



6. Project Management

Project management is the process of **planning, organizing, and managing resources** to achieve specific goals and objectives within a defined time frame. Here are some key concepts and principles of project management:项目管理是计划、组织和管理资源的过程，以在规定的时间内实现特定的目标和目的。以下是项目管理的一些关键概念和原则：

The **three primary constraints** of project management are commonly known as the "triple constraints" - **scope, time, and cost**. These constraints form the basis of the project management triangle, where each constraint represents one side of the triangle, and the quality of the project represents the area within the triangle. T

other constraints: risk, resources and quality

1. **Project scope 项目范围**: This refers to the specific goals and objectives of the project, as well as the deliverables that will be produced.： 这指的是项目的具体目标和目的，以及将产生的可交付成果。
2. **Project planning 项目规划**: This involves defining the project scope, identifying the resources required, estimating the time and cost involved, and developing a project schedule. 这包括界定项目范围，确定所需资源，估计所涉及的时间和成本，并制定项目时间表。
3. **Risk management 风险管理**: This involves identifying potential risks and developing strategies to mitigate them, as well as planning for unexpected events and changes in the project scope. 这涉及到识别潜在的风险，并制定战略来减轻风险，以及为突发事件和项目范围的变化进行规划。
4. **Communication沟通**: Effective communication is essential for project success, and involves clear and timely communication between team members, stakeholders,

and clients. 有效的沟通是项目成功的关键，它涉及到团队成员、利益相关者和客户之间清晰而及时的沟通。

5. Project execution项目执行: This involves putting the project plan into action, monitoring progress, and making adjustments as necessary to ensure that the project stays on track. 这涉及到将项目计划付诸实施，监测进展，并在必要时进行调整，以确保项目保持在正轨上。
6. Project monitoring and control 项目监测和控制: This involves tracking progress against the project plan, identifying potential problems, and taking corrective action as needed to keep the project on track. 这包括根据项目计划跟踪进度，发现潜在的问题，并根据需要采取纠正措施，以保持项目的正常运行。
7. Project closure 项目结束: This involves finalizing deliverables, completing documentation, conducting a final review, and transitioning the project to its operational phase. 这包括最终确定可交付的成果，完成文件，进行最终审查，并将项目过渡到其运行阶段。

Effective project management requires a combination of technical skills, such as knowledge of project management methodologies and tools, as well as soft skills, such as leadership, communication, and interpersonal skills. It also requires the ability to adapt to changing circumstances and to manage risks and uncertainties.

Measurement is Central to Quality

- How to plan for the project time and effort?

For the team?

For the customer?

- Which software/part of it needs more time for testing?
- Which developer should get a bonus payment for productivity?....

Measurement is a critical aspect of software quality, as it provides objective and quantitative data that can be used to plan, track, and improve software development processes. Here are some examples of how measurement can be used in software development:

1. **Project planning**: Measurement can be used to estimate the time and effort required for completing different tasks and activities during the software development life cycle. This can help project managers to allocate resources and plan project schedules more effectively.
2. **Test planning**: Measurement can be used to identify which parts of the software system or product require more testing, based on their complexity, criticality, or risk level. This can help testers to prioritize their testing efforts and focus on areas that are most likely to contain defects or issues.
3. **Productivity assessment**: Measurement can be used to assess the productivity and performance of individual developers or teams, based on their contributions to the software development process. This can help managers to identify areas for improvement and to reward high-performing developers or teams.
4. **Quality assessment**: Measurement can be used to assess the quality of the software system or product, based on various quality metrics such as defect density, code coverage, or customer satisfaction. This can help managers to identify areas for improvement and to track the progress of quality improvement initiatives.

Overall, measurement is essential for effective project management and software quality assurance. It provides objective data that can be used to make informed decisions, optimize resources, and improve the quality of software products.

What is “Measurement”?

Attributing values to objects 为对象赋值 is an important aspect of measurement in many domains, including software engineering. By measuring and comparing different values, we can make better decisions and optimize our resources more effectively.

In the context of software engineering, some examples of values that can be attributed to software products or processes include:

- Lines of code
- Cyclomatic complexity
- Code coverage
- Defect density

- Customer satisfaction ratings
- Response time
- Memory usage
- CPU utilization
- Number of downloads or installations

By measuring and comparing these values, we can identify areas for improvement and make informed decisions about how to allocate resources and optimize our development processes. For example, we can use code coverage and defect density metrics to identify areas of the software system that require more testing, or we can use customer satisfaction ratings to prioritize new features or enhancements.

In summary, measurement is a critical aspect of software engineering, and it enables us to make better decisions, optimize our resources, and improve the quality of our software products and processes.

测量是根据某些定义的标准或标准为对象或事件分配数值的过程。在软件工程中，度量用于评估软件开发过程的各个方面和由此产生的软件产品，例如它们的性能、可靠性、可用性和可维护性。

度量提供了一种方法来量化和分析软件开发过程和正在开发的软件产品的各个方面。它使我们能够确定需要改进的领域，设定目标和目的，并随着时间的推移监控进展情况。通过衡量各种指标并将它们与既定基准或标准进行比较，我们可以确定优势和劣势领域，并就如何分配资源和提高软件产品质量做出明智的决策。

度量是软件质量保证的一个关键方面，它用于整个软件开发生命周期，从需求收集和设计到测试、部署和维护。它有助于确保开发和维护软件产品以满足用户和利益相关者的需求和期望，并确保在预算范围内按时交付。

Measurement is Difficult in Software Engineering

- Most entities are difficult to measure reliably
- Difficult or impossible to “pin down” a single value

E.g., Software Quality (ISO/IEC 25010):

measurement is often difficult in software engineering due to the complexity and variability of software products and development processes. Here are some reasons why measurement can be challenging in software engineering: 由于软件产品和开发过程的复杂性和可变性，在软件工程中，测量往往是困难的。以下是测量在软件工程中具有挑战性的一些原因：

1. **Complexity** 复杂性: Software systems and products can be highly complex, with many interdependent components and interactions. This can make it difficult to measure the performance, reliability, or other characteristics of the software system or product. 软件系统和产品可能是高度复杂的，有许多相互依赖的组件和相互作用。这可能使测量软件系统或产品的性能、可靠性或其他特性变得困难。
2. **Variability** 可变性: Software development processes can vary widely across different organizations, projects, and teams. This can make it difficult to establish consistent and comparable measures across different contexts. 软件开发过程在不同的组织、项目和团队中可能有很大的不同。这就使得在不同情况下建立一致的、可比较的衡量标准变得困难。
3. **Subjectivity** 主观性: Software quality is often subjective and can vary depending on the needs and expectations of different stakeholders. This can make it difficult to establish objective and reliable measures of software quality. 软件质量往往是主观的，并可能因不同利益相关者的需求和期望而不同。这使得建立客观、可靠的软件质量衡量标准变得困难。
4. **Dynamic nature** 动态性: Software systems and products can change rapidly over time, with new features, enhancements, and bug fixes being introduced frequently. This can make it difficult to establish consistent measures over time. 软件系统和产品可能会随着时间的推移而迅速变化，新的功能、增强功能和错误修复经常被引入。这可能会使我们难以建立长期一致的衡量标准。
5. **Data availability** 数据的可用性: Collecting and analyzing data on software development processes and products can be challenging, especially when data is not readily available or when data is incomplete or unreliable. 收集和分析有关软件开发过程和产品的数据可能是一个挑战，特别是当数据不容易获得，或者数据不完整或不可靠时。

Despite these challenges, measurement remains an important aspect of software engineering, as it provides a way to evaluate and improve the quality of software products and processes. To overcome these challenges, software engineering

practitioners use a range of techniques and tools for collecting and analyzing data, such as software metrics, data mining, and machine learning. They also rely on their professional judgment and experience to interpret and apply the results of measurement activities in a meaningful way. 尽管有这些挑战，测量仍然是软件工程的一个重要方面，因为它提供了一种评估和改善软件产品和过程质量的方法。为了克服这些挑战，软件工程从业人员使用一系列技术和工具来收集和分析数据，如软件度量、数据挖掘和机器学习。他们还依靠自己的专业判断和经验，以有意义的方式解释和应用测量活动的结果。

Usual **Metrics**: Size and Complexity

- **After development ...**

1. How much effort will it require for maintenance?
2. Where should we direct testing effort?
3. How much effort was required for development?
4. Metrics are based upon source code ("white box")

- **Before development has started ...**

1. How much programming effort will module X require?
2. What will be the estimated cost of the final product?
3. Metrics are based upon requirements / specification ("black box")

Metrics are **essential** for evaluating software quality and for making **informed decisions** during the software development life cycle. **Size and complexity** are two common metrics used in software engineering to measure various aspects of software products and processes. 指标对于评估软件质量和在软件开发生命周期中做出明智的决策至关重要。规模和复杂性是软件工程中用来衡量软件产品和流程各个方面的两个常用指标。

Size metrics are based on the size of the software product or its components, such as lines of code, number of modules, or number of functions. These **metrics** can help us to estimate the effort required for software development and maintenance, to compare different software products or components, and to identify potential issues related to software size. 大小度量基于软件产品或其组件的大小，例如代码行数、模块数或功能

数。这些指标可以帮助我们估计软件开发和维护所需的工作量，比较不同的软件产品或组件，以及识别与软件规模相关的潜在问题。

Complexity metrics are based on the complexity of the software product or its components, such as the number of **control flow statements**, the depth of nesting, or the number of decision points. These metrics can help us to identify potential issues related to software complexity, such as high **defect rates** or **maintenance problems**, and to direct testing efforts more effectively. 复杂性指标基于软件产品或其组件的复杂性，例如控制流语句的数量、嵌套的深度或决策点的数量。这些指标可以帮助我们识别与软件复杂性相关的潜在问题，例如高缺陷率或维护问题，并更有效地指导测试工作。

Size and complexity metrics can be used both before and after development has started. Before development, size and complexity metrics can be used to **estimate** the effort required for software development, to evaluate different design options, and to **estimate** the cost of the final product. After development, size and complexity metrics can be used to evaluate the quality and maintainability of the software product, to identify potential issues related to software size and complexity, and to direct **testing and maintenance** efforts more effectively. 大小和复杂性指标可以在开发开始之前和之后使用。在开发之前，大小和复杂性指标可用于估计软件开发所需的工作量、评估不同的设计选项以及估计最终产品的成本。开发完成后，大小和复杂性指标可用于评估软件产品的质量和可维护性，识别与软件大小和复杂性相关的潜在问题，并更有效地指导测试和维护工作。

Overall, **size and complexity metrics** are important tools for software engineering practitioners, as they provide a way to measure and evaluate various aspects of software products and processes. By using these metrics, we can make more informed decisions, identify potential issues earlier, and improve the quality of our software products. 总的来说，规模和复杂性指标是软件工程从业者的重要工具，因为它们提供了一种衡量和评估软件产品和流程各个方面的方法。通过使用这些指标，我们可以做出更明智的决策，更早地发现潜在问题，并提高我们软件产品的质量。

White Box Complexity Metrics

White box complexity metrics are a type of software metric that are based on the internal structure and logic of the software product or system. These metrics are used to evaluate the complexity and maintainability of software code and to identify potential issues related to software quality.

白盒复杂性指标是一种基于软件产品或系统的内部结构和逻辑的软件指标。这些指标用于评估软件代码的复杂性和可维护性，并识别与软件质量相关的潜在问题。

Here are some examples of white box complexity metrics:

1. **Lines of code (LOC):** This metric counts the number of lines of code in a software program or module. This metric can be used to estimate the size and complexity of the software program and to compare different software components. 代码行数 (LOC)：此指标计算软件程序或模块中的代码行数。该指标可用于估计软件程序的大小和复杂性，以及比较不同的软件组件。
2. **Cyclomatic complexity:** This metric measures the number of independent paths through a software program or module. It is based on the control flow graph of the program and can be used to identify potential issues related to code quality, such as high defect rates or maintenance problems. 圈复杂度：该指标衡量通过软件程序或模块的独立路径的数量。它基于程序的控制流图，可用于识别与代码质量相关的潜在问题，例如高缺陷率或维护问题。
3. **Halstead complexity measures:** These measures are based on the number of distinct operators and operands in a software program or module. They can be used to estimate the effort required for software development and maintenance and to identify potential issues related to code quality. Halstead 复杂性度量：这些度量基于软件程序或模块中不同运算符和操作数的数量。它们可用于估计软件开发和维护所需的工作量，并识别与代码质量相关的潜在问题。
4. **Maintainability index:** This metric is based on a combination of various factors related to software code, such as complexity, size, and coupling. It is used to evaluate the maintainability of software code and to identify potential issues related to software quality and maintainability. 可维护性指数：该指标基于与软件代码相关的各种因素的组合，例如复杂性、大小和耦合。它用于评估软件代码的可维护性，并识别与软件质量和可维护性相关的潜在问题。

By using white box complexity metrics, software engineering practitioners can identify potential issues related to software quality and maintainability and take proactive steps to address them. These metrics can also be used to direct testing and maintenance efforts more effectively and to optimize the overall software development process.

通过使用白盒复杂性指标，软件工程从业者可以识别与软件质量和可维护性相关的潜在问题，并采取主动措施来解决这些问题。这些指标还可用于更有效地指导测试和维护工作，并优化整个软件开发过程。

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

The number of lines of code (LOC) is one of the most widely used metrics for measuring the size and complexity of software products. Here are some additional points to consider regarding this metric: 代码行数 (LOC) 是最广泛使用的衡量软件产品规模和复杂性的指标之一。以下是关于此指标的一些其他要点：

- The LOC metric is easy to compute and interpret, and it can provide an approximate measure of the size and complexity of a software product or system. LOC 度量易于计算和解释，它可以提供软件产品或系统的大小和复杂性的近似度量。
- However, the LOC metric has some limitations, as not all lines of code are created equal. For example, blank lines and comments can increase the number of lines without adding to the complexity of the code. 但是，LOC 指标有一些局限性，因为并非所有代码行都是一样的。例如，空行和注释可以增加行数而不增加代码的复杂性。
- The definition of a "line" of code can also vary depending on the programming language and coding style being used. For example, some programming languages may allow multiple statements on a single line, while others may require each statement to be on a separate line. 代码“行”的定义也可能因所使用的编程语言和编码

风格而异。例如，某些编程语言可能允许在一行中使用多个语句，而其他语言可能要求每个语句都在单独的行中。

- The LOC metric does not take into account the logical or architectural complexity of the software product or system. Therefore, it may not provide an accurate measure of the true complexity of the code. LOC 指标不考虑软件产品或系统的逻辑或架构复杂性。因此，它可能无法准确衡量代码的真实复杂性。
- Finally, the LOC metric is highly language-specific, as different programming languages may have different syntax and coding styles that affect the number of lines required to express a given piece of functionality. 最后，LOC 指标是高度特定于语言的，因为不同的编程语言可能具有不同的语法和编码风格，这会影响表达给定功能所需的行数。

In summary, while the number of lines of code can provide a useful approximation of the size and complexity of a software product or system, it is important to consider its limitations and to use other metrics and evaluation techniques to obtain a more complete understanding of the software product or system's quality and complexity. 总之，虽然代码行数可以提供对软件产品或系统的大小和复杂性的有用近似值，但重要的是要考虑其局限性并使用其他指标和评估技术来更全面地了解软件产品或系统的质量和复杂性。

Cyclomatic Complexity 循环复杂度

- Calculated from the control flow graph:

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

E – number of edges in the control flow graph;

N – number of nodes in the control flow graph;

P – number of procedures (usually 1)

The **resulting V(G)** value represents the number of independent paths through the code.

An independent path is any path that introduces at least one new statement or condition. In other words, an independent path is a path that is not already covered by

another path. By identifying the independent paths in a software program or module, we can determine the cyclomatic complexity of the code and gain insights into its logical complexity.

Overall, cyclomatic complexity is a useful metric for measuring the complexity and maintainability of software code, and it can help software engineering practitioners to identify potential issues related to code quality and to direct testing and maintenance efforts more effectively.

独立路径是指至少引入一个新语句或条件的任何路径。换句话说，一个独立的路径是一个没有被其他路径覆盖的路径。通过识别一个软件程序或模块中的独立路径，我们可以确定代码的循环复杂性，并深入了解其逻辑复杂性。

总的来说，循环复杂度是衡量软件代码复杂性和可维护性的一个有用指标，它可以帮助软件工程从业者识别与代码质量有关的潜在问题，并更有效地指导测试和维护工作。

- Number of independent paths through the code
- Independent path – any path that introduces at least one new statement/condition

Cyclomatic complexity is a **white box complexity metric** that measures the number of independent paths through a software program or module. It is based on the control flow graph of the program and provides a measure of the logical complexity of the code.

Here are some key points to consider regarding cyclomatic complexity:

- Cyclomatic complexity is based on the control flow graph of the software program or module. This graph shows the different control flow structures in the program, such as loops, conditionals, and subroutines, and the connections between them. 圈复杂度基于软件程序或模块的控制流图。该图显示了程序中不同的控制流结构，例如循环、条件和子例程，以及它们之间的连接。
- Cyclomatic complexity is computed by counting the number of independent paths through the control flow graph. Each path represents a possible sequence of program statements that can be executed during the execution of the program. 通过计算通过控制流图的独立路径的数量来计算圈复杂度。每个路径代表一个可能的程序语句序列，可以在程序执行期间执行。
- A higher cyclomatic complexity value indicates a higher degree of program complexity and a greater number of possible execution paths through the program.

更高的圈复杂度值表示更高程度的程序复杂度和更多可能的程序执行路径。

- Cyclomatic complexity can be used to identify potential issues related to code quality, such as high defect rates or maintenance problems. It can also be used to direct testing and maintenance efforts more effectively. 圈复杂度可用于识别与代码质量相关的潜在问题，例如高缺陷率或维护问题。它还可用于更有效地指导测试和维护工作。
- However, cyclomatic complexity does have some limitations. For example, it does not take into account the data complexity of the program, and it may not be applicable to all types of software programs or modules. 然而，圈复杂度确实有一些局限性。例如，它没有考虑程序的数据复杂性，它可能不适用于所有类型的软件程序或模块。
- Finally, it is important to note that cyclomatic complexity is a language-specific metric, as different programming languages may have different syntax and control flow structures that affect the cyclomatic complexity of a given piece of code. 最后，重要的是要注意圈复杂度是一种特定于语言的度量，因为不同的编程语言可能具有不同的语法和控制流结构，这些结构会影响给定代码段的圈复杂度。

Overall, cyclomatic complexity is a useful white box complexity metric that can provide insight into the logical complexity of software programs or modules. By using this metric, software engineering practitioners can identify potential issues related to code quality and maintainability and take proactive steps to address them.

Black Box Complexity Metrics

(If we don't have implemented code)

Black box complexity metrics are based on the external behavior of the software product or system, as observed through its inputs and outputs. These metrics are used to evaluate the complexity and quality of the software product or system from a user's perspective. 黑盒复杂性指标基于软件产品或系统的外部行为，如通过其输入和输出所观察到的那样。这些指标用于从用户的角度评估软件产品或系统的复杂性和质量。

Here are some examples of black box complexity metrics:

1. **Planning poker**: This is a technique used to estimate the effort required to complete a software development task. Team members are given a set of cards with different

point values, and they use these cards to estimate the complexity and effort required for each task. 计划扑克：这是一种用于估计完成软件开发任务所需工作量的技术。团队成员得到一组具有不同点值的卡片，他们使用这些卡片来估计每项任务所需的复杂性和工作量。

2. **Function points**: This metric measures the functionality provided by a software product or system, based on its inputs and outputs. It can be used to estimate the size and complexity of the software product or system and to compare different software components. 功能点：该度量标准根据软件产品或系统的输入和输出来衡量软件产品或系统提供的功能。它可用于估计软件产品或系统的大小和复杂性，以及比较不同的软件组件。
3. **User interface complexity**: This metric measures the complexity of the user interface of a software product or system, based on factors such as the number of screens, menu options, and input fields. It can be used to evaluate the usability and user-friendliness of the software product or system. 用户界面复杂性：该指标根据屏幕数量、菜单选项和输入字段等因素衡量软件产品或系统用户界面的复杂性。它可用于评价软件产品或系统的可用性和用户友好性。
4. **Performance metrics**: These metrics measure the performance of a software product or system, based on factors such as response time, throughput, and resource utilization. They can be used to identify potential issues related to software quality and to optimize the performance of the software product or system. 性能指标：这些指标根据响应时间、吞吐量和资源利用率等因素衡量软件产品或系统的性能。它们可用于识别与软件质量相关的潜在问题并优化软件产品或系统的性能。

By using black box complexity metrics, software engineering practitioners can gain insights into the complexity and quality of software products or systems from a user's perspective. These metrics can help to identify potential issues related to software quality, usability, and performance and to take proactive steps to address them.

Storey Points 层点

- 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**” 一起做决定
- The collective estimate of groups (i.e., of effort required for a story) is better than the estimate of an individual

Story points are an informal, agile unit of size measurement that is used to estimate the effort required to complete a software development task or user story. Here are some additional points to consider regarding story points:故事点是一种非正式的、灵活的规模度量单位，用于估计完成软件开发任务或用户故事所需的工作量。以下是关于故事点的一些其他要点：

- Story points are usually estimated on a scale from 1 to 10, with 1 representing a small and simple task and 10 representing a large and complex task.故事点通常按 1 到 10 的等级进行估算，1 代表小而简单的任务，10 代表大而复杂的任务。
- The estimation process typically involves the whole team, who provide their individual estimates and then discuss and negotiate until a consensus is reached.估算过程通常涉及整个团队，他们提供各自的估算，然后进行讨论和协商，直到达成共识。
- The estimation process is based on the idea of "wisdom of the crowds," which suggests that the collective estimate of a group is often more accurate than the estimate of an individual.估计过程基于“群体智慧”的理念，这表明群体的集体估计往往比个人的估计更准确。
- Story points are not a measure of time, but rather a measure of relative complexity and effort required to complete a task. • 故事点不是时间的衡量标准，而是完成任务所需的相对复杂性和工作量的衡量标准。
- Story points are often used in agile software development methodologies, such as Scrum, as a way to plan and track the progress of development tasks over time.故事点通常用于敏捷软件开发方法，例如 Scrum，作为一种随着时间的推移计划和跟踪开发任务进度的方法。

Planning Poker

Planning poker is a **technique** used in **agile software development** to estimate the effort required to complete a development task or user story. Here are some key points to

consider regarding planning poker:

- Planning poker is typically done during a sprint planning meeting, where the team discusses the tasks that need to be completed during the upcoming sprint. 计划扑克通常在冲刺计划会议期间完成，团队会在会上讨论在即将到来的冲刺期间需要完成的任务。
- Each team member is given a set of cards with point values, usually ranging from 1 to 10, that they use to estimate the effort required to complete a task. 每个团队成员都会得到一组带有点值的卡片，通常在 1 到 10 之间，他们用这些卡片来估计完成任务所需的工作量。
- The team members discuss the task and then choose a card that represents their individual estimate. 团队成员讨论任务，然后选择一张代表他们个人估计的卡片。
- Once all team members have chosen a card, the cards are revealed and the estimates are compared. 一旦所有团队成员都选择了一张卡片，就会展示这些卡片并比较估计值。
- If there is a wide range of estimates, the team members discuss the reasons for their estimates and try to reach a consensus on the estimated effort required. 如果估算范围很广，团队成员会讨论他们估算的原因，并尝试就估算所需的工作量达成共识。
- The process is repeated for each task or user story until all have been estimated. 对每个任务或用户故事重复该过程，直到对所有任务或用户故事进行估计。

Planning poker is based on the idea of "**wisdom of the crowds**," which suggests that the collective estimate of a group is often more accurate than the estimate of an individual. By using this technique, agile teams can leverage the expertise and knowledge of all team members to make more accurate estimates of development effort.

Process:

During a planning poker session, the whole agile development team is typically involved, and each team member is given a set of numbered cards to use for estimating the effort required to complete a development task or user story. Here are some additional points to consider:

- The numbered cards used in planning poker typically follow the Fibonacci sequence, with values such as 1, 3, 5, 8, 13, 20, and so on. 计划扑克中使用的编号牌通常遵循斐波那契数列，具有 1、3、5、8、13、20 等值。
- The use of larger values for more complex tasks helps to account for the uncertainty and variability involved in estimating development effort. 对更复杂的任务使用更大的值有助于解释估算开发工作量所涉及的不确定性和可变性。
- Each team member is also given a "?" card, which can be used if they are uncertain about the estimated effort required for a particular task. 每个团队成员也被赋予一个“？”卡，如果他们不确定某项特定任务所需的估计工作量，可以使用该卡。
- Once all team members have chosen a card, the cards are revealed simultaneously, and the estimates are compared. 一旦所有团队成员都选择了一张卡片，卡片就会同时展示，并比较估计值。
- If there is a wide range of estimates, the team members discuss the reasons for their estimates and try to reach a consensus on the estimated effort required. 如果估算范围很广，团队成员会讨论他们估算的原因，并尝试就估算所需的工作量达成共识。
- Planning poker helps to promote collaboration and communication within the development team, and it can help to improve the accuracy of development estimates over time. 规划扑克有助于促进开发团队内部的协作和沟通，并且随着时间的推移，它可以帮助提高开发估算的准确性。
- Low values - trivial to implement 数值越低越容易解决
- High values - difficult to implement 数值越高越难解决

1. 各自在卡片上写下自己认为的复杂指数
2. 公示鸽子的结果
3. 大家共同讨论为什么这样评分
4. 达成共识(同一个case最多讨论三次)

Team Velocity 团队速度

- 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.

Team velocity is a metric used in **agile software development** to measure the number of estimated story points implemented by a development team in a sprint. Here are some additional points to consider regarding team velocity:团队速度是敏捷软件开发中使用的一种度量标准，用于衡量开发团队在冲刺中实现的估计故事点的数量。以下是关于团队速度的一些其他要点：

- Team velocity can be calculated by measuring the number of story points completed by the team in a single sprint and then averaging that value over a set number of previous sprints (e.g., the previous 3 or 4 sprints).团队速度可以通过测量团队在单个冲刺中完成的故事点的数量，然后对该值在一组先前冲刺（例如，先前的 3 或 4 个冲刺）中的平均值进行计算。
- Team velocity is a useful metric for estimating the time required to complete a project, as well as for setting realistic targets for the number of stories that can be completed in a sprint.团队速度是一个有用的指标，可用于估算完成项目所需的时间，以及为可在冲刺中完成的故事数量设定切合实际的目标。
- By using team velocity to estimate the time required to complete a project, development teams can set more realistic project timelines and adjust their plans as needed based on actual progress.通过使用团队速度来估算完成项目所需的时间，开发团队可以设置更切合实际的项目时间表，并根据实际进度根据需要调整计划。
- Team velocity can also be used to identify potential areas for improvement in the development process, such as identifying bottlenecks or areas where the team may

be overcommitting or undercommitting. 团队速度还可用于识别开发过程中需要改进的潜在领域，例如识别瓶颈或团队可能过度承诺或承诺不足的领域。

- It's important to note that team velocity is a relative measure of development progress and does not directly correspond to actual time or effort required to complete a task or project. Instead, it provides a way to compare development progress over time and adjust plans accordingly. 请务必注意，团队速度是开发进度的相对衡量标准，并不直接对应于完成任务或项目所需的实际时间或工作量。相反，它提供了一种方法来比较一段时间内的开发进度并相应地调整计划。

Overall, team velocity is a valuable metric for agile development teams, as it can help to improve planning and decision-making and provide insight into the overall effectiveness of the development process. 总的来说，团队速度是敏捷开发团队的一个有价值的指标，因为它可以帮助改进规划和决策制定，并提供对开发过程整体有效性的洞察。

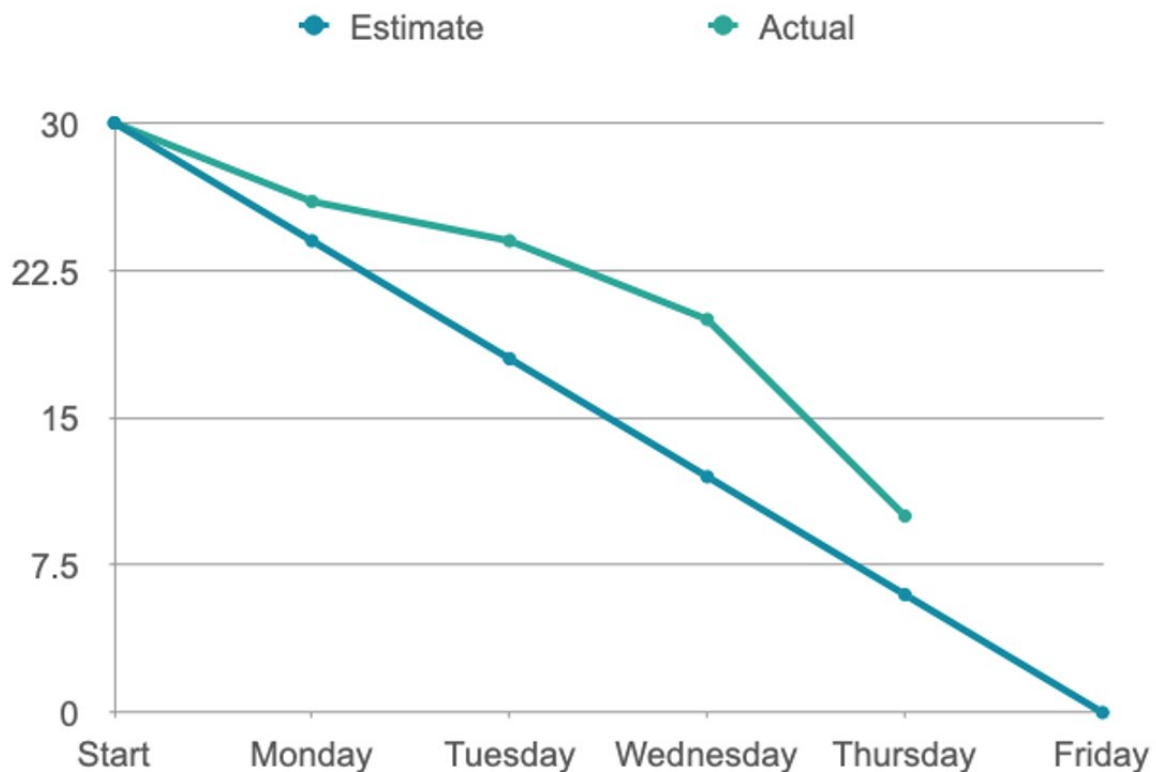
Burn Charts

Burn charts are a type of chart used in agile software development to visualize the progress of a development project over time. Here are some key points to consider regarding burn charts: 燃烧图是敏捷软件开发中使用的一种图表，用于可视化开发项目随时间的进展情况。以下是关于燃烧图需要考虑的一些要点：

- A burn chart typically shows the amount of work completed over time, usually in the form of a line chart. • 燃烧图通常以折线图的形式显示随时间推移完成的工作量。
- The **horizontal axis** of the chart represents time, usually in sprints or iterations. 图表的水平轴表示时间，通常以冲刺或迭代形式表示。
- The **vertical axis** of the chart represents the amount of work completed, usually in terms of story points or some other measure of development effort. 图表的纵轴表示已完成的工作量，通常以故事点或其他一些开发工作量度表示。
- As work is completed, the line on the chart moves up and to the right, indicating progress. • 工作完成后，图表上的线向上和向右移动，表示进度。
- A burn chart can be used to track the progress of individual sprints or of the project as a whole. • 燃烧图可用于跟踪单个冲刺或整个项目的进度。

- For example, if the line on the chart is not moving up as quickly as expected, the team may need to adjust their estimates or identify and address bottlenecks in the development process. 例如，如果图表上的线没有像预期的那样快速上升，团队可能需要调整他们的估计或识别和解决开发过程中的瓶颈。

Overall, burn charts are a useful tool for agile development teams, as they provide a clear and visual representation of development progress over time. By using burn charts, teams can better understand their progress and make more informed decisions about project planning and execution. 总的来说，燃烧图是敏捷开发团队的有用工具，因为它们提供了随时间推移的开发进度的清晰可视化表示。通过使用燃烧图，团队可以更好地了解他们的进度，并就项目规划和执行做出更明智的决策。



Software Laws: Patents, Copyright, Contract, Privacy

Petent law

Patent law is a type of software law that relates to the protection of software-related inventions. Here are some key points to consider regarding patent law:

- In many countries, software-related inventions can be protected by patents, which grant the holder exclusive rights to make, use, and sell the invention for a limited period of time (**usually 20 years**).在许多国家/地区，与软件相关的发明可以受专利保护，专利授予持有人在有限的时间段（通常为 20 年）内制造、使用和销售该发明的专有权。
- To be eligible for a patent, a software-related invention must **be new, non-obvious, and useful**.• 要获得专利资格，与软件相关的发明必须是新颖的、非显而易见的和有用的。
- In general, patent law is intended to encourage innovation by granting inventors exclusive rights to their inventions, which can help to offset the costs of research and development.一般来说，专利法旨在通过授予发明人对其发明的专有权来鼓励创新，这有助于抵消研发成本。
- However, patent law can also be controversial in the software industry, as some argue that it can stifle innovation by creating barriers to entry and limiting the ability of developers to build on existing ideas.然而，专利法在软件行业也可能引起争议，因为一些人认为它可以通过设置进入壁垒和限制开发人员在现有想法的基础上进行构建的能力来扼杀创新。
- Additionally, some have argued that software-related inventions are inherently difficult to patent, as software often builds on existing ideas and may be difficult to define and describe in a way that satisfies patent requirements.此外，一些人认为与软件相关的发明本质上很难申请专利，因为软件通常建立在现有想法的基础上，可能难以以满足专利要求的方式定义和描述。
- To be eligible for a software patent, an invention must typically be more than just a basic algorithm or mathematical formula. Instead, it must typically involve a specific application of the technology that is novel and non-obvious.要获得软件专利资格，一项发明通常必须不仅仅是基本算法或数学公式。相反，它通常必须涉及新颖且非显而易见的技术的特定应用。
- Patents can be expensive and time-consuming to obtain, and may not always be the best option for protecting intellectual property. In some cases, trade secrets or other forms of intellectual property protection may be more appropriate.获得专利可

能既昂贵又费时，而且可能并不总是保护知识产权的最佳选择。在某些情况下，商业秘密或其他形式的知识产权保护可能更合适。

Overall, patent law is an important consideration for software developers and companies, as it can impact their ability to protect and commercialize software-related inventions. By understanding the basics of patent law and working with qualified legal professionals, developers can help to protect their intellectual property and ensure that their innovations are properly recognized and rewarded. 总体而言，专利法是软件开发者和公司的重要考虑因素，因为它会影响他们保护软件相关发明并将其商业化的能力。通过了解专利法的基础知识并与合格的法律专业人士合作，开发人员可以帮助保护他们的知识产权并确保他们的创新得到适当的认可和奖励。

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 law is a type of software law that relates to the protection of original creative works, including software. Here are some additional points to consider regarding copyright: 著作权法是一种软件法，涉及对包括软件在内的原创作品的保护。以下是有关版权的一些其他要点：

- Copyright grants the creator of an original work the exclusive right to perform, copy, and adapt their work. This means that others must obtain permission from the creator in order to use or copy their work, and may be required to pay for the privilege.版权授予原创作品的创作者执行、复制和改编其作品的专有权利。这意味着其他人必须获得创作者的许可才能使用或复制他们的作品，并且可能需要为此特权付费。
- Copyright protection applies to a range of creative works, including literary, dramatic, musical, and artistic works. In the context of software, this includes the source code, object code, and user interface design of an application.版权保护适用于一系列创意作品，包括文学、戏剧、音乐和艺术作品。在软件环境中，这包括应用程序的源代码、目标代码和用户界面设计。
- Copyright protection is automatic and does not require registration. However, creators may choose to register their copyright with the government in order to gain additional legal protections.版权保护是自动的，不需要注册。但是，创作者可以选择向政府注册其版权以获得额外的法律保护。
- Copyright protection lasts for a fixed period of time, typically 70 years after the death of the author. However, there are many exceptions and limitations to copyright protection, particularly in the context of software.版权保护持续一段固定的时间，通常是作者去世后 70 年。但是，版权保护有许多例外和限制，特别是在软件方面。
- Copyright infringement can occur in a variety of ways, including through the unauthorized copying or distribution of software or through the use of someone else's code or design in a new application.侵犯版权的方式多种多样，包括未经授权复制或分发软件，或者在新应用程序中使用他人的代码或设计。

Overall, copyright is an important consideration for software developers and companies, as it can impact their ability to protect their intellectual property and prevent others from using their work without permission. By understanding the basics of copyright law and working with qualified legal professionals, developers can help to ensure that their software is properly protected and that they are not **infringing**侵犯;违反 on the rights of others.

总的来说，版权是软件开发商和公司的一个重要考虑因素，因为它会影响他们保护自己的知识产权并防止他人未经许可使用他们的作品的能力。通过了解版权法的基础知识并与合

格的法律专业人士合作，开发人员可以帮助确保他们的软件得到适当的保护，并且他们不会侵犯他人的权利。

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

Here are some additional points to consider regarding copyright theft: 以下是关于版权盗窃的一些其他要点：

- If you want to use someone else's creative work (including software), you typically need to obtain permission from the creator, either through a license agreement or some other type of legal arrangement. 如果您想使用他人的创意作品（包括软件），您通常需要通过许可协议或某种其他类型的法律安排获得创作者的许可。
- "Fair use" is a legal doctrine that allows limited use of copyrighted works without permission, typically for purposes such as criticism, comment, news reporting, teaching, scholarship, or research. However, the exact scope of fair use is often unclear and can depend on a variety of factors. “合理使用”是一种法律原则，允许在未经许可的情况下有限地使用受版权保护的作品，通常用于批评、评论、新闻报道、教学、奖学金或研究等目的。然而，合理使用的确切范围通常并不明确，可能取决于多种因素。
- Open source software is software that is made freely available for anyone to use, modify, and distribute. However, there may still be restrictions on how open source

software can be used or distributed, depending on the specific license agreement.开源软件是免费提供给任何人使用、修改和分发的软件。但是，根据特定的许可协议，开源软件的使用或分发方式可能仍然存在限制。

- Simply copying someone else's code or design is not acceptable and may be a violation of copyright law. However, if you independently create something that is similar to someone else's work, you may not be infringing on their copyright.简单地复制他人的代码或设计是不可接受的，并且可能违反版权法。但是，如果您独立创作与他人作品相似的作品，您可能不会侵犯他们的版权。
- In general, it's important to be aware of the legal implications of using someone else's creative work, and to seek legal advice if you're uncertain about whether your use of a particular work is legal.一般来说，了解使用他人创意作品的法律影响很重要，如果您不确定您对特定作品的使用是否合法，请寻求法律建议。

Contract Law 合同法

Employer contracts usually force an employee to:

- Not work for anyone else
- Hand over any ideas (Intellectual Property)
- Not disclose company secrets (Non-disclosure-agreements)

(even after you stop working for them)

Contract law is an important aspect of software law, particularly in the context of employment agreements and intellectual property ownership. Here are some additional points to consider regarding contract law:合同法是软件法的一个重要方面，特别是在雇佣协议和知识产权所有权方面。以下是关于合同法需要考虑的一些额外要点：

- Employment contracts often include provisions that limit an employee's ability to work for competitors, disclose company secrets, or use the company's intellectual property for their own purposes. These provisions may be enforceable under contract law, and employees who violate these provisions may be subject to legal penalties.雇佣合同通常包括限制员工为竞争对手工作、披露公司机密或为自己的目

的使用公司知识产权的能力的条款。这些规定可根据合同法强制执行，违反这些规定的员工可能会受到法律处罚。

- In addition to employment contracts, companies may also use non-disclosure agreements (NDAs) to protect their trade secrets and other confidential information. NDAs typically prohibit the recipient from disclosing the confidential information to others or using it for their own purposes, and may include provisions for legal remedies if the NDA is violated.除了雇佣合同，公司还可以使用保密协议 (NDA) 来保护他们的商业秘密和其他机密信息。NDA 通常禁止接收者向他人披露机密信息或将其用于自己的目的，并且可能包括在违反 NDA 时的法律补救措施条款。
- Contract law also applies to software development agreements, such as contracts between a client and a software development company. These agreements may include provisions related to intellectual property ownership, liability for defects or errors in the software, and dispute resolution mechanisms.合同法也适用于软件开发协议，例如客户与软件开发公司之间的合同。这些协议可能包括与知识产权所有权、软件缺陷或错误的责任以及争议解决机制相关的条款。
- Like other types of law, contract law can be complex and difficult to navigate. It's important for developers and companies to work with qualified legal professionals to ensure that their contracts are legally enforceable and protect their interests.与其他类型的法律一样，合同法可能很复杂且难以驾驭。开发商和公司必须与合格的法律专业人士合作，以确保他们的合同具有法律效力并保护他们的利益。

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**

Data protection is an important aspect of software law, particularly in the context of privacy and security. Here are some additional points to consider regarding data protection: 数据保护是软件法的一个重要方面，尤其是在隐私和安全方面。以下是有关数据保护的一些其他要点：

- Data protection laws are designed to protect individuals' personal data and privacy. These laws vary by country, but they generally require companies to obtain individuals' consent before collecting and processing their personal data, and to take measures to protect the security and confidentiality of that data. 数据保护法旨在保护个人的个人数据和隐私。这些法律因国家/地区而异，但它们通常要求公司在收集和处理个人数据之前获得个人的同意，并采取措施保护该数据的安全性和机密性。
- In the UK, the Data Protection Act sets out the principles for handling personal data, while in the EU, the Data Protection Directive provides similar guidance. In the US, there is no federal data protection law, but individual states have passed their own data protection laws. 在英国，《数据保护法》规定了处理个人数据的原则，而在欧盟，《数据保护指令》提供了类似的指导。在美国，没有联邦数据保护法，但各州已经通过了自己的数据保护法。
- The eight principles of data protection set out the basic requirements for handling personal data. These include processing data fairly and lawfully, using it for limited purposes, keeping it accurate and up-to-date, and ensuring that it is secure and not transferred to countries without adequate protection. 保障资料八项原则订明处理个人资料的基本要求。其中包括公平合法地处理数据，将其用于有限的目的，保持数据准确和最新，并确保数据安全且不会转移到没有充分保护的国家。

- Data protection laws can have significant implications for software development, particularly for companies that collect, store, or process large amounts of personal data. Developers may need to build in data protection features, such as encryption or access controls, to comply with these laws and protect individuals' privacy. 数据保护法可能会对软件开发产生重大影响，尤其是对于收集、存储或处理大量个人数据的公司而言。开发人员可能需要内置数据保护功能，例如加密或访问控制，以遵守这些法律并保护个人隐私。
- In addition to legal requirements, companies that handle personal data may also be subject to ethical considerations regarding privacy and data protection. These considerations may include respecting individuals' rights to privacy, being transparent about how data is collected and used, and taking steps to protect against unauthorized access or misuse of personal data. 除法律要求外，处理个人数据的公司还可能受到有关隐私和数据保护的道德考虑。这些考虑因素可能包括尊重个人的隐私权、对数据的收集和使用方式保持透明，以及采取措施防止未经授权的访问或滥用个人数据。

Review:

How can we measure complexity?

1. **Lines of Code (LOC):** This method involves counting the number of lines of code in a program. While this is a simple and easy-to-understand metric, it doesn't take into account the complexity of the code or the algorithms used.
2. **Cyclomatic Complexity:** This method measures the number of independent paths through a program's source code. It is calculated by constructing a control flow graph of the program and counting the number of regions in the graph.
3. **Halstead Metrics:** These metrics measure the complexity of a program based on the number of operators and operands used in the code. This method is based on the assumption that the more unique operators and operands used, the more complex the program is.
4. **Maintainability Index:** This metric is a composite measure of various factors that affect the maintainability of a program, including its size, complexity, and coupling.

5. **Function Points**: This method involves measuring the functionality provided by a program and using this as a basis for determining its complexity. This metric is particularly useful for large-scale software systems.

It's worth noting that these methods all have their own strengths and weaknesses, and none of them provides a perfect measure of complexity. However, they can be useful tools for evaluating the complexity of software systems and making decisions about testing, maintenance, and other aspects of software development.

Why do we use black box options?

Black box testing is a type of software testing in which the internal workings of the software are not known to the tester. Instead, the tester focuses on the inputs and outputs of the software, and tests it based on its functionality and behavior. Here are some reasons why we use black box testing:

1. **Simplicity**: Black box testing is often simpler and easier to perform than white box testing, because the tester doesn't need to have knowledge of the code or the internal workings of the software.
2. **Independence**: Black box testing is independent of the implementation of the software, which means that the tester can be someone other than the developer, and can provide an unbiased perspective on the software.
3. **Real-world testing**: Black box testing simulates real-world usage of the software, because it focuses on the inputs and outputs of the software, and how it behaves in response to different inputs.
4. **Comprehensive testing**: Black box testing can be used to test the entire system, including the user interface, input validation, data processing, and output generation.

Overall, black box testing is a useful technique for ensuring the functionality and quality of software, because it provides a way to test the software from the user's perspective, and to identify any issues or problems that may arise in real-world usage.

Regenerate response

What is a patent

A patent is a form of **intellectual property** that gives the inventor of a new and useful invention exclusive rights over it for a certain period of time. In other words, a patent is a government-granted license that allows the inventor to exclude others from making, using, or selling their invention for a set period, **usually 20 years** from the date of filing.

To be granted a patent, the invention must be new, useful, and non-obvious. This means that it must not have been publicly disclosed or available for public use before the filing of the patent application, and it must be different from existing inventions in the field. The inventor must also provide a detailed and specific description of the invention in the patent application, which must meet certain legal requirements.

Patents are often used to protect inventions in fields such as technology, medicine, and engineering, and can be a valuable asset for inventors, providing them with exclusive rights to their invention and the potential to license or sell it for profit.

What is the difference between patent and copyright?

Patents and copyrights are both forms of intellectual property protection, but they protect different things.

A patent is a government-granted license that gives an inventor the exclusive right to make, use, and sell their invention for a limited period of time, **usually 20 years** from the date of filing. Patents are typically used to protect new and useful inventions, processes, or machines. In order to be granted a patent, an invention must be novel, non-obvious, and useful.

Copyright, on the other hand, is a form of protection granted to the creators of original works of authorship, including literary, artistic, musical, and other intellectual works. Copyright gives the creator of a work exclusive rights to reproduce, distribute, and display the work, as well as the right to create derivative works based on the original. Copyright protection lasts for the lifetime of the creator plus a certain number of years after their death, **usually 70 years**.

In summary, patents protect new and useful inventions, while copyrights protect original works of authorship.

What do we learn about contract from Social Network?

The movie "The Social Network" is based on the real-life story of the founding of Facebook, and it portrays the legal disputes and conflicts that arose between the co-founders of the company, Mark Zuckerberg and Eduardo Saverin.

One of the key themes in the movie is the importance of having a clear and enforceable contract when starting a business with others. In the movie, Zuckerberg and Saverin initially agreed to a verbal contract in which they would split ownership of Facebook 50/50. However, when Facebook started to become successful and attract investors, Zuckerberg decided to change the terms of the agreement and dilute Saverin's ownership share.

This conflict led to a legal battle between Zuckerberg and Saverin, in which the validity and enforceability of the original contract were called into question. Ultimately, Saverin settled with Facebook for an undisclosed amount, but the case serves as a cautionary tale about the importance of having a written and enforceable contract when starting a business with others.

Some quizzes:

1. What is the purpose of measurement in software engineering?
 - a) To make sure the project is completed on time
 - b) To determine which team member should receive a bonus
 - c) To estimate the effort required for maintenance and testing
 - d) To ensure compliance with copyright and patent laws

Answer: c) To estimate the effort required for maintenance and testing

1. Which of the following is a white box complexity metric?
 - a) Lines of code
 - b) Planning Poker
 - c) Storey Points
 - d) Cyclomatic Complexity

Answer: d) Cyclomatic Complexity

1. What is Planning Poker?
 - a) A way to estimate the effort required for a software development task
 - b) A way to measure the number of lines of code in a software application
 - c) A method of protecting software from copyright infringement
 - d) A legal document used to establish ownership of a software application

Answer: a) A way to estimate the effort required for a software development task

1. Which of the following is a principle of data protection?
 - a) Process data for unlimited purposes
 - b) Keep data longer than necessary
 - c) Transfer data to countries without adequate protection
 - d) Process data in accordance with the data subject's rights

Answer: d) Process data in accordance with the data subject's rights

1. What is the difference between patent and copyright?
 - a) Patents protect the "stuff" created by someone else, while copyrights protect the "idea"
 - b) Patents last for 70 years after the author's death, while copyrights last for 20 years
 - c) Patents are granted by the government to stop others from exploiting an invention, while copyrights give the creator exclusive rights to their work
 - d) Patents are automatically owned, while copyrights must be granted

Answer: c) Patents are granted by the government to stop others from exploiting an invention, while copyrights give the creator exclusive rights to their work

More Quizzes

1. Which of the following is NOT a primary triple constraint of project management?

- a) Scope
- b) Time
- c) Cost
- d) Quality

2. What is the difference between a milestone and a task in project management?

- a) A milestone is a specific point in time when a task is completed, while a task is an action that needs to be completed in order to reach the milestone.
- b) A task is a specific point in time when a milestone is reached, while a milestone is an action that needs to be completed in order to reach the task.
- c) A milestone and a task are essentially the same thing, just different terms used by different project management methodologies.
- d) A milestone is a specific point in time when a task is started, while a task is an action that needs to be completed in order to reach the milestone.

3. What is a Gantt chart used for in project management?

- a) To track progress of the project against a timeline
- b) To assign tasks to team members
- c) To identify potential project risks
- d) To measure the complexity of the project

4. What is the critical path in project management?

- a) The sequence of tasks that must be completed in order to reach a specific milestone
- b) The longest sequence of tasks in a project that determines the earliest possible completion date
- c) The sequence of tasks that have the most potential to cause delays in the project
- d) The most complex sequence of tasks in a project

5. What is the purpose of using agile methodologies in project management?

- a) To deliver a product or service as quickly as possible
- b) To avoid using traditional project management methodologies
- c) To allow for frequent feedback and changes throughout the project
- d) To minimize the role of the project manager in the project

6. What is the difference between product backlog and sprint backlog in agile project management?

- a) Product backlog is a list of all the tasks that need to be done, while sprint backlog is a list of tasks for the current sprint.
- b) Sprint backlog is a list of all the tasks that need to be done, while product backlog is a list of tasks for the current sprint.
- c) Product backlog is a list of all the tasks that need to be done, while sprint backlog is a list of tasks for the next sprint.
- d) Sprint backlog is a list of all the tasks that need to be done, while product backlog is a list of tasks for the previous sprint.

7. What is the purpose of a risk management plan in project management?

- a) To identify potential risks to the project and develop a plan to mitigate them
- b) To assign responsibility for different aspects of the project to team members

- c) To track progress of the project against a timeline
- d) To measure the complexity of the project.

Answer:

1. Answer: d) Quality
2. Answer: a) A milestone is a specific point in time when a task is completed, while a task is an action that needs to be completed in order to reach the milestone.
3. Answer: a) To track progress of the project against a timeline.
4. Answer: b) The longest sequence of tasks in a project that determines the earliest possible completion date.
5. Answer: c) To allow for frequent feedback and changes throughout the project.
6. Answer: a) Product backlog is a list of all the tasks that need to be done, while sprint backlog is a list of tasks for the current sprint.
7. Answer: a) To identify potential risks to the project and develop a plan to mitigate them.

6b-Agile/planning.pocker