جامعة الحمدانية

كلية التربية للعلوم الصرفة

قسم علوم الحاسوب

المرحلة الثالثة

هندسة البرمجيات

Software Engineering

المقدمة

لم يعد خافيا علينا اهمية البرمجيات Software في حياتنا اليومية سواء في البيت أو المصنع أو المستشفى أو ... الخ، فنحن نتعامل يوميا مع العديد من الأجهزة والمعدات التي تعتمد في عملها على البرمجيات ومن المهم لنا أن تعمل هذه الأجهزة وبرامجها بالشكل والكفاءة التي نتوقعها منها. لذا فإن هندسة البرمجيات أصبحت اليوم أكثر أهمية من أي وقت مضى.

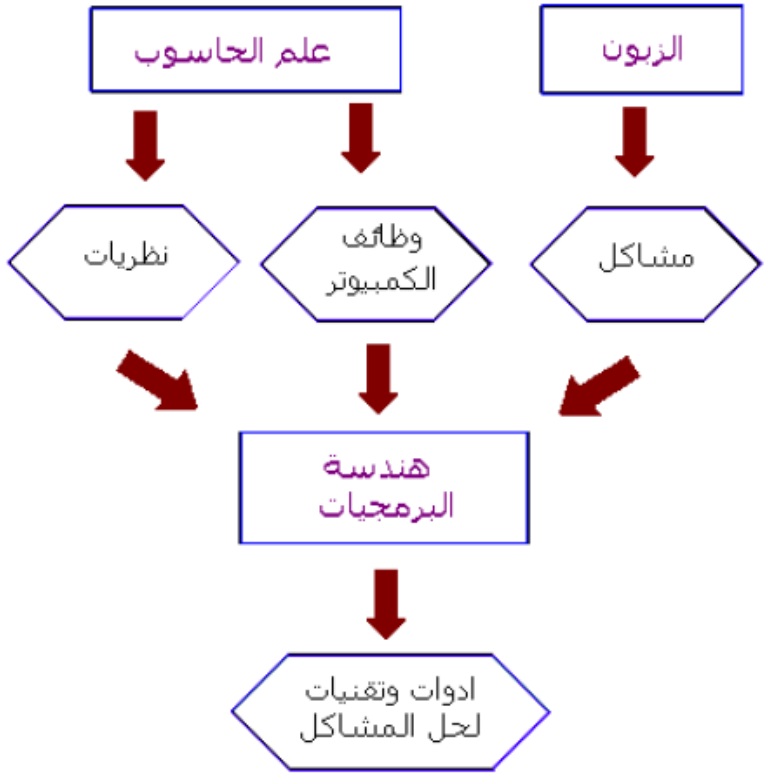
ما معنى هندسة؟

الهندسة Engineering : عبارة عن حل المشاكل. لأن المشكلة هي أساس أي عمل فإذا أردنا عمل مشروع فإنه لدينا مشكلة وهي كيفية عمل المشروع، لذا ظهرت كلمة مشكلة ومعالجة المشاكل مصاحبة للهندسة والمهندس هو الشخص الذي يقوم بحل المشاكل بطرقه العلمية المتقنة.

مهندس البرمجيات Software Engineer : يعتبر أن الحاسوب هي أداة لحل المشاكل problem - solving toolوعليه أن يستخدم معلوماته حول الحاسوب وعلم الحاسوب للمساعدة في حل المشكلة التي يطلب منه إيجاد حل لها.

اذا ما الفرق بين هندسة البرمجيات وعلوم الحاسوب

تعتبر هندسة البرمجيات هي فرع من علوم الحاسوب وكما في الشكل الاتي



شكل (1-1)

Software & Software Engineering

Software: a description of SW might take the following forms:: يمكن أن يأخذ ثلاث أشكال : SW

1. Instructions(computer programs) .

الايعازات أو برامج الحاسوب التي عند تنفيذها تجهز الوظائف والدوال المطلوبة

2. Data structure . التي تمكن البرنامج من معالجة البيانات بشكل ملائم

3. Documents . الوثائق التي تصف عمل النظام وكيفية استخدام

The software problems (crisis أزمة)

The problems that afflict software development can be characterized from a number of different perspectives:

1- Schedule and cost estimates are often inaccurate..الجدول الزمني وتخمين الكلفة غالبا تكون غير دقيقة

2- Productivity of SW people hasn't kept pace with the demand for their services.

إنتاجية الأشخاص الذين يعملون في تطوير الـSW ليست بنفس الكفاءة والسرعة عند طلب الخدمة .

3- Quality of SW is sometime less than adequate. نوعية SW أحيانا اقل من الحد الكافي.

4- Communication between customer and SW developer is often poor.

العلاقة بين الزبون ومصمم أو مطور النظام غالبا ما تكون ضعيفة.

5- Existing SW can be very difficult to maintain. صيانة الـSW الموجودة تكون صعبة.

Solutions:

The better techniques for SW quality assurance can achieve a discipline for SW development. A discipline is called ***Software engineering*(SWE).**

أفضل تقنية لضمان جودة الـ SW (لحل هذه المشاكل) وتطوير البرمجيات هي ما يدعى بـ SWE.

Definition of Software Engineering (SWE) :-

***Software engineering*(SWE)**: the technological discipline concerned with systematic production and maintenance of economically SW that is reliable, efficient, and modified with a minimum cost & effort.

SWE:هو فرع من فروع التكنولوجيا الذي يهتم بتطوير أو إنتاج وصيانة SW اقتصادي بحيث يكون معتمد عليه و كفوء وقابل للتحديث وبأقل جهد واقل كلفة .

SWE: The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.

أو تعريف أخر لـ SWE: هو المؤسسة التي تستخدم المبادئ الهندسية للحصول علىSW اقتصادي ويعمل بكفاءة عالية.

Software Engineering Goals:-

1. Low cost of production.

2. High performance.

3. Portability.

4. Low cost of maintenance. (Corrective , Adaptive and Enhancement Maintenance)

5. High reliability.

6. Delivery on time.

The factors that affect software productivity:- العوامل المؤثرة على انتاجية الـSW

The major factors that affect SW productivity are:-

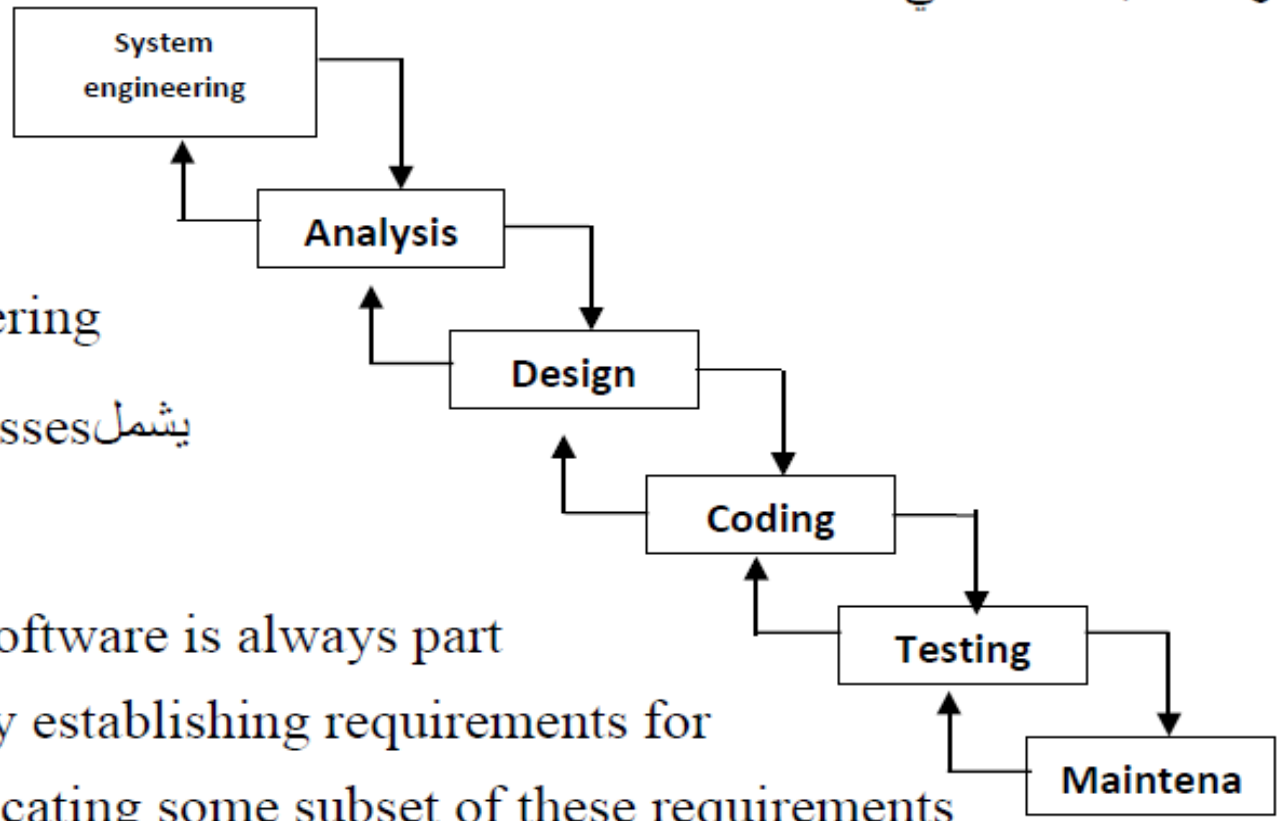
1. The size of the team and their experience. حجم الفريق(عدد أعضاء الفريق) ومدى خبرتهم.
2. The personality of the team members. شخصية أعضاء الفريق
3. The complexity of the problem. . درجة تعقيد المشكلة
4. If there are changes in requirements or design during the development cycle.
   إذا حدث تغيير بالمتطلبات أو التصميم خلال دورة التطوير.
5. The SW design techniques that are applied and the review process that is used.
   تقنيات تصميم SW وتنقيح العملية المستخدمة .
6. The implementation language, the software development tools, the computer HW, and other SW resources that are available.
   اللغة المستخدمة وأدوات تطوير الـSW والـHW ومصادر الـ SW المتوفرة.
7. If the SW under development has reliability or performance requirements.
   إذا كان SW الذي هو تحت التطوير له متطلبات تخص الوثوقية (المعولية) أو الأداء.

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 يتصـرف tests to uncoverيكشـف errors and ensure that defined input will produce actual results that agree with required results.

حالما تنتهي عملية كتابة البرنامج. نقوم بعملية الاختبار:

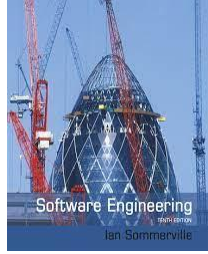
1- نتأكد من كل جملة في البرنامج قد تم اختبارها.

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.

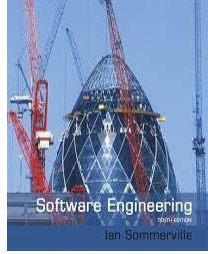
ال SW بلا شك سيخضع للتغييرات بعد تسليمه للزبون، والتغييرات ستحصل بسبب الأخطاء التي ستلاقى بسبب ان ال SW يجب ان يتكيف مع التغييرات في البيئة الخارجية.

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



- **Well Engineered Software**

- **Software VS Program**



Well Engineered Software

1. Maintainability

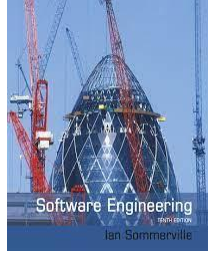
The software can be easily understood and changed over time if problems occur.

2. Reliability

The software performs as expected. Continuity of correct service.

3. Efficiency

The software is produced in the expected time. The software that is produced runs within the time expected for various computations to be completed.



Well Engineered Software

4. Usability

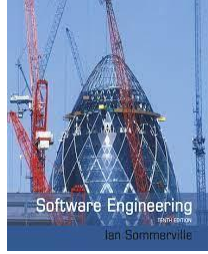
The software can be used properly.

5. Modifiability

The software can be easily change.

6. Portability

The software system can be ported to other computers or systems without major rewriting of the software.



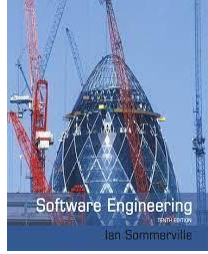
Well Engineered Software

7. Testability

The software can be easily tested.

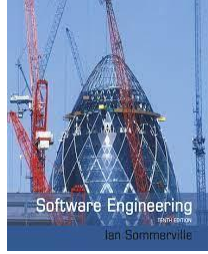
8. Reusability

Some or all the software can be used again in other projects.



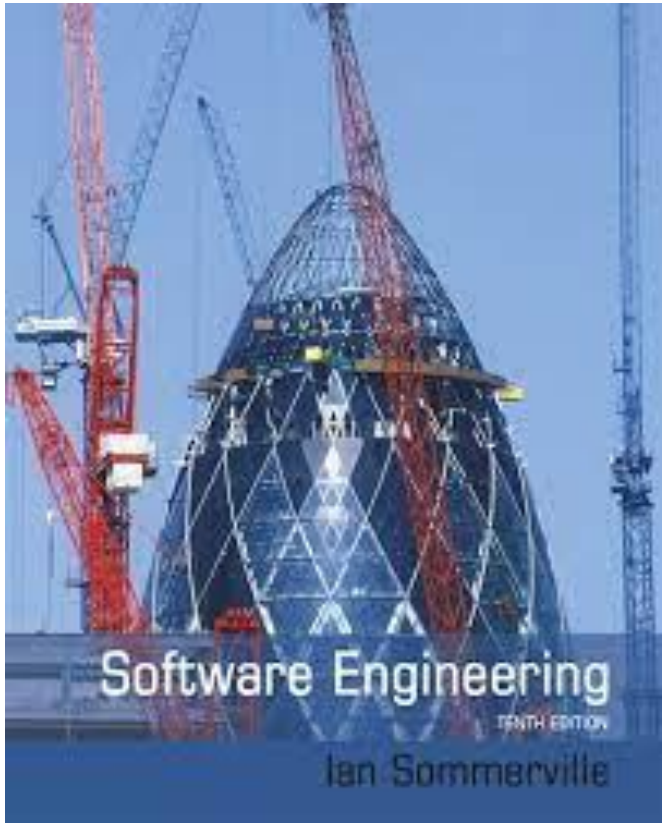
Software VS Program

	Program	Software
1	Programs are developed by individuals for their personal use.	A software is usually developed by a group of engineers working in a team.
2	Usually small in size	Usually large in size
3	Single user	Large number of users
4	Lacks proper documentation	Good documentation support
5	Lack of user interface	Good user interface



Reference

Sommerville, Ian “Software Engineering”, 9th edition, Addison-Wesley.



Thank You

Any Question / Suggestions