What dose software engineering focuses on?

ما الذى تهتم به هندسة البرمجيات؟

the cost-effective of high-quality software systems.

تهتم بانتاج أنظمة برمجية عالية الجودة لها أعلى كفأة وصممت بأقل سعر ممكن

Why software is abstract and intangible?

لماذا تعد البرمجيات قائمة على نفسها و غير معتمدة على البيئة المحيطة؟

what are the reasons of having extremely complex or difficult to under-stand software?

ماهى أسباب وجود برمجيات معقدة او صعبة الفهم؟

1- not constrained by materials

غير مقيدة بالمواد

2- not governed by physical laws

القوانين الفيزيائية لا تأثر بها

3- not governed by manufacturing processes

اليات المصنع لا تتحكم بها (مثال برامج عمل الغسالة هى نفس الاله ولكن اكثر من استخدام)

What is software crisis?

ماهى أزمة البرمجيات؟

problem resulted directly from the introduction of new computer hardware based on integrated circuits

مشكلة نتجت من ظهور اجزاء كمبيوتر قائمة على ال integrated circuits

Why informal software development wasn't good enough?

لماذا تطوير الانظمة بالمعلوماتية كان غير كافى؟

1- Major projects toke years for production

المشاريع الكبيرة تأخذ سنوات عديدة للانتاجها

2- software cost higher than predicted

تكلفة انتاج البرمجيات ترتفع عن التكلفة المتوقعة

3- unreliable

لايمكننا الاعتماد على هذه الانظمة لانها دائما تحتاج للتطوير ولديها قصور وأخطاء

4- difficult to maintain and poor performance

صعب التطوير عليها وأدائها منخفض

Can we have a single ideal approach? Why or why not?

هل يمكن ان يكون لدينا هيكل مثالى واحد لتطوير البرمجيات؟ لماذا ولماذا لا؟

No - Cause the wide Varity of the systems types and the organizations uses these systems requires different approaches of software development

لا – لتعددية انواع الانظمة والمؤسسات التى تستخدم هذه الانظمة

What is a software?

مامعنى برمجيات؟

software is a computer program and his associated documentations which may be developed for particular customer or developed for general market

هو برنامج كمبيوتر مع التوثيق المرفق له ويطور اما لعميل او للاستخدام العام بالاسواق الالكترونية

What is a software system? ما معنى نظام برمجيات؟

consists of number of separate programs, configuration files and associated documentations

يتكون من مجموعة من البامج المنفصلة وملفات اعداداتهم والتوثيقات الخاصة بكلا منهم

User documentation **توثيق للمستخدم**	Explains how to use the system and the website where you can download latest information تشرح للمستخدم كيف يستخدم النظام والموقع الذى يقوم بتحميل المستجدات الخاصة بالنظام منه
System documentation **توثيق للنظام**	Describe the structure of the system تصف هيكل النظام
Configuration files **ملفات الاعدادات**	Used to set up these programs تقوم باعداد بدء البنامج بما حدده المستخدم او الاعدادات الافتراضية

What are the two fundamental types of software products?

ما هما النوعان الاساسيان من المنتجات البرمجية؟

1- Generic products برمجيات للاستخدام العام

Generic products are a stand-alone systems developed by organizations and sold on open market such as **database** and **word processors**

هى برمجيات قائمة بذاتها طورت من منظمة ما ويتم بيعها فى الاسواق الالكترونية **مثل** برمجيات قواعد البيانات و معالجات النصوص

2- Customized (bespoke) products برمجيات للاستخدام الخاص

systems which are developed by a software contractor for a particular customer such as **control system for electronic devices** and **air traffic control system**

هى أنظمة تم تطويرها من قبل مقاول برمجيات لصالح مستخدم معين **مثل** أنظمة تحكم لأجهزة كهربائية و خطوط الطيران

What is software engineering? ماهى هندسة البرمجيات؟

software engineering is a sub science (discipline) from System Engineering which is concerned with all aspects of software production starting from system specifications until maintaining the system after the user receives it.

هى فرع من فروع هندسة النظم والتى تهتم بكل نواحى انتاج البرمجيات

What is the difference between software engineering and computer science? ماهو الفرق بين هندسة البرمجيات وعلم الحاسب ؟

Computer science is concerned with the fundamentals of software system

علم الحاسب يهتم بأساسيات انتاج النظام البرمجى

software engineering is concerned with delivering useful software and system practicality هندسة البرمجيات تهتم بانتاج برمجيات لها أهمية وتهتم بعمليتها فى يد المستخدم

What is the difference between software engineering and system engineering? ماهو الفرق بين هندسة البرمجيات و هندسة النظام ؟

software engineering is a part of the system engineering process

هندسة البرمجيات جزء من عملية تصميم النظام

system engineering concerned with all aspects of the development and evolution of complex computer-based systems and hardware development

هندسة النظام تهتم بكل نواحى تطوير النظام وتطور الانظمة المعقدة القائمة على الحاسب و تطور الاجزاء المادية (المعدات)

What is a software process? ماهى سير عملية (الية) البرنامج؟

software process is a set of activities their goal is the development and evolution of a software product هى مجموعة من الانشطة التى تهدف الى انتاج وتطوير منتج برمجى

list the fundamental software process activities?

اذكر الانشطة (الخطوات) الرئيسية فى سير عملية (اليه) البرنامج؟

software specification **مواصفات البرنامج**	Customers and engineers defines the constrains and how the software will be produced المهندسين والعملاء يحددوا كيف سيتم انتاج البرنامج
software development **تطوير البرنامج**	How software designed and programmed to meet the specifications كيف يتم تصميم وبرمجة البرنامج ليحقق المواصفات المطلوبة
software validation **التحقق من صحة البرنامج**	How software is checked to check if it has the Customer requirements كي يتم التحقق من البرنامج لضمان احتوائه على احتياجات العميل
software evolution **تطور البرنامج**	How the software is modified to adapt the changes required by market and customers كيف يتم تعديل البرنامج ليتماشى مع التعديلات المطلوبة من العميل او السوق

What is a software process model? ماهو نموذج سير عملية البرنامج؟

software process model is a simplified representation or description of a software process to present it in a specific perspective.

هى طريقة مبسطة للتمثيل او وصف سير عملية البرنامج من وجهة نظر محددة (المثال كل شركة ليها اسلوب لعرض نفس النظام)

Types of software process models? ماهى انواع نماذج سير عملية البرنامج؟

1- workflow model نموذج تدفق العمل

shows the sequence of activities, inputs, outputs and dependencies in a process

يعرض متسلسلة الانشطة والمدخلات والمخرجات واعتماد كل منهم على الاخر

Note: activities here represent human actions الانشطة هنا تمثل اعمال اشخاص

2- dataflow or activity model نموذج تدفق البيانات

represent the process as a set of activities each one of them carries out some data transformation it represents how inputs transformed to outputs

يمثل العملية على انها مجموعة من الخطوات وكل خطوة تقوم بعملت حويل لبعض البيانات من مدخلات لمخرجات

Note: activities here represent transformation carried out by people or by computers الانشطة هنا تمثل تحويل للبيانات بواسطة الانسان او الكمبيوتر

3- role/action model نموذج الادوار/ الاعمال

represent the roles of the people involved in the software process and activities they are responsible for تمثل دور الاشخاص المعنيين بسير عملية البرنامج والانشطة المسؤليين عنها

briefly mention Generic process models or paradigms of software development? اذكر باختصارنماذج تطوير البرمجيات؟

1- waterfall approach هيكل الشلال

takes the process fundamentals activities and represent them as separate process phases like (requirements specification, software design, implementation, testing ... etc.) when a step finished it **signs-off** and next step **initialized**

يعرض اللانشطة الرئيسية لعملية كمراحل عمليات منفصلة وعند انتهاء مرحلة تشير الى انها انتهت حتى يتم بدء المرحلة التالية

2- Evolutionary (iterative) development تطوير (بالتكرار) بالتطور

implement an initial system which is rapidly developed from very abstract specification then it will be refined by customer input to produce a system satisfies the customer needs

3- component-based software engineering (CBSE) هندسة البرمجيات القائمة على المكونات

This technique assumes that parts of the system already exist and the system developments process focuses on integrating these parts rather than developing them from the scratch.

هذا الاسلوب يعتمد على ان اجزاء النظام موجوده وعملية تطوير البرنامج تركز على دمج هذه الاجزاء بدلا من تطويرها من الاساس

What are the costs of software engineering? ماهى تكاليف هندسة البرمجيات؟

For generic software 60% of costs for development and the other 40% are testing (evolution) costs. تكلفة الانتاج 60% وتكلفة الختبار والتطور 40%

For Customized software evolution costs may exceed development costs.

تكلفة التطور قد تتعدى تكلفة التطوير

Costs vary depending on the type of system being developed and the requirements of system attributes such as performance and system reliability.

تختلف نسبة التكلفة اعتمادا على 1- نوع النظام الذى يتم تطويره 2- متطلبات النظام كالاداء ومدى الاعتماد عليه

Distribution of costs depends on the development model that is used.

كيفية توزيع التكلفة تعتمد على النموذج المستخدم فى التطوير

What are software engineering methods? ماهى اساليب هندسة البرمجيات؟

Structured approaches to software development which include System models, notations, rules, design advice and process guidance whose aim is to facilitate the production of high-quality software in a cost-effective way.

هى هياكل ممنهجة لتطوير البرمجيات تهدف الى تسهيل عملية انتاج برمجيات عالية الجودة وبأقل تكلفة

Model description **وصف النموذج**	Description of graphical models which should be produced وصف النموذج الذى سستم انتاجه بشكل رسومى
Rules **القواعد**	Constrains applies to the system model تطبيق القيود على النموذج الذى صمم
Recommendations **التوصيات**	Advice on good design practice نصائح عن التصميم الافضل
Process guidance **ارشادات العملية**	Which activities to follow ماهى الخطوة او النشاط التالى

List software engineering methods اذكر امثلة لأساليب هندسة البرمجيات

Mention some software development methodologies

اذكر بعض منهجيات تطوير البرمجيات

1- structured analysis

2- Jackson system development (JSD)

3- Unified Modeling Language (UML)

What is CASE (Computer-Aided Software Engineering)?

ماهى أنظمة هندسة البرمجيات بمساعدة الحاسب؟

Software systems which provide automated support for software process activities

هى أنظمة تقدم دعم ألى لأنشطة سير عملية البرمجيات

Upper-CASE are tools to support early activities of requirements and design

هى أدوات تدعم المراحل الاولية للانشطة كالمتطلبات والتصميم

Lower-CASE are tools to support later activities such as programming, debugging and testing

هى أدوات تدعم المراحل الاخيرة للانشطة كالبرمجة و تصحيح الاكواد واختبار النظام

Mention some of the supported programs in CASE Systems

اذكر بعض من البرامج المدعومة فى انظمة CASE

Requirements analysis, System modeling, debugging and testing

Mention some of the associated CASE technologies for methods

اذكر بعض من التقنيات الضمنية لأنظمة CASE للأساليب

Editors **المحررات**	For notations used in methods للرموز التى تستخدم فى لاساليب
Analysis modules **وحدات التحليل**	Check system model according to methods rules فحص نموذج النظام طبقا لقواعد الاساليب
Report generator **مولد التقارير**	Help while creating system documentation تساعد عند انشاء توثيق للنظام
Code generator **مولد الكواد**	Automatically generate source code from system model and some process guidance for the engineer تولد اجزاء من الكود اتوماتيكيا بناء على نموذج النظام واشادات العملية للمهندس

What are the attributes (characteristic) of good software?

ماهى خصائص او عوامل البرمجيات الجيدة؟

Maintainability **القابلية للصيانة**	Software should be written in a way that allow it to evolve to meet new requirements of customers due to changes in business environment يجب كتابة الكود بصيغة تسمح لله بالتطور ليحقق متطلبات العملاء المتغيرة طبقا لتغسرات بيئة العمل
Dependability **الاعتمادية**	Includes reliability, security and safety and no physical or economic damage during system failure تتضمن الاعتماد على النظام فى اتمام شيء و امن وامان النظام وضمان عدم وجود خسارة مالية او مادية عند حدوث فشل فى النظام
Efficiency **الكفأة**	Efficient use of system resources such as memory and processor Includes responsiveness, processing time and memory utilization الاستخدام الامثل لموارد النظام كوحدة الذاكرة والمعالج وتتضمن استجابة النظام فى اى وقت لعملية ما وقت معالجة كل عملية والاستخدام الامثل للذاكرة
Usability **سهولة الاستخدام**	Software should have an adequate documentation and appropriate user interface البرنامج يجب ان يتواجد له توثيق شامل و واجهه مناسبة

What are the key challenges facing software engineering?

ماهى التحديات الرئيسية التى تواجه هندسة البرمجيات؟

Increase of heterogeneity **تعددية الاجهزة و الانظمة او (زيادة عدم التجانس)**	A challenge to develop techniques for building dependable software and flexible enough to cope with heterogeneity التحدى هو تطوير تقنيات تسمح بانتاج برمجيات تعتمد على نفسها (مستقلة عن الالة التى تعمل عليها) ومرنة بشكل كافى لتلحق بتعددية الاجهزة والانظمة
Delivery time **وقت الانتاج**	A challenge to shortening the delivery time for large and complex systems without reducing quality التحدى هو التقليل من الوقت المطلوب لانتاج برمجيات كبية ومعقدة مع عدم تقليل كفائتها
Trustworthy software **برمجيات جديرة بالثقة**	A challenge to develop techniques that demonstrate that the software can be trusted by the customer التحدى هو تطوير اساليب تعرض للمستهلك مدى امان النظام وانه يمكنه الوثوق به

Mention some of the software engineering profession responsibilities?

اذكر بعض مسؤليات مهنة هندسة البرمجيات

Confidentiality **السرية**	software engineer should normally respect confidentiality of his employer or client مهندي البرمجيات يجب ان يحترم خصوصية وسرية العميل او رئيس عمله
Competence **المهارة**	software engineer shouldn't accept work outside his level of competence

	لايجب ان يقبل مهندس البرمجيات بالعمل خارج المستوى الذى يمهر فيه
Intellectual property rights **حقوق الملكية الفكرية**	software engineer should be aware of the local laws governing the use of intellectual property such as patents and copy rights to protect his employer or client rights مهندس البرمجيات يجب ان يكون على دراية بالقوانين المحلية التى تحكم استخدام الملكية الفكرية كبرائات الاختراع او حقوق الطبع لحماية العميل او ئيس عمله
Computer misuse **الاستخدام الخاطىء للكمبيوتر**	software engineer shouldn't use his technical skills to misuse other people's computers لايجب ان يستخدم مهندس البرمجيات مهاراته التقنية لسوء استخدام اجهزة الاشخاص الاخرين

Mention the principles (Code of Ethics) any software engineer must have?

اذكر بعض مبادىء (كود الاخلاقيات) التى يجب ان تكون لدى كل مهندس برمجيات

Public **العلنية**	software engineers must act for the public interest مهندس البرمجيات يجب ان يعمل من أجل الصالح العام (سواء لمؤسسة او عميل)
Client and Employer **العميل ورئيس العمل**	software engineers must act in a manner which is the best interest for Client or employer and consistent with the public interest مهندس البرمجيات سجب ان يعمل بأصح أسلوب يخدم صالح العميل او رئيس عمله بشط ان يكون مقيد بالصالح العام
Product **المنتج**	software engineers must ensure that their products and related modifications meets the possible highest professional standers مهندس البرمجيات يجب ان يتأكد من ان منتجاته والتعديلات المتعلقة بها تتوافق مع اعلى المعدلات الممكنة
Judgment **الحكم**	software engineers must maintain integrity and independence in their professional judgement مهندس البرمجيات يجب ان يحافظ على نزاهته واستقلال رائيهفى حكمهم المهنى
Management **الادارة**	software engineering managers and leaders should promote an ethical approach to the management of maintenance and development of software مهندس البرمجيات كمدير او قائد يجب ان يعزز نهج اخلاقى لادارة صيانة وتطوير المنتجات
Profession **المهنية**	software engineers should advance integrity and reputation of the profession consistent with the public interest يجب على مهندسي البرمجيات تعزيز النزاهة وسمعة المهنة بما يتفق مع المصلحة العامة
Colleagues **الزملاء**	software engineers must be fair and supportive for their colleagues يجب أن يكون مهندسى البرمجيات عادلون وداعمون لزملائهم
Self **النفس**	software engineers should participate in lifelong of learning regarding of practicing the profession يجب على مهندسي البرمجيات المشاركة في التعلم مدى الحياة فيما يتعلق بممارسة المهنة

RUP: **R**ational **U**nified **P**rocess

RUP Is an iterative software development process هي عملية تطوير برمجيات بتكرارية

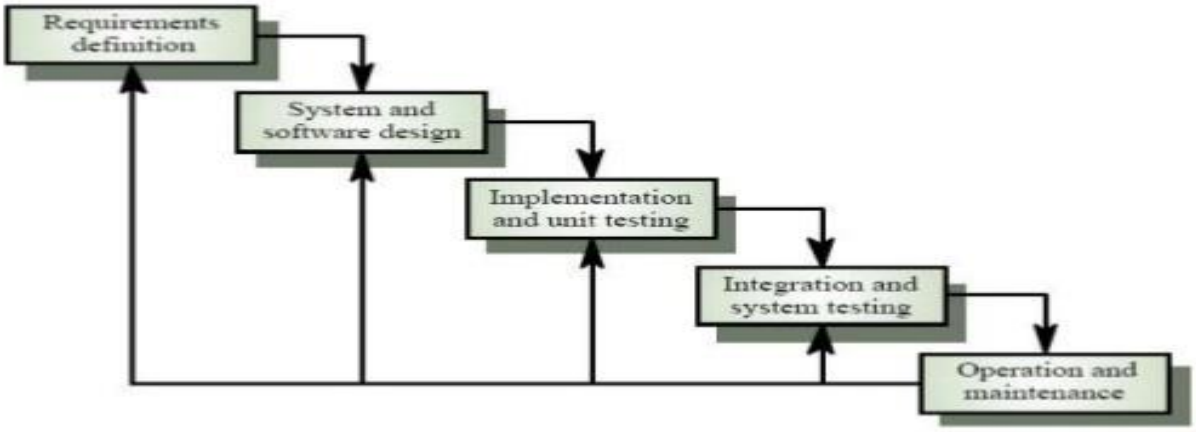
Software process models can be combined for designing large systems

يمكن دمج نماذج عملية سير البرمجيات لتصميم أنظمة كبيرة

Waterfall Model نموذج الشلال

transform mathematical model of a system into executable code using mathematical transformation that preserve its consistency.

تحويل نموذج رياضى لنظام الى كود يتم تنفيذه باستخادم تحويل رياضى للمحافظة على تناسقية النظام



Requirements and definition **المتطلبات والتعريفات**	The system contains, services and goals قيود النظام وخدماته وأهدافه
System and software design **تصميم النظام والبرنامج**	**System design** establish an overall architecture **تصميم النظام** يأسس هيكل عام **Software design** identify and describe the fundamental software system abstractions and their relationships **تصميم البرنامج** يعرف ويوصف المكونات الرئيسية المجردة للنظام البرمجى وعلاقاتهم
Implementation and unit testing **التطوير واختبار الوحدات**	**Implementation:** software design is realized as a set of programs or program units **التنفيذ**: تصميم البرنامج يدرك على انه مجموعة من البرامج او وحدات برامج **unit testing** verify that each unit meets its specifications **اختبار الوحدات**: يتأكد من ان كل وحدة تؤدى المواصفات المفروضة لها مسبقا
Integration and system testing **التكاملية واختبار النظام**	Individual program units or programs are integrated and tested as a complete system to ensure that the requirements have been meet وحدات البرنامج المنفصلة او البرامج يتم دمجها معا واختبارها كنظام كامل للتأكد من تطبيق كل المتطلبات **After testing** the software is delivered to customers **بعد الاختبار:** يتم تسليم البرنامج للعميل
Operation and maintenance **التشغيل والصيانة**	**Operation** the system installed and put into practical use **التشغيل:** النظام يتم تثبيته ووضعه فى موضع التنفيذ العملى **Maintenance** involves correcting errors which were not

	discovered in earlier stages of the life cycle الصيانة: تتضمن تصحيح الاخطاء التى لم تكتشف فى المراحل الاولية لدورة حياة البرنامج

Advantages of waterfall model **مميزات نموذج الشلال**

1- documentation is produces with each phase انتاج توثيق لنهاية كل مرحلة

2- fits with other engineering process models ملائم للاستخدام مع نماذج هندسة العمليات

Disadvantages (problems) of waterfall model **عيوب او مشاكل استخدام نموذج الشلال**

1- the difficulty of accommodating change after the process is underway

صعوبة اضافة تعديلات اثناء عمل المرحلة

2- One phase has to be complete before moving onto the next phase

مرحلة واحدة لابد ان تنتهى قبل الانتقال الى المرحلة التالية لها

3- partitioning of the project into distinct stages تجزئة المشروع الى مراحل مستقلة

this model is only appropriate when the requirements are well-understood and changes will be limited during the design process

هذا النموذج ملائم فقط عندما تكون المتطلبات مفهومة جيدا والتعديلات ستكون محدودة خلال مرحلة التصميم

Applicability of waterfall model **قابلية تطبيق نموذج الشلال**

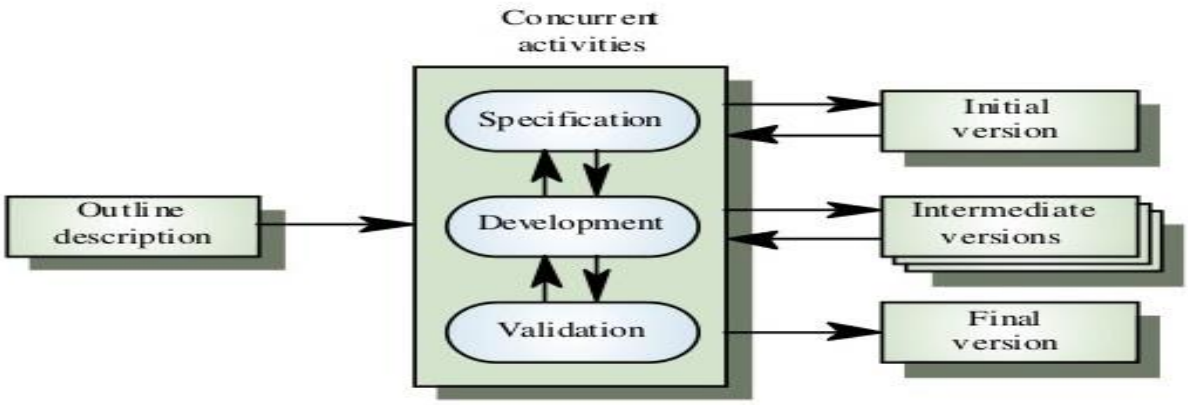
Used in some old large systems, now its recommended to design large systems if mixed with other models or sub-systems in large systems

استخدم فى الانظمة الكبيرة القديمة الان يوصى به لتصميم الانظمة الكبيرة بشرط ان يدمج مع نماذج اخرى او تصميم الانظمة الفرعية بالانظمة الكبيرة

Evolutionary (iterative) development التطوير بالتكرارية

Based on the idea of developing an initial implementation then refine it through customer comments until a stabile system has been developed.

قائم على فكرة تطوير نسخة ابتدائية للنظام ثم اعادة تحسينها من خلال تعليقات العميل عليها وتعاد العملية الا ان نطور نظام ثابت



Types of Evolutionary (iterative) development **انواع التطوير بالتكرار**

1- Exploratory development التطوير الاستكشافى

1.1 Objective of the process is to work with customers to explore their requirements and deliver a final system. هدف المرحلة العمل مع العميل لاستكشاف متطلباته وانتاج نظام نهائى

1.2 The development starts with the parts of the system that are understood.

التطوير يبدأ من أجزاء النظام المفهومة

1.3 the system evolves by adding new features proposed by the customer.

يتم تطوير النظام باضافة مميزات جديدة مقترحة من المستهلك

2- Throwaway prototyping النماذج المهملة

2.1 Objective is to understand the system requirements. الهدف فهم متطلبات النظام

2.2 Should start with poorly understood requirements to clarify what is really needed. يجب ان تبدأ من فهم غير صحيح للمتطلبات لاستيضاح ما يجب فعليا معرفته

Advantages of Evolutionary (iterative) development model مميزات التطوير بالتكرار

1- specification can be developed incrementally المواصفات يتم تطويرها تدريجيا

2- customers develop a better understanding of their problem العملاء ينسجو فهم افضل لمشكلاتهم

Disadvantages (problems) of Evolutionary (iterative) development model

عيوب التطوير بالتكرار

1- Lack of process visibility عدم وجود رؤية لاتمام العملية

2- Systems are often poorly structured الانظمة غالبا تصمم بشكل سيء (ضعيف)

Applicability of Evolutionary (iterative) development model التطبيق

Used for small and medium-size systems يستخدم فى الانظمة متوسطة او صغية الحجم

Component-based software engineering (CBSE)

هندسة البرمجيات القائمة على التجزئة

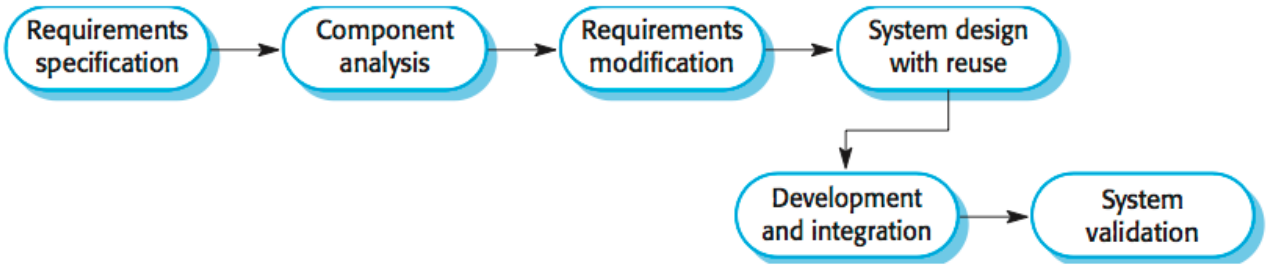
Based on systematic reuse where systems are integrated from existing components or **COTS** (**C**ommercial **O**ff-**T**he-**S**helf) systems

قائم على اعادة استخدام الانظمة بمعنى الانظمة يتم تكاملها من اجزاء موجودة او انظمة البرمجيات التجارية الجاهزة

COTS provide specific functionality such as text formatting or numeric calculation

البرمجيات التجارية الجاهزة تقدم وظائف محددة مثل تنسيق النصوص او حسابات عددية

Reuse-oriented development



Component analysis **تحليل المكونات**	A search is made for the components to implement the requirements specification يتم عمل بحث للمكونات التى سوف تنفذ مواصفات المتطلبات
Requirements modification **تعديل المتطلبات**	Requirements are analyzed using the information discovered about the components then modified المتطلبات يتم تحليلها باستخدام المعلومات المكتشفة عن الجزاء ثم يتم تعديلها
System design with reuse **تصميم النظام مع اعادة الاستخدام**	Design the framework of the system or existing framework is reused صمم هيكل النظام او اعد استخدام هيكل موجود مسبقا
Development and integration **التطوير والدمج**	**Develop** the system if it can't be externally developed **التطوير:** النظام لايمكن تطويره خارجيا (يعنى مفيش مكونات جاهزة مسبقا) **Integrate** the COTS and components to create the new system **الدمج:** البرمجيات التجارية الجاهزة والمكونات التى تستخدم لانشاء النظام الجديد

Advantages of Component-based software engineering model **المميزات**

1- reduce the amount of software to be produced تقليل وقت انتاج البرمجيات

2- reduce cost تقليل التكلفة

3- reduce risks تقليل المخاطر

4- faster delivery for software تسليم اسرع للبرمجيات

Disadvantages (problems) of Component-based software engineering model **العيوب**

1- compromises requirements تسوية المتطلبات (بمعنى الاتفاق على حل وسط للمتطلبات)

2- resulted system may not meet the customer needs النظام الناتج لايحقق احتياجات المستخدم

3- some control over the system evolution is lost as some reusable components are not under the control of the same organization using them

بعض التحكم فى تطوير النظام يتم فقده حيث بعض المكونات المعاد استخدامها ليست تحت تحكم نفس المؤسسة التى تستخدمهم

Applicability of Component-based software engineering model **التطبيق**

Used for large systems design تستخدم فى تصميم الانظمة الكبيرة

Process iteration تكرار العملية

System requirements **ALWAYS** evolve in large systems so process iteration where earlier stages are reworked is always part of the process for these systems

متطلبات النظام دائما تتطور فى الانظمة الكبيرة ولذلك تكرار العملية فى مراحل الابتدائية يتم اعادة صياغتها ويعد هذا جزء من عمليات هذه الانظمة

Types of Process iteration **أنواع تكرارات العمليات**

1- Incremental delivery التسليم التدريجى

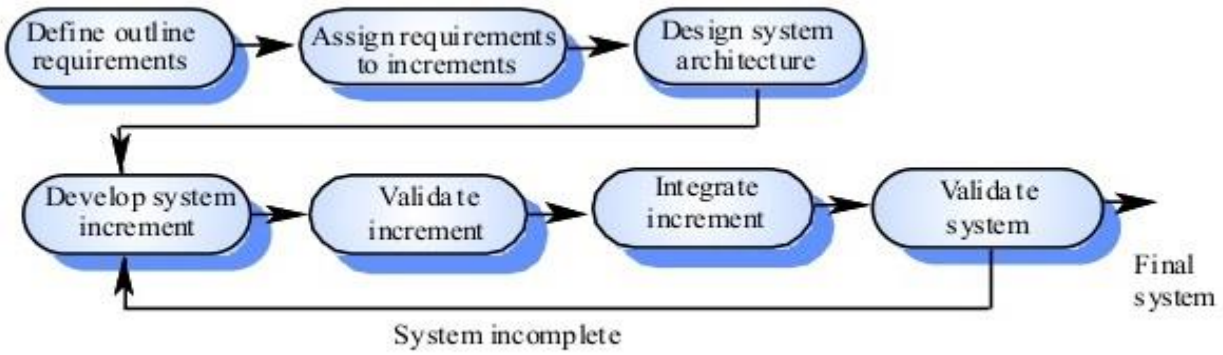
Software specification and implementation are broken down into a series of increments and each turn an increment is being developed

مواصفات البرنامج وتنفيذه يتم تجزئتهم الى سلسلة من التدرجات وفى كل خطوة يتم تطوير تدرج واحد

Customers define the priority of each increment the higher the priority is the earlier it will be included المستهلكون يحددوا اولوية كل تدرج كلما علت الاولوية يضاف التدرج باكرا

Once the development of an increment is started its requirements are frozen

عندما يبدأ تطوير تدرج ما تتجمد (لا يمكن التعديل فيها او الجوع اليها) متطلباته



Advantages of the Incremental delivery **المميزات**

1- Customers can benefit for the system while its being developed

العملاء يستفيدوا من النظام قبل انتهاء تطويره

2- Early increments act as a prototype to help elicit requirements for later increments التدرجات الباكرة تعمل كنماذج ابتدائية للمساعدة فى انتخاب (ترشيح) المتطلبات للتدرجات الاتية

3- Lower risk of overall project failure تقليل المخاطر لفشل النظام فى بشكل عام

4- The highest priority system services tend to receive the most testing

خدمات النظام الاعلى فى الاولوية تميل اى ان تمنح اختبارات مكثفة

Disadvantages of the Incremental delivery **العيوب**

1- Increments should be small التدرجات يجب ان تكون صغيرة

2- each increment should deliver some system functionality

كل تدرج يجب ان يمدنا ببعض وظائف النظام

3- difficulty of mapping user requirements onto increments of the right size

صعوبة توفيق متطلبات المستخدم مع تدرجات بحجم مناسب

4- hard to identify the common facilities which are needed by all increments

صعوبة التعرف على التسهيلات العامة التى يتم احتياجها لكل التدرجات

2- Spiral development التطوير الحلزونى

Development of the system spirals outwards from the initial outline through to the final developed system تطوير النظام يتدرج حلزونيا الى الخارج ابتداء من المخطط التمهيدى الى انتاج المشروع النهائى

Software Process is represented as a spiral rather than as a sequence of activities with backtracking from one activity to another

سير عملية البرمجة يمثل على شكل حلزونى بدلا من متتالية من الانشطة مع امكانية الجوع من نشاط الى نشاط اخر

Each loop in the spiral represents a phase in the process

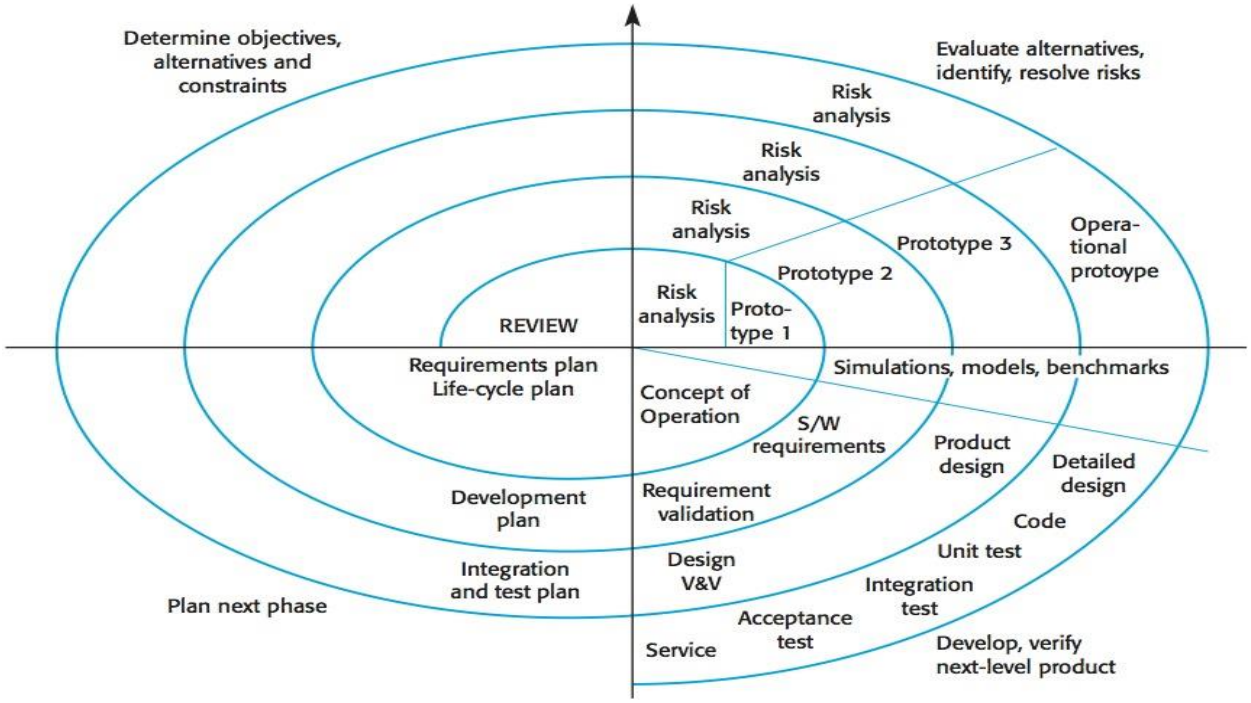
كل لفة فى النموذج الحلزونى تعبر عن مرحلة من مراحل العملية

loops in the spiral are chosen depending on the required phase

عدد اللفات فى النموذج الحلزونى تحدد بناء على عدد المراحل

Risks are explicitly assessed and resolved throughout the process

المخاطر تقيم بشكل واضح ويتم حلها خلال شير العملية



Spiral model sectors قطاعات النموذج الحلزونى

Objective setting **تحديد الاهداف**	Specific objectives for the phase are identified تحديد أهداف محددة للرمحلة
Risk assessment and reduction **تقييم المخاطر وتقليلها**	A detailed analysis is carried out for each identified risk then steps are taken to reduce this risk تحليل مفصل يتم عمله لكل خطأ يتم ايجاده
Development and validation **التطوير والتحقق من صحة النظام**	After risk evaluation a development model for the system is chosen using any of the generic models بعدما يقيم الخطأ يتم اختيار نموذج تطوير للنظام باستخدام اى نموذج من النماذج العامة
Planning **التخطيط**	The project is reviewed and the next phase of the spiral is planned if its required يتم استعراض المشروع و يتم البدء فى التخطيط للمرحلة التالية فى النموذج الحلزونى ان كانت مطلوبة

Advantages of the Spiral development **فوائد النموذج الحلزونى**

1- risk minimization يقلل من المخاطر

How Process Activities can be organized

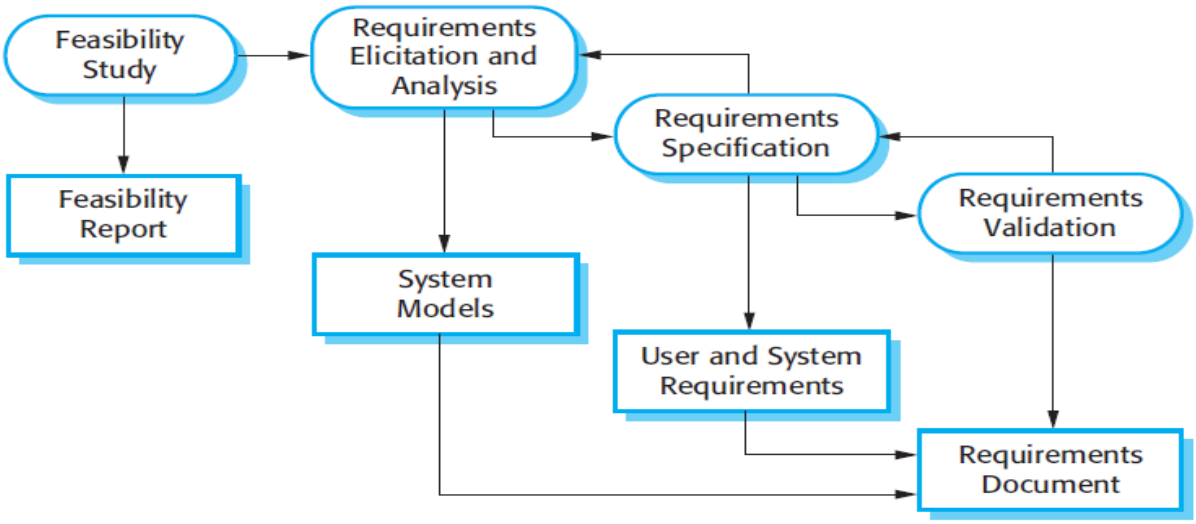
كيف يمكن تنظيم الانشطة الخاصة بسير عملية ما

1- Software specification (requirements engineering process)

توصيفات المتطلبات (متطلبات هندسة العملية)

Process of understanding and defining what services are required form the system as well as the constrains on the system operations

هى مرحلة فهم وتعريف الخدمات المراده من هذا النظام بالاضافة الى القيود الخاصة بعملياته



Main phases of the requirements engineering process

المراحل الرئيسية للمتطلبات

Feasibility study **دراسة الجدوى**	The study consider whither the proposed system will be cost-effective from a business point of view الدراسة تأخذ بعين الاعتبار ما اذا كان النظام المقدم عالى الكفأة بالنسبة لتكلفته من وجهة نظر استثمارية
Requirements elicitation and	Acquire the system requirements through the observation (analysis) of existing systems

analysis **انتخاب المتطلبات والتحليل**	الحصول على متطلبات النظام من خلال متابعة (تحليل) الانظمة الموجودة مسبقا
Requirements specification **مواصفات المتطلبات**	Translating the information gathered during the analysis into a document which defines the set of requirements Has two types **User requirements** and **system requirements** تحويل المعلومات التى جمعت خلال مرحلة التحليل الى توثيق يعرف مجموعة المتطلبات المراده وله نوعان توثيق للمستخدم وتوثيق للنظام
Requirements validation **التحقق من صحة المتطلبات**	The discovered errors in requirements document must be modified الخطاء التى تم اكتشافها فى وثيقة المتطلبات يجب ان يتم تعديلها

2- Software design and implementation تصميم وتنفيذ النظام

The process of converting the system specification into an executable system.

عملية تحويل مواصفات النظام الى نظام قابل للتنفيذ

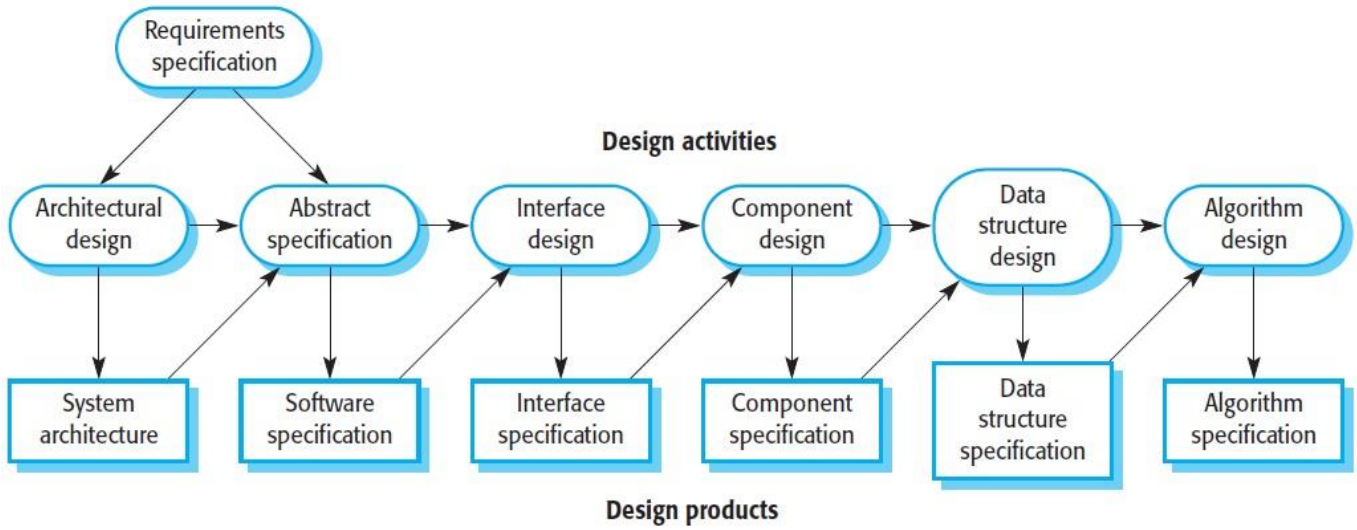
Software design: Design a software structure that realizes the specification

تصميم النظام: تصميم هيكل البرنامج الذى يحقق المواصفات

Implementation: Translate this structure into an executable program

التنفيذ: يترجم هيكل التصميم الى برنامج قابل للتنفيذ

The activities of design and implementation are closely related and may be interleaved الانشطة الخاصة بالتصميم والتنفيذ متقاربة جدا من بعضها وقد تكون متداخلة



Design process activities أنشطة مرحلة التصميم

Architectural design **التصميم المعمارى**	Subsystems makeup the system and identify the relationships and document them الانظمة الفرعية تصنع النظام وتعرف العلاقات والتوثيقات الخاصه به
Abstract specification **المواصفات المجرده**	Defines the abstract of each subsystem as well as its services and constrains which it must meet تعرف المجردات الخاصة بكل نظام فرعى كما انها تعرف الوظائف والقيود التى يجب تحقيقها
Interface design **تصميم الواجهه**	Each subsystem interface which work with other subsystems is designed and documented

	كل واجهه لنظام فرعى والتى تعمل مع انظمة فرعية أخرى تصمم وتوثق
Component design **تصميم المكونات**	Services are allocated to components and components interfaces are designed الخدمات تمنح للمكونات وواجهات المكونات تصمم
Data structure design **تصميم هياكل البيانات**	The data structures used in the system implementation are designed in detail هياكل البيانات المستخدمة فى تنفيذ النظام تصمم بالتفصيل
Algorithm design **تصميم الخوارزمية**	Algorithms used to provide services are designed in detail الخوارزميات تستخدم لتقديم الخدمات يتم تصميمها بالتفصيل

Algorithms adaptation methods **أساليب تأقلم الخوارزميات**

1- data structure design and algorithm design may be delayed until implementation process تصميم هيكل بيانات و تصميم خوارزمية قد يتأخر حتى مرحلة التنفيذ

2- interface may be designed after the data structure is specified if the design is exploratory الواجهه قد تصمم بعد تحديد هيكل البيانات اذا كان التصميم استكشافى

3- Abstract specification stage maybe skipped although it's an essential part of critical system design مرحلة المواصفات المجردة قد يتم تعديها بالرغم من اهميتها فى تصميم النظام

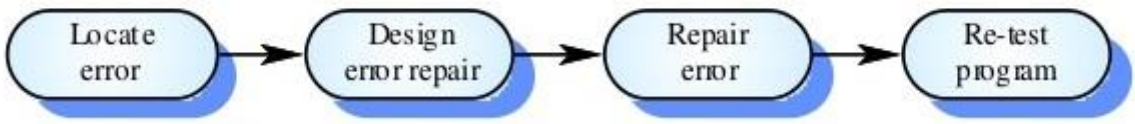
Programming and debugging **البرمجة وتصحيحها**

Translating a design into a program and removing errors from that program

ترجمة التصميم الى برنامج وازالة الاخطاء من البرنامج

Programmers carry out some program testing to discover faults in the program

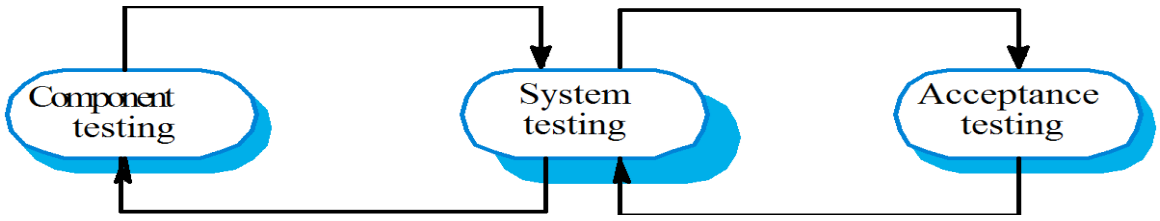
المبرمجين ينفذوا بعض برامج الاختبار لاكتشاف الاخطاء فى البرامج



3- software validation التحقق من البرامج

Involves checking and review processes and system testing to conforms if the system meets the requirements of customer

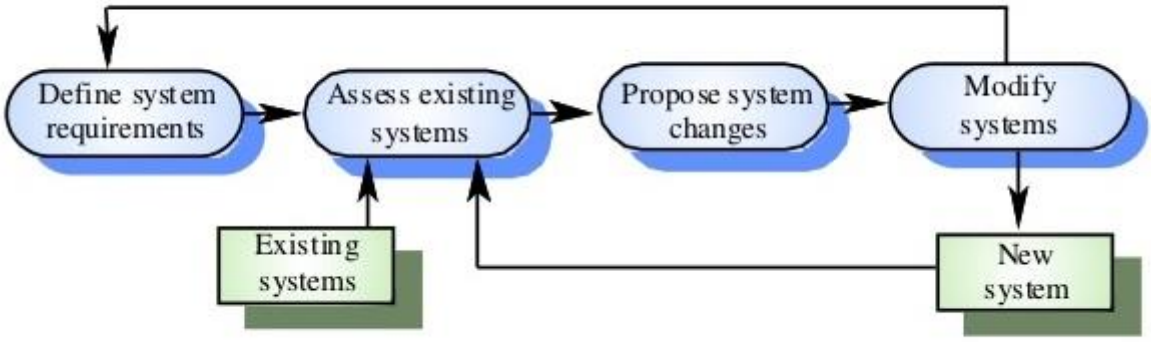
تتضمن التحقق من ومراجعة مراحل واختبارات النظام للتأكد من النظام يحقق متطلبات المستخدم



4- software evolution تطوير البرامج

As requirements change through changing business circumstances, the software that supports the business must also evolve and change.

بما ان المتطلبات تتغير بسبب تغير ظروف العمل كذلك البرامج الداعمة للعمل يجب ان تطور وتتغير



Computer-Aided Software Engineering (CASE)

هندسة البرمجيات بمساعدة الحاسوب

Automated activities using CASE استخدام الانشطة الالية فى هندسة البرمجيات بمساعدة الحاسوب

1- development of graphically system models as part of the requirements specifications تطوير نماذج انظمة رسومية كجزء من مواصفات المتطلبات

2- understanding a design using data dictionary فهم تصميم باستخدام قاموس بيانات

3- generation of user interface from a graphical interface توليد واجهه المستخدم من واجهه رسومية

4- program debugging using previous data about existing program

تصحيح اخطاء البرنامج باستخدام بيانات سابقة عن برنامج موجود مسبقا

5- translation of a program form an old programming language to a more recent version ترجمة برنامج من لغة برمجة قديمة الى اصدار احدث

improvements limitations in CASE قيود تطور هندسة البرمجيات بمساعدة الحاسوب

1-attempting to use artificial intelligence to provide support for design have not been successful. محاولة استخدام الذكاء الاصطناعى لتقديم الدعم فى التصميم لم تنجح

2- doesn’t support working as a team. لا تدعم العمل كفريق

Software Requirements متطلبات البرمجيات

Requirements engineering (RE) هندسة المتطلبات

The process of finding out, analyzing, documenting the services and the constraints that the customer requires from a system.

هى عملية ايجاد وتحليل وتوثيق الخدمات والقيود التى طلبها المستهلك فى النظام

What are requirements? ماهى المتطلبات؟

requirements are the descriptions of the system specification, services and constraints هى توصيفات لخصائص النظام وخدماته وقيوده

Types of requirement انواع المتطلبات

1- User requirements متطلبات للمسخدم

Statements in natural language plus diagrams of the services the system provides and its operational constraints (Written for customers.)

هى جمل مكتوبة بلغة طبيعة بالاضافة الى نماذج للخدمات التى يقدمها النظام وقيوده العاملة (المطبقه) (كتبت خصيصا للمستهلك)

2- System Requirements متطلبات للنظام

A structured document setting out detailed descriptions of the system's functions, services and operational constraints.

هى وثيقة مهيكلة لوضع تفصيلات مدققة لوظائف النظام وخدماته وقيوده العامله (المطبقه)

Classifications of System requirements تصنيفات متطلبات النظام

1- Functional requirements متطلبات وظيفية

1- Statements of services the system should provide هى تقارير للخدمات التى يقدمها النظام

2- how the system should react to particular inputs كيف يتفاعل النظام عند ادخال مدخل معين

3- how the system should behave in particular situations كيف يتعامل النظام فى موقف معين

Functional User requirements متطلبات المستخدم الوظيفية

maybe high-level statements of what the system should do

قد تكون تقارير عالية المستوى (سهلة وواضحة بدون تفاصيل) لما يقوم به النظام

Functional System requirements متطلبات النظام الوظيفية

should describe the system services in details. يجب ان تصف خدمات النظام بالتفصيل

Requirements imprecision عدم دقة المتطلبات

Problems arise when requirements are not precisely stated.

المشكلة تنتج عندما تكون المتطلبات غير مذكروة بالتحديد

Example: Consider the term **appropriate viewers** **مثال:** اعتبر مصطلح **عارض مناسب**

User intention - special purpose text viewer for each different document type.

نية المستخدم (لتفسير المصطلح) هو عارض نصوص ذات أغراض خاصة لمختلف أنواع الوثائق

Developer interpretation - Provide a text viewer that shows the contents of the document ترجمة المطور (للمصطلح) تقديم عارض نصوص يسمح للمستخدم بعرض محتوى اى وثيقة

Requirements completeness and consistency **تكاملية وتناسق المتطلبات**

Complete: They should include descriptions of all facilities required.

كاملة: يجب ان تحتوى على كل وسائل التسهيل المطلوبة

Consistent: There should be no conflicts in the descriptions of the system facilities.

متناسقة: يجب ان لايكون هناك تعارض فى توصيف وسائل تسهيل النظام

2- non-functional requirements **متطلبات غير وظيفية**

constraints on the services or functions offered by the system such as response time, constraints on the development process, standards, etc.

القيود الموجوده على الخدمات ووالوظائف التى يقدمها النظام مثل وقت استجابته, قيود مرحلة التطوير, المبادىء العامة وغيرها

non-functional requirements classifications (Types)

تصنيفات المتطلبات الغير وظيفية

Product requirements **متطلبات المنتج**

Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.

متطلبات تحدد ما اذا كان المنتج المطروح يجب ان يتعامل باسلوب معين كسرعة التنفيذ والاعتمادية وغيرها

Organisational requirements **متطلبات المؤسسة**

Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements, etc.

متطلبات تنتج من سياسات واجراءات المنظمة كمتطلبات التنفيذ و معايير العملية التى استخدمت وغيرها

External requirements **متطلبات خارجية**

Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc. متطلبات تظهر لعوامل خارجية عن النظام ومرحلة تطويره ك متطلبات التوافيقة و المتطلبات القانونية وغيرها

3- Domain Requirements **متطلبات المجال**

Requirements comes from the Application Domain of the system, they may be functional or non-functional متطلبات تاتى من مجال تطبيق النظام قد تكون وظيفية او غير وظيفية

Examples of functional requirements of the The LIBSYS system

مثال على المتطلبات الوظيفية فى نظام مكتبة (الكترونية)

1- The system shall provide appropriate viewers for the user to read documents in the document store. النظام يجب ان يقدم عارض مناسب للمستخدم ليطلع على الوثائق فى متجر الوثائق (الكترونى)

2- user shall be able to search either all of the initial set of databases or select a subset from it المستخدم يجب ان يكون قادر على البحث فى مجموعة ابتدائية من قواعد البيانات او قاعدة فرعية منهم

3- Every order shall be allocated a unique identifier (ORDER_ID) which the user shall be able to copy to the account's permanent storage area

لكل طلب شراء يجب ان يحدد معرف فريد (ORDER_ID) ويستطيع المستخدم نقله الى منطقة التخزين الدائم فى حسابه

Examples of Non-functional requirements of the The LIBSYS system

مثال على المتطلبات الغير وظيفية فى نظام مكتبة (الكترونية)

Product requirements **متطلبات المنتج**

1- User interface maybe implemented using HTML or Java Applets

واجهة المستخدم يتم انشائها باستخدام ال HTML او Java Applets

2- System Viewers will allow the user to view any document content in the store

عارض النظام يجب ان يسمح للمستخدم بعرض محتوى اى وثيقة فى المتجر

3- System must contain a search functionality either for all the initial set of databases or select a subset from it

النظام يجب ان يحتوى على اسلوب بحث اما فى مجموعة قواعد البيانات الاساسية او فى جزء منها

Organisational requirements **متطلبات المؤسسة**

The system development process and deliverable documents shall conform to the process and deliverables defined in XYZCo-SP-STAN-95.

مرحلة تطوير النظام والوثائق التى سيتم تسليمها يجب ان تتفق مع العمليات والنتائج المعرفة مسبقا فى XYZCo-SP-STAN-95

External requirements **متطلبات خارجية**

The system shall not disclose any personal information about customers apart from their name and reference number to the operators of the system

النظام يجب ان لا يسمح بعرض اى معلومات شخصية عن العميل بخلاف اسمه والرقم المرجعى (الكود الذى يعبر عنك) لمشغلى النظام

Goals and requirements الاهداف والمتطلبات

Goal A general intention of the user such as ease of use.

الهدف هو هدف عام للمستخدم مثل سهولة الاستخدام

Verifiable non-functional requirement A statement using some measure that can be objectively tested

التحقق من المتطلبات الغير وظيفية هو تقرير باستخدام بعض المقاييس (المعايير) التى يمكن اختبارها بشكل فعال

Example Traffic Control System **مثال نظام التحكم فى اشارات المرور**

system goal The system should be easy to use by experienced controllers and should be organized in such a way that user errors are minimized.

هدف النظام: النظام يجب ان يكون سهل الاستخدام من مراقبين ذوى الخبرة ويجب ان يكون منسق بشكل يحد من اخطاء المستخدم

verifiable non-functional requirement Experienced controllers shall be able to use all the system functions after a total of two hours training. After this training, the average number of errors made by experienced users won't exceed 2 per day.

التحقق من المتطلبات الغير وظيفية: المراقبين ذوى الخبرة يجب ان يكونوا قادرين على استخدام كل وظائف النظام بعد ساعتين من التدريب, بعد التدريب معدل متوسط الاخطاء لكل منهم يجب ان لا يتعدى 2 لكل يوم

How you can specify non-functional system properties

(Requirements measures)

كيف نقوم بتحديد خصائص النظام الغير وظيفية

As Non-functional requirements maybe very difficult to state precisely and imprecise requirements may be difficult to verify

المتطلبات الغير وظيفية قد تكون صعبة للغاية عند محاولة ذكرها بالتحديد كما ان اسلوب عدم دقة البيانات قد يكون صعب التحقق منه

Property	Measure
Speed	Response time Screen refresh time
Size	K bytes Number of RAM ships
Ease of use	Training time Number of help Frames
Reliability	Availability Rate of failure occurrence
Robustness	Time to restart After Failure Percentage of events causing failure
Portability	Number of target systems

Requirements documents often include statements of goals mixed with requirements

وثائق المتطلبات غالبا تحتوى على تقارير للاهداف مختلطة مع المتطلبات

Conflicts between non-functional requirements and functional or non- functional requirements are common in complex systems

التتضارب بين المتطلبات الغيروظيفية ونفسها او الوظيفية شائع فى الانظمة المعقدة

Example Spacecraft system **مثال نظام المركبات الفضائية**

To minimise weight, the number of separate chips in the system should be minimised. لتقليل الوزن عدد الشرائح المنفصلة فى النظام يجب تقليله

To minimise power consumption, lower power chips should be used.

لتقليل استهلاك الطاقة يجب استخدام شرائح ذات طاقة منخفضة

However, using low power chips may mean that more chips have to be used. Which is the most critical requirement

على الرغم من ذلك استخدام شرائح ذات طاقة منخفضة قد يعنى الحاجة لاستخدام شرائح اكثر وهذا يعد اكثر مطلوب حرج

Domain requirements متطلبات المجال

1- Derived from the application domain (Not from the user requirements)

تشتق من مجال التطبيق و ليس من متطلبات المستخدم

2- describe system characteristics and features تصف خصائص النظام و مميزاته

3- they maybe new functional requirements قد تكون متطلبات وظيفية

4- define specific computations تعرف حسابات محددة

If domain requirements are not satisfied, the system maybe unworkable

اذا لم تحدد متطلبات المجال قد يصبح النظام غير عملى

User requirements متطلبات المستخدم

1- Should describe functional and non-functional requirements so they can be understandable by system users يجب ان تصف المتطلبات الوظيفية والغير وظيفية لكل يفهمهم مستخدم النظام

2- they should only specify the external behaviour of the system

يجب ان يحددوا فقط التصرفات الخارجية للنظام (ليس كيف يعمل)

3- they should avoid system design يجب ان تتجنب تصميم النظام

4- must be written using natural language phrases يجب ان تكتب كجمل باستخدام لغة طبيعية

Problems with natural language **مشاكل اللغات الطبيعية**

Lack of clarity Precision is difficult without making the document difficult to read.

قلة الوضوح صعوبة الدقة بدون جعل الوثيقة (الى بتخرج للمستخدم) صعبة القرأة

Requirements confusion Functional and non-functional requirements tend to be mixed-up. **خلط المتطلبات** المتطلبات الوظيفية وغير الوظيفية تميل الى ان تدمج معا

Requirements amalgamation Several different requirements may be expressed together as single requirement. **دمج المتطلبات** كثير من المتطلبات قد يعبر عنها كمطلوب واحد

System requirements متطلبات النظام

1- A more detailed version from user requirements used by software engineers

نسخة مفصلة من متطلبات المستخدم تستخدم من قبل مهندسى البرمجيات

2- They are considered to be a basis for designing the system

يتم اعتبارهم ك أسس لتصميم النظام

3- Explains how the user requirements should be provided by the system

تشرح كيف يتم تقديم متطلبات المستخدم من خلال النظام

4- They may be used as a part of the system contract قد يتم استخدامهم كجزء من عقد النظام

Requirements and design المتطلبات والتصميم

requirements should state what the system should do and the design should describe how it does this المتطلبات يجب ان تحدد ما سيقوم به النظام والتصميم يصف كيف يقوم النظام بتنفيذها

In practice, requirements and design are inseparable فى الحياة العملية التصميم والمتطلبات لاينفصلوا

1- A system architecture may be designed to structure the requirements then organize them according to different subsystems

هيكل النظام قد يصمم لهيكلة المتطلبات ثم تنسقها تبعا للانظمة الفرعية المختلفة

2- The system may deal with other existing systems

قد يتعامل النظام مع انظمة اخرى موجوده مسبقا

3- The use of a specific Architecture to satisfy functional and non-functional requirements. استخدام هيكل معين لايفاء المتطلبات الوظيفية والغير وظيفية

Problems with Natural Language specification

مشاكل مواصفات اللغات الطبيعية

Ambiguity The readers and writers of the requirement interpret the same words in the same way. NL is naturally ambiguous so this is very difficult.

الغموض الكتاب و القراء للمتطلبات يفسروا نفس الكلمات بنفس الاسلوب لان اللغة الطبيعية غامضة بطبيعتها ولذلك نجد صعوبة

(ملخص الكلام ان المستخد والمطور هيفهموا نفس الحاجة وده خطأ لانه لاما المبرمج هيفهم والمستخدم لا او العكس)

Over-flexibility The same thing may be said in a number of different ways in the requirements specification.

الافراط فى المرونة نفس الشيء يمكن قوله باكثر من طريقة فى توصيف المتطلبات

Lack of modularization NL structures are unsuitable to structure system requirements

عدم وجود نمطية اللغة الطبيعية غير صالحة لهيكلة متطلبات نظام

Alternatives to NL specification بدائل مواصفات اللغات الطبيعية

(Notations for requirement specifications) ملاحظات لمواصفات المتطلبات

Notation	Description
Structured Natural Language **اللغة الطبيعية المهيكلة**	Express the requirement specifications as standard forms or templates تعبر عن مواصفات المتطلبات كاستمارات عامة او قوالب
Design Description Language **لغة التصميم الوصفى**	Define an optional model of the system using a language like a programming language تعرف نموذج احتياطى للنظام باستخدام لغة مثل لغة برمجة **Useful** for interface specifications مفيد فى تحديد مواصفات واجهة المستخدم
Graphical Notations **الرسوم البيانية**	Define functional requirements for the system using a graphical language and text notations تعرف المتطلبات الوظيفية للنظام باستخدام لغة تصميم وملاحظات نصية **Like** use-case description and sequence diagrams
Mathematical Specifications **المواصفات الرياضية**	Notations based on mathematical concepts such as sets or finite state machine هى ملاحظات قائمة على مبادىء رياضية مثل المجموعات

Structured language specifications مواصفات اللغات الهيكلية

The freedom of the requirements writer is limited by a predefined template for requirements. حرية كاتب المتطلبات مقيدة بقالب معرف مسبقا للمتطلبات

1- requirements are written in a standard way. المتطلبات تكتب بشكل عام

2- The terminology used in the description may be limited.

المصطلحات التى تكتب فى الوصف تكون مقيدة

Advantage is that maintains most of the expressiveness of natural language is but ensures that a degree of uniformity is imposed on the specification.

تتميز بانها تحافظ على اكبر قدر من التعبير المكتوب بلغة طبيعية ولكن تتأكد من عدم وجود توحد فى المواصفات

Form-based approach النهج القائم على الاستمارة

Specify system requirements by defining one or more standard forms or templets to express these requirements

يحدد مواصفات النظام بتعريف واحد او اكثر من الاستمارات او القوالب العامة للتعبير عن هذه المواصفات

Form-based specifications مواصفات النموذج القائم على الاستمارة

1- Description of the function or entity. وصف عن كيان الوظيفة

2- Description of inputs and where they come from. وصف عن المدخلات ومن اين تأتى

3- Description of outputs and where they go to. وصف عن المخرجات والى اين تذهب

4- Indication of what other entities are required. الاشارة الى ما اذا كان هناك كيانات اخرى مرادة

5- Description of the action to be taken. وصف للتحرك الواجب تنفيذه

6- Pre and post conditions (if appropriate). قبل التنفيذ وبعد التنفيذ ان كان النظام يحتاج اليهم

7- Description of the side effects (if any) of the operation. وصف للاثار الجانبية لاى عملية بالنظام

Form-based node specification المواصفات

Function, Description, Inputs, Source, Outputs, Destination, Action, Requires, Pre-Condition, Post-Condition, Side-Effects

requirements document وثيقة المتطلبات

(SRS: Software Requirements Specification)

وثيقة متطلبات المواصفات

The requirements document is the official statement of what the system developers should Implement. وثيقة المتطلبات هى تقرير رسمى لما يجب على مطورى النظام تطبيقه

Should include both a definition of user requirements and detailed specification of the system requirements. يجب ان تتضمن تعريف لكلا من متطلبات المستخدم وتوصيف مدقق لمتطلبات النظام

It is NOT a design document. هى ليست وثيقة تصميم

As far as possible, it should set of WHAT the system should do rather than HOW it should do it على قدر المستطاع يجب ان تحدد ماذا يجب ان ينفذ النظام بدلا من كيف يقوم بتنفيذه

Requirements Engineering Processes

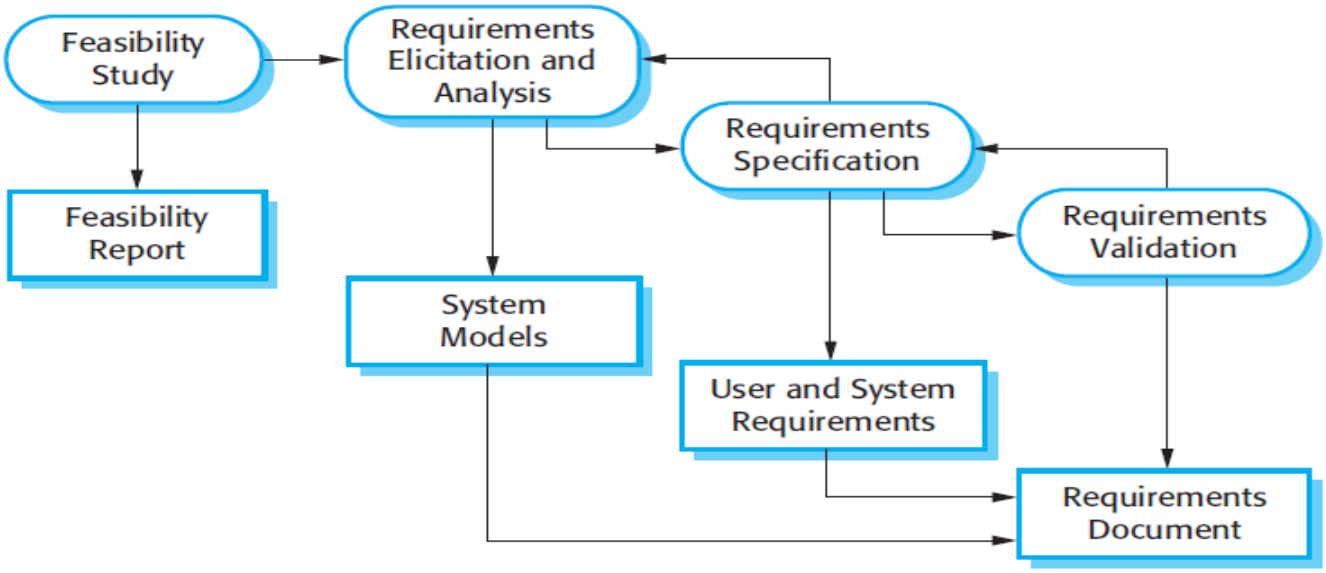
generic activities common to all processes

1- Requirements elicitation.

2- Requirements analysis.

3- Requirements validation.

4- Requirements management.



Feasibility studies

decides whether the proposed system is worthwhile or not.

A short focused study that checks

1- If the system contributes to organizational objectives.

2- If the system can be engineered using current technology and within budget.

3- If the system can be integrated with other systems that are used.

Feasibility study implementation

Based on information evaluation, information collection and report writing

Questions for people in the organisation

What if the system wasn't implemented?

What are current process problems?

How will the proposed system help?

What will be the integration problems?

Is new technology needed? What skills?

What facilities must be supported by the proposed system?

1,2_ Requirements Elicitation (Discovery) and analysis

Technical staff working with customers to find out about the application domain, the services and operational constraints that the system should provide

stakeholders include end-users, managers, maintenance engineers, domain experts, trade unions, etc.

Problems of requirements analysis

(difficulty of Understanding the stockholder requirements)

1- Stakeholders don't know what they really want from the computer system.

2- Stakeholders express requirements in their own terms.

3- Different stakeholders may have conflicting requirements.

4- Organisational and political factors may influence the system requirements.

5- The requirements change during the analysis process as New stakeholders may emerge or the business environment changes.

Process activities (Using Spiral development)

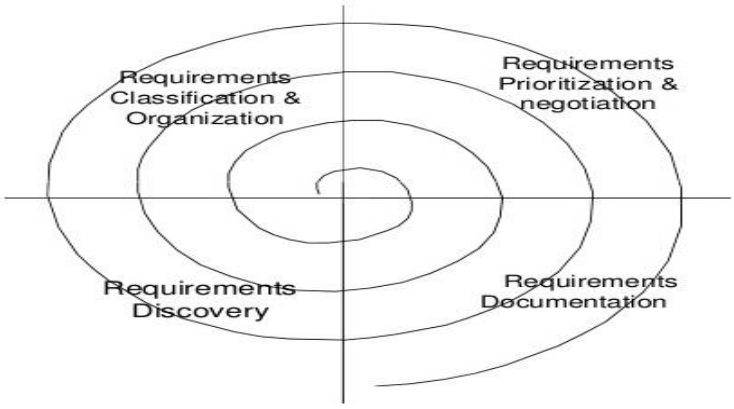
1- Requirements discovery documentation, interacting with stakeholders to discover the requirements also Domain requirements are discovered at this stage.

2- Requirements classification and organization Groups related requirements and organizes them into coherent clusters.

3- Requirements Prioritization and negotiation Prioritizing requirements and resolving requirements conflicts.

4- Requirements Documentation Requirements are documented and input into the next round of the spiral.

Requirements Elicitation (Discovery) and analysis process



Example: system stakeholders for a bank ATM

Bank customers, Representatives of other banks, Bank managers, Counter staff Database administrators, Security managers, Marketing department, Hardware and software maintenance engineers, Banking regulators

Viewpoints

Viewpoints are a way of structuring the requirements to represent the perspectives of different stakeholders.

Stakeholders may be classified under different viewpoints.

This multi-perspective analysis is important as there is no single correct way to analyze system requirements.

Generic Types of viewpoint

Interactor viewpoints People or other systems that interact directly with the system.

In an ATM, the customer's and the account database.

In an LIBSYS, Users and Library staff

Indirect viewpoints Stakeholders who do not use the system themselves but who influence the requirements.

In an ATM, management and security staff.

In an LIBSYS, Finance, Library manger and Article Providers

Domain viewpoints Domain characteristics and constraints that influence the requirements.

In an ATM, standards for inter-bank communications.

In an LIBSYS, UI standards and Classification System

Viewpoint specific identification

1- Providers and receivers of system services.

2- Systems that interact directly with the system being specified.

3- Regulations and standards.

4- Sources of business and non-functional requirements.

5- Engineers who have to develop and maintain the system.

6- Marketing and other business viewpoints.

Interviewing

formal or informal interviewing with stockholders

the RE team puts questions to stakeholders about the system that they use and the system to be developed.

Types of interview

Closed interviews where a pre-defined set of questions are answered.

Open interviews where there is no pre-defined agenda and a range of issues are explored with stakeholders.

Interviews in practice

Normally a mix of closed and open-ended interviewing.

Interviews are good for getting an overall understanding of what stakeholders do and how they might interact with the system.

Interviews are not good for understanding domain requirements

Interviews Problems with Application Domain

1- Requirements engineers cannot understand specific domain terminology.

2- Some domain knowledge is so familiar that people find it hard to articulate or think that it isn't worth articulating.

Effective interviewers

Interviewers should be open-minded, willing to listen to stakeholders and should not have preconceived ideas about the requirements.

They should prompt the interviewee with a question or a proposal and should not simply expect them to respond to a question such as "what do you want".

Scenarios

Scenarios are real-life examples of how a system can be used.

Useful for adding detail to an outline requirement description

They may include

1- A description of the starting situation.

2- A description of the normal flow of events.

3- A description of what can go wrong.

4- Information about other concurrent activities.

5- A description of the state when the scenario finishes.

Example **LIBSYS scenario**

Initial assumption [1]: The user has logged on to the LIBSYS system and has located the journal containing the copy of the article.

Normal [2]: The user selects the article to be copied. He or she is then prompted by the system to either provide subscriber information for the journal or to indicate how they will pay for the article. Alternative payment methods are by credit card or by quoting an organisational account number.

The user is then asked to fill in a copyright form that maintains details of the transaction and they then submit this to the LIBSYS system.

The copyright form is checked and, if OK, the PDF version of the article is downloaded to the LIBSYS working area on the user's computer and the user is informed that it is available. The user is asked to select a printer and a copy of the article is printed. If the article has been flagged as 'print-only' it is deleted from the user's system once the user has confirmed that printing is complete.

What can go wrong [3]: The user may fail to fill in the copyright form correctly. In this case, the form should be re-presented to the user for correction. If the resubmitted form is still incorrect then the user's request for the article is rejected.

The payment may be rejected by the system. The user's request for the article is rejected.

The article download may fail. Retry until successful or the user terminates the session.

It may not be possible to print the article. If the article is not flagged as 'print-only' then it is held in the LIBSYS workspace. Otherwise, the article is deleted and the user's account credited with the cost of the article.

Other activities [4]: Simultaneous downloads of other articles.

System state on completion [5]: User is logged on. The downloaded article has been deleted from LIBSYS workspace if it has been flagged as print-only.

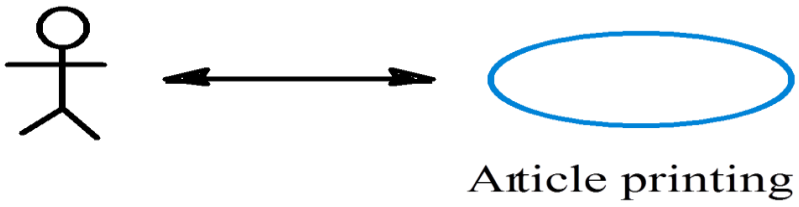
Use cases

Use-cases are a scenario based technique in the UML which identify the actors in an interaction and which describe the interaction itself.

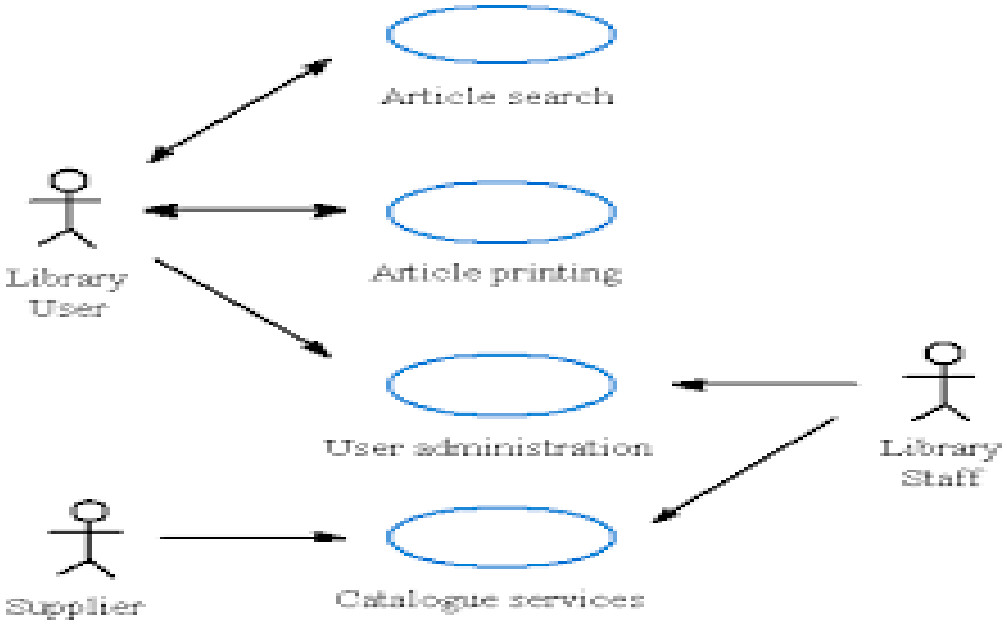
A set of use cases should describe all possible interactions with the system.

Sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system.

Article printing use-case



LIBSYS use cases



3_ Requirements validation

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

Requirements error costs are high so validation is very important

Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error

Requirements checking

Validity: Does the system provide the 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 validation techniques

Requirements reviews Systematic manual analysis of the requirements.

Prototyping Using an executable model of the system to check requirements.

Test-case generation Developing tests for requirements to check testability.

Requirements reviews

Regular reviews should be held while the requirements definition is being formulated.

Both client and contractor staff should be involved in reviews.

Reviews may be formal (with completed documents) or informal. Good communications between developers, customers and users can resolve problems at an early stage.

Review checks

Verifiability: Is the requirement realistically testable?

Comprehensibility: Is the requirement properly understood?

Traceability: Is the origin of the requirement clearly stated?

Adaptability: Can the requirement be changed without a large impact on other requirements?

4_ Requirements management

Requirements management is the process of managing changing requirements during the requirements engineering process and system development.

Requirements are inevitably incomplete and inconsistent

New requirements emerge during the process as business needs change and a better understanding of the system is developed;

Different viewpoints have different requirements and these are often contradictory.

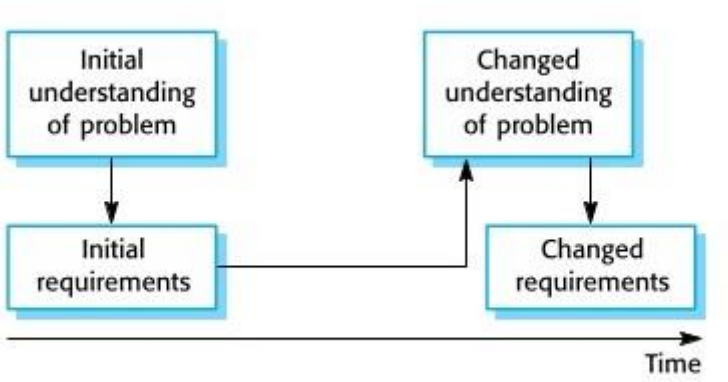
Requirements change

The priority of requirements from different viewpoints changes during the development process.

System customers may specify requirements from a business perspective that conflict with end-user requirements.

The business and technical environment of the system changes during its development.

Requirements evolution Diagram



Enduring and volatile requirements

Enduring requirements: Stable requirements derived from the core activity of the customer organization. E.g. a hospital will always have doctors, nurses, etc. May be derived from domain models

Volatile requirements: Requirements which change during development or when the system is in use. In a hospital, requirements derived from health-care policy

Requirements classification

Type	Description
Mutable requirements	Requirements that change because of changes to the environment in which the organisation is operating.
Emergent requirements	Requirements that emerge as the customer's understanding of the system develops during the system development.
Consequential requirements	Requirements that result from the introduction of the computer system.
Compatibility requirements	Requirements that depend on the particular systems or business processes within an organization.

Requirements management planning

During the requirements engineering process, you have to plan:

Requirements identification: How requirements are individually identified.

A change management process: The process followed when analyzing a requirements change.

Traceability policies: The amount of information about requirements relationships that is maintained.

CASE tool support: The tool support required to help manage requirements change.

Traceability

Traceability is concerned with the relationships between requirements, their sources and the system design

Source traceability: Links from requirements to stakeholders who proposed these requirements.

Requirements traceability: Links between dependent requirements.

Design traceability: Links from the requirements to the design.

CASE tool support

Requirements storage: Requirements should be managed in a secure, managed data store.

Change management: The process of change management is a workflow process whose stages can be defined and information flow between these stages partially automated.

Traceability management: Automated retrieval of the links between requirements

Requirements change management

Should apply to all proposed changes to the requirements.

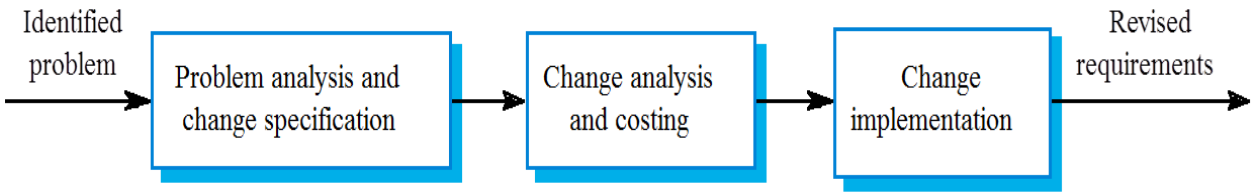
Principal stages

Problem analysis: Discuss requirements problem and propose change;

Change analysis and costing: Assess effects of change on other requirements;

Change implementation: Modify requirements document and other documents to reflect change.

Change management Diagram



Formal Specification

Formal methods

Formal specification is part of a more general collection of techniques that are known as “formal methods”.

These are all based on mathematical representation and analysis of software.

Specification in the software process

1- Specification and design are inextricably intermingled.

2- Architectural design is essential to structure a specification and the specification process.

3- Formal specifications are expressed in a mathematical notation with precisely defined vocabulary, syntax and semantics.

4- One of the main benefits of formal specification is its ability to uncover problems and ambiguities in the system requirements.

Formal specification in the software process

1- If a formal specification of the software is developed, this usually comes after the system requirements have been specified but before the detailed system design. There is a tight feedback loop between the detailed requirements specification and the formal specification.

2- One of the main benefits of formal specification is its ability to uncover problems and ambiguities in the system requirements.

3- The involvement of the client decreases and the involvement of the contractor increases as more detail is added to the system specification.

Use of formal specification

1- Formal specification involves investing more effort in the early phases of software development.

2- This reduces requirements errors as it forces a detailed analysis of the requirements.

3- Incompleteness and inconsistencies can be discovered and resolved.

4- Hence, savings as made as the amount of rework due to requirements problems is reduced

Cost profile

The use of formal specification means that the cost profile of a project changes

There are greater upfront costs as more time and effort are spent developing the specification.

However, implementation and validation costs should be reduced as the specification process reduces errors and ambiguities in the requirements.

Specification techniques

1- Algebraic specification: The system is specified in terms of its operations and their relationships.

2- Model-based specification: The system is specified in terms of a state model that is constructed using mathematical constructs such as sets and sequences. Operations are defined by modifications to the system's state.

The structure of an algebraic specification

< SPECIFICATION NAME>

sort < name >

Imports < LIST OF SPECIFICATION NAMES>

Informal description of the sort and its operations

Operation signatures setting out the names and the types of

the parameters to the operations defined over the sort

Axioms defining the operations over the sort

Specification components

Introduction Defines the sort (the type name) and declares other specifications that are used.

Description Informally describes the operations on the type.

Signature Defines the syntax of the operations in the interface and their parameters.

Axioms Defines the operation semantics by defining axioms which characterize behavior.

Systematic algebraic specification

Algebraic specifications of a system may be developed in a systematic way

1- Specification structuring.

2- Specification naming.

3- Operation selection.

4- Informal operation specification.

5- Syntax definition.

6- Axiom definition.

Specification operations

Constructor operations: Operations which create entities of the type being specified.

Inspection operations: Operations which evaluate entities of the type being specified.

To specify behavior, define the inspector operations for each constructor operation.

Operations on a list ADT

1- Constructor operations which evaluate to sort List Create, Cons and Tail.

2- Inspection operations which take sort list as a parameter and return some other sort Head and Length.

3- Tail can be defined using the simpler constructors, Create and Cons. No need to define Head and Length with Tail.

Computer Science

Chapter (6)

Use of application architectures:

- As a starting point for architectural design.
  كنقطة انطلاق للتصميم المعماري.
- As a design checklist. كقائمة مراجعة التصميم
- As a way of organising the work of the development team.
  كطريقة لتنظيم عمل فريق التطوير.
- As a means of assessing components for reuse.
  كوسيلة لتقييم مكونات لإعادة الاستخدام.

Application types:

1- **Data processing applications**
Data driven applications that process data in batches without explicit user intervention during the processing.
التطبيقات المستندة إلى البيانات التي تعالج البيانات على دفعات دون تدخل صريح من المستخدم أثناء المعالجة.

Ex: Billing systems, Payroll systems

2- **Transaction processing applications**
Data-centered applications that process user requests and update information in a system database.
التطبيقات التي تتمحور حول البيانات والتي تعالج طلبات المستخدمين ومعلومات التحديث في قاعدة بيانات النظام."تعديل البيانات الموجوده في "database

Ex: E-commerce systems, Reservation systems

Transaction management middleware or teleprocessing: monitors handle communications with different terminal type
عبارة عن وسيط لكي يتمكن من التعامل مع الاختلاف بين الاجهزة او البرمجيات لكي يتم التعامل مع ال database.

3- **Event processing systems**
Applications where system actions depend on interpreting events from the system's environment.
التطبيقات التي تعتمد فيها إجراءات النظام على تفسير الأحداث من بيئة النظام." نظام يعتمد علي حدث معين "

Ex: Word processors, Real-time systems – temperature system

4-Language processing systems

Applications where the users' intentions are specified in a formal language (such as JAVA) that is processed and interpreted by the system. (**complier**).

التطبيقات التي يتم فيها تحديد نوايا المستخدمين بلغة رسمية مثل (JAVA) والتي تتم معالجتها وتفسيرها بواسطة النظام. "عمل لغة عن طريق كتابة مجموعة اوامر ويتم ترجمتها الي لغة الالي للجهاز"

Ex: Compilers, Command interpreters

2.1 Information systems architecture:

Layers include:

- The user interfaces
- User communications
- Information retrieval
- System database

2.2 Resource allocation systems:

- A resource database.
- A rule set describing how resources are allocated.
- A resource manager.
- A resource allocator.
- User authentication.
- Query management.
- Resource delivery component.
- User interface.

2.3 Layered system implementation:

- Each layer can be implemented as a large-scale component running on a separate server.

  يمكن تنفيذ كل طبقة كمكون واسع النطاق يعمل على خادم منفصل

- On a single machine, the middle layers are implemented as a separate program that communicates with the database through its API

على جهاز واحد ، يتم تطبيق الطبقات المتوسطة كبرنامج منفصل يتصل بقاعدة البيانات من خلال API الخاص به.

- Fine-grain components within layers can be implemented as web services.

Language processing components:

- Lexical analyzer
- Symbol table
- Syntax analyzer
- Syntax tree
- Semantic analyzer
- Code generator
- تحليل – جدول للرموز – محلل بناء الجمله – النحو – المعني – الكود .