Identifying the relative importance of the 6 dimensions on a user's perceived workload
2. Rating each of the 6 dimensions on a scale

● 用户完成任务后回答 NASA TLX。 这是必要的，因为要求他们在任务期间完成它通常是不可能的。 然而，这可能意味着用户忘记了感知到的工作负载的详细信息。

● 调查问卷分两步进行评分：

1. 确定 6 个维度对用户感知工作负载的相对重要性
2. 在量表上对 6 个维度中的每一个维度进行评级

NASA TLX Weighting Summary:

- After a task, users make 15 paired comparisons of 6 workload dimensions.
- Each chosen dimension earns 1 point, with a max of 5 points per dimension.
- Total dimension scores will sum up to 15, indicating their workload impact.
- The raw TLX score is used when weighting is skipped for simplicity.
- Studies on weighting's effectiveness are inconclusive, varying by context.

完成一项任务后，用户对 6 个工作负载维度进行 15 次配对比较。

每个选择的维度可获得 1 分，每个维度最多可获得 5 分。

维度总分总计为 15，表明其对工作负载的影响。

为简单起见，跳过加权时使用原始 TLX 分数。

关于加权有效性的研究尚无定论，因情况而异。

NASA TLX Dimension Rating:

- Users rate each dimension on a scale from 0 to 100, marked on a line with 21 ticks.
- Ratings are in 5-point increments; marking between ticks defaults to the higher value.
- Scores are calculated by (tick position - 1) * 5, with the fifth tick equating to a score of 20.
- This method is consistent across different formats, like paper or mobile apps.

用户对每个维度进行评分，评分范围为 0 到 100，在一条线上标有 21 个刻度。

评级以 5 分为增量； 刻度之间的标记默认为较高值。

分数的计算方式为（勾号位置 - 1）* 5，第五个勾号相当于 20 分。

此方法在不同格式（例如纸质或移动应用程序）中是一致的。

NASA TLX What do the scores tell us?

● If the weights are used then the individual ratings on each of the dimensions are multiplied by their respective weights, summed and divided by 15, resulting in an aggregate perceived workload score for a task ranging from 0 – 100.
● If the weights are not used then the individual ratings on each of the dimensions can be summed and divided by 6, resulting in an aggregate perceived workload score ranging from 0 – 100.
● The individual ratings on the 6 dimensions also give some insight in to where the workload is coming from. This can be helpful for developers hoping to improve their design.
● 如果使用权重，则将每个维度的单独评分乘以各自的权重，求和并除以 15，得出任务的感知工作负载总分，范围为 0 – 100。
● 如果不使用权重，则可以将每个维度的单独评分相加并除以 6，从而得出 0 – 100 之间的总体感知工作负载分数。
● 6 个维度的单独评级还可以让您了解工作负载的来源。 这对于希望改进设计的开发人员很有帮助。

Two widely used questionnaires - System Usability Survey (SUS)

● The System Usability Scale (SUS) provides a “quick and dirty”, reliable tool for measuring usability.
● It was created by **John Brooke in 1986.**
● It consists of a 10 item questionnaire with five response options for each item ranging from Strongly agree to Strongly disagree.
● It enables the evaluation of a wide variety of products and services, including hardware, software, mobile devices, websites and applications
● 系统可用性量表(SUS) 提供了一种“快速而简单”的可靠工具来衡量可用性。
● 由 John Brooke 于 1986 年创建。
● 它由 10 个项目的调查问卷组成，每个项目有五个回答选项，范围从“非常同意”到“非常不同意”。
● 它可以评估各种产品和服务，包括硬件、软件、移动设备、网站和应用程序

System Usability Survey (SUS) - **benefits**

● SUS has become an industry standard, with references in over 1300 articles and publications.
● The noted benefits of using SUS include:
    ● It is a very easy scale to administer to participants
    ● It can be used on small sample sizes with reliable results
    ● The SUS has been validated and shown to effectively differentiate between usable and unusable systems
● 这是一个非常容易对参与者进行管理的量表
● 可用于小样本量，结果可靠
● SUS 已经过验证并显示可以有效地区分可用和不可用的系统

System Usability Survey (SUS) - scale

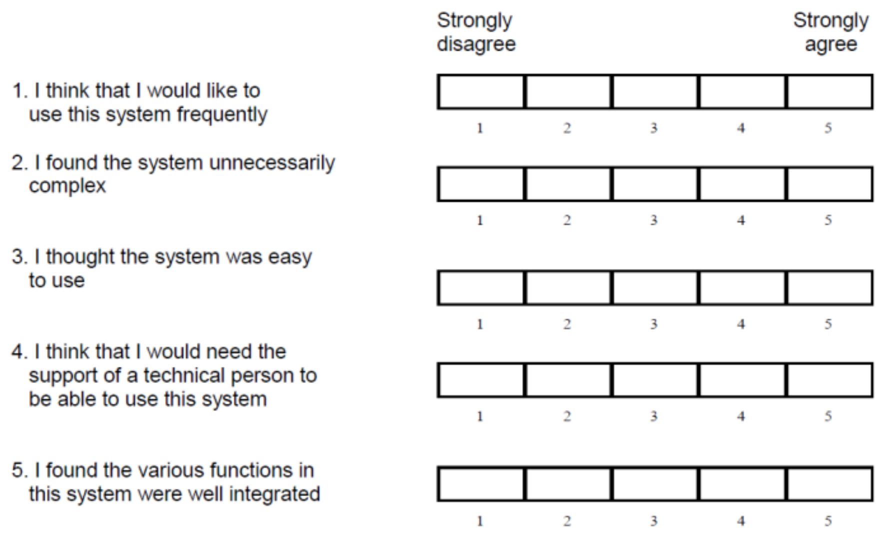
● When an SUS is used, participants are asked to score the 10 items with one of five responses that range from Strongly Agree to Strongly disagree i.e. using a five point Likert scale

当使用 SUS 时，参与者被要求对 10 个项目进行评分，并选择从“非常同意”到“非常不同意”的五种回答之一，即使用五点李克特量表

10 items(question)

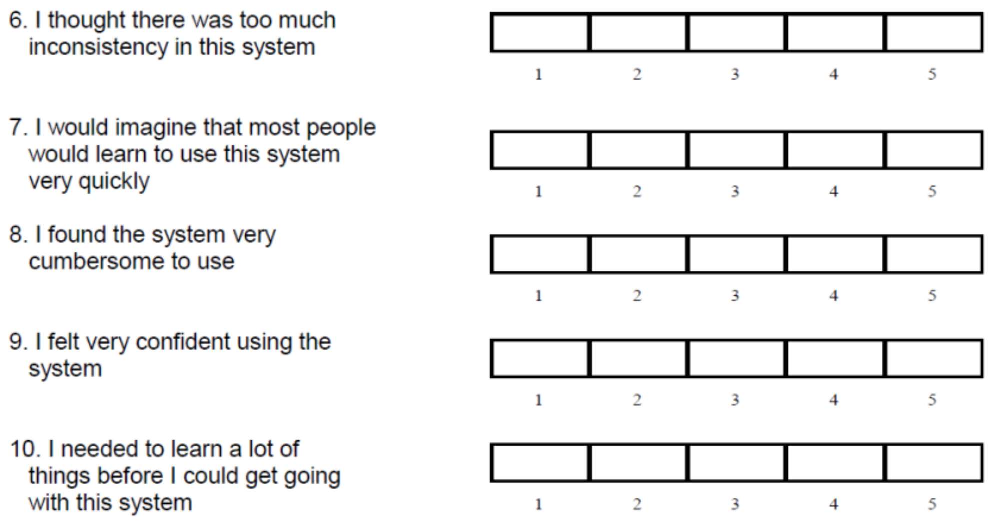
1. I think that I would like to use this system frequently
2.I found the system unnecessarily complex
3. I thought the system was easy to use
4. I think that I would need the support of a technical person to be able to use this system
5.I found the various functions in this system were well integrated
6.I thought there was too much inconsistency in this system
7.1 would imagine that most people would learn to use this system very quickly
8.I found the system very cumbersome to use
9. I felt very confident using the system
10.I needed to learn a lot of things before I could get going with this system
1. 我想我会经常使用这个系统
2.我发现系统不必要地复杂
3.我认为该系统易于使用
4.我认为我需要技术人员的支持才能使用这个系统
5.我发现这个系统的各种功能集成得很好
6.我认为这个系统有太多不一致的地方
7. 我想大多数人会很快学会使用这个系统
8.我发现系统使用起来很麻烦
9. 我对使用该系统感到非常有信心
10.在开始使用这个系统之前，我需要学习很多东西

System Usability Survey (SUS) – scale 2



	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	1	2	3	4	5
2. I found the system unnecessarily complex	1	2	3	4	5
3. I thought the system was easy to use	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
5. I found the various functions in this system were well integrated	1	2	3	4	5

System Usability Survey (SUS) – scale 3

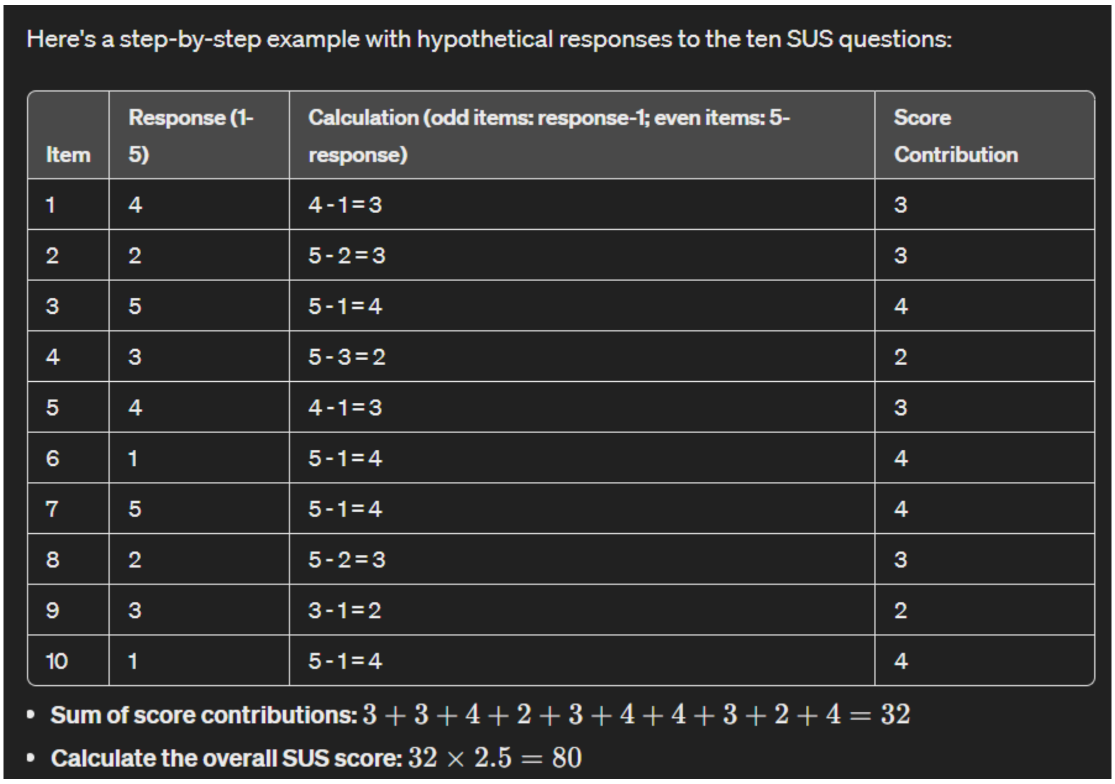


6. I thought there was too much inconsistency in this system	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
8. I found the system very cumbersome to use	1	2	3	4	5
9. I felt very confident using the system	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system	1	2	3	4	5

System Usability Survey (SUS) – scoring 1

● The SUS is given to users when they have completed using the system which is being evaluated
● They score each of the 10 items by marking one of the five boxes
● The SUS yields a single number representing a composite measure of the overall usability of the system being studied. ***Note that scores for individual items are not meaningful on their own.***

● To calculate the SUS score, first sum the score contributions from each item. Each item's score contribution will range from 0 to 4.
● For items 1,3,5,7,and 9 (the odd numbered items) the score contribution is the scale position minus 1. For items 2,4,6,8 and 10 (the even numbered items) the contribution is 5 minus the scale position.
● Multiply the sum of the scores by 2.5 to obtain the overall score.
● SUS scores have a range of 0 to 100.
● Based on research, a SUS score **above 68** would be considered above **average** and anything below 68 is below average.



Here's a step-by-step example with hypothetical responses to the ten SUS questions:

Item	Response (1-5)	Calculation (odd items: response-1; even items: 5-response)	Score Contribution
1	4	4-1=3	3
2	2	5-2=3	3
3	5	5-1=4	4
4	3	5-3=2	2
5	4	4-1=3	3
6	1	5-1=4	4
7	5	5-1=4	4
8	2	5-2=3	3
9	3	3-1=2	2
10	1	5-1=4	4

- **Sum of score contributions:** $3+3+4+2+3+4+4+3+2+4=32$
- **Calculate the overall SUS score:** $32 \times 2.5 = 80$

Statistical testing

● You might get a user to rate the SUS of two different designs and want to know if one design is significantly better than the other.
● Similarly, you might want to know if two levels of difficulty in your game are significantly different so you get a user to rate the workload of both levels.
● To determine whether the differences in scores are significantly different we can use a statistical test
● 您可能会让用户对两种不同设计的 SUS 进行评分，并想知道一种设计是否明显优于另一种。
● 同样，您可能想知道游戏中的两个难度级别是否显着不同，以便让用户对两个级别的工作量进行评分。
● 为了确定分数差异是否显着不同，我们可以使用统计检验
● When comparing two designs with the System Usability Scale (SUS) or two difficulty levels in a game, statistical tests determine if one is significantly better or more difficult

than the other.
● The Wilcoxon Signed Rank Test is suitable for paired comparisons like SUS or NASA TLX scores.
● Use it when a single user rates two conditions (e.g., two game difficulty levels).
It's effective even with small user groups (minimum of 5), but more reliable with larger groups.
● 当使用系统可用性量表(SUS) 比较两种设计或游戏中的两种难度级别时，统计测试可确定一种设计是否比另一种明显更好或更困难。
● Wilcoxon 签名排名测试适用于配对比较，例如 SUS 或 NASA TLX 分数。
● 当单个用户评价两个条件（例如，两个游戏难度级别）时使用它。
即使对于小用户组（至少 5 个）它也很有效，但对于较大的组更可靠。

● Look up the calculated W test statistic in the table of critical values
● To do this you need to know N, which is the **number of users**, and the significance level, which we will set at 0.05
● This means that if a significant difference is found then it is 95% certain that this is a real difference rather than due to randomness

● If we are comparing two sets of values generated by two different groups e.g. experienced gamers and novice gamers then we use a different test to see if they are significantly different
● This is known as the Mann-Whitney U test.
● 如果我们比较两个不同组生成的两组值，例如 经验丰富的玩家和新手玩家然后我们使用不同的测试来看看他们是否有显着差异
● 这称为 Mann-Whitney U 检验。

Week 9 Software Quality

Overview

● Software quality and how to get to it
● Test-driven development
  ○ White box testing
  ○ Black box testing

Why is Software Quality relevant?

● Reputation
● Software Certification
● Legality
● Cost of Product and Maintenance
● Organizational Certification
● Moral/ethical codes of practice
● 声誉
● 软件认证
● 产品和维护成本
● 组织认证

● 合法性　　● 道德/伦理行为准则

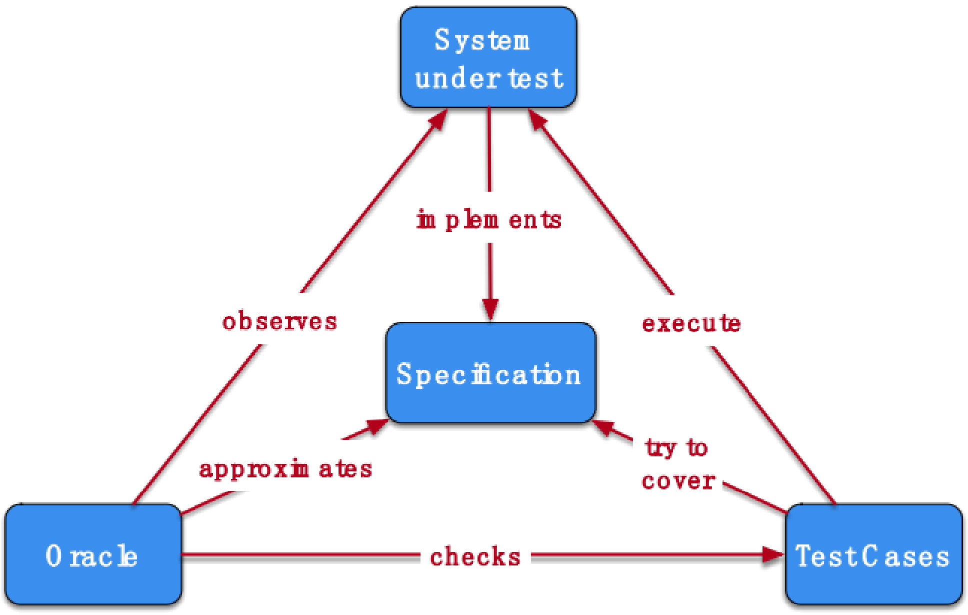
Software Quality is Multi-dimensional

● Subjective or “fitness for use”: as perceived by an individual user (e.g., aesthetics of GUI, missing functionality…)
● Objective or “conformance to requirements”: can be measured as a property of the product (e.g., detailed documentation, number of bugs, compliance with regulations ….)
● Practical: what does it mean to your team and your clients?
● 主观或“适用性”：单个用户的感知（例如 GUI 的美观、功能缺失……）
● 目标或“符合要求”：可以作为产品的属性来衡量（例如，详细文档、错误数量、遵守法规……）
● 实用：这对您的团队和客户意味着什么？

Steps Towards Software Quality:

● Use a standard development process
● Use a coding standard
  ● Compliance with industry standards (e.g., ISO, Safety, etc.) ● 符合行业标准（例如 ISO、安全等）
  ● Consistent code quality 代码一致性
  ● Secure from start
  ● Reduce development costs and accelerate time to market
○ Define and monitor metrics (defect metrics and complexity metrics) ○ 定义和监控指标（缺陷指标和复杂性指标）
● High complexity leads to higher number of defects ● 高复杂性导致更多缺陷
○ Identify and remove defects
  ● Conduct manual reviews 人工检查
  ● Use Testing

Testing process: key elements and relationships



White Box Testing

- Access to software ``internals'':
  - Source code
  - Runtime state
  - Can keep track of executions.
- White box testing exploits this to
  - Use code to measure coverage
    - Many different ways
  - Drive generation of tests that maximize coverage (推动生成最大化覆盖范围的测试)

- Coverage Metrics:
  - Statement coverage
  - Branch coverage

  ~~//● Def-Use or Dataflow coverage~~

  ~~//● MC/DC (Modified Condition / Decision Coverage)~~

  ~~//● Mutation coverage…~~
- Prescribed metrics, e.g., DO178-B/C standard for civilian aircraft software

    - non-critical - statement coverage
    - safety-critical - MC/DC coverage
- 规定的指标，例如民用飞机软件的 DO178-B/C 标准
- 非关键-语句覆盖
- 安全关键 - MC/DC 覆盖范围

Statement Coverage

- Test inputs should collectively have executed each statement
- If a statement always exhibits a fault when executed, it will be detected
- 测试输入应共同执行每个语句
- 如果一条语句在执行时总是出现错误，则会被检测到
- Computed as:

$$Coverage = \frac{|Statement\ execuated|}{|Total\ statement|}$$

Branch Coverage

- Test inputs should collectively have executed each branch
- Subsumes statement coverage
- 测试输入应共同执行每个分支
- 包含报表覆盖范围
- Computed as:

$$Coverage = \frac{|Branch\ execuated|}{|Total\ statement|}$$

Black Box Testing

- No access to “internals”
    - May have access, but don’t want to
- We know the interface
    - Parameters ○ 参数
    - Possible functions / methods ○ 可能的功能
- We may have some form of specification document

Testing Challenges

- Many different types of input
- Lots of different ways in which input choices can affect output
- An almost infinite number of possible inputs & combinations

Equivalence Partitioning (EP) Method

Identify tests by analyzing the program interface

1. Decompose program into “functional units”
2. Identify inputs / parameters for these units
3. For each input
    a) Identify its limits and characteristics
    b) Define “partitions” - value categories
    c) Identify constraints between categories
    d) Write test specification

通过分析程序接口来识别测试

1. 将程序分解为“功能单元”
2. 确定这些单元的输入/参数
3. 对于每个输入
    a) 确定其局限性和特征
    b) 定义“分区”—— 值类别
    c) 确定类别之间的约束
    d) 编写测试规范

EP – 1. Decompose into Functional Units

- Dividing into smaller units is good practice
    - Possible to generate more rigorous test cases.
    - Easier to debug if faults are found.
- E.g.: dividing a large Java application into its core modules / packages
- 分成更小的单元是很好的做法
    - 可以生成更严格的测试用例。
    - 发现故障时更容易调试。
- 例如：将大型 Java 应用程序划分为其核心模块/包
- Already a functional unit for the Grading Component example

EP – 2. Identify Inputs and Outputs

- For some systems this is straightforward
    - E.g., the Triangle program:
        - Input: 3 numbers,
        - Output: 1 String
    - E.g., Grading Component
        - Input: 2 integers: exam mark and coursework mark
        - Output: 1 String for grade
- For others less so. Consider the following:
    - A phone app.
    - A web-page with a flash component.

Boundary Values

- Most frequently errors occur in "edge" cases
    - Test just under boundary value
    - Test just above the boundary value
    - Test the boundary value

How do we go about using this?

- Testing applied in Java: Use JUnit
    - uses “Assertions” to test the code
    - Allow us to state what should be the case
    - If assertions do not hold, JUnit’s logging mechanisms reports failures
    - Various types of assertion are available, e.g., assertEquals(expected, actual); assertTrue(condition); assertFalse(condition); assertThat(value, matchingFunction)

Review

- What is Software Quality?
- What are key elements and relationships for test specifications?
- How do we carry out white-box testing?
- How do we carry out black-box testing?

What is Software Quality?

Software quality is a multi-dimensional concept that includes:

- Subjective or "fitness for use": This aspect is based on individual user perception, such as the aesthetics of the GUI or specific functionalities.
- Objective or "conformance to requirements": This can be measured objectively through various metrics such as detailed documentation, the number of bugs, and compliance with regulatory standards.
- Practical considerations: This refers to the relevance and importance of the software quality to your team and your clients, taking into account factors like maintainability, usability, and performance.

What are key elements and relationships for test specifications?

The key elements and relationships in the testing process are centered around the System Under Test (SUT), Test Cases, and Oracles:

- System Under Test (SUT): This is the software system being tested. It can be a white-box system (where internal structures are known and accessible) or a black-box system (where only external interfaces are accessible).

- Test Cases: These are specific conditions or variables used to determine if the SUT behaves as expected. Test cases should ideally cover all functionalities and potential paths in the software.
- Oracles: Oracles are mechanisms by which the correctness of the SUT’s output is judged. They can be actual expected outputs, heuristic rules, or other criteria that define what is considered a correct or incorrect behavior.

How do we carry out white-box testing?

White-box testing involves looking into the internal structures or workings of an application:

- Access the internals: Testers need access to the source code, runtime state, and sometimes even the development environment.
- Measure and maximize coverage: Use different metrics such as statement coverage, branch coverage, and mutation coverage to ensure that most or all parts of the code are executed by the test cases.
- Generate tests to maximize coverage: Develop tests that cover as many different pathways and conditions in the code as possible, often focusing on edge cases and potential error conditions.

How do we carry out black-box testing?

Black-box testing focuses on the external behavior of the software without knowledge of its internal workings:

- Understand the interface and specifications: Testers work with the known interfaces of the software—inputs and expected outputs as defined by the specifications.
- Design test cases based on functionality: Test cases are developed to check all functions and responses of the software according to what it is supposed to do, without regard to how these functions are implemented.
- Execute and observe outcomes: Testers input data through the defined interfaces and observe if the outputs match the expected results as defined by the specifications, thus validating the functionality and reliability of the software.

What is Software Quality?

- Subjective: User’s perception, like GUI aesthetics.
- Objective: Measurable by standards, e.g., bug count, documentation quality.
- Practical: Importance to your team and clients, covers usability and performance.

Key Elements for Test Specifications

- System Under Test (SUT): Software being tested.
- Test Cases: Conditions to check if SUT behaves as expected.
- Oracles: Criteria used to judge correctness of SUT outputs.

White-Box Testing

- Access Internals: Testers need source code access.
- Coverage Metrics: Use metrics like statement and branch coverage.
- Maximize Coverage: Create tests that cover all code paths and conditions.

Black-Box Testing

- Interface Knowledge: Focus on inputs and expected outputs based on specifications.
- Functionality Tests: Develop tests based on what the software is supposed to do.
- Outcome Observation: Input data and check if outputs match expectations.

什么是软件质量?

主观：用户的感知，如 GUI 美观。

目标：可以通过标准来衡量，例如错误数量、文档质量。

实用：对您的团队和客户的重要性，涵盖可用性和性能。

测试规范的关键要素

被测系统（SUT）：正在测试的软件。

测试用例：检查 SUT 是否按预期运行的条件。

Oracles：用于判断 SUT 输出正确性的标准。

白盒测试

访问内部：测试人员需要访问源代码。

覆盖率指标：使用语句和分支覆盖率等指标。

最大化覆盖范围：创建覆盖所有代码路径和条件的测试。

黑盒测试

接口知识：关注基于规范的输入和预期输出。

功能测试：根据软件的预期功能开发测试。

结果观察：输入数据并检查输出是否符合预期。