AL- Mustansirya University
College of Education
Department of Computer Science

Course: Software Engineering
Lecturer: Iman Hussein
Third Class

CHAPTER 2

Software Development Models

نماذج إعداد المنظومات

Topics:

2.1 Software Lifecycle
2.2 Software development
2.3 Software Development Models

2.1 Software Development Life cycle (دورة حياة إعداد المنظومة)

Each software product proceeds to a number of distinct stages, these are:

- **Requirements engineering**
- **Software design**
- **Software construction**
- **Validation and verification**
- **Software testing**
- **Software deployment**
- **Software maintenance**

Depending the software process used for the development of the software product, these stages may occur in different orders, or frequency.

2.1.1 Requirements Engineering (**requirement analysis and definition** by using engineering approach):

Requirements engineering is the interface between customers and developers on a software project. Requirements should make explicit the ideas of the customer about the prospective system.

2.1.2 Software Design

The designers convert the logical software requirements from stage (1) into a technical software design by describe the software in such a way that programmers can write line of code that implement what the requirements specify.

2.1.3 Software Construction

Software construction is concerned with implementing the software design by means of programs in one or more programming languages and setting up a build management system for compiling and linking the programs.

This stage content several steps, these are :

a. Software reuse

1. Component based software engineering
2. Software product lines

b. Security and reliability

c. Software documentation

d. Coding standards

a. Software Reuse

The goal of software engineering is to achieve many features with little effort and few defects. Software reuse is believed to play an important role in achieving this goal by encapsulating effort in units of source code, which can be reused in other projects. However, the effort needed to make something reusable may not be worth it, if it is only reused few times, or needs extensive adaptation for each reuse.

a.1. Component Based Software Engineering

Building software systems from prefab software components is an old dream of software engineering.

a.2. Software Product Lines

Software systems are often part of a family of similar systems. The goal of software product line is to maintain a set of reusable core artifacts that are common to all systems in the product line. Thus, code for a specific product can focus on the specifics of that product, reusing the common functionality.

b. Security and Reliability

Software must be dependable by making it reliable (software should work very well under any environments), secure and safety (by verifying from user authentication to using any system).

c. Software Documentation

- User documentation?
- Technical documentation?
- Documentation generation?

d. Coding Standards

Coding standards are important to ensure portability and make code maintainable by others than the original developer.

2.1.4 Validation and Verification

- Software inspection
- Software testing

2.1.4. A) Software Inspection

Software inspections are reviews of the code with the purpose of detecting defects. In an inspection someone other than the programmer reads a program unit of limited size to determine whether it satisfies the requirements and specification. A formal process and checklist are used to ensure that no aspects are forgotten.

2.1.4. B) Software Testing

Testing each unit founded in this software, follow by testing software integration.

2.1.5 Software Deployment

After development, software should be put to use. That is, it should be released and made available to users, who can then download, install, and activate it. These activities are captured under the common term software deployment. Richard S. Hall in the 'Software Deployment Information Clearinghouse' defines software deployment as follows: "The term software deployment refers to all of the activities that occur after a software system has been developed and made available for release. As such, software deployment includes activities such as packaging, releasing, installing, configuring, updating, and uninstalling a software system." and "Software deployment is the assembly and maintenance of the resources necessary to use a version of a system at a particular site".

The following deployment activities make up the software deployment process:

Release
Packaging
Transfer
Installation
Configuration
Activation
De-activation
Update
Adapt
De-installation
De-release

These activities are not necessarily performed sequentially. Many phases of the deployment process are often performed manually.

For example, downloading, building and installing a source distribution of a software package, requires a number of commands to be formulated and executed. Each such command requires knowledge of some sort about the activity.

Manual deployment does not scale when deploying:

many applications
applications composed from separately deployed components
on multiple machines
on different types of machines

2.1.6 Software Maintenance

As software evolves after its first release, software maintenance is needed to improve it, i.e., repair defects, and to extend it, i.e., add new functionality.

2.1 Software Development Life cycle (دورة حياة إعداد المنظومة)

منذ بداية السبعينات بدا إعداد المنظومات يتم بطريقة صحيحة وفي شكل مشروع أسوة بالمشاريع الهندسية الاخرى ،وهذا المشروع يتكون من مجموعة مراحل عرفت باسم دورة حياة إعداد المنظومات (Software Development Life cycle).

- دورة حياة البرمجيات – Software lifecycle: يخضع تطوير البرمجيات إلى دورة حياة، وهي عبارة عن مجموعة أنشطة مرتبطة يُدار عبرها أي مشروع تطوير، وتشكل الإجرائيات والطرق آلية تحقيقها، وتعرّف دورة الحياة المراحل التي على المنتج البرمجي أن يجتازها بدءاً من الاستطلاع الأولي وحتى النهاية.

و دورة حياة إعداد اي منتج تمر بعدد من المراحل المتميزة، وهذه هي:

- Requirements engineering (المتطلبات الهندسية)
- Software design(تصميم المنظومة)
- Software construction(بناء المنظومة)
- Validation and verification(المصادقة والتحقق)
- Software testing(اختبار المنظومة)
- Software deployment(نشر او توظيف المنظومة)
- Software maintenance(صيانة المنظومة)

2.1.1 Requirements Engineering (requirement analysis and definition by using engineering approach):

ان الهدف الاساسي من مرحلة التحليل هو فهم متطلبات المستخدم (user requirement) وتنظيم وتمثيل modeling هذه المتطلبات في شكل مخططات وجداول لتتم الموافقة عليها مع الزبون واعتبارها اساسا للمرحلة اللاحقة ألا وهي مرحلة التصميم .

وفي مرحلة التحليل يتم ايضا جمع الحقائق حول النظام الحالي وفهم المشاكل التي تعترض سير العمل ، وتستخدم عدة طرق للوصول لهذا الفهم مثل (**المقابلة الشخصية والاستبيان والملاحظة والعرض التجريبي** وغيرها) ومن ثم المتطلبات الوظيفية (Functional requirement) وغير وظيفية (non-functional requirement) للمنظومة المقترح تنفيذها.

ويمكن إعداد المنظومة باستخدام أحد المنهجيات الاتية :

- المنهجية الهيكلية (Structured Methodology) : وتقسم هذه المنهجية المنظومة الى أجزاء وظيفية في شكل هرمي وتستخدم أدوات هيكلية في التحليل والتصميم والبرمجة مثل مخطط انسياب البيانات DFD والمخطط الهيكلي Structure chart.

- المنهجية الشيئية (Object Oriented Methodology) : وتقسم هذه المنهجية النظام الى كائنات لها وظائف وخصائص شبيهة بالكائنات التي نتعامل معها في حياتنا اليومية .وتستخدم ادوات شيئية مثل مخطط الحالة UCD ومخطط الفصيلة class diagram.

- منهجية فصائل المكتبة (Class library) : حيث يستطيع من خلالها معد المنظومة تجهيز منظومة تجريبية او إعداد منظومة متكاملة في خلال فترة وجيزة (ساعات أو أيام).وتحتاج بالطبع الى التدريب عليها ،وتحتوي على الادوات الضرورية لتحليل وتصميم وتنفيذ المنظومات مشتملة ايضا على كيفية تصميم واجهة المستخدم .ومن امثلة هذه الفصائل ما يسمى الفصائل الاساسية لميكروسوفت (Microsoft Foundation) Class MFC : وهي عبارة عن مكتبة فصائل وتسمى ايضا (Application Frameworks).

وتتلخص أهم نشاطات التحليل فيما يلي :

- اكتشاف المجهول من قبل محلل النظم .لهذا وجب علية في البداية الاستماع للزبون والمستخدم وملاحظة مجريات الامور في النظام الحالي .
- الاستقصاء والتنقيح والفهم لمشاكل النظام الحالي وتحديد المتطلبات للنظام الحالي .
- تمثيل ما فهمه بالمخططات وشاشات الحوار والتثبيت من البيانات المؤخذة بعناية.
- بعد فهم المسألة تأتي كتابة وثيقة مواصفات المتطلبات لتصبح على شكل عقد او اتفاقية بين المحلل والزبون للأشياء المطلوب تنفيذها في المنظومة المقترحة .

2.1.2 Software Design

مرحلة التصميم : تأتي هذه المرحلة بعد تحديد المتطلبات من قبل الزبون ،وإعداد مواصفات دقيقة لهذه المتطلبات .حيث يتم في هذه المرحلة (مرحلة التصميم) تمثيل الحل في شكل خرائط وجداول ومخططات تفصيلية نهائية (blueprint) للمنظومة المقترحة التي تلبي المتطلبات المدونة في وثيقة المتطلبات في المرحلة السابقة لها .
وتستخدم هذه المرحلة ادوات تماثل وتشبة الخرائط الانشائية والمعمارية التي تسبق عملية البناء وفي هذه المرحلة يتم

- تحديد المدخلات
- تحديد المخرجات
- تحديد الواجهات interface مع المنظومة التي تشتمل علي
  - واجهة software
  - واجهة hardware
  - واجهة المستخدم
- كتابة للخوارزميات Algorithm
- تصميم البيانات الخاصة بالمنظومة Data design

وفي نهاية مرحلة التصميم يتم إعداد وثيقة تسمى وثيقة مواصفات التصميم (Design specification document) لتتم مراجعتها من قبل الإدارة واعتمادها لتصبح الاساس للمرحلة اللاحقة .

2.1.3 Software Construction

يتم في هذه المرحلة برمجة المنظومة باستخدام إحدى لغات قاعدة البيانات واللغات الأخرى المناسبة لنوع التطبيق تحت الإعداد .

وهذه المرحلة تتم بعدة خطوات وهي :

This stage content several steps, these are :

a. Software reuse

1. Component based software engineering
2. Software product lines

b. Security and reliability

c. Software documentation

d. Coding standards

a. Software reuse

إعادة استخدام البرمجيات : الهدف من هندسة البرمجيات هو تحقيق العديد من الميزات مع القليل من الجهد وقليل من العيوب. ويعتقد إعادة استخدام البرمجيات يلعب دورا هاما في تحقيق هذا الهدف عن طريق غمد الجهد في وحدة شفرة المصدر(كتابة الكود)، والتي يمكن إعادة استخدامها في مشاريع أخرى. ومع ذلك، فإن الجهد اللازم لجعل شيء يمكن إعادة استخدامها قد لا يكون يستحق ذلك، إذا كان يتم استخدامها فقط بضع مرات، أو يحتاج التكيف الواسع لكل إعادة استخدامها.

A.1. عنصر بناء (المركب الاساسي) هندسة البرمجيات

Component based software engineering: بناء نظم البرنامج من مكونات البرامج الجاهزة هو الحلم القديم في هندسة البرمجيات.

A.2. خطوط البرنامج من المنتجات

Software product lines: نظم البرمجيات غالبا ما تكون جزءا من عائلة أنظمة مماثلة (متشابهة). الهدف من خط انتاج البرنامج هو للحفاظ على مجموعة من القطع الأثرية الأساسية القابلة لإعادة الاستخدام التي هي مشتركة بين كل الأنظمة في خط الانتاج. وبالتالي، يمكن كود لمنتج معين تركز على تفاصيل هذا المنتج، وإعادة استخدام الوظيفة المشتركة.

b. Security and reliability

ب. الأمن والموثوقية :يجب أن تكون البرمجيات يمكن الاعتماد عليها من خلال جعلها موثوقة (البرمجيات يجب أن تعمل بشكل جيد جدا تحت أي بيئة) وآمن وسلامة (من خلال التحقق من مصادقة المستخدم لاستخدام أي نظام).

c. Software documentation

ج. توثيق البرمجيات

- وثائق المستخدم؟
- الوثائق التقنية؟
- انشاء الوثائق؟

d. Coding Standards

د. ترميز المعايير: معايير الترميز مهمة لضمان قابلية وجعل رمز للصيانة من قبل الآخرين من المطور الأصلي.

2.1.4 Validation and Verification

المصادقة (Validation) والتحقق (Verification):

- **فحص البرمجيات او تفتيش البرمجيات (Software inspection)**
- **اختبار البرمجيات(Software testing)**

• فحص البرمجيات (Software inspection)

التفتيش في البرنامج لاستعراض الكود والغرض من ذلك للكشف عن العيوب. الشخص في التفتيش من غير المبرمجين يقرأ وحدة البرنامج لتحديد الحجم المحدد ما إذا استوفى الشروط والمواصفات. وتستخدم عملية رسمية وقائمة مراجعة لضمان عدم وجود جوانب تنسى.

• اختبار البرمجيات(Software testing)

اختبار كل وحدة تأسست في هذا البرنامج، واتباع طريق اختبار تكامل البرمجيات.

2.1.5 Software Deployment

بعد تطويرها، ينبغي وضع البرامج للاستخدام. وينبغي اطلاقها وإتاحتها للمستخدمين، الذين يمكن بعد ذلك تحميلها وتركيبها، وتفعيلها. يتم التقاط هذه الأنشطة في إطار مصطلح شائع وهو نشر البرمجيات (software deployment).

ريتشارد Richard S. Hall (نشر البرامج لتبادل المعلومات) يعرف نشر البرامج على النحو التالي:

(يشير مصطلح نشر البرامج لجميع الأنشطة التي تحدث بعد أن تم وضع نظام البرمجيات وإتاحتها للاعلان عنها، نشر البرنامج يتضمن. أنشطة مثل التعبئة والتغليف، والإعلان والتركيب والتكوين والتحديث، وإلغاء نظام البرمجيات "الغاء التنصيب"). و (نشر البرنامج هو تجميع وصيانة الموارد اللازمة لاستخدام إصدار(اصدار نسخة) من نظام في موقع معين).

<u>أنشطة النشر التالية تشكل عملية نشر البرامج:</u>

Release (الاصدار)
Packaging (التعبئة والتغليف)
Transfer (النقل)
Installation (التنصيب او التثبيت)
Configuration (الاعدادات او التكوين)
Activation (التفعيل)
De-activation (الغاء التفعيل)
Update (التحديث)
Adapt (التكيف)
De-installation (الغار التنصيب)
De-release (الغاء الاصدار)

لا يتم تنفيذ هذه الأنشطة بالضرورة بالتسلسل. وغالبا ما تجرى العديد من مراحل عملية النشر يدويا.

على سبيل المثال، التحميل، البناء والتركيب وتوزيع مصدر حزمة برامج، يتطلب عددا من الأوامر ليتم صياغتها وتنفيذها. يتطلب كل هذا الأمر معرفة نوعا عن النشاط.

<u>دليل النشر لا مقياس له عند نشر:</u>

- العديد من التطبيقات
- تطبيقات تتألف من مكونات يتم الاعلان عنها بشكل منفصل
- على آلات المتعددة
- على أنواع مختلفة من الآلات

2.1.6 Software Maintenance

<u>صيانة البرمجيات :</u> مع تطور البرمجيات بعد أول اصدار لها، هناك حاجة إلى برامج الصيانة لتحسينها، أي إصلاح العيوب، وتمديده، أي إضافة وظائف جديدة.

2.2 Software Development

Three phases to develop the software (ثلاث مراحل لتطوير البرمجيات)

1- Definition (التعريف)
2- Design (التصميم)
3- Maintenance (الصيانة)

1- Definition

1- What information to be processed
2- What design constrains exist.
3- What function and performance desired.
4- What interfaces are desired.
5- What validation criteria are required?
6- What is modeling?

2- Design

1- How data structures to be designed.
2- How procedural details to be implemented.
3- How design to be translated into language.
4- How testing is performed.

3- Maintenance

1- Error
2- Adaptation
3- Modification

ثلاث مراحل لتطوير البرمجيات (**Software Development**)

1- التعريف **(Definition)**

2- التصميم **(Design)**

3- الصيانة **(Maintenance)**

1- التعريف (Definition)

1- ما هي المعلومات التي سيتم تجهيزها (معالجتها)

2- ما هي المعوقات الموجودة في التصميم .

3- ما وظيفة وأداء المطلوب(المتطلبات).

4- ما هي الواجهات المطلوبة .

5- ما هي المعايير (الشروط) المطلوبة للتحقق من الصحة ؟.

6- ما هي النماذج؟

2-التصميم (Design)

1- كيف تصمم هياكل البيانات .

2- كيف تنفذ تفاصيل الإجرائية .

3- كيفية تصميم الترجمة إلى اللغة.(كيف يترجم التصميم الى لغة).

4- كيف يتم تنفيذ الاختبار.

3- الصيانة (Maintenance)

1- الخطأ(الاخطاء) (Error)

2- التكيف (Adaptation)

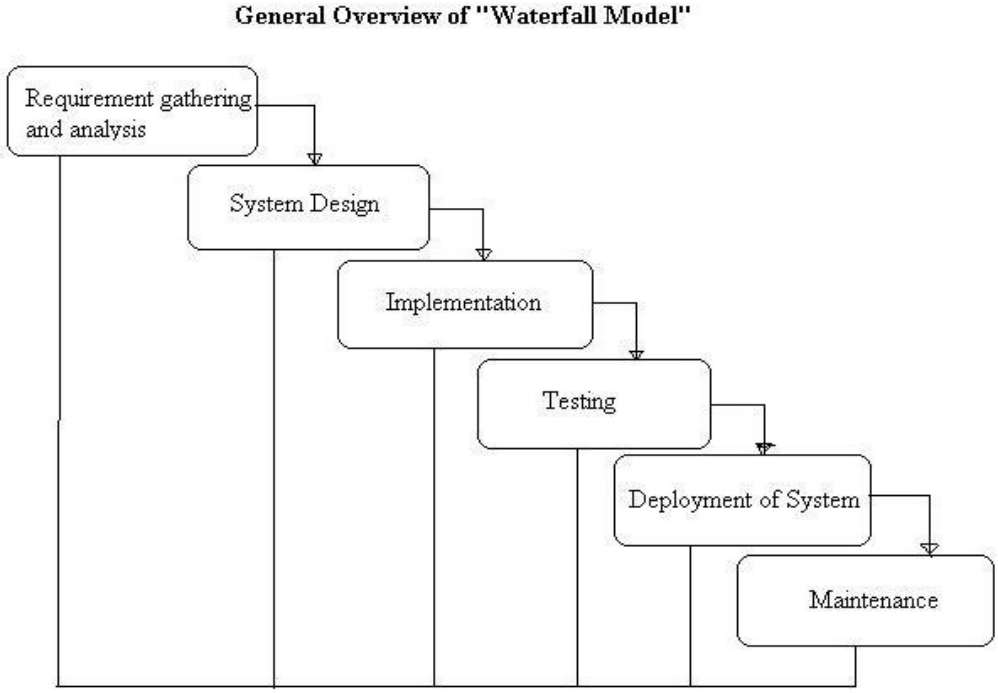
3- التعديل (Adaptation)

2.3 Software Development Models

1- Waterfall Model

The Waterfall Model was first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It is very simple to understand and use. In a waterfall model, each phase must be completed fully before the next phase can begin. This type of model is basically used for the for the project which is small and there are no uncertain requirements. At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project. In this model the testing starts only after the development is complete. In **waterfall model phases** do not overlap.

Diagram of Waterfall-model:



Advantages of waterfall model:

- This model is simple and easy to understand and use.
- It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- In this model phases are processed and completed one at a time. Phases do not overlap.
- Waterfall model works well for smaller projects where requirements are very well understood.

Disadvantages of waterfall model:

- Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.

When to use the waterfall model:

- This model is used only when the requirements are very well known, clear and fixed.
- Product definition is stable.
- Technology is understood.
- There are no ambiguous requirements
- Ample resources with required expertise are available freely
- The project is short.

Very less customer enter action is involved during the development of the product. Once the product is ready then only it can be demoted to the end users. Once the product is developed and if any failure occurs then the cost of fixing such issues are very high, because we need to update everywhere from document till the logic.

النموذج التدفقي Waterfall model

يسمى النموذج الشلال او التدفقي waterfall model (وهو اول نموذج تم استخدامه بداية من السبعينيات) ايضا يعرف بالنموذج التسلسلي Sequential model نظرا لتتابع المراحل وكان طوق النجاة في نهاية الستينيات وله الفضل في حل الكثير من المشاكل التي تمت مواجهتها عند اعداد المنظومات وساهم الى حد ما في حل مشكلة ازمة البرمجيات software crisis حيث كانت المنظومات يتم اعدادها بدون توثيق وتستخدم ادوات بدائية في التحليل والتصميم .

وكان المخطط الانسيابي flowchart الاداة الوحيدة المستخدمة وبذلك كانت الجودة متدنية والاخطاء كثيرة والزبون في اغلب الاحيان غير راض عن المنظومة النهائية .

(متى يجب استخدام نموذج الشلال) When to use the waterfall model

يشترط عند استخدام هذا النموذج :
- عندما تكون المتطلبات معروفة جيدا، وواضحة وثابتة وكاملة.
- تعريف المنتج مستقر.
- التكنولوجيا مفهومة.
- لا توجد متطلبات غامضة
- تتوفر موارد (المصادر وغيرها) متوفرة و ذوي الخبرة المطلوبة متوفرة
- يجب ان يكون المشروع قصير.

وقد تم استخدام هذا النموذج كثيرا في السابق في اعداد المنظومات وهو مقتبس من الطريقة الهندسية في التصنيع والانتاج والبناء من حيث عملية المراحل واستخدام مفهوم المشروع .

اما المراحل التي تخص النموذج التدفقي فهي :

1- مرحلة التخطيط Planning phase
2- مرحلة التحليل Analysis phase
3- مرحلة التصميم Design phase
4- مرحلة التنفيذ Implantation phase
5- مرحلة الاختبار Testing phase
6- مرحلة الصيانة Maintenance phase

Advantages of waterfall model: (المزايا)

- هذا النموذج هو بسيط وسهل الفهم والاستعمال.
- فمن السهل إلادارة نظرا لصلابة هذا النموذج – لان كل مرحلة لها مخرجات محددة وعملية تنقيح (عملية مراجعة)
- في هذا النموذج المراحل يتم معالجتها والانتهاء منها في وقت واحد. اي ان المراحل لايحدث فيها عملية تتداخل (overlap).
- نموذج الشلال يعمل بشكل جيد للمشاريع الصغيرة عندما يكون هناك فهم جيد ودقيق للمتطلبات

Disadvantages of waterfall model: (عيوب نموذج التدفقي)

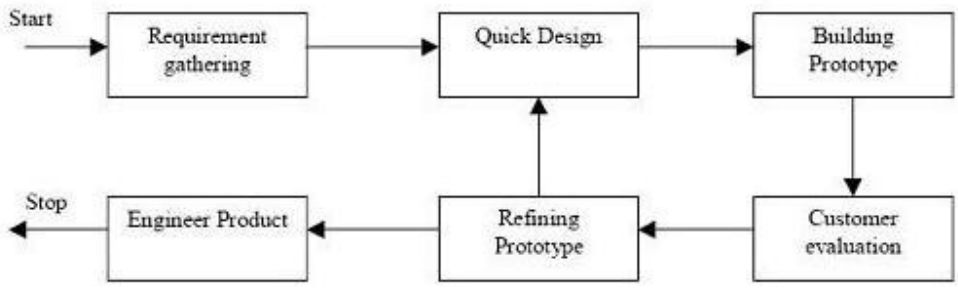
- بمجرد التطبيق في مرحلة الاختبار، فإنه من الصعب جدا الرجوع وتغيير شيء ما لم يكن مدروسة جيدا في مرحلة الفهم (مرحلة التحليل وجمع المتطلبات).
- لايتم إنتاج أي نسخة من المنظومة الا في مرحلة متاخرة من دورة حياة اعداد المنظومة .
- في هذا النموذج يوجد الكثير من المخاطر وعدم اليقين (الشك).
- ليس نموذجا جيدا للمشاريع المعقدة ومشاريع المنهجية الشيئية (object-oriented projects).
- نموذج ضعيف غيرمناسب للمشاريع الطويلة والمستمرة.
- لا تصلح للمشاريع حيث المتطلبات معتدلة (محدودة المدى- متوسطة) إلى عالية و المخاطر متغيرة.

ان هذا النموذج Waterfall Model وكما اشرنا سابقا متداول ومستخدم في اغلب المشاريع منذ بداية السبعينيات .
اما الان فاصبح هذا النموذج لا يواكب تعقد وتنوع المنظومات خصوصا بعد ظهور الشبكات والانترنت والوسائط المتعددة ونظم قواعد البيانات .
والاهم من ذلك لم يحل مشكلة المتطلبات التي في اغلب الاحيان لا تكون دقيقة ولا كاملة . لهذا تم تعديل هذا النموذج بحيث يمكن الرجوع الى المراحل السابقة في حالة تعديل او حدوث تغيير في المتطلبات . وهذا النموذج المعدل هو ايضا لم يحقق الغرض المطلوب .لهذا اصبحت الحاجة لاستخدام نماذج اكثر تطورا ضرورة حتمية ، وسارع المختصون بهندسة البرمجيات الى البحث عن حل بديل وذلك باعداد نماذج اخرى اكثر تطورا وتراعي مشكلة عدم اكتمال ودقة المتطلبات .

2- Prototype model:

The basic idea here is that instead of freezing the requirements before a design or coding can proceed, a throwaway prototype is built to understand the requirements. This prototype is developed based on the currently known requirements. By using this prototype, the client can get an "actual feel" of the system, since the interactions with prototype can enable the client to better understand the requirements of the desired system. Prototyping is an attractive idea for complicated and large systems for which there is no manual process or existing system to help determining the requirements. The prototype are usually not complete systems and many of the details are not built in the prototype. The goal is to provide a system with overall functionality.

Diagram of Prototype model:



Prototyping Model

Advantages of Prototype model:

- Users are actively involved in the development
- Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.
- Errors can be detected much earlier.
- Quicker user feedback is available leading to better solutions.
- Missing functionality can be identified easily
- Confusing or difficult functions can be identified Requirements validation, Quick implementation of, incomplete, but functional, application.

Disadvantages of Prototype model:

- Leads to implementing and then repairing way of building systems.
- Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.
- Incomplete application may cause application not to be used as the full system was designed Incomplete or inadequate problem analysis.

When to use Prototype model:

- Prototype model should be used when the desired system needs to have a lot of interaction with the end users.
- Typically, online systems, web interfaces have a very high amount of interaction with end users, are best suited for Prototype model. It might take a while for a system to be built that allows ease of use and needs minimal training for the end user.
- Prototyping ensures that the end users constantly work with the system and provide a feedback which is incorporated in the prototype to result in a useable system. They are excellent for designing good human computer interface systems.

Prototyping Model نموذج العرض التجريبي

جاء نموذج العرض التجريبي Prototyping Model كمحاولة لحل قصور (عدم نجاح) النموذج التدفقي في حل مشكلة المتطلبات الخاصة بالزبون . ويقوم معد المنظومة باعداده ثم بعد ذلك يعرض منظومة تجريبية (نموذج تجريبي pilot system) كعينة sample على الزبون (والمستخدم) يمثل او يشابه المنظومة النهائية المطلوبة ولكن بصورة اصغر (اي صورة مبدئية) ومدة قصيرة في الاعداد. وتستخدم هذه العينة عادة اما لتوضيح المتطلبات او كبداية لاصدار عدة نسخ متتالية للوصول للمنظومة النهائية التي ستسلم للزبون فيما بعد . وفي هذا النموذج يتم عرض وظائف محددة من المنظومة على الزبون في اقرب وقت ممكن وان كان ذلك في مستوى منخفض من الجودة ، والفكرة هنا هي جعل الزبون يرى عينة من المنظومات النهائية في مرحلة مبكرة .

When to use Prototype model: متى نستخدم العرض التجريبي ؟

نستخدم العرض التجريبي Prototype للاسباب الاتية :

1 - هذا النموذج يستخدم عندما يحتاج النظام المرغوب الكثير من التفاعل مع المستخدمين النهائيين.
6- عادة ، انظمة الانترنت (online systems, web interfaces) تحتوي على تفاعل عالي مع المستخدمين النهائيين، فهذا النموذج مناسب لها .
3- لتوضيح وتحديد وتنقيح المتطلبات .
4- اقناع الزبون بالجدوى الفنية عندما تكون المخاطرة الفنية عالية .
5- محاولة فهم المسالة المراد حلها قبل الشروع في رصد الموارد والمستلزمات .
6- نسخة بدائية للتوصل الى النسخة النهائية او ما يسمى بالاصدار (المنظومة النهائية).

والميزة هنا ان الزبون يعطى فرصة لتغيير متطلباته في مرحلة مبكرة من المشروع حيث تعتبر تكلفة التغيير قليلة نسبيا. وتستخدم طريقة العرض المبدئي كحل لأهم مشكلة في النموذج التدفقي (Waterfall model) وهي مشكلة عدم اكتمال المتطلبات .

او يصبح العرض التجريبي هو البداية في اعداد المنظومة من حيث تجهيز اول عرض تجريبي وعرضه على الزبون والمستخدم واخذ الملاحظات والتغيرات بشأنه ليتم بعد ذلك تجهيز اول نسخة first version لعرضها . وتبدا عملية اعداد المنظومة لتشمل الملاحظات السالفة الذكر حيث يتم مرة اخرى عرض النسخة الثانية ليتم اخذ المزيد من الملاحظات والاضافات ليتم تعديل المنظومة.
وتصبح النسخة الثانية second version بعد التعديل جاهزة للتحميل على اجهزة الزبون وتسمى هذه الحالة بالاصدار الاول first release وتستمر عملية عمل النسخ وتحميل الاصدارات الى ان يتم تجهيز المنظومة بالكامل وينتهي عمل اخر اصدار وبذلك تسلم الى الزبون نهائيا .والهدف النهائي منه هو الوصول الى المنظومة النهائية ابتداء من نموذج تجريبي محدود .

ويمكن تلخيص نشاطات نموذج العرض التجريبي Activities في الخطوات التالية :

1- يتم التعرف على المتطلبات من قبل معد المنظومة وبمساعدة المستخدم .
2- يتم تصميم وتنفيذ منظومة عرض مبدئي تقوم باداء وظائف محدودة كعينة (limited functions). ويمكن هنا استعمال لغة برمجة مرئية مثل فيجوال بيسك ولغة قاعدة البيانات مثل اكسس نظرا لسهولة برمجتها .
3- تتم تجربة منظومة العرض التجريبي من قبل المستخدم بحضور معد المنظومة ثم بمفرده في موقع الزبون .
4- يعطى الزبون (المستخدم) فرصة استعمال المنظومة التجريبية pilot system ونتوقع منه في المقابل ان يعطي ملاحظات (تغييرات) حول المنظومة .
5- من التغييرات المطلوبة في الخطوة الرابعة انه يمكن لمعد البرمجيات ان يعدل المنظومة ويجهز النسخة اللاحقة .
6- يتم تكرار الخطوات الثالثة والرابعة والخامسة حتى يتم تحقيق متطلبات المستخدم تماما وبالتالي تصبح المنظومة جاهزة للاستخدام .

Advantages of Prototype model:

مزايا العرض التجريبي :

1- يقلل من مشاكل تحديد المتطلبات .
2- يوفر مشاركة ودعم اكبر من طرف الزبون في المراحل المبكرة من المشروع .
3- يقلل من مجازفة تصحيح الاخطاء في المراحل المتاخرة .
4- يمكن استعماله كقاعدة لكتابة المواصفات بجودة اعلى .
5- يمكن استعماله لتدريب المستخدمين من قبل تسليم المنظومة .
6- يقلل من وقت اعداد المنظومة .
7- يقلل من التكلفة الاجمالية لاعداد المنظومة .
8- يقلل من مخاطر المشروع .
9- تحسين عملية التواصل بين المستخدم ومعد المنظومة .

Disadvantages of Prototype model:

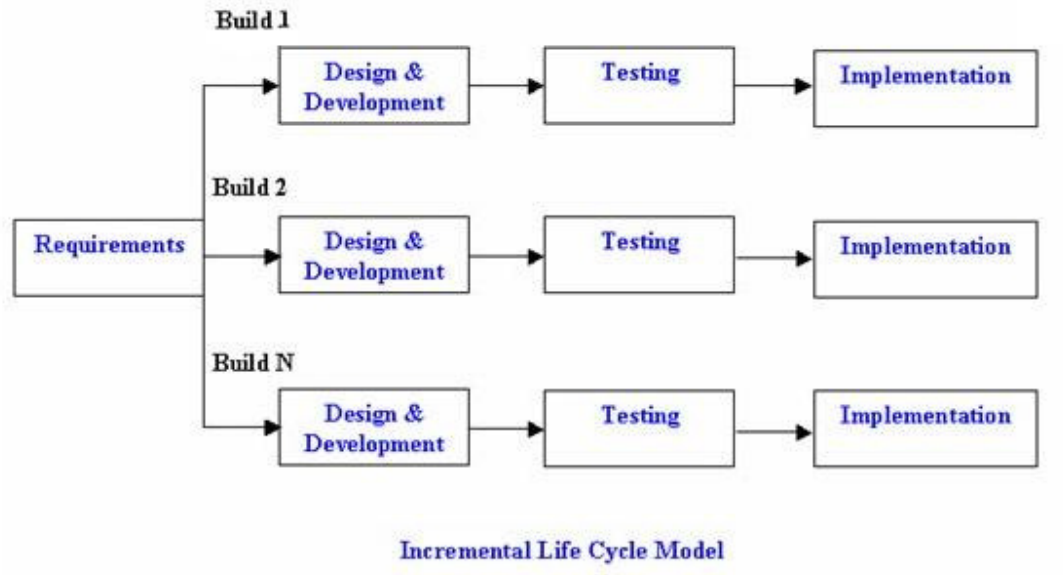
عيوب العرض التجريبي :

1- يؤدي إلى تنفيذ ومن ثم إصلاح طريق أنظمة البناء.
2- عمليا ،هذا المنهجية قد يزيد من تعقيد النظام ونطاق النظام قد يتوسع خارج الخطط الاصلية .
3- طلب غير مكتمل قد يتسبب بتطبيقات لا تستخدام مثلا تصميم نظام كامل ع اساس تحليل غير كافي وغير كامل للمشكلة.
4- احيانا يكون من الصعب اقناع الزبون بأن المنتوج المبدئي هو للعرض فقط (نظام تجريبي) وقد يطلب الاسراع في استلام المنتوج ونتيجة لذلك يستلم منظومة غير مكتملة الجودة لتبدا مرحلة الصيانة مبكرا.
5- لاعداد منظومة تجريبية سريعة قد يستخدم المبرمج خوارزميات قليلة الكفاءة ويستخدم ادوات بسيطة في التحليل والتصميم والبرمجة وهذه الادوات عادة ما تصبح هي الادوات الاساسية في اعداد المنتوج النهائي مما قد يؤدي الى اعداد منتوج قليل الجودة.
6- تخلق احساسا خادعا من التفاؤل (تفاؤل ان المنظومة تم الانتهاء من اعدادها).
7- قلة استخدام التوثيق.

3- Incremental model

In incremental model the whole requirement is divided into various builds. Multiple development cycles take place here, making the life cycle a "multi-waterfall" cycle. Cycles are divided up into smaller, more easily managed modules. Each module passes through the requirements, design, implementation and testing phases. A working version of software is produced during the first module, so you have working software early on during the software life cycle. Each subsequent release of the module adds function to the previous release. The process continues till the complete system is achieved.

Diagram of Incremental model:



Incremental Life Cycle Model

Advantages of Incremental model:

- Generates working software quickly and early during the software life cycle.
- This model is more flexible – less costly to change scope and requirements.
- It is easier to test and debug during a smaller iteration.
- In this model customer can respond to each built.
- Lowers initial delivery cost.
- Easier to manage risk because risky pieces are identified and handled during it'd iteration.

Disadvantages of Incremental model:

- Needs good planning and design.
- Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
- Total cost is higher than waterfall.

When to use the Incremental model:

- This model can be used when the requirements of the complete system are clearly defined and understood.
- Major requirements must be defined; however, some details can evolve with time.
- There is a need to get a product to the market early.
- A new technology is being used
- Resources with needed skill set are not available
- There are some high risk features and goals.

3- Incremental Model النموذج التزايدي

ان الهدف من النموذج التزايدي هو تقليص وقت انتظار الزبون للحصول على المنظومة النهائية خصوصا المعقدة والكبيرة التي تستغرق مدة طويلة (قد تستغرق سنوات) لاعدادها بالكامل . والحل يكمن في تسليم المنظومة الى الزبون على دفعات وذلك بتقسيم المشروع (المنظومة) الى اجزاء يتم تسليمها للزبون على اجزاء تباعا في فترات زمنية محددة ومتلاحقة.
بمعنى ان يتم تقسيم المشروع الى مجموعة مشاريع جزئية حيث كل مشروع ينهي جزءا من المشروع الكلي (وتمر عليه جميع المراحل) ليتم تسليمه وتشغيله فيما بعد في موقع الزبون .ثم يتم اعداد المشروع الجزئي التالي وهكذا الى ان يتم تسليم الجزء الاخير من المشروع.

فمثلا يمكن تقسيم المنظومة المصرفية الى عدة اجزاء : الحسابات الجارية وحسابات التوفير والحوالات والصرف الاجنبي . فيتم اولا اعداد منظومة الحسابات الجارية وتسليمها للزبون ومن ثم البدء في اعداد الجزء الثاني وهو حسابات التوفير وهكذا الى ان يتم تنفيذ كل الاجزاء الخاصة بالمنظومة المصرفية بالكامل.

وهناك مثال اخر وهو منظومة المبيعات حيث يتم اعداد وتسليم الجزء الخاص بعمل الطلبيات اولا ثم يتم تسليم اعداد فواتير الشراء في الجزء الثاني بعد فترة زمنية وهكذا .

يبدا هذا النموذج بتحديد الاهداف وتحويلها الى متطلبات ليتم تصميمها وتنفيذها واختبارها ثم تسليمها للزبون من اجل البدء في تشغيلها على ارض الواقع .

- وهذا النموذج هو عبارة عن **خليط من النموذج التدفقي ونموذج العرض التجريبي** (waterfall + prototype).

- ويستخدم اذا كانت هناك مخاطرة في تنفيذ المنظومة بالكامل دفعة واحدة اي يستخدم هذا النموذج لتقليل هذه المخاطرة .

خصائص النموذج التزايدي :

- يتكون المنتوج عادة من عدة اجزاء او بناء Build
- كل جزء (بناء) يتم تصميمه وكتابته واختباره ثم تسليمه للزبون.
- يتكون البناء من قطع برمجية تتفاعل لتعطي وظيفة معينة .
- يعطى المنتوج للزبون تدريجيا (على دفعات) جزءا بعد الاخر .
- بخلاف نموذج العرض المبدئي يعتبر كل جزء في هذا النموذج منتوجا (اصدار Release) قابلا للعمل في موقع الزبون.

مزايا النموذج التزايدي Advantages of Incremental model:

1- لا يحتاج الى الكثير من اعضاء فريق العمل مما يؤدي الى توفير التكلفة (هذا النموذج هو أكثر مرونة).
2- يؤدي الى اتصال جيد بين اعضاء الفريق.
3- يستطيع الزبون ان يرى منظومة قابلة للعمل مبكرا.
4- يقلل خطر المجازفة وذلك باستخدام العرض المبدئي .
5- يمكنك تسليم الاجزاء البرمجية العاجلة وتاجيل الباقي فيما بعد.
6- يستفاد من الخبرة التي تم الحصول عليها في اعداد الاجزاء السابقة عند بناء الاجزاء التالية .

Disadvantages of Incremental model: **عيوب النموذج التزايدي**

- الاحتياج الى التخطيط الجيد والتصميم.
- يحتاج الى تعريف واضح وكامل للنظام كله قبل أن يتم التقسيم والبناء بشكل متزايد.
- التكلفة الإجمالية أعلى من نموذج الشلال (waterfall model)

When to use the Incremental model: **متى يستخدم نموذج التقديم التزايدي**

- يمكن استخدام هذا النموذج عندما متطلبات نظام كامل تكون واضحة التعريف ومفهومة.
- يجب تحديد المتطلبات الرئيسية؛ ومع ذلك، يمكن لبعض التفاصيل تتطور مع مرور الوقت.
- وهناك حاجة للحصول على المنتج إلى السوق (انزاله الى السوق) في وقت مبكر.
- تكنولوجيا جديدة بدأت تستخدم .
- الموارد مع مجموعة المهارات اللازمة غير متوفرة
- هناك بعض الميزات عالية المخاطر والأهداف.

4- Spiral model:

The spiral model is similar to the **incremental model**, with more emphasis placed on risk analysis. The spiral model has four phases: **Planning, Risk Analysis, Engineering and Evaluation**. A software project repeatedly passes through these phases in iterations (called Spirals in this model). The baseline spiral, starting in the planning phase, requirements is gathered and risk is assessed. Each subsequent spiral builds on the baseline spiral.

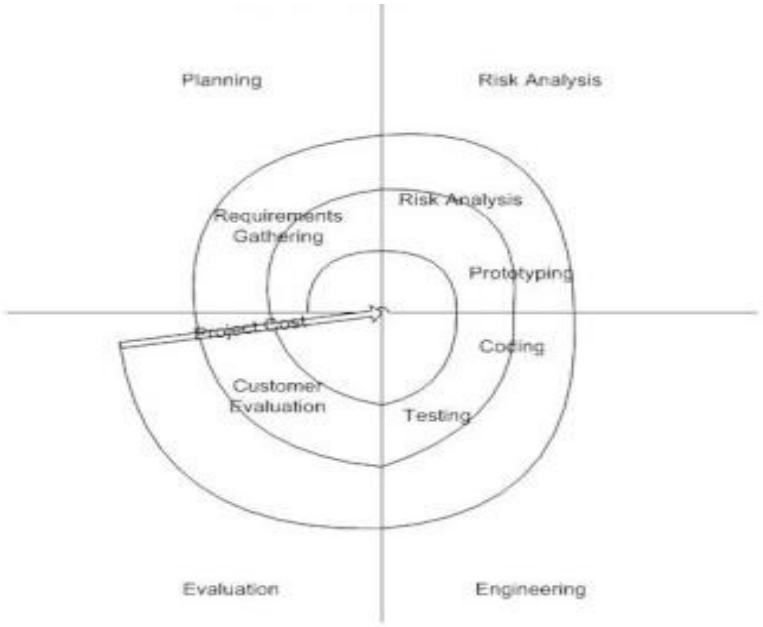
Planning Phase: Requirements are gathered during the planning phase. Requirements like **'BRS'** that is 'Business Requirement Specifications' and **'SRS'** that is 'System Requirement specifications'.

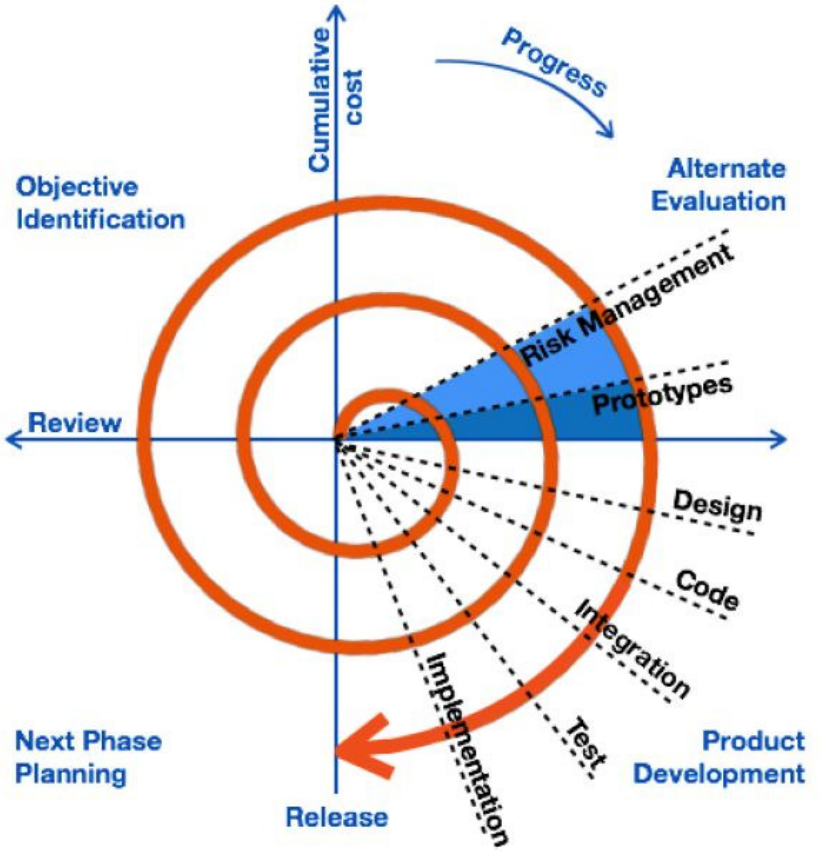
Risk Analysis: In the risk analysis phase**,** a process is undertaken to identify risk and alternate solutions. A prototype is produced at the end of the risk analysis phase. If any risk is found during the risk analysis then alternate solutions are suggested and implemented.

Engineering Phase: In this phase software is **developed**, along with testing at the end of the phase. Hence in this phase the **development and testing** is done.

Evaluation phase: This phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral.

Diagram of Spiral model:





Advantages of Spiral model:

- High amount of risk analysis hence, avoidance of Risk is enhanced.
- Good for large and mission-critical projects.
- Strong approval and documentation control.
- Additional Functionality can be added at a later date.
- Software is produced early in the software life cycle.

Disadvantages of Spiral model:

- Can be a costly model to use.
- Risk analysis requires highly specific expertise.
- Project's success is highly dependent on the risk analysis phase.
- Doesn't work well for smaller projects.

When to use Spiral model:

- When costs and risk evaluation is important
- For medium to high-risk projects
- Long-term project commitment unwise because of potential changes to economic priorities
- Users are unsure of their needs
- Requirements are complex
- New product line
- Significant changes are expected (research and exploration)

4-Spiral Model النموذج اللولبي

يعتبر النموذج اللولبي Spiral Model خليطا بين النموذج التدفقي والعرض التجريبي وهذا النموذج يشابه الى حد بعيد **النموذج التزايدي** . وفي هذا النموذج يتم التعامل مع المخاطر Risks التي تهدد المشروع في دورة معينة من هذا اللولب قبل البدء في الدورة التالية من اللولب وهكذا .ويستخدم هذا النموذج اذا لم توجد ضمانة (وجود مخاطر) كاملة في تنفيذ ونجاح المشروع ، ووجود تغيير واضح في المتطلبات حيث يتم التعامل مع هذه المشكلة بواسطة العرض التجريبي(Prototype model) .وباستعمال العرض المبدئي(Prototype model) تتم مشاركة المستخدم في كل مرحلة من مراحل اعداد المنظومة .وهذا النوع من النماذج ثبت انه يلائم المشاريع الكبيرة بصورة خاصة .

يتكون النموذج اللولبي من اربع مراحل وهي :

1- مرحلة التخطيط Planning Phase
2- مرحلة تحليل المخاطر Risk Analysis Phase
3- مرحلة الهندسة Engineering Phase
4- مرحلة تقييم الزبون Customer Evaluation Phase

وفيما يلي تفاصيل لهذه المراحل :

1- مرحلة التخطيط Planning Phase

وتجمع المتطلبات خلال مرحلة التخطيط: متطلبات مثل" BRS " هذا هو متطلبات الأعمال المواصفات 'Business Requirement Specifications' و " SRS "هذا هو" مواصفات متطلبات النظام (System Requirement specifications) .

وفي هذه المرحلة نقوم بالنشاطات التالية :

- عرف المشكلة (Define the problem)
- حدد الاهداف (Determine objectives)
- حدد القيود(Determine constraints)
- عرف المواصفات(Define specifications)
- عرف البدائل (Determine alternatives)

2- مرحلة تحليل المخاطر Risk Analysis Phase

في هذه المرحلة يجب ان نقوم بنشاطين :

- نموذج تجريبي لحل مشكلة عدم التاكد من المتطلبات
- وايضا التعرف على المخاطر والحلول البديلة وتحليلها

3- مرحلة الهندسة Engineering Phase

الهدف الرئيسي من هذه المرحلة هو اعداد الاصدار الموالي من المنظومة .

وفي هذه المرحلة يجب انجاز النشاطات التالية :

- التصميم المفصل (Detailed design)
- التشفير (Coding)
- اختبار لوحدة (Unit testing)
- اختبار التكامل (Integration testing)

4- مرحلة تقييم الزبون Customer Evaluation Phase

يتم في هذه المرحلة تقييم المنظومة من حيث سهولة الاستعمال للحصول على النتائج الصحيحة .فيقيم الزبون المنظومة ويعطي ملاحظات تؤخذ في الاعتبار في الجولة الموالية من التخطيط

Advantages of Spiral model: مزايا النموذج اللولبي

1- يستطيع المستخدم ان يقيم ويغير المتطلبات في مرحلة مبكرة من المشروع (موافقة على المنظومة وسيطرة على عملية التوثيق).
2- مناسب للمشاريع الكبيرة والمعقدة (ذات المهام الحرجة) .
3- كل من المستخدم ومعد المنظومة يستجيب لانواع مختلفة من المخاطر في كل تركيبة (دورة في اللولب) وعليه فأن المخاطر ستتناقص شيئا فشيئا (كمية عالية من تحليل المخاطر وبالتالي يعزز تجنب المخاطر) .
4- يتم إنتاج منظومة في مرحلة اعداد المشروع في وقت مبكر.
5- وظائف إضافية يمكن إضافتها في وقت لاحق.

Disadvantages of Spiral model: **عيوب النموذج اللولبي**

1- يمكن أن تكون نموذجا مكلفا للاستخدام .
2- تحليل المخاطر يتطلب خبرة خاصة
3- نجاح المشروع يعتمد إلى حد كبير على مرحلة تحليل المخاطر.
4- لا تعمل بشكل جيد للمشاريع الصغيرة.

When to use Spiral model: **متى يستخدم النموذج اللولبي**

- عندما تكاليف ومخاطر التقييم هو المهم
- مناسب للمشاريع ذات المخاطر المتوسطة و العالية
- التزام المشروع على المدى الطويل غير حكيم بسبب التغيرات المحتملة للأولويات الاقتصادية
- المستخدمين غير متأكدين من احتياجاتها
- متطلبات معقدة
- خط إنتاج جديد
- تغيرات هامة ومن المتوقع (البحث والتنقيب)

5- Fourth generation techniques (4GT)

The term "fourth generation techniques" (4GT) encompasses a broad array of software tools that have one thing in common. Each enables the software engineer to specify some characteristic of software at a high level, the tool then automatically generates source code based on the developer's specification.
The 4GT paradigm for software engineering focuses on the ability to specify software using specialized language forms or a graphic notation that describes the problem to be solved in terms that the customer can understand.
Currently, a software development environment that supports the 4GT paradigm includes some or all of the following tools:

- Non-procedural languages for database query
- Report generation
- Data manipulation
- Screen interaction
- Definition & code generation; high-level graphics capability
- Spreadsheet capability

Like other paradigms, 4GT begins with a requirements gathering step, the customer would describe requirements and these would be directly translated into an operational prototype.

Advantages of (4GT):

- Simplified the programming process.
- Use non-procedural languages that encourage users and programmers to specify the results they want, while the computers determines the sequence of instruction that will accomplish those results.
- Use natural languages that impose no rigid grammatical rules.

Disadvantages of (4GT):

- Less flexible that other languages
- Programs written in 4GLs are generally far less efficient during program execution that programs in high-level languages. Therefore, their use is limited to projects that do not call for such efficiency.

نموذج طريقة الجيل الرابع 4GT Model

يستخدم نموذج طريقة الجيل الرابع Fourth Generation Techniques ادوات الجيل الرابع لاعداد المنظومات. وفكرة هذا النموذج هي تمكين مهندس البرمجيات من تحديد بعض الخصائص للمنظومة وبناء على ذلك يتم تكوين شفرة المصدر تلقائيا .

الادوات المستخدمة :

1- شاشة تفاعلية Screen interaction
2- مكون تقارير Report Generator
3- مكون شفرة Code Generators
4- لغات قواعد البيانات DMBS
5- لغات شيئية Object _Oriented Languages
6- محرر رسومات Graphic Editors
7- محرر نص Text Editors
8- جداول الكترونية Spread Sheets

مراحل الجيل الرابع :

يتكون نموذج الجيل الرابع من مجموعة من المراحل وهي :

1- مرحلة تحديد المتطلبات Requirements phase
2- مرحلة التصميم Design phase
3- مرحلة التنفيذ Implementation phase
4- مرحلة الاختبار Testing phase

Advantages of (4GT): مزايا نموذج الجيل الرابع

1- تقليص الوقت اللازم لاعداد البرمجيات (المنظومات) والسبب في ذلك يعود الى تبسيط عملية البرمجة.
2- تحسن من عملية انتاجية البرمجيات وذلك بسبب استخدام لغات غير الإجرائية (لغات قواعد البيانات :SQL,Oracle) التي تشجع المستخدمين والمبرمجين لتحديد النتائج التي تريد، في حين أن أجهزة الكمبيوتر يحدد تسلسل التعليمات التي من شأنها تحقيق تلك النتائج.
3- رفع جودة منتوج البرمجيات وذلك بسبب استخدام اللغات الطبيعية التي لا تفرض أي قواعد نحوية جامدة.

Disadvantages of (4GT):
العيوب او الصعوبات في استخدام النموذج تتمثل في ان :

1- تكون ادوات الجيل الرابع (4GT) اقل مرونه من اللغات الاخرى والسبب في ذلك لان ادوات 4GT صعبة الاستعمال على المبرمجين الذين تعودوا على استعمال اللغات البرمجة القديمة .

2- كفاءة شفرة المصدر الناتجة تلقائيا من هذه الادوات code generator ليست في المستوى المطلوب وتحتاج الى تعديل . والسبب ان البرامج المكتوبة في (4GT) عادة ما تكون أقل كفاءة أثناء تنفيذ البرنامج (أن البرامج في اللغات عالية المستوى) . وبالتالي، فإن استخدامها يقتصر على المشاريع التي لا تستدعي مثل هذه الكفاءة.

3- استخدام هذه الادوات في المنظومات الكبيرة قد يسبب مشاكل في عملية الصيانة .

6- Rapid Application Development model (RAD)

RAD model is Rapid Application Development model. It is a type of incremental model. In RAD model the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype. This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.

The phases in the rapid application development (RAD) model are:

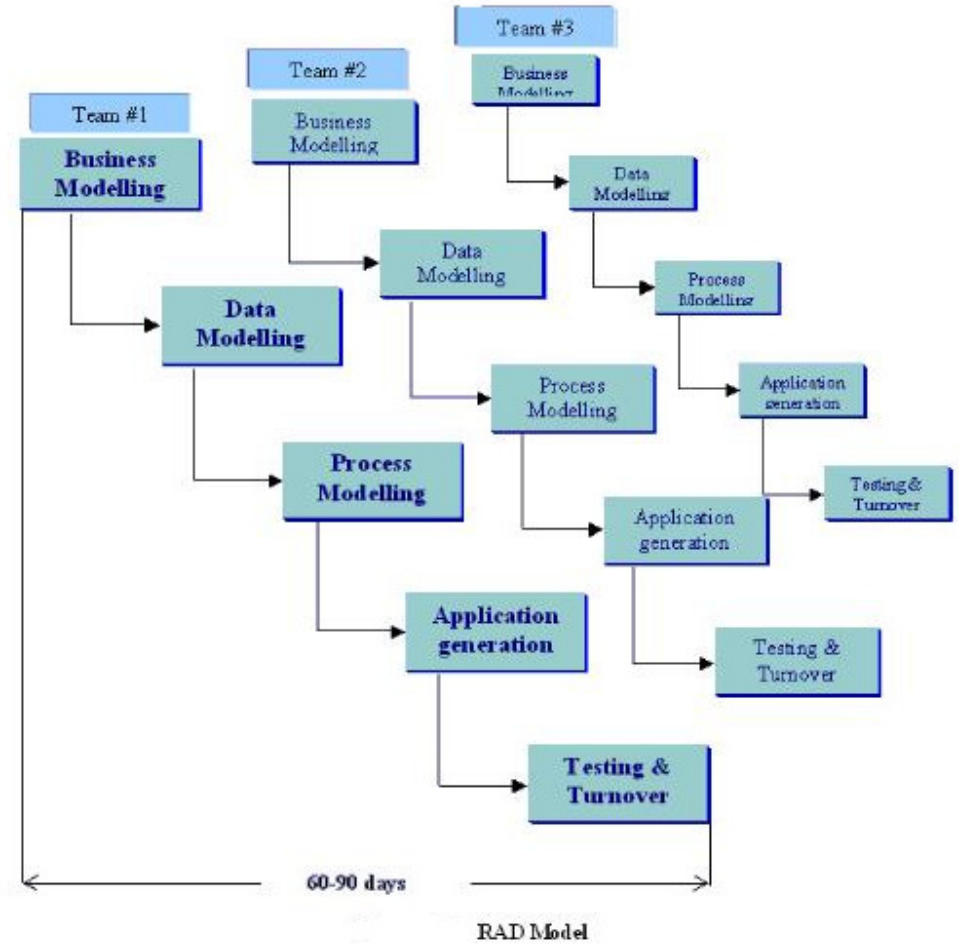
1- Business modeling: The information flow is identified between various business functions.
2- Data modeling: Information gathered from business modeling is used to define data objects that are needed for the business.
3- Process modeling: Data objects defined in data modeling are converted to achieve the

business information flow to achieve some specific business objective. Description are identified and created for CRUD of data objects.

4- Application generation: Automated tools are used to convert process models into code and the actual system.

5- Testing and turnover: Test new components and all the interfaces.

Diagram of RAD Model:



RAD Model

Advantages of the RAD model:

- Reduced development time.
- Increases reusability of components
- Quick initial reviews occur
- Encourages customer feedback
- Integration from very beginning solves a lot of integration issues.

Disadvantages of RAD model:

- Depends on strong team and individual performances for identifying business requirements.
- Only system that can be modularized can be built using RAD
- Requires highly skilled developers/designers.
- High dependency on modeling skills
- Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.

When to use RAD model:

- RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.
- It should be used if there's high availability of designers for modeling and the budget is high enough to afford their cost along with the cost of automated code generating tools.
- RAD SDLC model should be chosen only if resources with high business knowledge are available and there is a need to produce the system in a short span of time (2-3 months).

Rapid Development Model (RAD) نموذج الاعداد السريع

وهو نوع من نموذج تدريجي (incremental model).ويستخدم هذا النموذج لاعداد البرمجيات التي تعتبر من النوع الخطي .حيث يتم اعداد المنتوج النهائي باستخدام طريقة التنفيذ على دفعات للمنظومة .والوقت المتوسط لبناء منظومة متكاملة قابلة للتشغيل قد يتطلب مدة من 60 الى 90 يوما .ويعتمد هذا النموذج على اعادة استعمال المكونات (Component Reuse).

مراحل نموذج الاعداد السريع :

1- مرحلة نمذجة الاعمال Business modeling phase
2- مرحلة نمذجة البيانات Data modeling phase
3- مرحلة نمذجة العمليات Process modeling phase
4- مرحلة تكوين التطبيق Application generation phase
5- مرحلة الاختبار Testing phase

1- مرحلة نمذجة الاعمال Business modeling phase : يتم هنا تعريف انسياب البيانات بين مختلف وظائف النظام (التعرف على تدفق المعلومات بين القطاعات المختلفة).

2- مرحلة نمذجة البيانات Data modeling phase: في هذه المرحلة يتم تعرف الكائنات وخصائص كل منها والعلاقات بينها (يتم استخدام المعلومات التي تم جمعها من مرحلة نمذجة الأعمال لتحديد كائنات البيانات اللازمة).

3- مرحلة نمذجة العمليات Process modeling phase : يتم هنا تحديد العمليات على كل كائن (تم تعريفه في مرحلة نمذجة البيانات). قد تشمل هذه العمليات الإضافة والإلغاء والتعديل واسترجاع كائن بيانات.

4- مرحلة تكوين التطبيق Application generation phase : تستعمل هذه المرحلة (مرحلة البرمجة) لغة الجيل الرابع كأداة لبناء البرمجيات .

5- مرحلة الاختبار Testing phase : تعتبر عملية الاختبار عملية بسيطة لأنه كما ذكرنا سابقا يركز هذا النظام على إعادة استعمال الأجزاء وهو مفهموم ذو علاقة بالبرمجة الشيئية مما يمكننا من الاستفادة من الأجزاء التي تم اختبارها في منظومة سابقة .

مزايا نموذج الاعداد السريع Advantages of the RAD model:

1- يمكن استعماله في جميع انواع التطبيقات بشرط ان تكون المنظومة المقترحة مقسمة الى اجزاء مناسبة (لكي تمكنا من المعاينة الأولية التي تحدث بسرعة).
2- يتعامل مع التقنية الشيئية Object Technology التي تركز على اعادة استعمال الاجزاء .
3- يستغرق وقتا اقل في اعداد البرمجيات من معظم النماذج الاخرى .
4- باستخدام RAD نحصل على منظومات ذات جودة عالية .والسبب في ذلك التكامل من البداية يحل الكثير من لقضايا التكامل.
5- يزيد من انتاجية المبرمجين .
6- المنظومات المنفذة باستخدام هذا النموذج ذات تكلفة منخفضة .

عيوب نموذج الاعداد السريع Disadvantages of RAD model:

1- يحتاج الى عدد اكبر من المبرمجين (حتى نشكل فرق عمل RAD) وذلك بسبب ان كل وظيفة او جزء يتم اعداده بواسطة فريق منفصل .
2- فقط الانظمة التي تكون قابلة للتجزئه يمكن إنشاؤها باستخدام RAD .
3- يتطلب مطورين ومصممين ذو مهارات عالية .
4- الاعتماد الكبير على مهارات النمذجة
5- غير قابل للتطبيق على مشاريع رخيصة من ناحية تكلفة النمذجة وتوليد البرمجة (الشفرة) التلقائية العالية المستوى .
6- يحتاج الى مشاركة وتعاون كامل بين الزبون ومعد البرمجية والا فان المشروع سيفشل .

When to use RAD model:

متى يتم استخدام النموذج RAD

1- RAD ينبغي أن يستخدم عندما تكون هناك حاجة إلى إنشاء نظام يمكن تجزئته في 2-3 أشهر من الزمن.
2- وينبغي أن يستخدم إذا كان هناك وفرة عالية من المصممين للنمذجة والميزانية مرتفعة بما يكفي لتتحمل التكلفة مع تكلفة ادوات الشفرة الناتجة التلقائية .
3- يجب اختيار نموذج RAD SDLC فقط إذا كانت الموارد مع المعرفة التجارية العالية متوفرة (متاحة) وهناك حاجة لإنتاج النظام في فترة قصيرة من الزمن (2-3 أشهر).