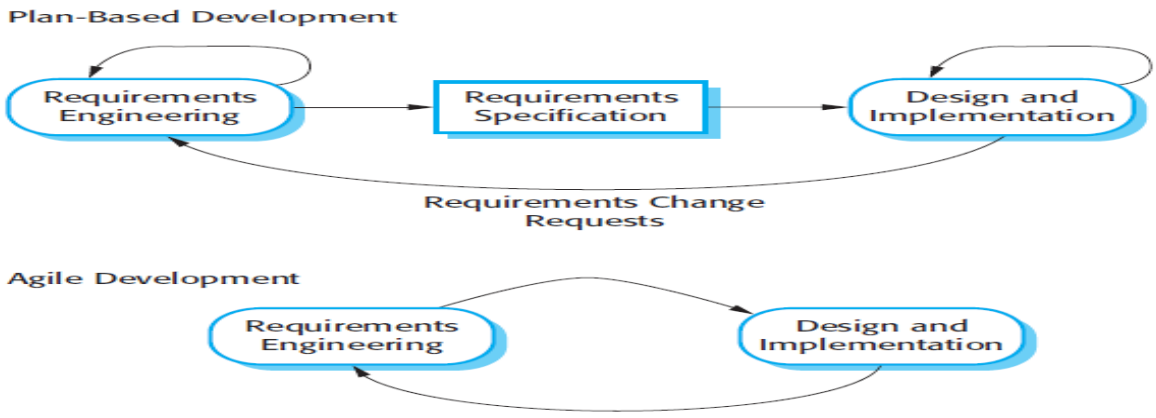
Plan-driven and agile development

- Agile approaches to software development consider design and implementation to be the central activities in the software process. They incorporate other activities, such as requirements elicitation استنباط/توضيح and testing, into design and implementation. By contrast, a plan-driven approach to software engineering identifies separate stages in the software process with outputs associated with each stage. The outputs from one stage are used as a basis for planning the following process activity. Figure below shows the distinctions between plan-driven and agile approaches to system specification. In a plan-driven approach, iteration occurs within activities with formal documents (Every project manager should create a small core set of formal documents defining the project objectives, how they are to be achieved, who is going to achieve them, when they are going to be achieved, and how much they are going to cost. These documents may also reveal inconsistencies that are otherwise hard to see) used to communicate between stages of the process. For example, the requirements will evolve تطور and, ultimately, a requirements specification will be produced. This is then an input to the design and implementation process. In an agile approach, iteration occurs across activities. Therefore, the requirements and the design are developed together, rather than separately. A plan-driven software process not necessarily waterfall model – plan-driven, incremental development and delivery is possible. It is perfectly feasible to allocate requirements and plan the design and development phase as a series of increments. An agile process is not inevitably code-focused and it may produce some design documentation.



Plan-driven and agile specification

Most projects include elements of plan-driven and agile processes. Deciding the balance depends on Technical, human, organizational issues:

1. Is it important to have a very detailed specification and design before moving to implementation? If so, software engineer probably needs to use a plan-driven approach.
2. Is an incremental delivery strategy, where software engineer delivers the software to customers and get rapid feedback from them, realistic? If so, consider using agile methods.
3. 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 may not be possible for large systems that require larger development teams so a plan-driven approach may have to be used.
4. What type of system is being developed? Plan-driven approaches may be required for systems that require a lot of analysis before implementation (e.g. real-time system with complex timing requirements).
5. What is the expected system lifetime? Long-lifetime systems may require more design documentation to communicate the original intentions of the system developers to the support team.
6. What technologies are available to support system development? Agile methods rely on good tools to keep track of an evolving design
7. How is the development team organized? If the development team is distributed or if part of the development is being outsourced, then software engineer may need to develop design documents to communicate across the development teams.
8. Are there cultural or organizational issues that may affect the system development? Traditional engineering organizations have a culture of plan-based development, as this is the norm/standard in engineering.
9. How good are the designers and programmers in the development team? It is sometimes argued that agile methods require higher skill levels than plan-based approaches in which programmers simply translate a detailed design into code
10. Is the system subject to external regulation? If a system has to be approved by an external regulator then software engineer will probably be required to produce detailed documentation as part of the system safety case.

Requirements Engineering

Requirements Engineering (RE) is the process of finding out/discover, analyzing, documenting and checking the services and constraints.

Requirements are a statements range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification.

There are two types of requirements:

1. User requirements: Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.
2. System requirements: A structured document setting out يحددdetailed descriptions of the system's functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.

The next figure illustrates the distinction between user and system requirements for a Mental Health Care -Patient Management System (MHC-PMS).

User Requirement Definition

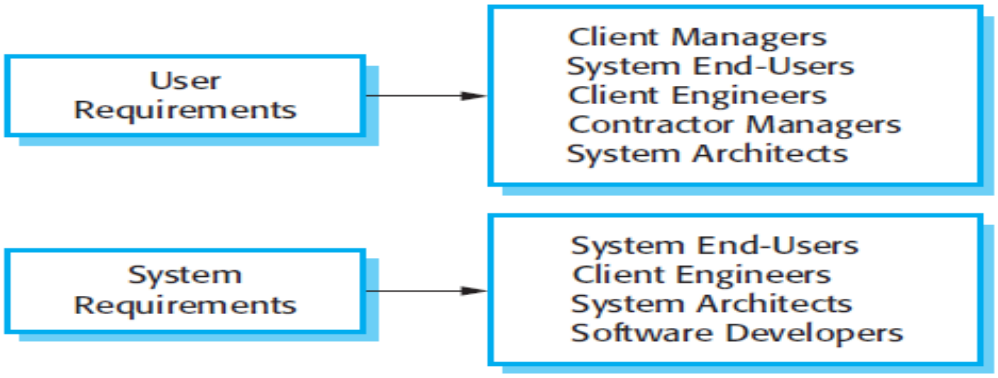
1. The MHC-PMS shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

System Requirements Specification

1.1 On the last working day of each month, a summary of the drugs prescribed, their cost, and the prescribing clinics shall be generated.
1.2 The system shall automatically generate the report for printing after 17.30 on the last working day of the month.
1.3 A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed, and the total cost of the prescribed drugs.
1.4 If drugs are available in different dose units (e.g., 10 mg, 20 mg) separate reports shall be created for each dose unit.
1.5 Access to all cost reports shall be restricted to authorized users listed on a management access control list.

The SWE/RE needs to write requirements at different levels of detail because different readers use them in different ways. Next figure shows possible readers of the user and system requirements. The readers of the user requirements are not usually concerned with how the system will be implemented and may be managers who are not interested in the detailed facilities of the system. The readers of the system requirements need to know more precisely what the system will do because they are concerned with how it will support the business

processes or because they are involved in the system implementation.



Readers of different types of requirements specification

Functional and non-functional requirements

Software system requirements are often classified as functional requirements or nonfunctional requirements:

1. Functional requirements: Statements of services the system should provide how the system should react to particular inputs and how the system should behave in particular situations. Also, it is may state what the system should not do.
2. Non-functional requirements: Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc. it is often apply to the system as a whole rather than individual features or services.

Functional requirements

It is describe functionality or system services. These requirements depend on the type of software, expected users and the general approach taken by the organization when writing requirements. Functional user requirements may be high-level statements of what the system should do. Functional system requirements should describe the system services in detail.

Functional system requirements vary from general requirements covering what the system should do to very specific requirements reflecting local ways of working or an organization's existing systems. For example, here are examples of functional requirements for the MHC-PMS system, used to maintain information about patients receiving treatment

for mental health الصحة العقلية problems:

1. A user shall be able to search the appointments lists for all clinics عيادات.
2. The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.
3. Each staff member using the system shall be uniquely identified by his or her 8-digit employee number.

Imprecision الغموض in the requirements specification occurs when requirements are not precisely stated and ambiguous requirements may be interpreted in different ways by developers and users. For example, the first example requirement for the MHC-PMS states that a user shall be able to search the appointments lists for all clinics. The rationale for this requirement is that patients with mental health problems are sometimes confused. They may have an appointment at one clinic but actually go to a different clinic. If they have an appointment, they will be recorded as having attended, irrespective of بصَرْف النظر عنthe clinic. So, the term 'search' in requirement 1

- User intention/ medical staff – search for a patient name across all appointments in all clinics;
- System developer interpretation تَفْسِير– search for a patient name in an individual clinic. User chooses clinic then search.

The functional requirements specification of a system should be both complete and consistent. Completeness means that all services required by the user should be defined. Consistency means that requirements should not have contradictory متناقض definitions. In practice, for large, complex systems, it is practically impossible to achieve requirements consistency and completeness. One reason for this is that it is easy to make mistakes and omissions اغفال/سهو when writing specifications for complex systems. Another reason is that there are many stakeholders in a large system. A stakeholder is a person or role that is affected by the system in some way. Stakeholders have different and often inconsistent needs. These inconsistencies may not be obvious when the requirements are first specified, so inconsistent requirements are included in the specification.