جامعة الحمدانية

كلية التربية للعلوم الصرفة

قسم علوم الحاسوب

المرحلة الثالثة

المحاضرة الرابعة

هندسة البرمجيات

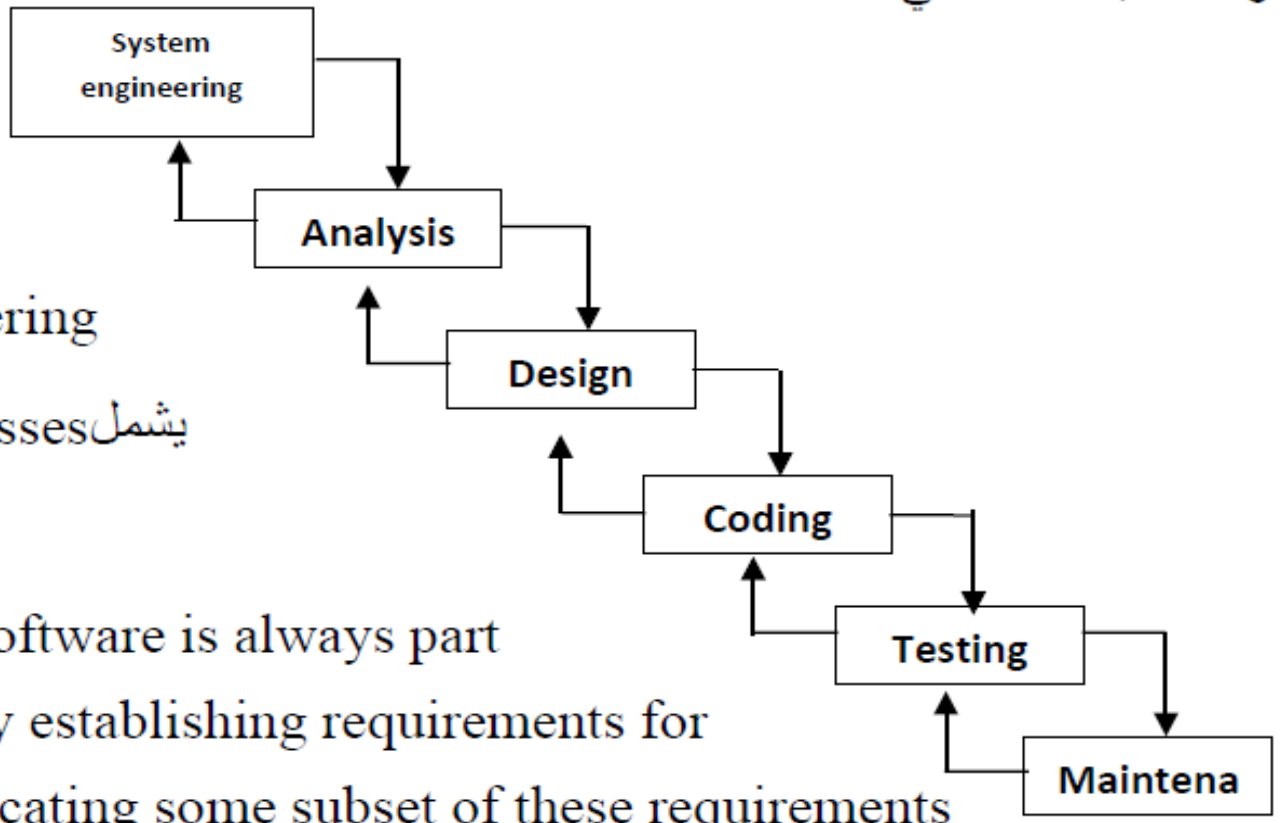
Software Engineering

Software Engineering Paradigms (Software Life Cycle):-

1. The classic life cycle (Sometimes called the “waterfall model”):-

the life-cycle paradigm demands a systematic, sequential approach to software development that begins at the system level and progresses through analysis, design, coding, testing and maintenance.

هو نموذج شائع الاستخدام في هندسة البرمجيات يتبع طريقة أو خطوات منظمة ومتسلسلة لتطوير SW وهذه الخطوات موضحة بالشكل الأتي:



Modeled after the conventional engineering cycle, the life-cycle paradigm encompassesيشمل the following activities:

1. **System engineering** :- Because software is always part of a larger system, work begins by establishing requirements for all system elements and then allocating some subset of these requirements to software; such as hardware, people and databases.

بسبب ان الـSW هو دائما جزء من نظام كبير، فان العمل يبدا بتهيئة المتطلبات لكل عناصر النظام ثم تخصيص بعض من المجاميع الثانوية للمتطلبات للـ SW مثل قواعد البيانات والناس و HW. (أي نحدد وظيفة كل جزء من أجزاء النظام أي تحدد وظيفة الـSW).

2. Software requirements analysis:- The requirements gathering process is intensified and focused specifically on software. To understand the nature of the program(s) to be built, the software engineer (“analyst”) must understand the required function, performance, and interface. Requirements are documented and reviewed with the customer.

تصميم الـSW في الحقيقة تتكون من خطوات متعددة وتركز على تصميم هيكل البيانات ومعمارية الـSW وتفاصيل الإجراء وخصائص الواجهة. يتم تحويل المتطلبات لتمثيل الـSW بصيغة تسهل عملية البرمجة ويجب أن يوثق التصميم ويصبح جزء من شكل الـSW.

3. **Design:-** software design is actually a multistep process that focuses on four distinct attributes of the program: data structure, software architecture, procedural detail, and interface characterization. The design process translates requirements into a representation of the software that can easily coded. The design is documented and becomes part of the SW configuration.

تصميم الـSW في الحقيقة تتكون من خطوات متعددة وتركز على تصميم هيكل البيانات ومعمارية الـSW وتفاصيل الإجراء وخصائص الواجهة. يتم تحويل المتطلبات لتمثيل الـSW بصيغة تسهل عملية البرمجة ويجب أن يوثق التصميم ويصبح جزء من شكل الـSW.

4. **Coding:-** This steps translates design into programming language.

5. **Testing:-** once code has been generated, program testing begins. The testing process focuses on the logical internals of the software, ensuring that all statements have been tested, and on the functional externals, that is, conducting يتصرف tests to uncoverيكشف errors and ensure that defined input will produce actual results that agree with required results.

حالما تنتهي عملية كتابة البرنامج. نقوم بعملية الاختبار:

1- نتأكد من كل جملة في البرنامج قد تم اختبارها.

2- نعطي input ونأخذ output ونلاحظ هل أن الـoutput مطابقة للنتائج المتوقعة أو قريبة منها.

6. **Maintenance:-** software will undoubtedlyمن غير شك undergoيتحمل changeتغيير after it is delivered to the customer. Change will occur because errors have been encountered, because the software must be adaptedيتكيف to accommodateيلائم changes in its external environment.

ال SW بلا شك سيخضع للتغييرات بعد تسليمه للزبون، والتغييرات ستحصل بسبب الأخطاء التي ستلاقى بسبب ان ال SW يجب ان يتكيف مع التغييرات في البيئة الخارجية.

The classic life cycle is the oldest and the most widely used paradigm for software engineering.

نماذج عمليات البرمجيات Software Process Models

- **ايجابيات النموذج (Model Advantages)**

١. نموذج سهل للفهم
٢. سهل للإدارة
٣. المراحل تكتمل وتعالج مرحلة تلو الاخرى
٤. العمل يقسم الى مشاريع صغيرة حيث المتطلبات تصبح سهلة للفهم
٥. يفضل في المشاريع حيث الجودة هي اكثر اهمية بالمقارنة مع الجدول الزمني والتكلفة

نماذج عمليات البرمجيات Software Process Models

- **السلبيات النموذج (Model Disadvantages)**

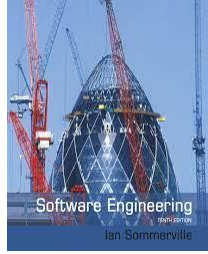
١. لا يمكن ان تعود خطوة ، اذا مرحلة ما يوجد بها خطأ لا يمكن الرجوع اليها وتعديلها لا نها تصبح العمليات معقدة في المرحلة، حتى لو تغيير بسيط في اي مرحلة سابقة يمكن ان يسبب مشاكل كبيرة للمراحل اللاحقة حيث ان جميع المراحل تعتمد على بعضها.

٢. نسبة كبيرة من المخاطر واحتمال حصول اخطاء

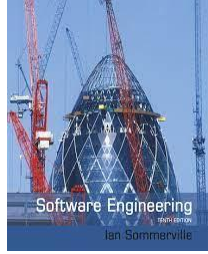
٣. ليس نموذج جيد للمشاريع المعقدة

٤. نموذج ضعيف للمشاريع الطويلة والمستمرة

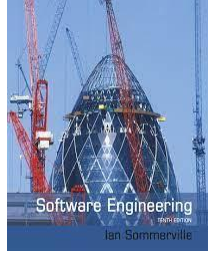
٥. غير مناسب للمشاريع حيث المتطلبات فيها مخاطر عالية من التغيير



Prototyping Model



Contents



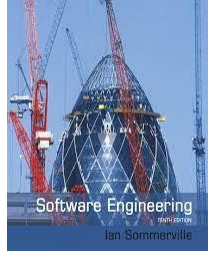
What is Prototyping Model?

Prototyping is development of a preliminary version of a software in order to allow certain aspects of that that software to be investigated.

Often the primary purpose of a prototype is to obtain feedback from the intended users; the requirements specification for the software can then be updated to reflect this feedback, and so increase confidence in the final software.

النماذج الأولية هي تطوير نسخة أولية من البرنامج للسماح بدراسة جوانب معينة من هذا البرنامج.

غالبًا ما يكون الغرض الأساسي من النموذج الأولي هو الحصول على تعليقات من المستخدمين المقصودين؛ يمكن بعد ذلك تحديث مواصفات متطلبات البرنامج لتعكس هذه الملاحظات، وبالتالي زيادة الثقة في البرنامج النهائي.



What is Prototyping Model?

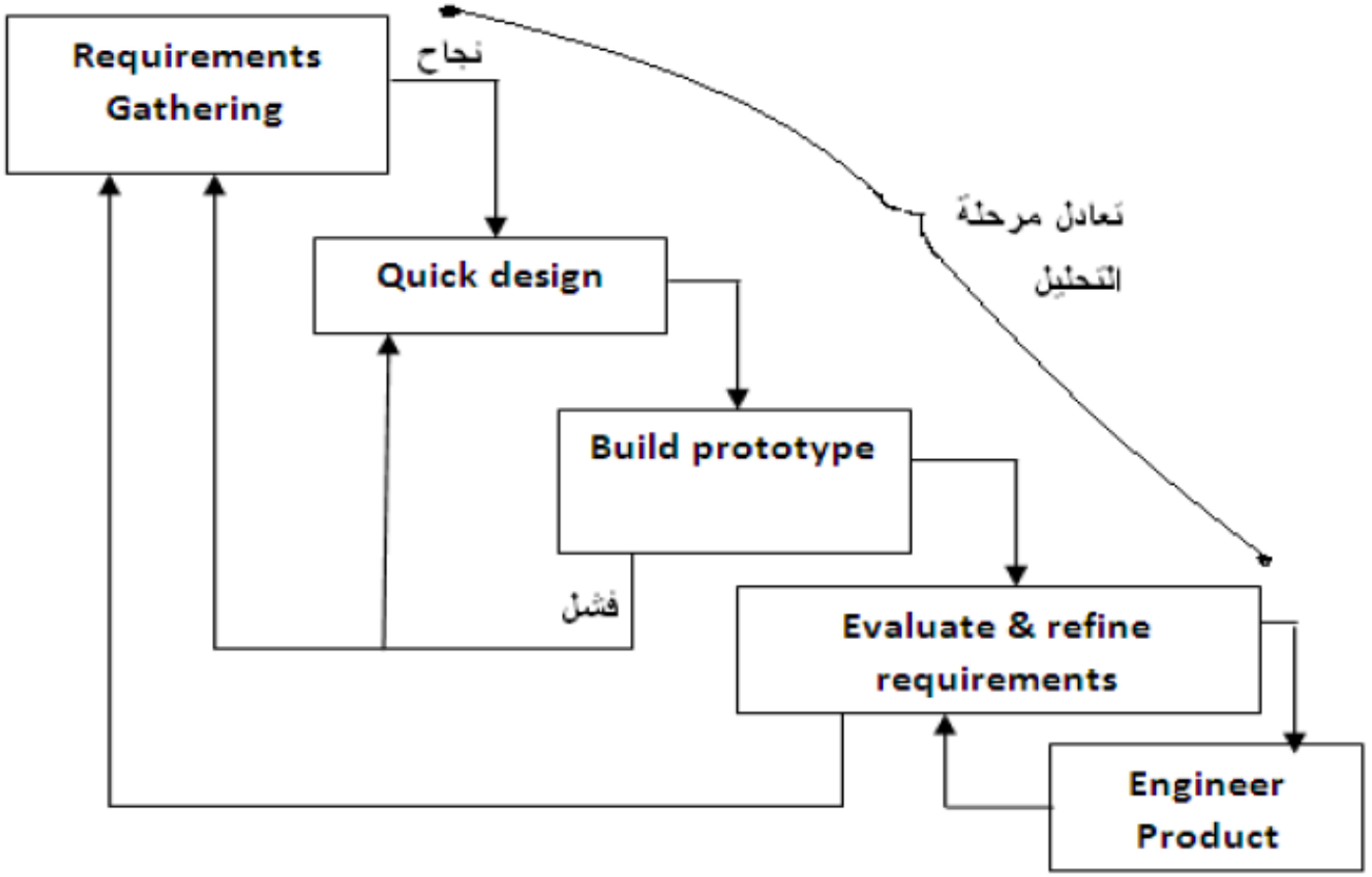
Prototyping is a type of evolutionary development, the method of building a software where developers get the general idea of what is needed by the users, and then build a fast, high-level version of the software as the beginning of the project.

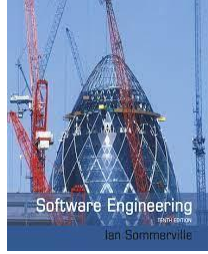
The idea of prototyping is to quickly get a version of the software in the hands of the users and to jointly evolve the system through a series of iterative cycle of design.

النماذج الأولية هي نوع من التطوير التطوري، وهي طريقة بناء برنامج حيث يحصل المطورون على فكرة عامة عما يحتاجه المستخدمون، ثم يقومون ببناء نسخة سريعة وعالية المستوى من البرنامج كبداية للمشروع.

تتمثل فكرة النماذج الأولية في الحصول بسرعة على نسخة من البرنامج في أيدي المستخدمين وتطوير النظام بشكل مشترك من خلال سلسلة من دورات التصميم التكرارية.

Phases of prototyping Model





Strengths of Prototyping Model

- **Improved user communications.**
- **Users like it.**
- **Speeds up development process.**
- **Good for eliciting software requirements.**
- **Provides a tangible model to serve as basic for production version.**
- **This is useful when requirements are changing rapidly.**

تحسين اتصالات المستخدم.

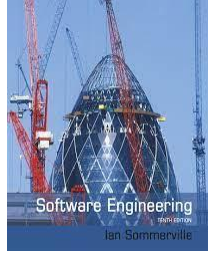
المستخدمين يحبون ذلك.

يسرع عملية التطوير.

جيد لاستنباط متطلبات البرمجيات.

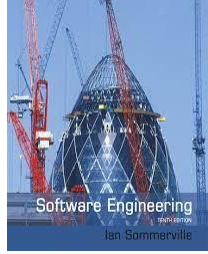
يوفر نموذجًا ملموسًا ليكون بمثابة الأساس لإصدار البرنامج النهائي.

وهذا مفيد عندما تتغير المتطلبات بسرعة.



Weaknesses of Prototyping Model

- **It is impossible to know at the outset of the project how along it will take.**
- **There is no way to know the number of iterations that will be required.**
- **It is difficult to build an accurate cost estimate.**
- **Documentation may be more difficult.**
- **This approach is not suitable for all software.**
- **The user sees what appears to be a working version of software. Actually, it is only mock-ups of software.**



نقاط الضعف في نموذج النماذج الأولية

من المستحيل أن نعرف في بداية المشروع المدة التي سيستغرقها.

لا توجد طريقة لمعرفة عدد التكرارات المطلوبة.

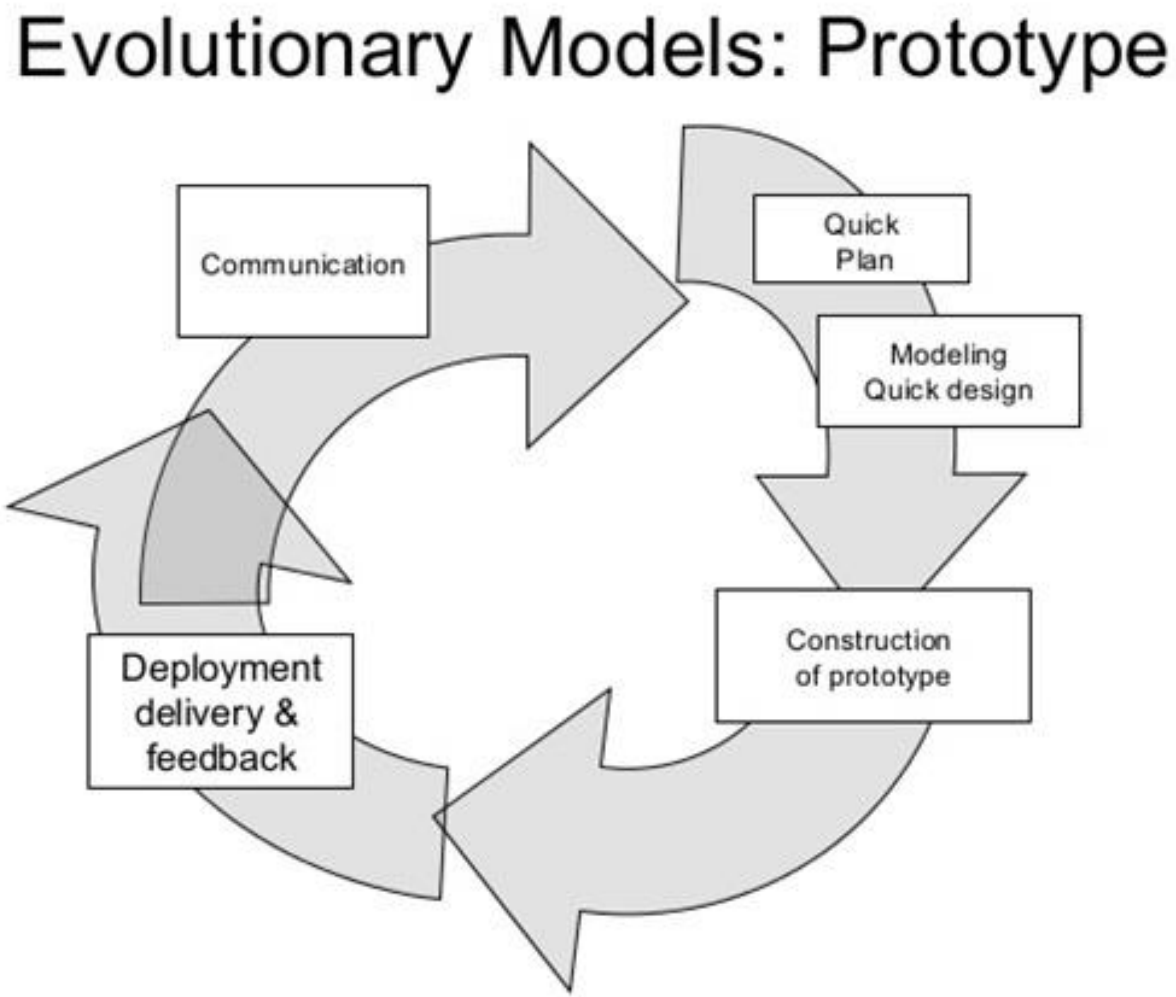
من الصعب بناء تقدير دقيق للتكلفة.

قد يكون التوثيق أكثر صعوبة.

هذا الأسلوب غير مناسب لجميع البرامج.

يرى المستخدم انها نسخة غير صالحة للعمل مثل البرنامج النهائي.

Prototyping Model



Thank You

Any Question / Suggestions