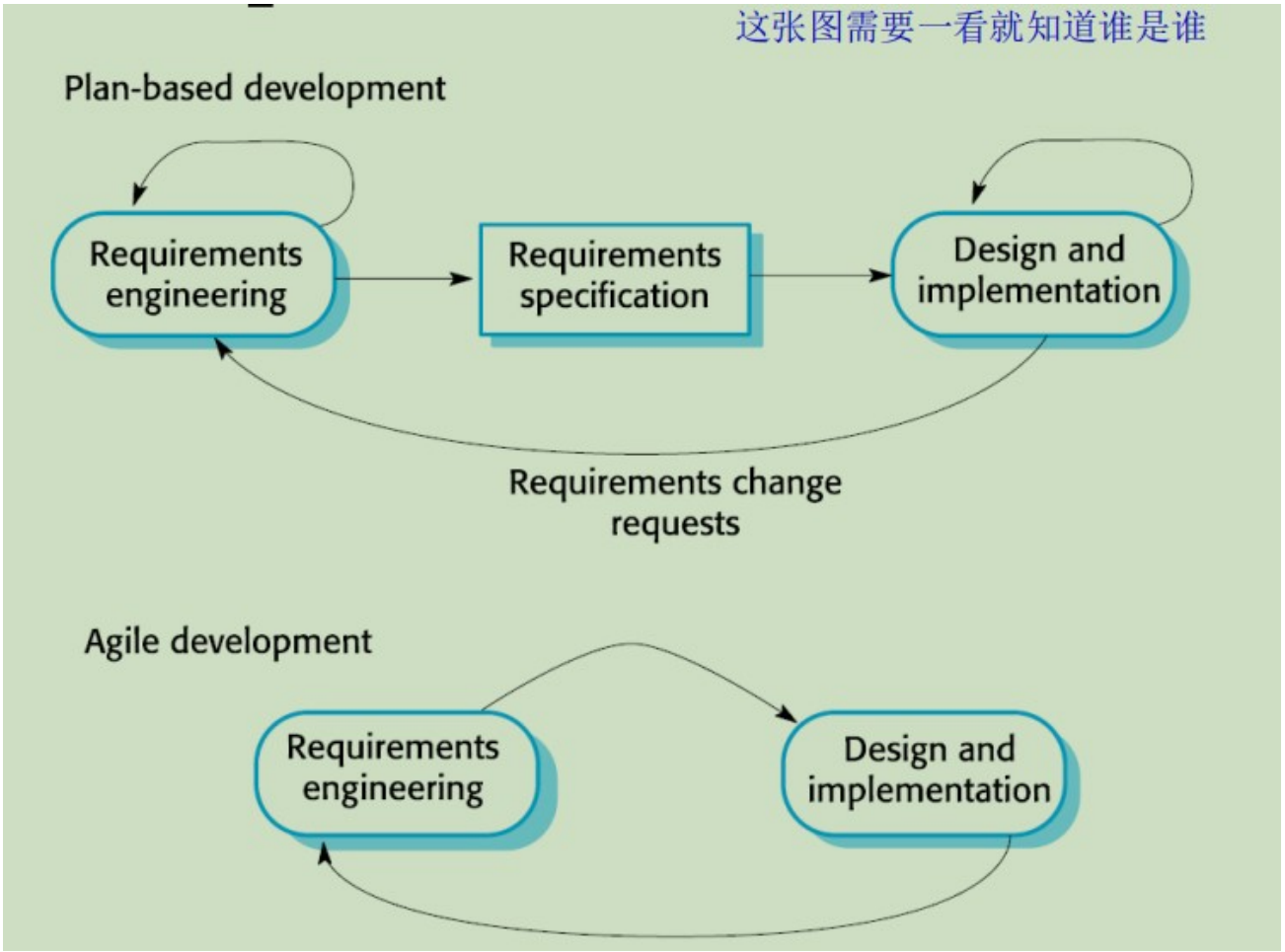
L3 Agile Software Development

- *The system is developed as a series of versions/increments with stakeholders involved in version specification and evaluation*
- Plan-driven development
  - *a plan-driven approach is based around separate development stages with the outputs to be produced at each of these stages planned in advance*
  - *not necessarily waterfall model-plan driven, incremental development is possible*
- Agile development
  - *specification, design, implmentation and testing are interleaved*
  - *the outputs from the development process are decided through a process of negotiation during the software development process*



Agile methods

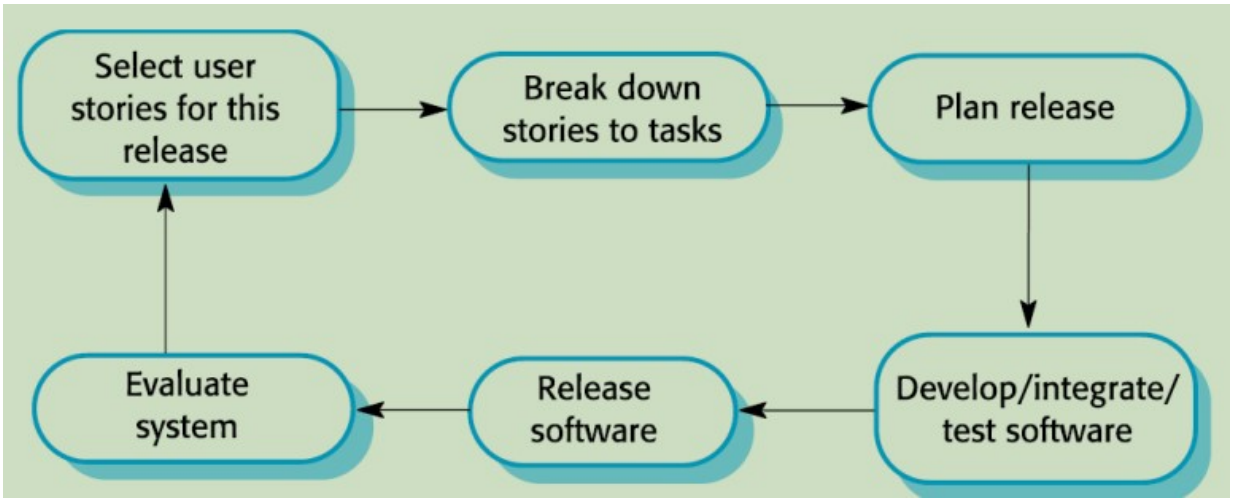
- reduce overheads in SW process; be able to respond quickly to changing requirements without excessive rework

- customer involvement; incremental delivery; people not process; embrace change; maintain simplicity

Agile development techniques

Extreme Programming (XP)

- *A very influential agile method*



Influential XP practices

- **User stories for specification**
  - User requirements are expressed as user stories or scenarios
  - These are written on cards and development team break them down into implementation tasks. These tasks are the basis of schedule and cost estimates
  - The customer choose the stories for inclusion in the next release based on their priorities and the schedule estimates
- **Refactoring**
- 传统的SE is design for change. 但是XP认为这不值得因为changes can't be reliable anticipated预测. 它认为constant code improvement (refactoring) to make changes easier
- **Test-first development**
  - 在代码编写前明确测试可以帮助需求理解；客户参与其中；
  - test automation
    - tests are written as executable components before the task is implemented
  - problems
    - 比起测试，程序员更喜欢编程
    - 有些测试很难增量地编写

        - 很难判断一组测试地完整度

- **Pair programming**
    - 程序员结对坐在一起，互相审查review

Agile project management

- The principal responsibility of SW project managers
    - *to manage the project so that the SW is delivered on time and within the planned budget for the project*
- The standard approach to project management is plan-driven
    - *manager draw up a plan for the project*
- Agile project management requires a different approach adapted to
    - *incremental development*
    - *the practices used in agile methods*

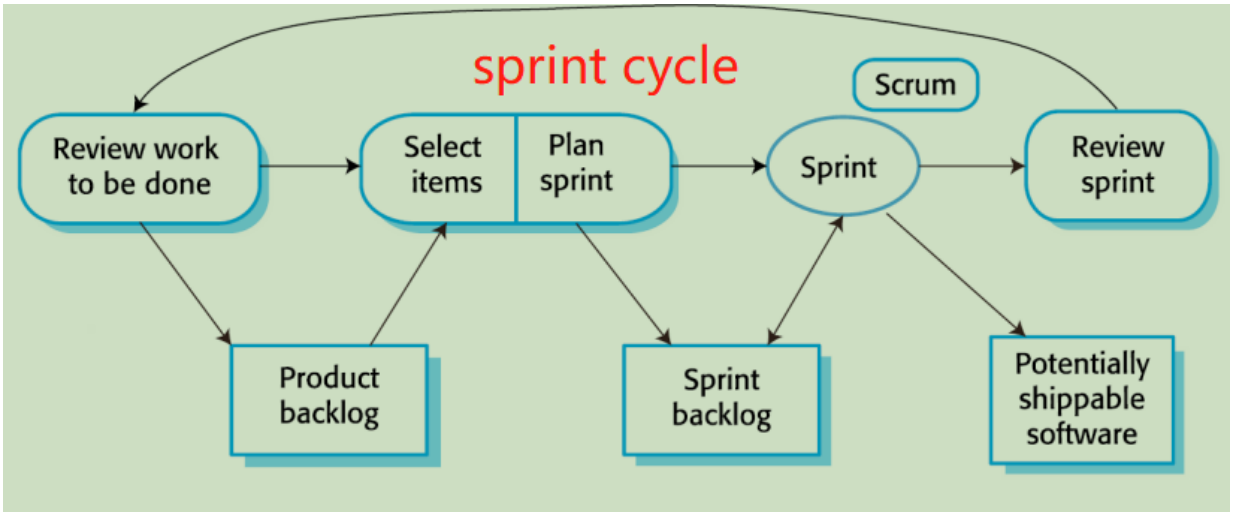
Scrum

- an agile method that focus on managing iterative development
- Three phases
    - *initial phase: outline planning phase. establish the general objectives*
    - *followed by a series of **sprint cycle**, where each cycle develops an incremental of the system*
    - *closure phase: wraps up the project, completes required documentation*
- Benefits:
    - *product is broken dwon into a set of manageable and understandable chunks*
    - *unstable requirements don't hold up process*
    - 团队沟通提高
    - 客户按时得到交付，并且可以得到feedback
    - 客户和开发者之间的信任

Scrum sprint cycle

- fixed length, 2-4weeks. starting point: product backlog

- Scrum master: *protect the development team from external distractions*



Scaling agile methods

- Scaling up agile methods involves changing these to cope with larger, longer projects where there are multiple development teams, perhaps working in different locations
- **Scaling up**: using agile methods for developing large software systems that can't be developed by a small team
- **Scaling out**: how agile methods can be introduced across a large organization with many years of software development experience
- agile fundamentals: *flexible planning, frequent system releases, continuous integration, test-driven development and good team communications*

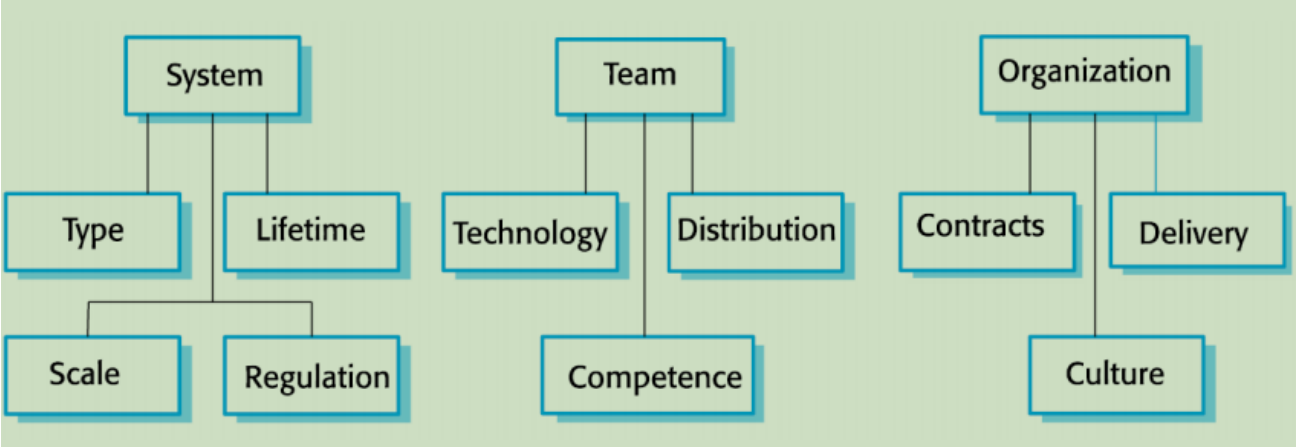
pratical problems with agile methods

- contractual issues合同问题
  - the informality of agile development is incompatible with the legal approach to contract definition that is commonly used in large companies
- Maintenance
  - agile are most appropriate for new SW development rather than maintenance
  - key problems
    - *lack of product documentation*
    - *kepping customers involved in the development process*
    - *maintaining the continuity of the development team*
  - 依赖于开发团队必须知道和理解要做什么，对于long-life系统来说，员工可能会辞职
- design for small co-located teams

Agile and plan-driven methods

- deciding on the balance of plan-driven and agile processes:
  - *is it important to have a very detailed specification and design before moving to implementation? If so, you probably need to use a plan-driven approach*
  - *is an increnmental delivery strategy, where you deliver the software to customers and get rapid feedback from them, realistic? If so, consider using agile methods.*
  - *how large is the system that is being developed? agile methods are most effective when the system can be developed with a small co-located team who can communicate informally. This way not be possible for large systems that require larger development teams so a plan-driven approach may have to be used*

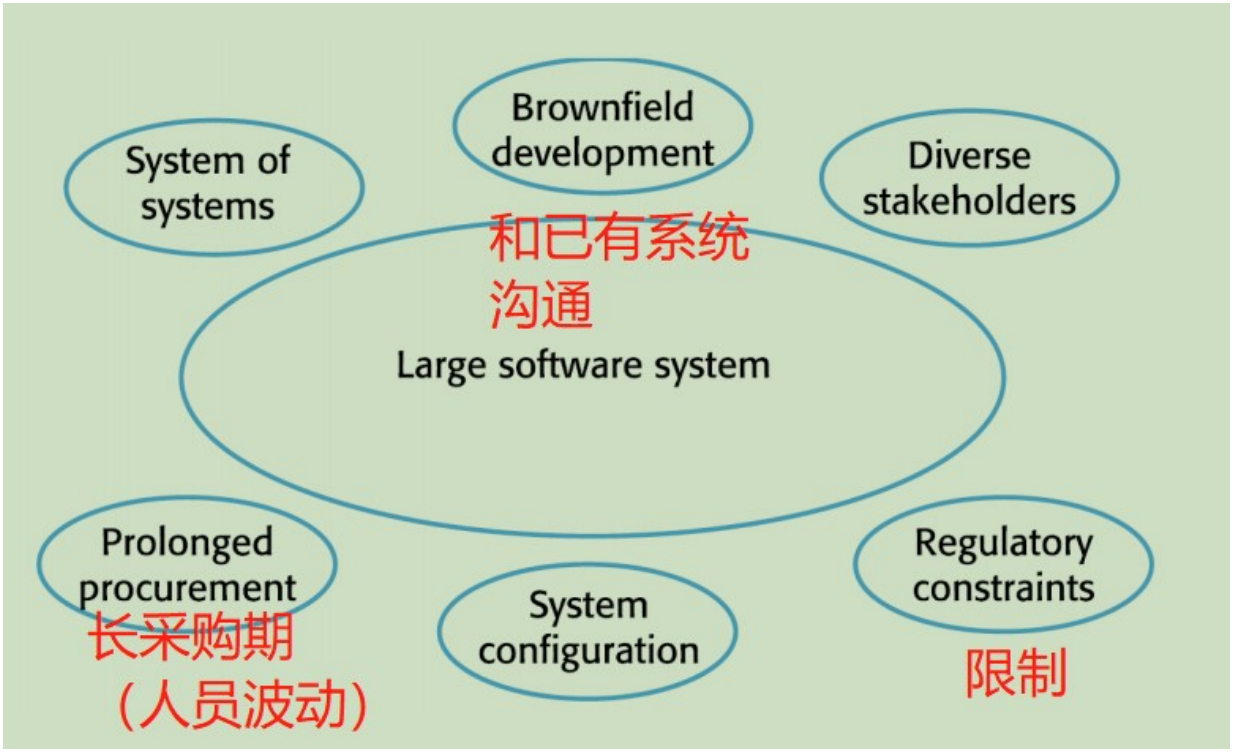
- Agile and plan-based factors
  - long-lifetime systems require documentation to communicate the intentions of the system developers to the support team



Agile methods for large systems

- 团队人员分散，沟通很少
- 大型系统分散为多个部分，和已存在的系统有交互。not really lend themslves to 灵活性和增量开发
- where several systems are integrated to create a system, a significant fraction of the development is concerned with system configuration rather than original code

development



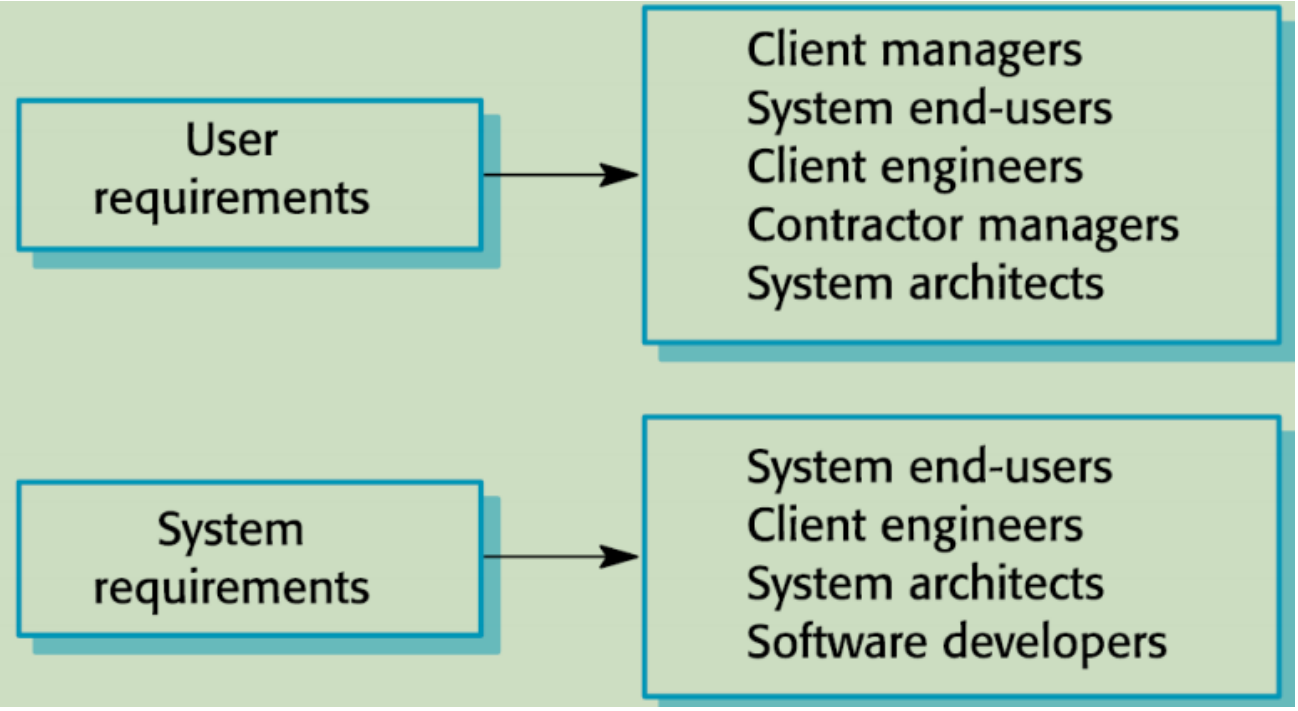
Key points

- Agile methods are incremental development methods that focus on rapid SW development, frequent SW releases, reducing process overheads by minimizing documentation and producing high-quality code
  - *agile development practices include*
  - *user stories for system specification*
  - *frequent releases of the software*
  - *continuous software improvement*
  - *test-first development*
  - *customer participation in the development team*
- Scrum is an agile method that provides a project management framework
- Many practical development methods are a mixture of plan-based and agile development
- Scaling agile methods for large systems is difficult
  - *large systems need up-front design and some documentation and organizational practice may conflict with the informality of agile approaches*

L4 Requirement Engineering

- 抽象的需求和具体的定义，都可以称作requirements

- user requirements
  - *written for customers* Statements in natural language plus diagrams of the services thesystem provides and its operational constriaints. 用自然语言和图表描述，说明系统必须提供哪些服务、运行系统要受到哪些约束
- System requirements
  - a structured document setting out detailed descriptions of the systems functions, services, and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor. 详细说明系统将要提供的服务以及系统受到的约束。精确描述软件的功能，系统买方和软件开发者签订合同的重要内容。



Functional and non-functional requirement

Functional requirements

- describing the behavior of the system, 描述系统提供的服务
- ambiguous模糊不清的 requirements may be interpreted in diffrent ways by developers and users
- 原则上，需求应该完整且一致，但实际上由于复杂性，不可能

Non-functional requirements

- 判断一个系统的操作的标准，而不是对系统行为的描述. define properties and Constraints.
- Non-functional classifications

- product requirements: *product must behave in a particular way eg. speed*
- organisational requirements: *organizational policies and procedures*
- external requirements: *external to the system and its development process*

Domain requirements

- Constraints on the system from the domain of operation

Requirements engineering process

- RE根据情况有很大不同，但有一些generic activities. 实践中都是交错的
  - *Requirements elicitation* 需求诱导
  - *Requirements analysis*
  - *Requirements validation*
  - *Requirements management*

Requirements elicitation

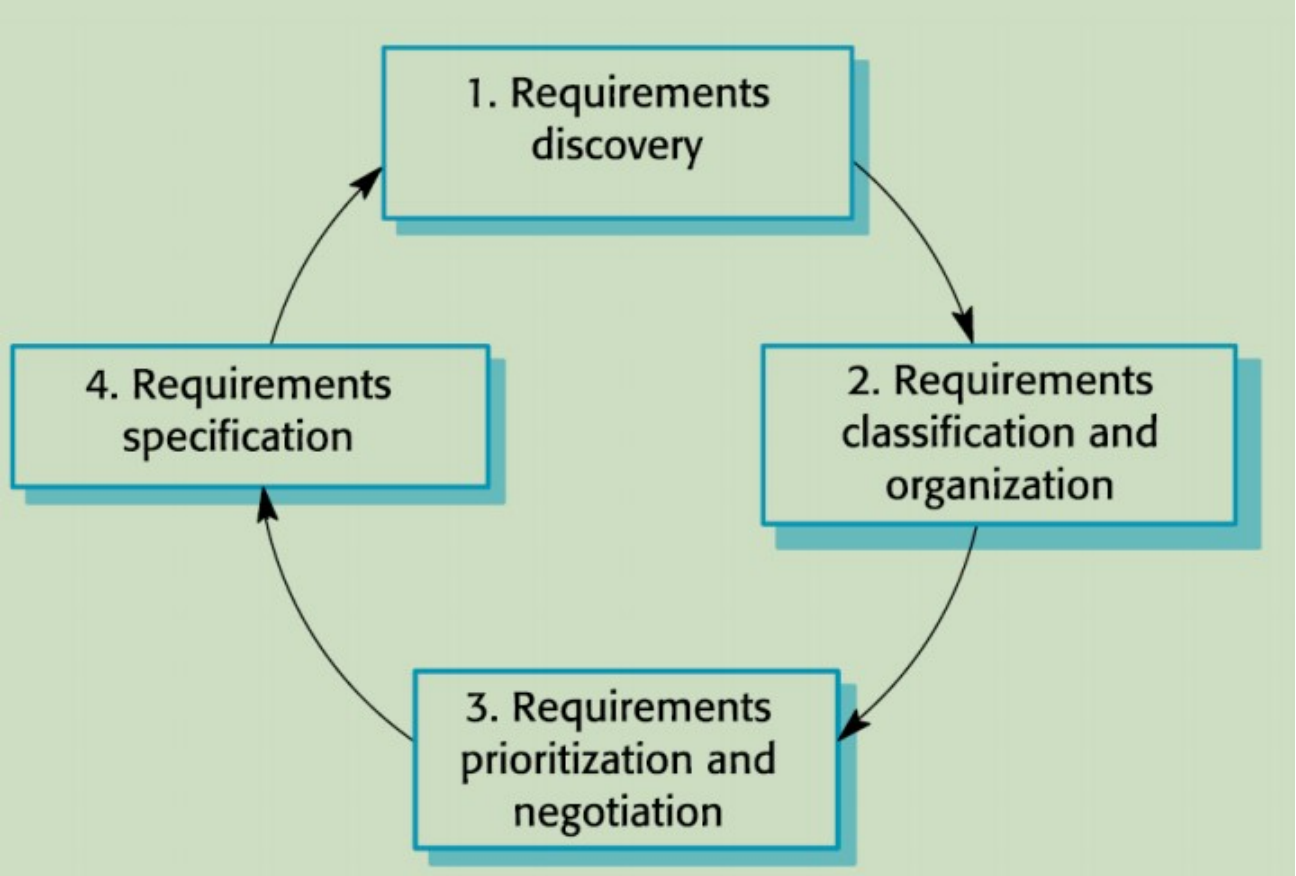
- Sometimes called requirement elicitation or requirement discovery
- May involve end-users, managers, engineering involved in maintenance, domain experts, trade unions, etc. These are called *stakeholders*.
- 软件工程师与stakeholder一起find out应用领域，系统提供的服务，要求的性能，约束
- Stages include
  - *Requirements discovery*
  - *Requirements classification and organization*
  - *Requirements prioritization and negotiation*
  - *Requirements specification*
- problems: stakeholder用自己语言表述，不知道自己真正需求，需求冲突， 需求有新的变化

Process activities

- Requirements discovery
  - *Interacting with stakeholders to discover their requirements*
  - *Domain requirements are also discovered at this stage*
  - Interview: not good for understanding domain requirements
  - Stories and scenarios (使用场景): real-life examples

        - 启动情况的说明；正常事件流； 所出现的错误描述；其他并发活动；场景结束状态描述

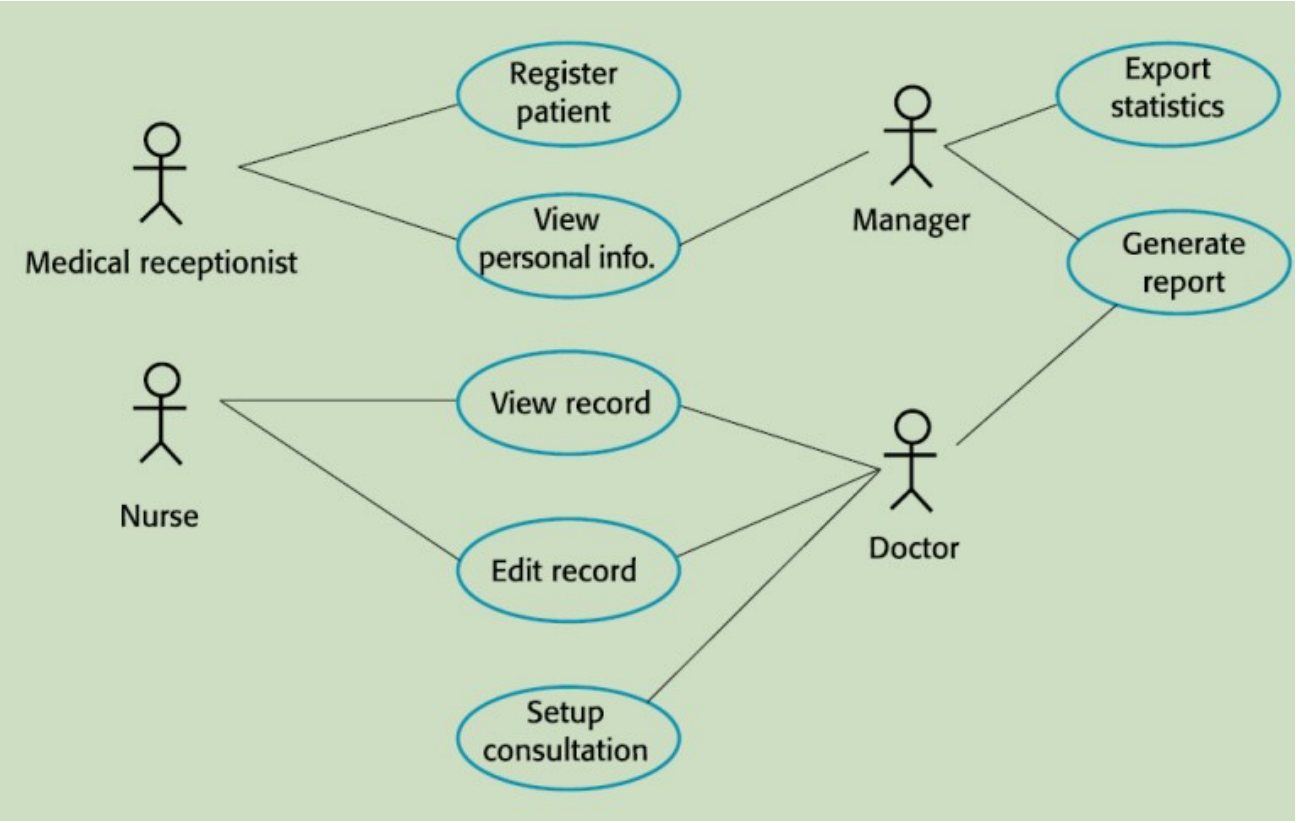
- Requirements classification and organisation
    - *Groups related requirements and organises them into coherent clusters*
- Prioritisation and negotiation
    - *Prioritising requirements and resolving requirements conflicts*
- Requirements specification
    - Requirements are documented and input into the next round of the spiral



Requirements specification

- The process of writing the user requirements(要让没有任何背景的客户明白) and system requirements in a requirements document
- ways of writing a system requirements specification
    - natural language; structured natrual language; design description language(like 编程语言); graphical notations; mathematical specifications
- In principle, *requirements* should state **what the system should do**, and the *design* should describe **how it does this**

Use cases



- software requirements documents: NOT design document. 对系统开发人员所需要的东西的官方声明

Requirements validation

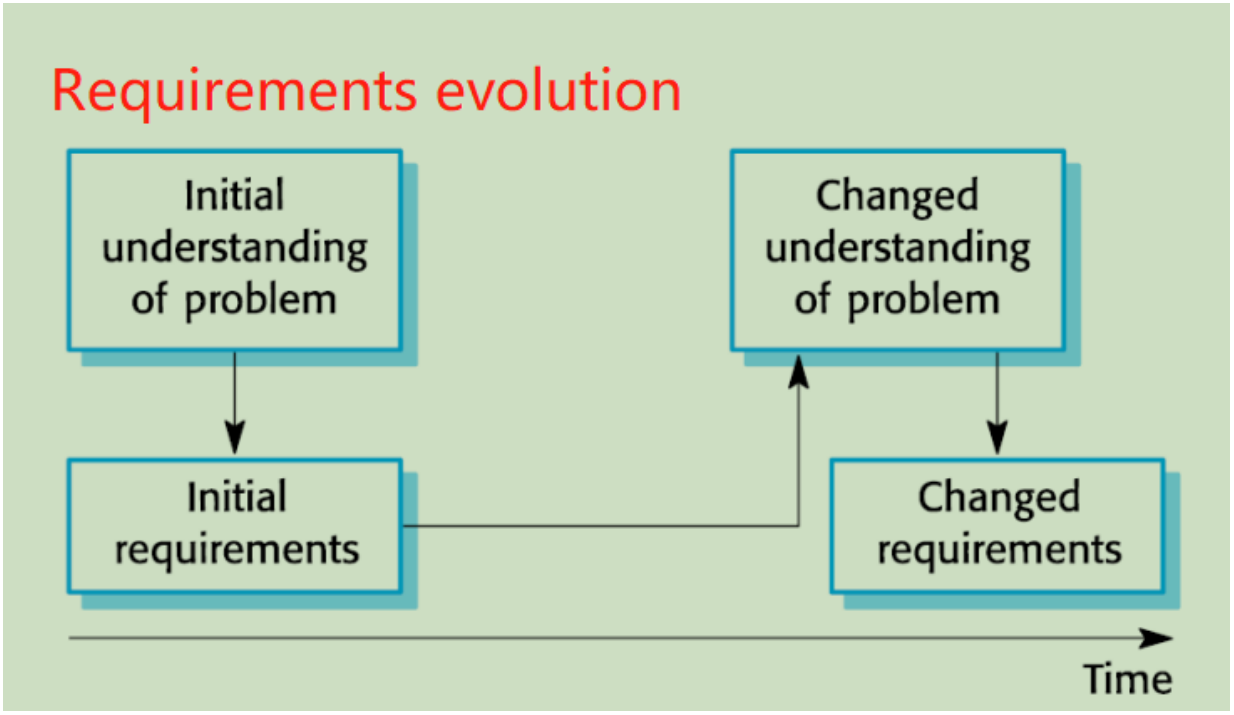
- Concerned with demonstrating that the requirements define the system that the customer really wants

Requirements checking

- Validity: * Does the system provide functions which best support the customer's needs?*
- Consistency: * Are there any requirements conflicts?*
- Completeness: *Are all functions required by the customer included?*
- Realism: *Can the requirements be implemented given available budget and technology?*
- Verifiability: *Can the requirements be checked?*

Requirements change

- **Requirement management** is the process of managing changing requirements during the requirements engineering process and system development.



Key Points

- Requirements for a software system set out what the system should do and define constraints on its operation and implementation
- Functional requirements are statements of the services that the system must provide or are descriptions of how some computations must be carries out
- Non-functional requirements often constrain the system being developed and the development process being used
- The requirements engineering process is an iterative process that includes requirements elicitation, specification and validation
- Requirements elicitation is an iterative process that can be represented as a spiral of activities - requirements discovery, requirements classification and organization, requirements negotiation and requirements documentation
- Requirements specification is the process of formally documenting the user and system requirements and creating a software requirements document
- Requirements validation is the process of checking the requirements for validity, consistency, completeness, realism and verifiability
- Requirements management is the process of managing and controlling these changes.