1. 产品的功能性：概括而言，即产品必须完成的工作（或任务），例如：办公软件必须完成文档编辑、存储和打印等；
   产品的可用性：概括而言，意指交互式产品易于人类用户所使用，例如：易于学习、易于理解、易于操作。
   说明功能性和可用性是两个不同的概念
   功能性：系统必须提供的服务，缺乏这些服务用户无法完成其工作
   可用性：如何将这些服务提供给用户，以便于用户使用
2. 交互设计的中心问题：按照需要支持或扩充的用户活动，选择适当的方法和技术，以优化用户和系统、环境或产品之间的交互。
3. 什么是交互设计：设计支持人们日常工作和生活的交互式产品。
4. 交互设计原则上包括 4 项活动，这些活动通常相互之间具有联系，且本身是迭代过程
   1） 识别用户需要并建立需求
   2） 开发满足需求的候选设计方案
   3） 构建用于交流和评估的交互式原型
   4） 评估整个过程的设计结果
5. 过程需采用用户为中心设计（UCD）的方法
   1） 一种有效的方法是用户参与式设计
   2） 对原型进行评估是保证设计可用性的核心环节
   3） 理解用户的活动是提取相关需求的重要步骤
6. 基本的设计原理包括可视性、反馈、限制、映射、一致性和启发性等（ Donald A. Norman 详细描述了这些原理）
   1） 可视性：系统呈现下一步可执行操作，以及系统状态改变
   2） 反馈：操作结果（系统状态改变）的可视性
   3） 限制：即对用户在特定时刻可执行操作的限制（物理、逻辑和文化限制）
   4） 一致性：在 UI 设计中，类似的任务应使用类似的操作和表示
   5） 启示性：事物可能通过其形状或其他属性建议可以对他们作什么
7. Nielsen, *et al* 提出了 10 个主要的可用性原理
   1） 系统状态的可视性：适时提供适当反馈，便于用户了解系统的状态
   2） 系统应与真实世界相符合：使用用户的语言而非面向系统的概念和术语
   3） 用户的控制和自主权：提供醒目的出口，便于用户退出异常状况
   4） 一致性和标准化：避免用户无法确定不同的词汇、情景、动作是否表示相同的含义
   5） 帮助用户识别、诊断和修复错误：使用简明语言，描述问题的性质并提供解决的方法
   6） 预防出错：应尽可能地避免错误的发生
   7） 依赖识别而非记忆：使得对象、动作和选项清晰可见
   8） 使用的灵活性和有效性：为熟练用户提供快捷键
   9） 最小化设计：避免使用无关或极少使用的信息
   10）帮助及文档：提供易于检索、便于学习的帮助文档

1. 一个 Web 浏览器改进设计实例

相关情节（scenario）

设计者假定需要开发更好、功能更强的浏览器

调查并分析现有浏览器在使用方面的问题

发现：用户无法有效使用书签功能，因为其限制太强

原因：层次化的目录结构不能有效管理大量网址

结论：需开发存储和检索网址的新方法

在分析现有书签功能时，进一步明确了相关问题的假定

若无意中将网址放入错误的目录，有可能丢失该网址

网址不易在不同的目录之间移动

系统没有醒目提示如何将多个网址不同的目录之间移动

系统没有醒目提示如何将已存储的网址重新排序

据此，为了更有效的支持，提出了有关用户需要的假定

若改进书签功能，则用户认为更为实用且会更多地使用

用户需要灵活的网址管理方式，以支持参考和转发的目的

事实上，理解问题空间通常称为识别和指定使用上下文

2. 什么是概念模型（conceptual model）（名词解释）

简单说，即所感兴趣事物的概念以及它们之间关系的描述

在软件工程中，概念模型通常用来描述系统应做什么

例如：环境中的什么信息和关系系统应维护（领域模型）

环境中的什么活动系统应当自动化（过程模型）

旨在与管理者或专家讨论系统的特征并指定需求

在 HCI 中，概念模型通常用于描述用户界面应做什么

例如：系统能够让用户看见并操作什么信息

系统能够让用户执行什么活动

3. 概念模型的分类基于活动的概念模型和基于对象的概念模型

基于活动的概念模型是以用户活动为基础。基于对象的概念模型以物理世界隐喻（物理现象、人工制品）为基础。后者侧重于具体表示用户可视、可操作的信息及操作方式，信息的表示通常是对物理世界的模拟

4. 最常用的人机交互方式有：

指令型（instructing）用户指示系统做某事

会话型（conversing）用户与系统的实时信息交换

操作型（manipulating）用户直接操作虚拟空间中的对象

探索型（exploring）用户穿越虚拟（或物理）空间

5. 会话：类似于人们之间的会话，用户与系统之间的双向信息通信，用户与系统交互旨在获取所需的信息，而不是命令系统做某事。

6. 界面隐喻(比拟)：隐喻是一种短语，它通过陈述另一种事物来描述一件事物，而不使用词语就可以对它进行比较。（名词解释）

7. 隐喻和类比：

隐喻：通过人们已理解的概念来解释新的概念

类比：人们通过两者之间类似性的比较来理解和解决问题

隐喻和类比在交互设计中的用法

概念化某种具体的交互技术：如将系统作为一种工具

作为概念模型，并实例化为界面的一部分，如桌面隐喻

作为描述计算技术的一种方式，如 Internet highway
作为操作名，如 cut and paste
作为培训教材的内容，如比较字处理器和打字机

8. 界面隐喻的优点：减少学习使用计算机所需的努力，人们可以使用实际领域（而非计算）知识来描述系统功能

界面隐喻的争论

过分强调忠实于物理上下文可能忽视新方法和技术的使用
可能导致文化和逻辑上的冲突
物理限制使得某些活动在所使用的隐喻上无法有效完成
满足物理限制可能违背设计规则（如一致性）
用户难以理解界面隐喻之外的系统功能
采用不良设计作为隐喻
限制了设计者的创新思维

9. 交互范型指的是我们在构思交互设计时的主导思想或思考方式，其目的是指导设计人员询问什么问题。（名词解释）一些相关的交互范型，以满足不同问题的需要：

1) 无处不在计算
2) 普适计算（应属于无处不在计算范畴）
3) 可穿戴计算
4) 实物用户界面、增强现实、混合现实、虚拟现实
5) 上下文感知计算
6) 工作日世界（应属于计算机支持的协同工作）

10. 为何需要迭代：无论何类问题，均需要进行设计迭代，对设计问题，需要按照概念模型改进界面和交互设计，对建模问题，需要进一步提取用户信息，求精假设或提出新的用户需要假设，并据此改进概念模型。迭代不可避免。设计不可能一次成功
11. 在设计和评估概念模型的初步原型时，需处理许多问题

1) 界面上的信息表示方式及交互方式
2) 需要使用何种类型的多媒体（如语音或动画）
3) 对用户的操作需要提供何种类型的反馈
4) 需要使用何种类型的 I/O 设备
5) 是否需要为用户提供界面代理
6) 是否需要特殊的输入设备，还是表示为虚拟操作
7) 需要提供何种类型的帮助

12. 物理设计与概念设计的关系

对这些问题作出决策的同时需要考虑物理设计的问题

信息表示：
采用何种具体的交互形式或技术（如菜单、表格等）
如何组织界面的布局（如窗口、对话框、菜单的排列）

反馈表示
反馈信息的具体形式（如图标、音频等）
反馈信息的机制（如词法、语法或语义反馈）

多媒体表示
媒体应用的具体形式（如动画和语音的结合）

在考虑物理设计决策时通常会发现某些概念决策问题

概念决策与实际情况不符（如交互方式不支持用户活动）

技术或物理限制不支持概念决策

此时需要返回上一阶段改进概念设计

1. 认知：认知指的是与 knowing 相关的能力、行为和过程，如何感知物理刺激？如注意、知觉等；如何认识自我、他人以及环境？如意识、记忆等；如何利用这些知识进行思维？如推理、问题求解等。
2. 感知指的是人们如何使用感官从环境中获取信息，并把它转变成对物品、事件、声音和味觉的体验。
3. Norman 按照思维方式将认知划分为两种模式，经验式认知（experiential cognition mode），思考式认知（reflective cognition mode）
4. 概念框架：即支撑一个学科体系的基本理论结构，包括概念、原理、模型等。三种方法：心理（mental）模型、信息处理（使用比拟和类比把大脑的工作方式概念化）、外部认知。
5.
6. 外部认知：人们需要同各种外部表示相交互，并且使用它们来创建信息。外部表示与物理工具相结合大大增强了人们的认知能力。包括：
   1) 外部化以减少记忆负载
   2) 减少计算负载
   3) 标注与认知跟踪
7. 感知机制：是通过感官训练所获得的对外部表示的意识简单说，即对感知范围内的环境状态的理解，并据此来调整自己的行为。
8.

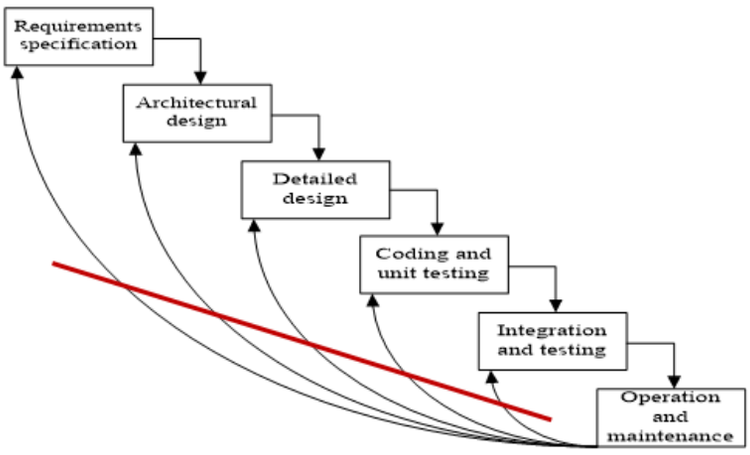
1. 协作：是指在目标实施过程中，部门与部门之间、个人与个人之间的协调与配合。
2. 主要的社会机制类型
   1) 会话机制（conversational mechanism）
   2) 协调机制（coordination mechanism）
   3) 感知机制（awareness mechanism）
3. 同步通信（支持实时的对话，允许用户只通过声音或者文字进行交谈，不同程度地支持非语言的通信），异步通信（支持非同步的远程通信，异步通信不要求严格的时间次序，它由一方发起通信过程，通信者能够决定何时恢复。）
4. 协调机制：需要考虑在执行任务时，应如何与其他人交互。为此，我们就需要引入协调机制。包括：
   1) 言语和非言语的通信
   2) 时间表、规定和约定
   3) 共享外部表示
5. 以下从社会学和人类学的角度，引入了有关社会性的概念框架：
   1) 言语动作理论
   2) 分布式认知理论

1. 情感因素：包括神经、发声、肌肉、心跳、呼吸等的反应
2. 富有表现力的界面方式：
   1） 使用富含表情的图标和其他图形元素来表达感情状态
   2） 使用动态图标
   3） 使用动画
   4） 使用语言信息
   5） 使用不同声音表示不同的动作和事件
3. 改善用户挫折感：
   1） 花招百出，不适用借口去掩饰
   2） 错误提示，说明引起错误的原因和解决问题的方法
   3） 用户负担过重，软件升级应是毫不费力的自动过程
   4） 外观，界面设计应做到简单、优雅、让人一目了然，并且符合可用性设计原则、图形设计原则以及人类工程学原则
4. 什么是代理，代理种类

   人类代理在物理世界中代表着客户的利益，例如：旅行社代理、税务代理等软件代理在虚拟世界中代表着用户的利益，电子邮件代理、搜索代理、上网助手等。

   合成角色、动画代理、情感代理、形象化的对话界面代理

1. 交互设计的内容：交互设计需要考虑产品应用、目标应用域和相关的实际限制，从而提出设计方案。而且，我们需要提出多个候选方案，让用户进行评估。评估若要成功，就必须把设计表示为合适的形式，以便于用户理解。
2. 交互设计的 4 个基本活动：
   1) 识别用户需要并建立需求
   2) 设计侯选方案
   3) 建立设计的交互式版本
   4) 评估设计
3. 交互设计过程的 3 个关键特征
   1) 以用户为中心的设计
   2) 特定的可用性标准
   3) 迭代是必然，设计不可能一次成功
4. 一种典型的参与者分类，前三类即甲方，后一类为乙方
   主要方：即最终用户
   第二方：间接使用系统的人员
   第三方：他们不属于以上两类，但受到系统成败的影响
   提供方：包含在系统设计、开发和维护中的人员
5. 设计决策的依据
   可用性规约由 6 项组成，包括
      1) 属性：错误操作的可撤销性
      2) 度量概念：撤销一个错误操作序列
      3) 度量方法：撤销所需的用户操作的个数
      4) 当前水平：所有当前产品都不允许这样的撤销
      5) 计划水平：最多使用两步操作
      6) 最坏情况：与错误操作同样多的操作
      7) 最好情况：一步操作
   如何按照可用性目标来指定度量是一个复杂问题
      1) 有效性：对任务合适的支持？任务覆盖程度、可用信息
      2) 效率：所使用的资源？时间、操作个数
      3) 实用性：所有功能都是必要的？
      4) 安全性：可靠地执行？出错的几率、撤销方式
6. 5 个生命周期模型，3 个来自软件工程，2 个来自 HCI，
   一个简单的交互设计过程模型（用户为中心的设计，迭代，评估依赖于可用性目标）。
   瀑布模型：
   优点：
      1) 为项目提供了按阶段划分的检查点。
      2) 当前一阶段完成后，您只需要去关注后续阶段。
      3) 可在迭代模型中应用瀑布模型。
   缺点：
      1) 在项目各个阶段之间极少有反馈。
      2) 只有在项目生命周期的后期才能看到结果。
      3) 通过过多的强制完成日期和里程碑来跟踪各个项目阶段。



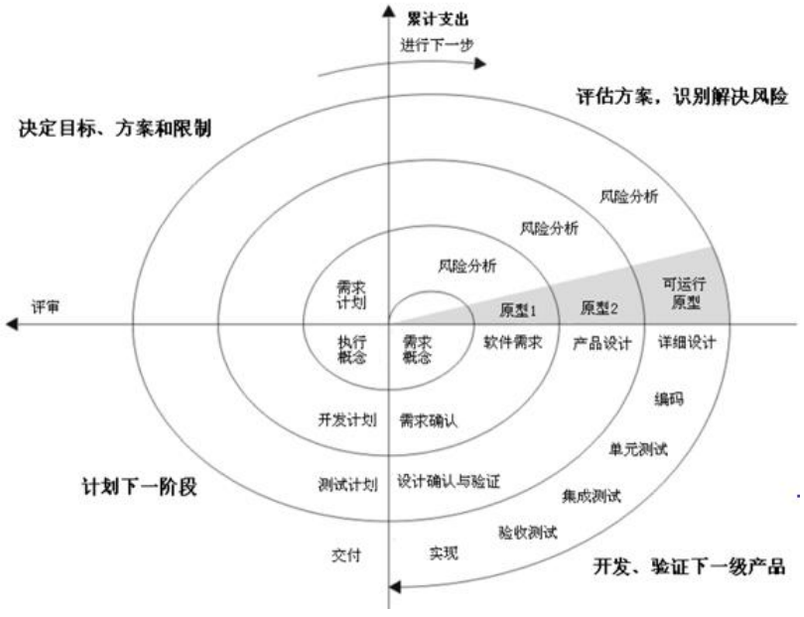
螺旋模型：

优点：

1) 设计上的灵活性,可以在项目的各个阶段进行变更。
2) 以小的分段来构建大型系统,使成本计算变得简单容易。
3) 客户始终参与每个阶段的开发,保证了项目不偏离正确方向以及项目的可控性。
4) 随着项目推进,客户始终掌握项目的最新信息 ， 从而他或她能够和管理层有效地交互。
5) 客户认可这种公司内部的开发方式带来的良好的沟通和高质量的产品。

缺点：

1) 采用螺旋模型需要具有相当丰富的风险评估经验和专门知识，在风险较大的项目开发中，如果未能够及时标识风险势必造成重大损失。
2) 过多的迭代次数会增加开发成本，延迟提交时间。



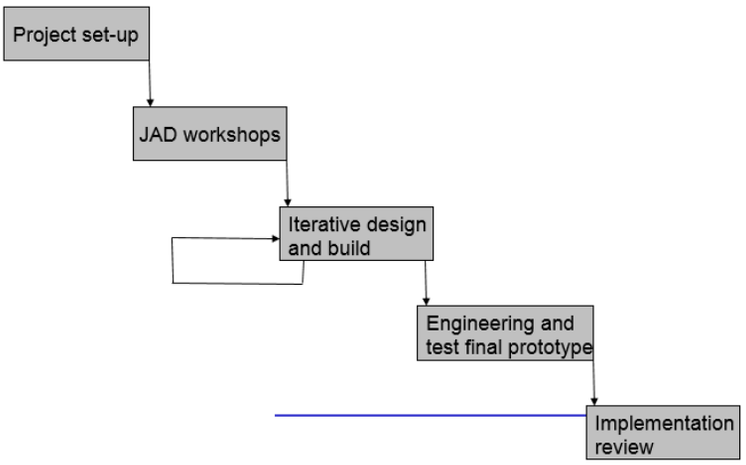
快速应用开发模型：

优点：

1) 克服瀑布模型的缺点，减少由于软件需求不明确带来的开发风险。

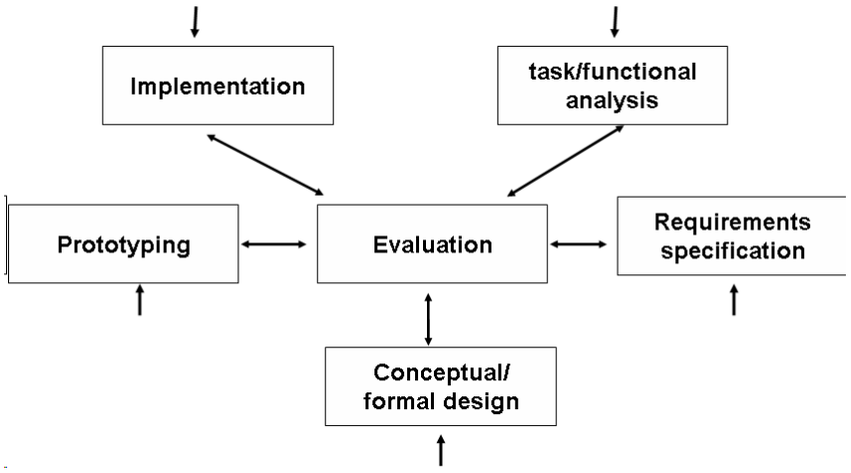
缺点

1) 所选用的开发技术和工具不一定符合主流的发展；
2) 快速建立起的系统结构加上连续的修改可能会导致产品质量低下；



星形模型表示了有经验设计者的一种自然行为的抽象

1) 强调评估在对于产生一个较好的解的重要性
2) 未明确指定各个活动的生命周期
3) 不能用来规划和管理整个开发过程



可用性工程生命周期模型

1) 体现了可用性工程的整体概念
2) 强调如何将可用性目标加入软件工程方法（如 OOSE）
3) 与作者提出的简单交互设计过程有类似之处，同样包括需求、设计、评估和原型阶段
4) 提议使用风格指南作为可用性目标的表示机制
5) 其灵活性允许裁剪为适合小项目开发的轻量级过程模型

1. 需求：有关目标产品的陈述或规约，需求应描述系统做什么，但不是系统如何做。
2. 需求活动：识别需要并建立需求
3. 避免产品的需求错误一个解决方法：以用户为中心的开发方法。
4. 数据搜集技术：问卷调查、访谈、专题组和研讨会、自然观察、研究文档
5. 解释和分析应在收集过程后立即展开，目的：提取、组织和描述需求
6. 用户为中心的设计需要一个面向用户的数据解释
7. 不同的任务描述方法
   1) 情节，一种非形式的叙事性描述（又称用户故事(User Story)）
   2) 用例，按照 UML，用例是一个系统执行的动作序列（包括替换序列）集合，对行为者产生可观察的结果
   3) 基本用例，在一个抽象层次上指定用户和系统的交互，描述用户想要做什么，以及系统响应，避免不成熟设计的相关技术假定，允许设计者考虑不同的交互设计方案

   区别（给一个软件如何区分）：
   1) 情节：识别需要（当前）
      帮助建立需求（未来，设计前）
      说明设计（未来，设计中及设计后）
   2) 用例：帮助建立交互需求（设计前）
      描述系统功能需求（设计中）
   3) 基本用例：描述交互需求（交互设计的内容）
8. 层次化任务分析（会画图）

1. 设计活动包括两个方面：

   概念设计：概念建模，描述用户如何使用产品（界面呈现内容、任务流程、交互方式、交互范型及隐喻等），从需求至初步设计。

   物理设计：界面的细节设计，如布局、结构、交互技术，具体化。（菜单设计，图标设计，屏幕设计，信息显示）
2. 特征：迭代，用户参与
3. 什么是原型：对产品概念的形象化和具体化，是设计师构想的体现，不是产品，是产品的一种近似或受限表示
4. 为什么建立原型：评估设计，发现问题，与文档相比，更容易观察和与原型进行交互，有效地交流，设计思想，支持在候选方案中做出选择
5. 高保真模型和低保真模型优缺点：

类型	优点	缺点
低保真原型	• 开发成本低 • 可评估多个设计概念 • 是有用的交流设施 • 可解决屏幕布局问题 • 适用于识别市场需求 • 可证明设计概念	• 可捕获的错误有限 • 不能作为详细规范用于指导编程 • 受制作介质的影响 • 对可用性测试的作用有限 • 不便于说明过程流
高保真原型	• 包含完整功能 • 完全可交互 • 用户驱动的 • 明确定义了过程流 • 适用于详细设计和测试 • 可获得最终产品的使用体验 • 可作为详细规范 • 可作为销售的支持工具	• 开发成本高 • 制作耗时 • 不能有效证明设计概念 • 不适合于收集需求

低保真模型（卡片式原型）：用简单的方式来快速、近似地表示产品概念。

注重内容结构，成本低，迭代，开发阶段。

高保真模型（软件原型）：构建的原型更接近于最终产品。

注重评估测试，时间长，维护阶段。

垂直原型和水平原型。

6. 开发概念模型的三个方面
   1) 交互方式对用户的活动提供支持
   2) 界面隐喻使用户易于理解产品的使用
   3) 交互范型为具体化概念模型提供了指导
7. 屏幕设计涉及两个问题：如何把任务分解为多个屏幕显示；如何设计个别的屏幕显示。

1. 用户参与设计的重要性：

为了充分考虑用户的需要，最好让实际用户参与开发。采用用户参与式开发开有两个与系统功能无关的原因：期望管理（使得用户对新产品有切合实际的期望）；所有权（使得用户感到对产品具有所有权）。

2. 用户参与的不同形式

1） 作为设计组成员

全职参与：可提供持续的输入，也可深入理解系统及原理

兼职参与：不能提供连贯的输入，且会有较大的工作压力

短期地轮流参与：可能造成不一致的输入

长期参与：可提供一致的输入，但可能失去与其他用户的联系

2） 参与专题讨论或项目会议

3） 如果具有大量用户，则需采用折衷的方法

各个目标用户组的代表全职参与设计活动

其他用户参与专题讨论、评估或其他的数据搜集活动

3. 什么是以用户为中心的方法

用户为中心表示了一种设计思想，而非纯技术方面。产品开发以实际用户的需要和目标驱动，而不仅是技术。良好的设计系统应能利用用户的能力，帮助他们克服限制。系统应对用户的任务完成提供支持，而不是限制

4. 迭代设计：设计过程通常是设计—测试—评估—再设计的过程

5. 以下指出以研究用户和他们任务为基础的五项具体原理

1） 开发以用户任务和目标来驱动

2） 系统需对用户的行为和使用上下文提供支持

3） 捕捉用户的特征并将其应用于设计

4） 在整个开发过程中咨询用户，并认真考虑他们的意见

5） 应在使用的上下文中做出所有的设计决策

6. 理解用户的工作：现场研究法（如何研究）

7. 上下文相关设计的步骤：上下文质询、工作建模、合并、工作再设计、用户环境设计、纸模及用户测试、付诸实践

8. 上下文质询依赖于四项主要原理

上下文原理：观察用户的工作场景并深入理解发生的事件

合作关系：设计者应与用户密切合作，以达到理解的目的

解释原理：观察结果的解释应由用户和设计者结合完成

焦点原理：调查应集中在与系统设计相关的问题上

9. 上下文询问与访谈：

上下文询问可以视为一种“无计划”的访谈，但具有更广的含义。在“无计划的访谈”中，访谈者不必预先确定一组问题，可根据具体情形而发挥。“上下文询问”是在受访人的工作环境中进行的，受访人课继续工作，他继承了其他一些数据搜集技术（观察）。当然，也可以考虑对不同的受访人使用不同的访谈技术。

10. 上下文质询与现场研究的区别

1） 时间简短，上下文访谈仅需 2-3 小时，而后者需数周或数月

2） 重点明确，集中，而后者需更广泛地研究工作情景

3） 只是观察和提问，并不实际参与用户的工作

4） 以系统设计为目的，而后者可能并不具有特定地开发计划