

2010

SOFTWARE ENGINEERING

Ala' kh. Showish  
Basrah University

## SOFTWARE ENGINEERING (SE)

### هندسة البرمجيات:

#### برمجيات الحاسوب: The Computer Software

عبارة عن المنتج الذي يصممه ويبنيه مهندسو البرمجيات ، ويشمل البرامج القابلة للتنفيذ ضمن الحاسوب من أي حجم وأي معمارية ، والبيانات التي تضم أرقام ونصوص ، وال software تأخذ الصيغ التالية:

#### 1- الايعازات: Instructions

برامج الحاسوب ، والتي عندما تنفذ تعطي الوظيفة المطلوبة.

#### 2- هياكل البيانات: Data structures

والتي تمكن البرامج لكي تعالج المعلومات على نحو كافٍ.

#### 3- الوثائق: Document

والتي تصف تشغيل واستخدام البرنامج.

#### مفهوم هندسة البرمجيات: Software Engineering

نحن نستخدم معرفتنا بالحاسوب والتخمين لكي تساعدنا على حل المشاكل ، والتي كثيرا ما تكون تلك المشاكل التي نتعامل معها لها علاقة بالحاسوب او بنظام الحاسبة الموجود ، لذلك من الضروري ان نفهم أولا طبيعة المشكلة.

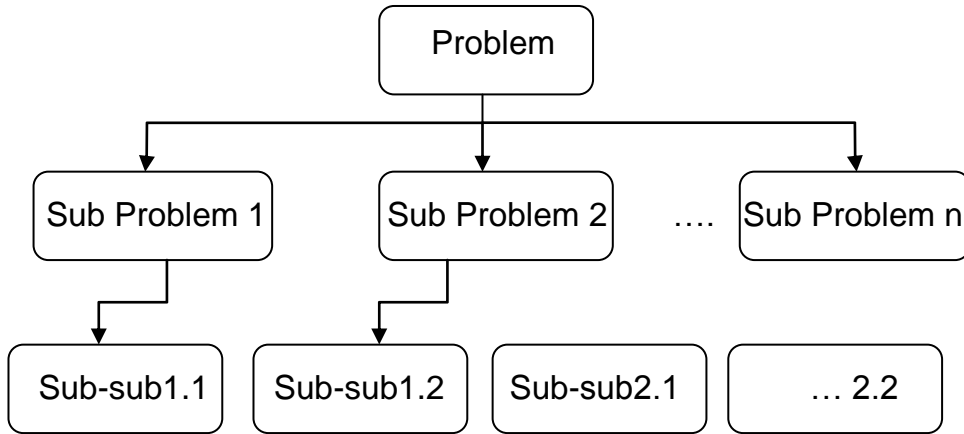
#### حلول المشاكل: Solving Problems

اغلب المشاكل تكون كبيرة وبعض الاحيان تحتاج حيلة ، خصوصا اذا كانت تمثل شيئا ما جديد لم يحل مسبقا.

كذلك يجب ان تستقصى من خلال التالي: So we must begin investigate it by

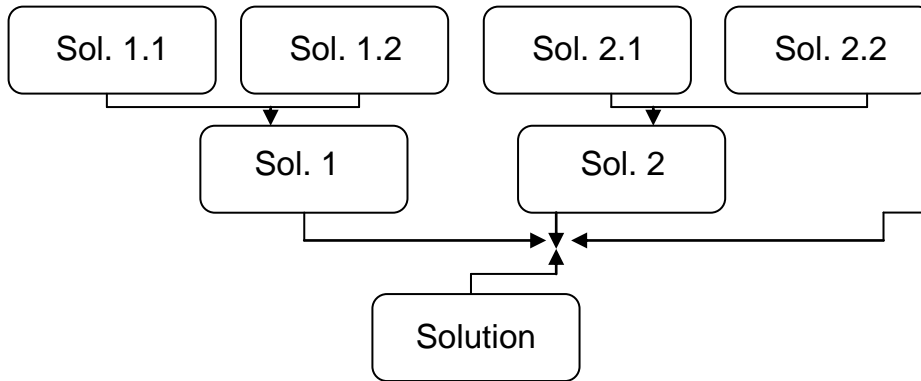
#### 1- تحليل المسألة: Analyzing

نفكك المشكلة الى اجزاء بحيث نستطيع ان نفهمها ، ونحاول البيانات التي نتعامل معها نقسمها بحيث يمكن وصف المشاكل الأكبر كمجموعة من المشاكل الصغيرة المرتبطة مع بعضها ، كما في المخطط التالي:



## 2- تركيب الحلول: Synthesis

بناء الحل من مركبات (مكونات) تعنون المظاهر المتعددة للمشكلة ، كما في الشكل التالي:



إن هندسة البرمجيات تعنى بتصميم وتطوير برامج ذات جودة عالية.

والمشاركون في عملية صناعة البرنامج عادة ما يندرجون تحت ثلاث مجموعات:

- الزبون Customer : وهو الشركة (أو الشخص) الممولة لمشروع تطوير البرنامج المطلوب.
- المستخدم User : الشخص (أو الأشخاص) الذي سوف يقوم فعلا باستعمال البرنامج.
- المطور Developer : وهو الشركة (أو الشخص) الذي سوف يقوم بتطوير البرنامج لصالح الزبون.

الفرق بين علوم الحاسبات وهندسة البرمجيات:

## Difference between Computer Science and Software Engineering

علوم الحاسبات تعنى بالنظريات والطرق التي تشكل الاساس للحاسوب وأنظمة الحاسبات ، بينما هندسة البرمجيات تعنى بالمشاكل العملية في إنتاج الـ software .

### معالجة البرمجيات: Software Process

معالجة الـ software عبارة عن مجموعة من الأنشطة المتبعة لتصميم أو إنتاج software معين . وهي عبارة عن عمليات عقلية وإبداعية تعتمد على الإنسان وتحتاج الى تفكير وتدقيق وإدراك المسألة لإيجاد الحل ، وليس من الممكن ان تكون عملية أوتوماتيكية ( automate ) من خلال نظام معين كآلة أو معمل لإنتاج software ، والمحاولات لجعل عملية إنتاج software أوتوماتيكياً واجهت نجاح محدود جداً .

وتوجد أربع أنشطة معالجة أساسية ، وهي عامة في أغلب عمليات إنتاج software :

#### ١ - تحديد الأساسيات: Software Specification

الزبون والمهندسون يحدد الـ software الذي تحت البناء والقيود التي توضع على عملياته ، وعمليات الإدخال والإخراج والوظائف وغيرها .

#### ٢ - التطوير: Software Development

حيثما الـ Software هو مصمم ومبرمج.

#### ٣ - التحقق: Software Validation

حيثما الـ Software هو محقق ومنفذ لتأكد من انه هذا الذي يطلبه الزبون.

#### ٤ - التعديل: Software Evolution

حيثما الـ Software هو معدل ومصحح الأخطاء لموائمة الزبون الجديد ومتطلبات السوق.

### نموذج المعالجة: The Software Process Model

هي طريقة وصف وتمثيل مجموعة الأنشطة لتصميم Software معين حتى أبدأ بتنفيذها فعلياً ، وكل نمط معالجة يمثل المعالجة من منظور خاص ، و نماذج المعالجة تتضمن فعاليات ذلك الجزء من معالجة الـ Software المنتج ودور الأشخاص الذي يعملون في هندسة البرمجيات.

بعض الأمثلة على أنواع طرق المعالجة Process Model :

### ١ - نموذج سير العمل: A workflow model

عبارة عن التتابع من الفعاليات في طول المعالجة مع وراثته الإدخال ، والإخراج ، والاعتماديات (dependencies) ، والفعاليات في هذا النموذج تمثل النشاط الإنساني.

### ٢ - سير البيانات (نموذج نشاط): Data Flow / activity model

تعرض المعالج كمجموعة من الفعاليات ، كل واحدة منها تنفذ بعض عمليات نقل البيانات (transformation) ، و عملية سير البيانات الـ Data flow تبين كيفية وصول المدخلات حتى تصبح مخرجات.

### ٣ - الأدوار (الأفعال): A role / action model

تمثل ادوار الأشخاص الذين يعملون بهذا الـ Software وكذلك الفعاليات المرتبطة به او التي تقع على عاتقه.  
اغلب الموديلات للـ Software تكون معتمدة على واحد من ثلاث موديلات عامة (نماذج تطوير الـ Software):

- ١ - الطريقة الشلالية : -The Waterfall approach
- ٢ - طريقة التطوير التكرارية : - iterative development
- ٣ - هندسة البرمجيات المبنية على أساس المكونات : - Component – Based Software Engineering (CBSE)

### ١ - الطريقة الشلالية : -The Waterfall approach

هذه الطريقة تأخذ الفعاليات وتعتبر كل واحدة على شكل طور منفصل وكل مرحلة تنتهي وتبدأ المرحلة التي بعدها ، مثال ذلك: تحديد المتطلبات ( requirement ) ، تحديد الأساسيات ( specification ) ، التصميم ( software design ) ، التنفيذ ( implementation ) ، الاختبار ( testing ) ، الخ.  
يتمنى النموذج الانحداري بالبساطة، ولذا فإنه يسهل على المطور توضيح لهدف سير العمل بالمشروع للعميل والمراحل المتوقعة من العمل ، وقد كان هذا النموذج أساس عمل كثير من

المؤسسات لفترة طويلة مثل وزارة الدفاع الأمريكية ، إلا أن لهذا النموذج العديد من العيوب ، أهمها أنه لا يعكس الطريقة التي يعمل بها المطورون في الواقع. باختصار النموذج الانحداري سهل الفهم و بسيط في إدارته. لكن مميزاتة تبدأ في التداخي بمجرد أن يزداد تعقيد المشروع. (( راجع الرسم في الصفحة 20 ))

## ٢ - طريقة التطوير التكرارية : iterative development -

هذه الطريقة تختلف عن الأولى حيث يحصل هنا تداخل ، أي قبل أن تنتهي المرحلة الحالية ممكن أن تبدأ مرحلة جديدة ، مثال ذلك ممكن أن تتداخل فعاليات تحديد الأساسيات (specification) ، التطوير (development) و التحقق (validation) . هذه الطريقة اسرع من الطريقة الشلالية ، وممكن عرضها على الزبون لتصل الى مرحلة تلبي متطلباته ، أي أنه يمكن المطورين من الحصول على ملاحظات وتقييم الزبون مبكرا و بصورة منتظمة ، ورصد الصعوبات المحتملة قبل التماخي بعينها في عمليات التطوير ، كما أنه يمكن من اكتشاف مدى حجم و تعقيد العمل مبكرا. ممكن حل بديل لهذه الطريقة باستخدام طريقة أكثر مهيكلة وصولا لمنتج أكثر متانة ونظام قابل للدعم .

## ٣ - هندسة البرمجيات المبنية على أساس المكونات :

### - Component – Based Software Engineering (CBSE)

هذه الطريقة تفترض وجود أجزاء من النظام و عملية التطبيق تعتمد على عملية تكامل هذه الأجزاء مع النظام . (التفصيل صفحة 22).

### طرق هندسة البرمجيات : Software engineering methods

عبارة عن طرق مهيكلة لتطوير Software معين والهدف منها هو عملية تسهيل انتاج Software عالي الجودة وبتكلفة بسيطة.

وطرق الهندسة Methods : هي الطرق أو الخوارزميات المختلفة لإنشاء وتصميم النظام أو البرنامج والتحقق تدفق البيانات من وجهة نظر المهندس.

### مُنتج البرمجيات : Software Product

نظام ال- Software عادة ما يتركب من عدد من البرامج المنفصلة ( separate- programs ) ، والملفات الشكلية ( configuration ) ، والتوثيقات (Documentation) ، وهي على نوعين : التي تصف تركيب النظام ، والتوثيقات التي توضح للمستخدم كيف يستعمل النظام.

### أنواع البرمجيات : Software types

توجد نوعين أساسية من مُنتج ال- software حسب الاستخدام :

#### ١ - مُنتج عام : Generic Products

وهي برامج مُنتجة لأغراض عامة تطور وتباع في السوق ، ويستطيع اي زبون (شركة) ان يشتريه ويعمل عليه ، ومن الأمثلة عليه :

- مجموعة الأوفس Microsoft Office
- مضاد الفيروس Anti – Virus
- Auto Ran Virus Remover
- مُسرّع تحميل الانترنت Download Manager
- منظم الريجستري Registry Cleaner

#### ٢ - مُنتج مخصص : Customized Products

برامج مُعدة خصيصا للمستخدم حيث يتم التعاقد مع جهة ال- software لتطويره بشكل خاص للزبون او الشركة ، ومن أمثله :

- نظام إدارة الكلية College Administration system
- نظام التسجيل Registration System
- تصميم صفحات ويب Web Sites
- نظام مكتبة Library System

### المواصفات الجيدة لل- Software

لا توجد خصائص محددة مئة بالمئة تحدد ان ذلك ال- software جيد او لا ولكن توجد بعضها مثل ما هي الخدمات التي يقدمها البرنامج ، والخصائص المرتبطة بالبرنامج والتي تعكس جودة البرنامج ، ومن هذه الخصائص :

## Attributes of Good Software

### ١ - قابلية الصيانة : Maintainability

المرونة الكافية للتعدّل في العمليات أو إضافتها أو تغيير الصلاحيات ..

### ٢ - الاعتمادية : Dependability

تنفيذ الأعمال دون أخطاء وعند حصول خلل في الـ software لا يؤثر فيه.

### ٣ - الكفاءة : Efficiency

تنفيذ أكبر قدر ممكن من العمليات في أقصر وقت.

### ٤ - إمكانية الاستخدام : Usability

أن يكون مرن و سهل التعلم و جدي التصميم.

وتكون الأنظمة متأرجحة ما يعني تلك المعايير بين صعود و هبوط وبالتالي على مدى إمكانية توفيق تلك المعايير تكون الأنظمة أفضل.

### المسؤوليات الأخلاقية والمهنية (الاحترافية) :

يجب على مهندس البرمجيات الاهتمام بالنقاط التالية :

### ١ - الخصوصية : Confidentiality

خصوصية العملاء الذين نتعامل معهم حيث يجب علينا نحن المهندسين الحفاظ على أسرارهم ، فعند تصميم النظام يجب علينا مراعاة ذلك .

### ٢ - التخصصية : Competence

لا تضع نفسك في مكان ليس مكانك فحسب عليك تحديد للعميل تخصصك .

### ٣ - الحفاظ على حقوق الملكية : Intellectual Property Rights

مثل عدم نسخ العمل من دون إذن مسبق من المنتج.

### ٤ - سوء استعمال الحاسوب : Computer Misuse

لا تستخدم الحاسوب إلا في الشيء الذي اعد من اجله .



## التحديات التي تواجه هندسة البرمجيات : Key Challenge Facing Software Engineering

### ١ - تحدي عدم التجانس : The Heterogeneity Challenge

يحدث في حال تصميم software لينتكامل مع نظام قديم وليس مع نظام متكامل جديد.

### ٢ - تحدي التسليم : The Delivery Challenge

أي توصيل الـ software بالشكل الصحيح الى الناس وكيفية حفظ حقوق الملكية وكذلك قصر وقت التوصيل.

### ٣ - تحدي الثقة : The Trust Challenge

الثقة بـ software وإمكانياته حيث لا نستطيع ضمان الثقة فكل software يحتوي على ثغرات كما في إصدار نظام التشغيل Windows Vista .

## النظام التقني الاجتماعي : Socio – Technical System

هي مجموعة مكونات مرتبطة فيما بينها وتعمل مع بعضها لانجاز هدف معين وليس بالضرورة ان تعمل مع بعضها لانجاز ذلك الهدف.

الأنظمة التي تتضمن الـ software ممكن ان تقسم الى قسمين:

### ١ - تقنيات أساسها فقط الحاسوب : Technical Computer – Based

### ٢ - النظام التقني الاجتماعي : Socio – Technical System

### ١ - تقنيات أساسها فقط الحاسوب : Technical Computer – Based

هذا النظام يتضمن Hardware و Software لكن مكوناته لا تعتمد على إجراءات procedures و معالجات processes معينة .  
ومن أمثلتها التلفزيون ، TV ، الهاتف الجوال Mobile Phone ، واكثر برمجيات الحاسبات الشخصية (PC) Personal Computer .

### ٢ - النظام التقني الاجتماعي : Socio – Technical System

يتكون من software و hardware و الإنسان people ، وهذا النظام يتأثر بالبيئة الخارجية وبما أن الإنسان احد عناصره إذن سوف تتغير سلوكيات هذا النظام من وقت لآخر تبعا لتغير سلوك الإنسان .

من أمثاله نظام سيطرة و أوامر الشرطة (Police command & Control System) .

### خصائص النظام التقني الاجتماعي :

#### Essential Characteristics of socio – technical system

- ١ - الخصائص الواضحة تكون للنظام ككل ، أي ليست لها علاقة بجزء مفرد من النظام وإنما بأكمله ، بمعنى آخر هذه الصفات لا تتعلق بأجزاء النظام.
- ٢ - هذه الخصائص كثيرا ما تكون غير مقيدة ، أي عند تسليط مدخلات على النظام ليس بالضرورة أن ينتج نفس الإخراج (الإخراج غير محدد).
- ٣ - أهداف النظام والعلاقات بين مكونات النظام وطريقة تفسيرها تحدد فشل أو نجاح النظام.

#### النظام الجزئي : Sub – System

النظام : هو مجموعة وحدات تعمل معا لانجاز وظيفة معينة ، و النظام الفرعي : هو نظام قائم بذاته ويعتمد عليه نظام آخر ويكون جزء من نظام متكامل ، و من أمثاله نظام تحديد المواقع الجغرافية Geographical info system (GIS) ، وكذلك نظام سيطرة و أوامر الشرطة Police command & Control System .

#### الخصائص البارزة للنظام : Emergent System Properties

- هذه المواصفات البارزة ليست بالضرورة ان تكون خاصة لبعض أجزاء النظام .
- بعض هذه المواصفات ممكن أن نشتمها من النظام الجزئي ، وقد يحتوي هذا الجزء على أخطاء تؤثر على النظام الذي نعتمده.

#### أمثلة على المواصفات البارزة : Examples of some emergent properties

##### ١ - الحجم : Volume

الحيز الذي يتقبله النظام ، فكل جزء من النظام له حجم معين .

##### ٢ - التعويل على النظام : Reliability

إمكانية الاعتماد على بعض أجزاء النظام .

### ٣ - الأمانة : Security

يكون له القابلية على مقاومة الهجوم ( فيروس virus ، منتحلي الشخصية hiker ).

### ٤ - إمكانية التصليح : Reparability

### ٥ - إمكانية الاستخدام : Usability

أن يكون بسيط سهل الاستخدام .

توجد نوعين من الخصائص البارزة :

## There are Two Types of emergent properties

### ١ - الخصائص الوظيفية : Functional emergent properties

كل وظيفة لها علاقة بالوظيفة الرئيسية ويعتمد عليها النظام ، أي وظيفة كل جزء يتم تنفيذها حتى ينفذ النظام ككل الوظيفة الرئيسية له.

### ٢ - الخصائص الغير وظيفية : Nonfunctional emergent properties

لها علاقة بالسلوكيات فوظيفة جزء معين لا تؤثر على الوظيفة الرئيسية للنظام ولكن تؤثر على بعض الخصائص مثل الإنجازية (يؤدي الوظيفة مع خلل بسيط) ، والحماية ، والتعويل على البرنامج.

هناك ثلاث تأثيرات متعلقة بإمكانية التعويل او الاعتماد على النظام:

### ١ - اعتمادية القطع المادية : Hardware Reliability

إمكانية حدوث خلل Hardware في النظام وكم يستغرق من الوقت لتصليح هذا الخلل .

### ٢ - اعتمادية البرمجيات : Software Reliability

احتمالية حدوث خطأ بسبب الـ software في النظام .

### ٣ - اعتمادية المشغل : Operator Reliability

إمكانية حدوث خطأ بسبب المشغل لهذا النظام ، قد يكون إدخال خطأ للبيانات او تنفيذ البرنامج بشكل غير صحيح ، وهذا التأثير هو مركب فقد يكون بسبب ضغوط نفسية على المشغل او بسبب الخلل في الـ Hardware او Software وبالتالي فإن مجموع هذه الأخطاء يسبب توقف بالنظام System Shutdown .

## نظام الحماية : Security System

**النظام الأمين :** هو النظام الذي لا يسمح بالوصول غير المصرح به لبيانات هذا النظام ، لكن ليس بالضرورة ان نعرف كل طرق الوصول لكي نمنعها وانما توجد آلية لمنعه مثال ذلك منع مواصفات لموقع معين حيث لا يمكن ذلك فلربما يظهر موقع جديد ليس لدينا مواصفاته.

## هندسة النظم : System Engineering

**هندسة النظم :** تهتم بكل جوانب تطوير النظام وتتضمن الـ Hardware و Software و هندسة المعالج process engineering ، وكذلك تهتم بتفاعل النظام مع المستخدم والبيئة ، كما في النظام التقني الاجتماعي socio – technical system .

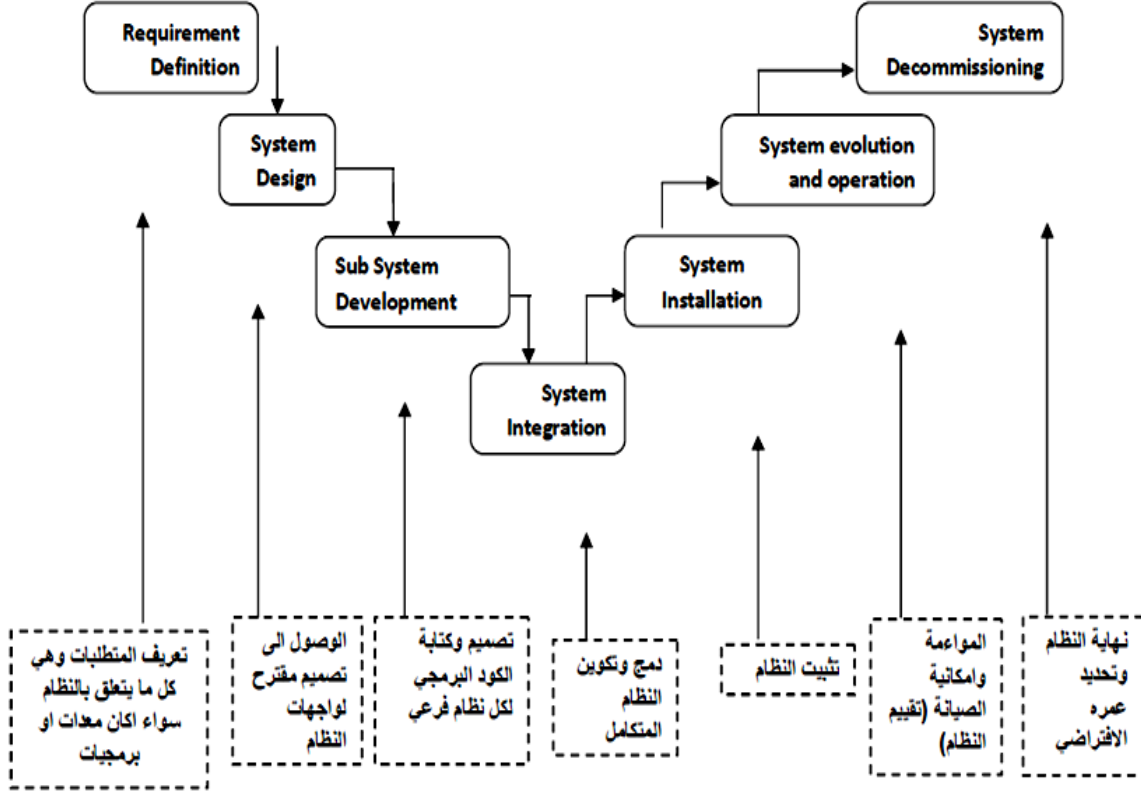
أما هندسة البرمجيات : فهي جزء من هذه المعالجة.

**وهندسة النظم :** هي عبارة عن فعاليات التحديد Specifying والتصميم Designing والتطبيق implementing والتحقق Validating والصيانة Maintaining .

مهندس البرمجيات يحتاج إلى فهم هندسة النظم بسبب مشاكل هندسة البرمجيات والتي كثيرا ما تكون نتيجة لقرارات هندسة النظم .

## أطوار هندسة النظم : The Phases of System Engineering

ويمكن توضيح تلك المراحل بالشكل التالي:



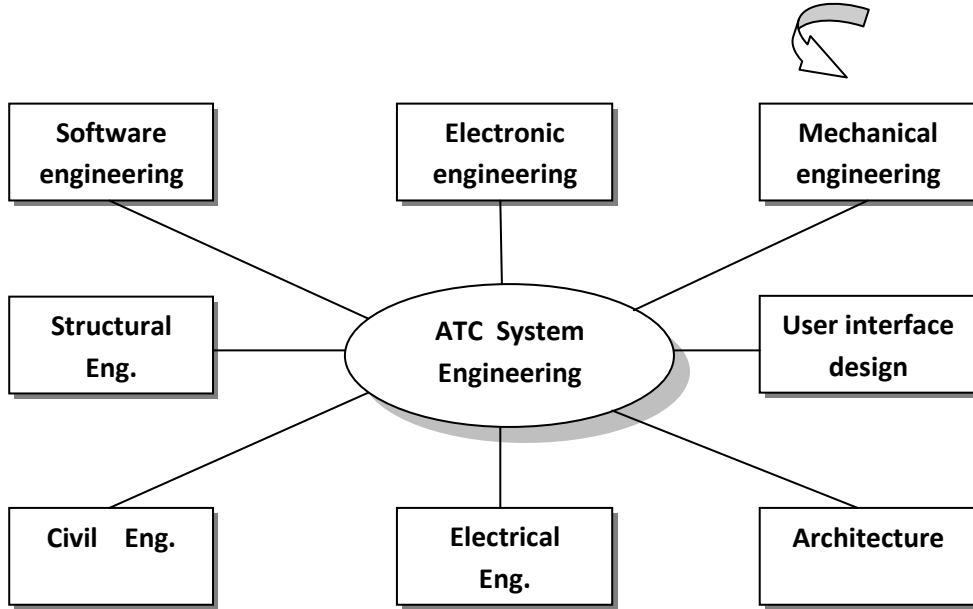
توجد فواصل وفروقات بين أنظمة هندسة المعالج وتطوير برمجيات بمعالج معين:

١ - مجال عملية التغيير محددة خلال تطوير النظام ، عملية إعادة النظر بالنظام خلال تصميمه ليتلائم مع المتطلبات الجديدة تكون سهلة مع أنظمة هندسة المعالج ولكنها صعبة مع تطوير النظام.  
 مثال : لو كان عندنا تصميم شبكة الهاتف الجوال والذي يتكون من معدات Hardware و software ، فتغيير أماكن الأبراج وتقريب المسافات ليس بالشئ السهل وتحتاج الى إمكانيات عالية ، بينما الـ software فتغييره سهل ولا يحتاج الى تلك الإمكانيات .

٢ - نظام الهندسة اكثر انضباطية وأكثر حساسية وتحتاج الى مجال واسع للفهم وليس هناك مجال للاختلاف وسوء الفهم بين المهندسين الذين يستعملون مصطلحات واتفاقيات مختلفة.

### مثال ذلك: نظام التحكم بالرحلات الجوية ATC

حيث يتطلب تصميمه مجموعة من المهندسين باختصاصات مختلفة ( هندسة الميكانيك ، هندسة الإلكترونيات ، هندسة البرمجيات ، هندسة البناء ، الهندسة المدنية ، هندسة الكهرباء ، الهندسة المعمارية ، تصميم واجهات المستخدمين ) والترتيب من اليمين إلى اليسار ← ، وكما في المخطط التالي:



### مرحلة (طور) تعريف المتطلبات : System Requirements Definition

هذا الطور يختص بمسألتين:

١ - ما هي وظيفة النظام.

٢ - خصائص النظام الأساسية والضرورية.

وهذه المرحلة تحدد وظيفة كل نظام وما هي مواصفاته وهو يركز على اشتقاق ثلاث أنواع من المتطلبات :

أ - المتطلبات المجردة لكل وظيفة دون الدخول إلى التفاصيل Abstract Functional Requirements

ب - خصائص النظام الغير وظيفية System Properties .

ت - المميزات Characteristics في النظام والتي يجب أن لا تظهر .

مثال عن هذا الطور :

### نظام إنذار من الحرائق والدخلاء Fire and intruder alarm system

تصميم نظام تحذير من الحرائق والدخلاء لبناية معينة حيث يوفر تحذير داخلي internal و خارجي external للحرائق والدخول غير المصرح به ، هنا حددنا وصف عام لهذا النظام وبعد ذلك نحدد المواصفات الأخرى ومشاكل النظام المتوقعة والصعبة Wicked Problem .

### مرحلة تصميم النظام : System Design

وظيفة هذه المرحلة هي تحديد وظيفة كل جزء من اجزاء النظام ، بعد تقسيم النظام الى عدة اجزاء .  
والمعالجة في هذه المرحلة تتضمن الانشطة التالية:

#### ١ - متطلبات الاجزاء: Partition Requirement

تحليل المتطلبات وتنظيمها في مجموعات ذات علاقة مع بعضها سواء كانت تلك المتطلبات معدات كالطابعات وكاميرات المراقبة او برمجيات معينة فمثلا لو كان النظام كبير الحجم فان الشركة المصنعة له سوف تقسم النظام وتعطي كل فريق عمل من المبرمجين او المحللين جزء من النظام الكلي .

#### ٢ - تعريف اجزاء النظام : identify sub – system

تحديد او تعريف الانظمة الفرعية وذلك حسب تقسيمها للمتطلبات.

#### ٣ - اسناد المتطلبات الى الجزء الفرعي الذي حددناه : Assign Requirement. To sub – system

تأثير تلك المتطلبات المحددة على الانظمة الفرعية بمعنى هل تلك المتطلبات تواءم الانظمة الفرعية اذا كان نعم فننتقل الى المرحلة التالية واذا كان لا فنعود لنحدد ونقسم المتطلبات والانظمة الفرعية لاحظ اننا وضعنا سهم ذات جهتين حيث بإمكاننا ان نعود ان اردنا ذلك.

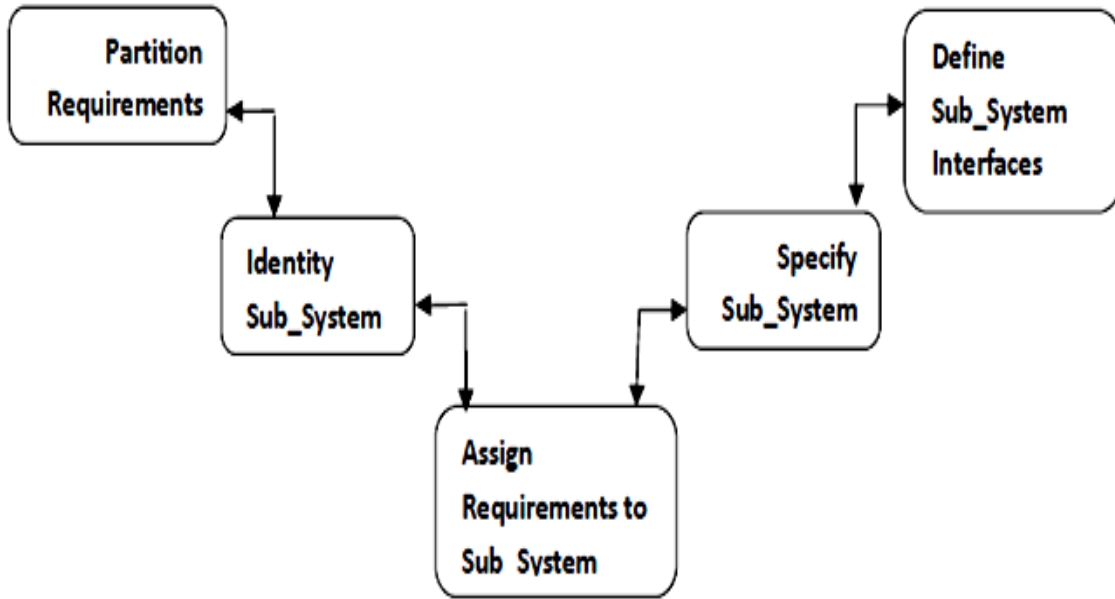
#### ٤ - تحديد وظيفة كل جزء من النظام Specify sub-system Functionality

نحدد ونعطي مواصفات لكل نظام فرعي اي اننا سوف نحدد وظيفة كل جزء من اجزاء النظام الفرعي فلا يكون هناك نظام فرعي دون عمل مثلا.

#### ٥ - تحديد واجهة النظام الفرعي Define sub-system interface

تحديد الشكل والواجهات الخاصة بالنظام ليتسنى تصميم الأنظمة الفرعية على التوازي والتي ممكن ان اربطها مع أي جزء آخر من النظام.

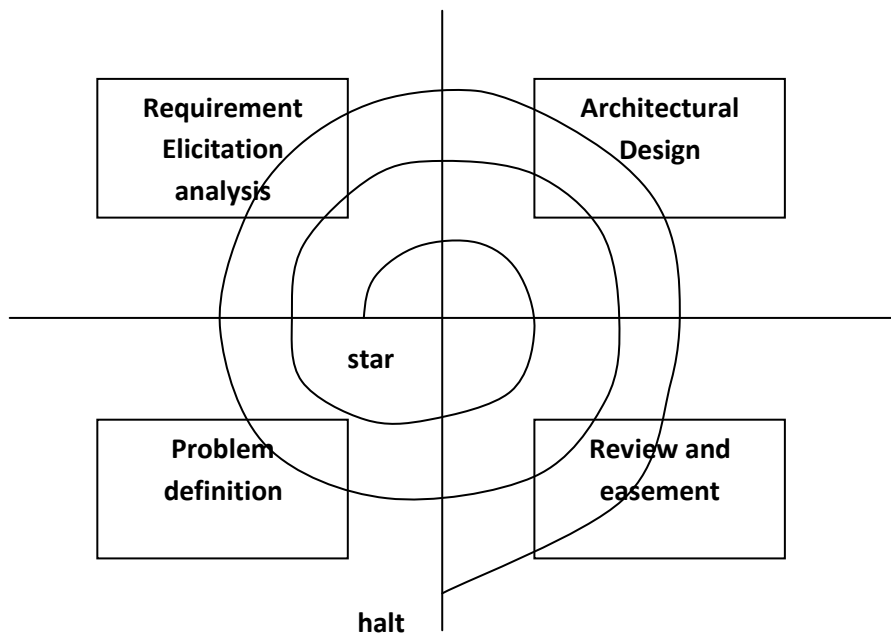
ويمكن توضيح مراحل التصميم بالشكل التالي:



الأسهم الراجعة تساعد في الرجوع الى حل مسألة تم اكتشافها لاحقاً.

النموذج الحلزوني للمتطلبات والتصميم:

### A spiral Model of Requirement and design





ويشمل المراحل التالية :

- ١ - مرحلة التصميم Architectural Design
- ٢ - مرحلة تحليل المتطلبات Requirement Elicitation analysis
- ٣ - مرحلة تحديد المسألة Problem definition
- ٤ - مرحلة المراجعة والتقييم Review and easement

وتستمر هذه المراحل بالدوران متتالين بين الأنظمة الفرعية للنظام حتى نصل الى النسخة النهائية وفي كل مرحلة نضيف أشياء الى ان يكتمل النظام .

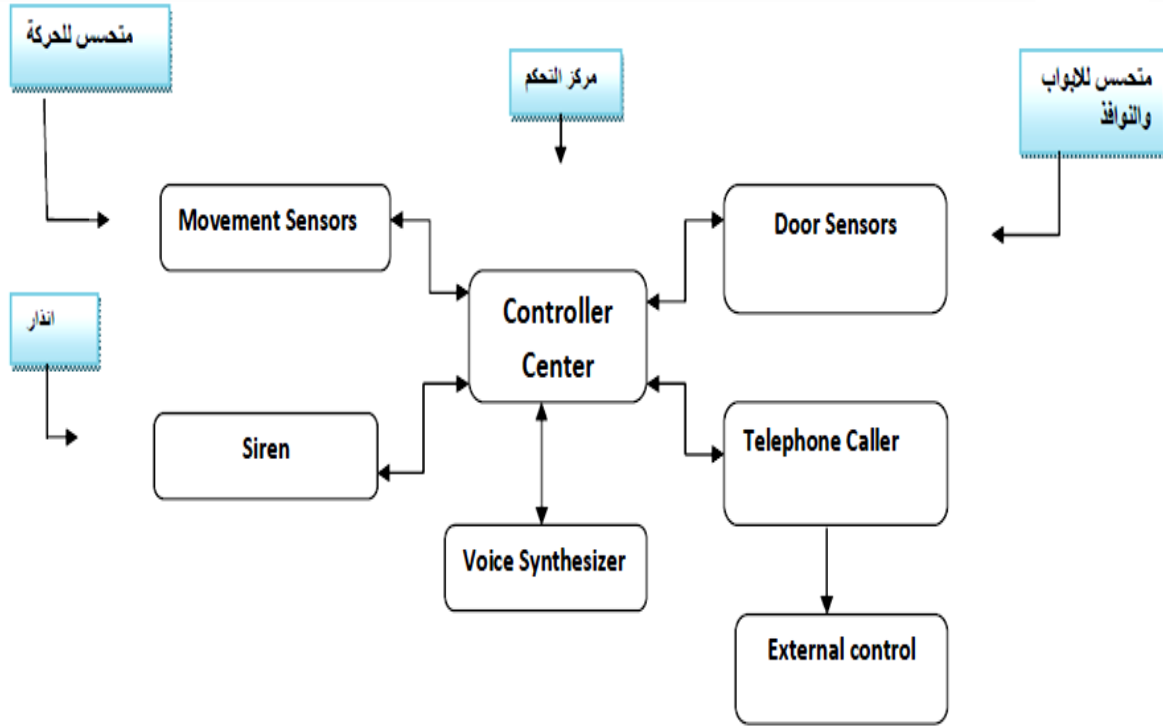
### مرحلة تشكيل (نمذجة) النظام System Modeling

خلال مرحلتي المتطلبات والتصميم ؛ النظام ممكن ان يُشكل كمجموعة عناصر وعلاقات بينها ، وتكون مشروحة رسومياً بصورة اعتيادية وفي معمارية اي نظام ، بحيث تعطي القارئ نظرة عامة عن النظام ، وتنظيمها بشكل رسوم يسمى (النمذجة – Modeled) .

مثال ذلك : نظام التنبيه عن الدخلاء ، وهو نظام فرعي وجزء من نظام التنبيه عن الحرائق والدخلاء:

#### Simple Alarm System :- ( intruder sub – system )

الشكل التالي يوضح النظام الالي وظيفته وتفاعل مع النتيجة الخارجة وتحسس لها كان يكون نظام بنكي يتحسس لاي فتح لاي باب أو نافذة خارج الدوام الرسمي وبالتالي يقوم باجراء معين كان يقوم بالاتصال بالشرطة أو بمدى البنك أو أن يقوم باطلاق جرس للإنذار.



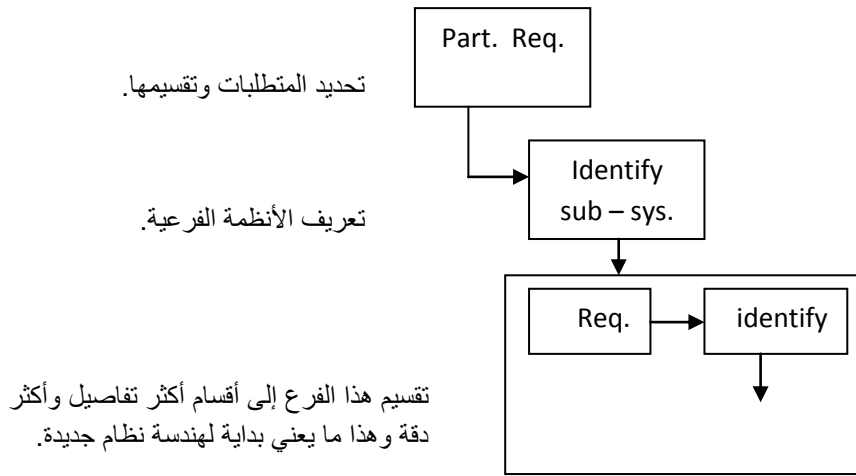
- **Controller Center** يقوم بالسيطرة على التنبيه حيث يستقبل إشارة من المتحسسات او يرسل إشارة الى الأجهزة المرتبطة به.
- **Door Sensor** متحسس للأبواب والنوافذ ويقوم بإرسال الإشارة الى مركز السيطرة.
- **Movement Sensors** متحسسات في الأرضية وتقوم بإرسال الإشارة الى مركز السيطرة.
- **Siren** صفارة الإنذار تطلق صوت الإنذار بعد استلام الإشارة من مركز التحكم.
- **Voice Synthesizer** صوت مركب وتعمل بعد استلام الإشارة من مركز التحكم.
- **Telephone Caller** جهاز التلفون بعد ان يستلم الإشارة من مركز التحكم يقوم بالاتصال بالشخص او بمركز الشرطة المحدد.
- **External Control Center** مركز سيطرة خارجي .

تم تحرير هذا المقرر ليلة العاشر من محرم 1431 ، ليلة استشهاد أبو الأحرار الإمام الحسين (عليه السلام) والموافق 2009/12/26 .

علاء الزبيدي/ علوم الحاسبات

### مرحلة تطوير النظام الفرعي: sub – System Development

خلال عملية التطوير النظام الفرعي يُحدد خلال عملية التصميم ، أي عملية التطوير تجري على النظام الفرعي الذي تم تحديده مسبقا .  
 وهذا قد يتضمن بداية لعملية هندسة نظام أخرى لنظام فرعي معين.  
 وإذا كان هذا النظام الفرعي عبارة عن software فقد يتضمن إحدى عمليات المعالجة مثل تحديد المتطلبات ، التصميم ، التطبيق ، الاختبار .. الخ.  
 والمخطط التالي يوضح ذلك:



النظام الفرعي عادة ما يُطور على التوازي ، أي لا نستطيع تطوير Sub2 قبل Sub1 لأنه قد تعتمد مدخلات Sub2 على مخرجات Sub1 .

وخلال عملية فحص النظام من المؤكد أن تواجهنا بعض المشاكل وبالتالي نحتاج الى تعديل لبعض أجزاء النظام وهذا التعديل يجب ان تسري نتائجه على بقية أجزاء النظام الأخرى.

### مرحلة تكامل النظام : System integration

هي عملية دمج فروع أجزاء النظام مع بعضها للحصول على النظام المتكامل ، وخلال هذه المرحلة تأخذ الأجزاء التي تم تطويرها وتضعهم سويةً لإكمال النظام ككل ، وهذا التكامل بعد إتمامه ممكن أن نسميه ؛ "big bang" الضربة الكبرى .

## مرحلة تطوير النظام : System Evolution

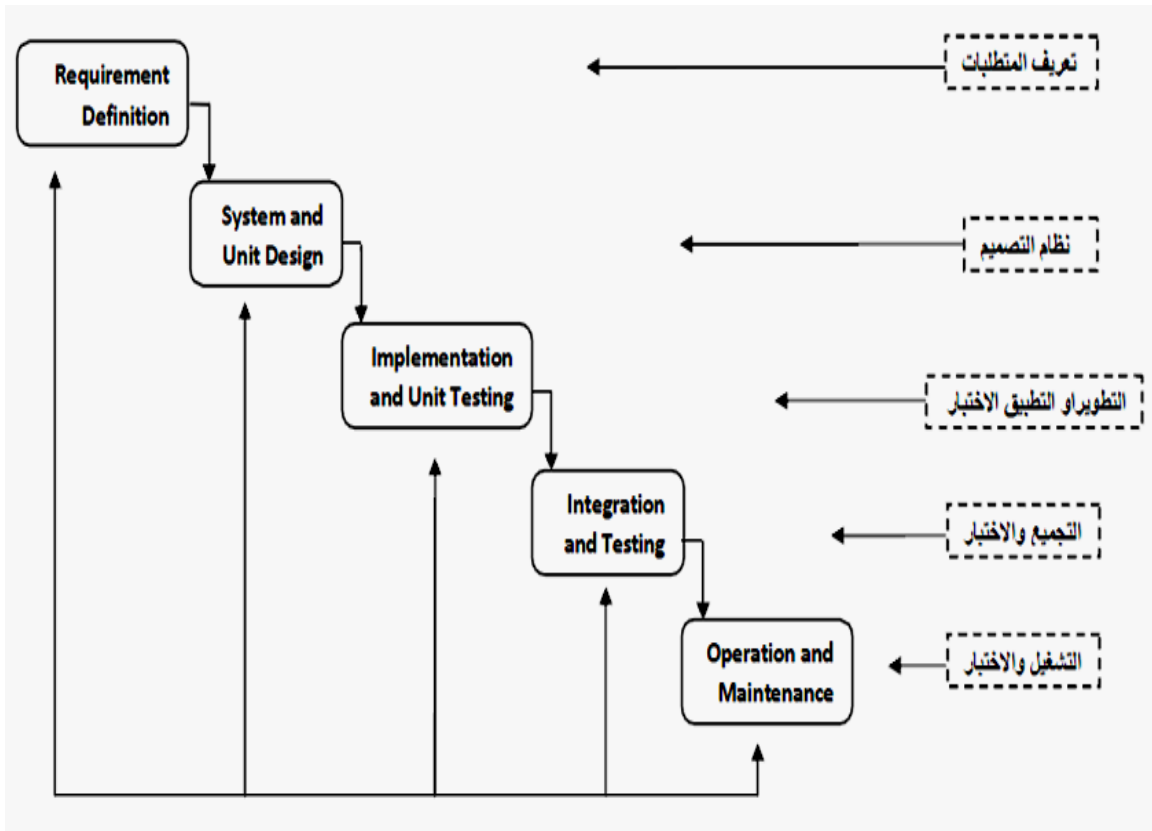
عملية استبدال الأنظمة الكبيرة ليس بالشئ البسيط مثل ATC نظام الملاحة الجوية ، او غيرها من الأنظمة التي نحتاج الى تطويرها.

وتعد من أصعب المراحل لأنه عند تصميم النظام وأردنا إدخال تعديلات جديدة عليه هذه العملية تتطلب منا دراسة أو مراجعة النظام من جديد لأنه قد يكون مر وقت طويل على عملية تصميمه فيجب ان نتذكره ومن ثم نبدأ بإكماله أو إدخال التعديلات عليه ، مع ملاحظة ان النظام الذي نعمل عليه الآن ليس النظام الذي صمّمناه سابقا بل هناك بعض التغييرات كتغيير المتطلبات وغيرها ، كل هذا يضاف الى الصعوبات التي ممكن ان نواجهها في عملية تعديل او تطوير النظام.

## خروج النظام عن الخدمة : System Decommissioning

خروج النظام عن الخدمة ، في الـ Hardware تكون فقط عملية إتلاف الجهاز وهي سهلة نوعا ما ، اما في الـ Software فعملية تدمير النظام بسيطة ولكن قد تكون هناك بيانات مهمة يجب الاحتفاظ بها وخرزنها وقد تحتاج الى تكلفة او Software إضافي ، وهذه تُعد أيضا تكلفة إضافية الى جانب المشاكل الأخرى.

(( رسم تابع للطريقة الشلالية : The Waterfall approach صفحة 5 ))



توضيح لرسم (الطريقة الشلالية) والتي هي احد موديلات الأنشطة SW. Process Models :

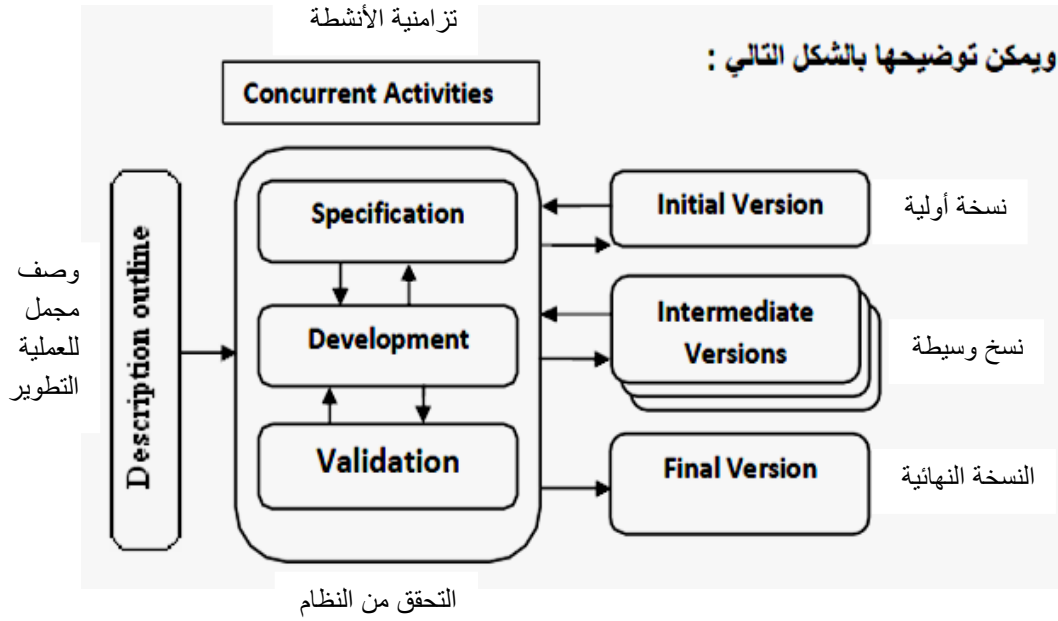
- ١ - تحليل وتعريف المتطلبات: وتشمل خدمات النظام و الأهداف والقيود على النظام التي تحدد مع الزبون.
- ٢ - تصميم برمجيات النظام: تقسيم المتطلبات الى الأنظمة البرمجية والأنظمة المادية ، تتضمن هذه المرحلة تعريف و وصف نظام البرمجيات أو العلاقات بين أجزاء نظام البرمجيات الأساسية.
- ٣ - تطبيق (تنفيذ) العمل واختبار الوحدات (أجزاء النظام): خلال هذه المرحلة يكون المنتج البرمجي على شكل مجموعة برامج أو وحدات وعملية فحص الوحدات هي عملية فحص اذا ما كانت كل وحدة توافق المتطلبات.
- ٤ - عملية تجميع النظام واختباره: تتم عملية تجميع هذه الوحدات الى نظام متكامل للتأكد من ان النظام يوافق المتطلبات الأساسية.
- ٥ - العمل والصيانة : (العمل) هو طور طويل الأمد ، وهي عملية نصب المنتج في الواقع العملي ، (الصيانة) تتضمن تصحيح الأخطاء المكتشفة في مراحل متقدمة من حياة النظام وهذا ما يؤدي الى تحسين وحدات النظام لتوافق المتطلبات الجديدة المكتشفة.

محاسن ومساوي النموذج الشلالي:

- عملية التوثيق تنجز في كل مرحلة ، وهو سهل وبسيط ، ولكنه غير مرن في تقسيم العمل الى مراحل منفصلة .
- ويستخدم هذا النموذج عندما تكون المتطلبات واضحة وسهلة الهم وعدم احتمالية تغييرها خلال العمل .

## 2- طريقة التطوير الارتقائي : Evolutionary Development

وهو أحد طرق SW. Process Models ، ويعتمد هذا النظام على فكرة تطوير تطبيق ابتدائي وعرضه على المستخدم للحصول على تعليقات وتصفيه خلال مجموعة نسخ (إصدارات) Version لحين الوصول الى نظام متكامل .



عملية التطوير في هذه الطريقة تبدأ من المستثمر الذي يقوم بوضع الخطوط العريضة للنظام ومن ثم تقوم الشركة بعملية التحليل للنظام الحالي سواء أكان يوي ام آلي حتى تصل إلى مواصفات النظام الجديد وتحدد المتطلبات والجهود للنظام وبالتالي نحصل على نسخة أولية للنظام وقد يكون النظام القديم هو النسخة الأولية للنظام الجديد في هذه الطريقة وبعد ذلك يمكننا ان ننتقل الى التطوير Development فتصبح لدينا نسخة نسميها بالوسرطة ومن ثم الاختبار للنظام والاختبار هو اختبار للشروط الموضوعه في النظام اي هل يؤدي الوظائف والأغراض المحددة له مسبقا فان كان نعم فننتقل الى النسخة النهائية وإلا فنعود الى مرحلة التطوير وبالتالي تصبح لدينا نسخة وسرطة أخرى فذلك سميها نسخ وسرطة اي قد تكون أكثر من نسخة ومن ثم نعود الى مرحلة الاختبار مرة أخرى وهكذا حتى يتم وضع النسخة النهائية للنظام ، والمسئول عن عملية الانتقال من مرحلة الاختبار الى الوسرطة او النهائية هو فريق الفرقي في التطوير.

يوجد نوعين من هذه الطريقة :

### ١ - الطريقة الاستكشافية : Exploratory Development

يقصد بها عندما تدرس الأهداف والمتطلبات أو تستكشف مع الزبون تبدأ عملية التطوير من أجزاء النظام المفهومة.

### ٢ - الطريقة المباشرة : Throw away , Prototyping

يكون فيها هدف مرحلة التطوير الارتقائي هو فهم المتطلبات من الزبون ، ويكون التركيز على العمل مع الزبون بالمتطلبات الصعبة الفهم .

### محاسن الطريقة الارتقائية :

- ١ - تعتبر أكثر فعالية من الطريقة الشلالية في إنتاج أنظمة تلبي احتياجات الزبون بشكل مباشر .
- ٢ - يلاحظ فيها اختصار الوقت .
- ٣ - اختصار عدد البرمجيات وهو يؤدي إلى تقليل التكلفة وتقليل المخاطرة .

### مساوئ الطريقة الارتقائية :

- ١ - العملية في هذه الطريقة غير واضحة (غير مرئية) أي لا توجد إمكانية لقياس نسبة الانجاز.
- ٢ - لا يوجد توثيق لكل مرحلة من مراحل تطوير النظام .
- ٣ - النظام المصمم بهذه الطريقة يفتقر الى الهيكلية ، وعند التغيير بشكل عام على Software يكون صعب ومكلف .

### الاستخدامات :

- ١ - يستخدم في الأنظمة الصغيرة والمتوسطة .
- ٢ - في الأنظمة الكبيرة ممكن ان ندمج الطريقتين الارتقائية والشلالية .

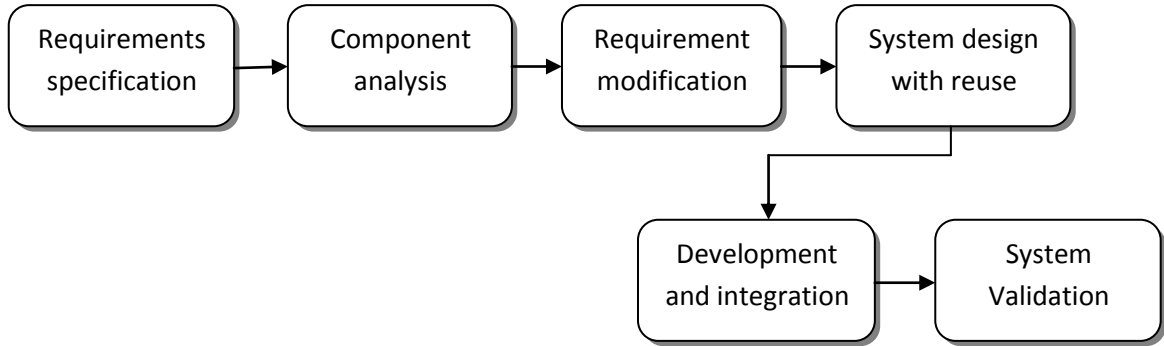
### ٣ - هندسة البرمجيات المبنية على أساس المكونات :

#### **Component – Based Software Engineering (CBSE)**

هذه الطريقة تفترض وجود أجزاء من النظام وعملية التطبيق تعتمد على عملية تكامل هذه الأجزاء مع النظام .

يعتبر مفهوم إعادة الاستخدام Reusability من المفاهيم الأساسية في تصميم البرمجيات ويكون ذلك عندما تكون جزء من متطلبات نظام هي نفس جزء من متطلبات نظام آخر . والعمل بهذه الطريقة يتطلب قاعدة كبيرة من المكونات البرمجية القابلة للاستخدام .

الرسم التالي يوضح هذه الطريقة :



#### -1 Requirements specification

تحديد المتطلبات لمعرفة ما هو المطلوب .

#### -2 Component analysis

يتم فيها البحث عن المكونات لتلبية المتطلبات ، ليس بالضرورة ان تكون متطابقة .

#### -3 Requirement modification

يتم تحليل المتطلبات اعتمادا على معلومات المكونات الموجودة .

#### -4 System design with reuse

يتم تصميم الإطار العام للنظام أو استخدام إطار موجود مسبقا ، مع الأخذ بالحسبان المكونات التي سوف تستخدم .

#### - 5 Development and integration

يتم فيها تجميع المكونات الموجودة وتطوير المكونات الجديدة والتي لا يوجد ما يقابلها حيث يتم تصميم العناصر الجديدة وربطها مع العناصر الموجودة سابقا .

#### - 6 System Validation

يتم فيها التحقق من النظام .

إلى هنا انتهى مقرر الفصل الأول بحول الله وتوفيقه .

علاء الزيايدي/ علوم الحاسبات