What is Software?

***Software** is:*

*1- **instructions** (computer programs) that when executed provide desired features, function, and performance;*

1- تعليمات (برامج الكمبيوتر) التي عند تنفيذها توفر المبز ات المطلو بة، و ظبفة، و الأداء

*2- **data structures** that enable the programs to adequately manipulate information and*

2- هياكل البيانات التي تمكن البرامج من التلاعب بشكل كاف بالمعلومات و

*3- **documentation** that describes the operation and use of the programs.*

3- الوثائق التي تصف تشغيل واستخدام البرامج.

What is Software?

Software is a **logical rather than a physical system element**. Therefore, software has characteristics that are considerably different than those of hardware:

البرنامج هو عنصر منطقي بدلا من عنصر فيزيائي. ولذلك، فإن للبرنامج خصائص تختلف اختلافا كبيرا عن خصائص الأجهزة:

- *Software is developed or engineered, it is not manufactured in the classical sense.*

تم تطوير أو هندسة البرمجيات، فإنه لم يتم تصنيعها بالمعنى الكلاسيكي.

- *Software doesn't "wear out."*

البرنامج لا "يرتدي".

- *Although the industry is moving toward component-based construction, most software continues to be custom-built.*

على الرغم من أن الصناعة تتحرك نحو البناء القائم على المكون، فإن معظم البرامج لا تزال مصممة خصيصا.

Software Applications

- system software
- engineering/scientific software
- embedded software
- product-line software
- Web / Mobile applications
- AI software

س \ تطبيقات البرمجيات ؟

- نظام البرمجيات
- تطبيق البرمجيات
- الهندسة / البرمجيات العلمية
- جزءا لا يتجزأ من البرمجيات
- خط الانتاج البرمجيات
- الويب / تطبيقات الجوال
- أي البرمجيات

Legacy Software

برنامج تراث

Legacy software systems . . . were developed decades ago and have been continually modified to meet changes in business requirements and computing platforms. The proliferation of such systems is causing headaches for large organizations who find them costly to maintain and risky to evolve.

أنظمة البرمجيات القديمة. . . وقد تم تطويرها منذ عقود وتم تعديلها باستمرار لمواجهة التغيرات في متطلبات الأعمال ومنصات الحوسبة. إن انتشار مثل هذه النظم يسبب صداعا للمنظمات الكبيرة التي تجدها مكلفة للحفاظ عليها ومخاطرها في التطور.

Why must it change?

س \ لماذا يجب أن تتغير؟

- software must be adapted to meet the needs of new computing environments or technology.

يجب تكييف البرمجيات لتلبية احتياجات بيئات الحوسبة الجديدة أو التكنولوجيا.

- software must be enhanced to implement new business requirements.

يجب تعزيز البرمجيات لتنفيذ متطلبات العمل الجديدة.

- software must be extended to make it interoperable with other software must be re-architected to make it viable within a network environment.

يجب تمديد البرنامج ليصبح قابلا للتشغيل المتبادل مع أنظمة أو قواعد بيانات أخرى أكثر حداثة.

Characteristics of WebApps - I

خصائص تطبيقات الويب - إي

The following attributes are encountered in the vast majority of WebApps:

تمت مصادفة السمات التالية في الغالبية العظمى من تطبيقات الويب:

- Network intensiveness. A WebApp resides on a network and must serve the needs of a diverse community of clients.

الشبكة الكثافة : ويب التطبيق يقيم على شبكة ويجب أن تخدم احتياجات مجتمع متنوع من العملاء

- **Concurrency**. A large number of users may access the WebApp at one time.

التزامن: عدد كبير من المستخدمين قد الوصول إلى ويباب في وقت واحد.

- **Unpredictable** load. The number of users of the WebApp may vary by orders of magnitude from day to day.

تحميل غير متوقعة. قد يختلف عدد مستخدمي ويباب بأوامر من يوم لآخر.

- **Performance**. If a WebApp user must wait too long (for access, for server-side processing, for client-side formatting and display), he or she may decide to go elsewhere.

الأداء: إذا كان المستخدم ويباب يجب الانتظار وقتا طويلا (للوصول، لمعالجة من جانب الخادم، لتنسيق جانب العميل وعرض)، وقال انه قد انها قررت الذهاب إلى أي مكان آخر.

- **Availability**. Although expectation of 100 percent availability is unreasonable, users of popular WebApps often demand access on a "24/7/365" basis.

التوفر: على الرغم من أن توقع توافر 100 في المئة غير معقول، مستخدمي تطبيقات ويب شعبية في كثير من الأحيان تتطلب الوصول على أساس "365/7/24".

Characteristics of WebApps - II

خصائص تطبيقات الويب - إي

- **Data driven.** The primary function of many WebApps is to use hypermedia to present text, graphics, audio, and video content to the end-user.

البيانات مدفوعة: الوظيفة الأساسية للعديد من تطبيقات الويب هي استخدام الوسائط الفائقة لعرض النص والرسومات والصوت ومحتوى الفيديو إلى المستخدم النهائي.

- **Content sensitive:** The quality and aesthetic nature of content remains an important determinant of the quality of a WebApp.

محتوى حساس: نوعية وطبيعة الجمالية للمحتوى لا تزال عاملا هاما في تحديد نوعية من تطبيق ويب.

- **Continuous evolution:** Unlike conventional application software that evolves over a series of planned, chronologically-spaced releases, Web applications evolve continuously.

التطور المستمر: على عكس برامج التطبيقات التقليدية التي تتطور عبر سلسلة من الإصدارات المخطط لها زمنيا، تتطور تطبيقات الويب بشكل مستمر .

- **Immediacy:** Although *immediacy*—the compelling need to get software to market quickly—is a characteristic of many application domains, WebApps often exhibit a time to market that can be a matter of a few days or weeks.

الفورية: على الرغم من أن الفورية - الحاجة الملحة للحصول على البرمجيات لتسويق بسرعة - هو سمة من سمات العديد من مجالات التطبيق، تطبيقات الويب غالبا ما تظهر وقتا للسوق التي يمكن أن تكون مسألة بضعة أيام أو أسابيع.

- **Security:** Because WebApps are available via network access, it is difficult, if not impossible, to limit the population of end-users who may access the application.

الأمن: لأن هي تطبيقات الويب المتاحة عبر الوصول إلى الشبكة، فإنه من الصعب، إن لم يكن من المستحيل، للحد من عدد السكان من المستخدمين النهائيين الذين قد الوصول إلى التطبيق.

- **Aesthetics:** An undeniable part of the appeal of a WebApp is its look and feel.

جماليات: جزء لا يمكن إنكاره من نداء من التطبيق على شبكة الإنترنت هو مظهرها ويشعر.

Software Engineering

هندسة البرمجيات

Some realities:

- *a concerted effort should be made to understand the problem before a software solution is developed*
- *design becomes a pivotal activity*
- *software should exhibit high quality*

بعض الحقائق:

ينبغي بذل جهود متضافرة لفهم المشكلة قبل تطوير حل برمجي

يصبح التصميم نشاطا محوريا

البرمجيات يجب أن يحمل جودة عالية

يجب أن تكون البرمجيات قابلة للصيانة

The seminal definition:

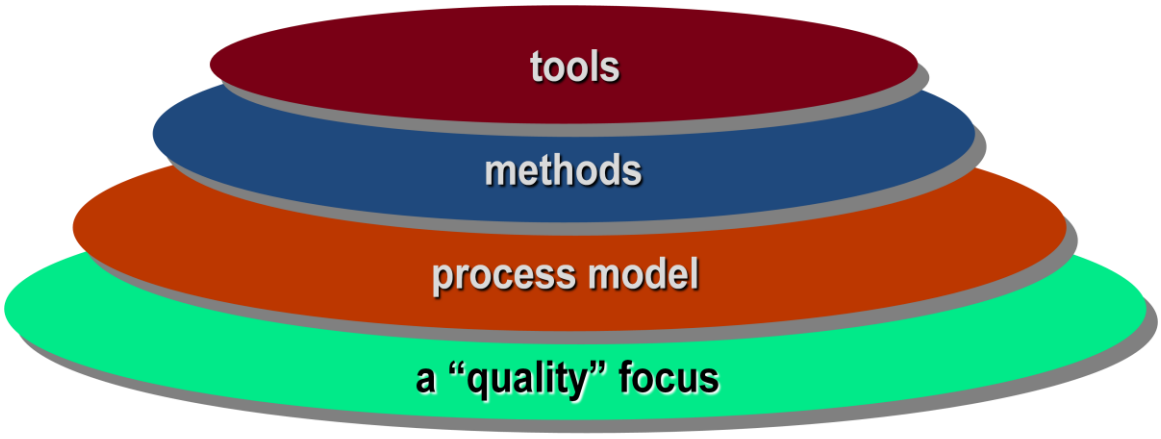
- *[Software engineering is] the establishment and use of engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.*

التعريف الأساسي:

[هندسة البرمجيات] إنشاء واستخدام المبادئ الهندسية من أجل الحصول على البرمجيات الاقتصادية التي يمكن الاعتماد عليها ويعمل بكفاءة على آلات حقيقية

A Layered Technology

تقنية الطبقات



S/W Process

- In the context of software engineering, a process is not a rigid prescription for **how to build computer software**. Rather, it is an **adaptable approach that enables the people doing the work** (the software team) to pick and choose the appropriate set of work actions and tasks.

في سياق هندسة البرمجيات، عملية ليست وصفة جامدة لكيفية بناء برامج الكمبيوتر. بل هو نهج قابل للتكيف تمكن الناس من القيام بالعمل (فريق البرمجيات) لاختيار واختيار مجموعة مناسبة من الإجراءات والمهام العمل.

- The intent is always to **deliver software** in a **timely manner** and with **sufficient quality** to **satisfy** those who have **sponsored its creation** and those who will **use it**.

القصد هو دائما لتقديم البرامج في الوقت المناسب وبجودة كافية لتلبية أولئك الذين رعوا إنشائها وأولئك الذين سوف تستخدمه.

A Process Framework

إطار عمل

A ***process framework*** establishes the *foundation for a complete software engineering process* by identifying a **small number of framework activities** that are applicable to all software projects, regardless of their size or complexity.

ويضع إطار العمليات الأساس لعملية هندسة برمجيات كاملة عن طريق تحديد عدد صغير من الأنشطة الإطارية التي تنطبق على جميع مشاريع البرمجيات بغض النظر عن حجمها أو تعقيدها.

In addition, the process framework encompasses a set of umbrella activities that are applicable across the entire software process. *A generic process framework for software engineering* encompasses ***five activities***:

وبالإضافة إلى ذلك، فإن إطار العملية يشمل مجموعة من الأنشطة الشاملة التي تنطبق على كامل عملية البرمجيات. ويشمل إطار العمليات العامة لهندسة البرمجيات خمسة أنشطة:

Framework Activities

أنشطة الإطار

- Communication
- Modeling
  - Analysis of requirements
  - Design
- Construction
  - Code generation
  - Testing
- Deployment

الاتصالات

تخطيط

النماذج

تحليل المتطلبات

التصميم

اعمال بناء

رمز الجيل

اختبارات

نشر

Umbrella Activities

أنشطة مظلة

- Software project tracking and control
- Risk management
- Software quality assurance
- Measurement
- Software configuration management
- Reusability management
- Work product preparation and production

تتبع مشروع البرمجيات والتحكم فيها

إدارة المخاطر

ضمان جودة البرمجيات

مراجعات فنية رسمية

قياس

إدارة تكوين البرامج

إدارة إعادة الاستخدام

العمل إعداد المنتج والإنتاج

The Essence of Practice

جوهر الممارسة

- George Polya suggests:

1- Understand the problem

(communication and analysis).

2- Plan a solution

(modeling and software design).

3- Carry out the plan

(code generation).

4- Examine the result for accuracy

(testing and quality assurance).

1- فهم المشكلة

(التواصل والتحليل).

2- التخطيط لحل

(النمذجة وتصميم البرمجيات).

3- تنفيذ الخطة

(توليد الرموز).

4-فحص النتيجة للتأكد من دقتها
(اختبار وضمان الجودة).

Understand the Problem

فهم المشكلة

- *Who has a stake in the solution to the problem?* That is, who are the stakeholders?

من لديه مصلحة في حل المشكلة؟ أي من هم أصحاب المصلحة؟

- *What are the unknowns?* What data, functions, and features are required to properly solve the problem?

ما هي المجهول؟ ما هي البيانات والوظائف والميزات المطلوبة لحل المشكلة بشكل صحيح؟

- *Can the problem be compartmentalized?* Is it possible to represent smaller problems that may be easier to understand?

هل يمكن تقسيم المشكلة؟ هل من الممكن تمثيل مشاكل أصغر قد يكون من الأسهل فهمها؟

- *Can the problem be represented graphically?* Can an analysis model be created?

هل يمكن تمثيل المشكلة بيانيا؟ هل يمكن إنشاء نموذج تحليل؟

Plan the Solution

تخطيط الحل

- *Have you seen similar problems before?* Are there patterns that are recognizable in a potential solution? Is there existing software that implements the data, functions, and features that are required?

هل رأيت مشاكل مماثلة من قبل؟ هل هناك أنماط يمكن التعرف عليها في حل محتمل؟ هل هناك برامج موجودة تقوم بتنفيذ البيانات والوظائف والميزات المطلوبة؟

- *Has a similar problem been solved?* If so, are elements of the solution reusable?

هل تم حل مشكلة مشابهة؟ إذا كان الأمر كذلك، هل عناصر الحل قابلة لإعادة الاستخدام؟

- *Can subproblems be defined?* If so, are solutions readily apparent for the subproblems?

هل يمكن تعريف المشاكل الفرعية؟ إذا كان الأمر كذلك، هل الحلول واضحة بسهولة للمشاكل الفرعية؟

- *Can you represent a solution in a manner that leads to effective implementation?* Can a design model be created?

هل يمكن أن تمثل حلا بطريقة تؤدي إلى التنفيذ الفعال؟ هل يمكن إنشاء نموذج تصميم؟

Carry Out the Plan

تنفيذ الخطة

- *Does the solution conform to the plan?* Is source code traceable to the design model?

هل يتوافق الحل مع الخطة؟ هل يمكن تتبع شفرة المصدر إلى نموذج التصميم؟

- *Is each component part of the solution provably correct?* Has the design and code been reviewed, or better, have correctness proofs been applied to algorithm?

هل كل جزء مكون من الحل صحيح بشكل صحيح؟ هل تمت مراجعة التصميم والرمز، أو أفضل، تم تطبيق البراهين الصحيحة على الخوارزمية؟

Examine the Result

فحص النتيجة

- *Is it possible to test each component part of the solution?* Has a reasonable testing strategy been implemented?

هل من الممكن اختبار كل جزء مكون من الحل؟ هل تم تنفيذ استراتيجية اختبار معقولة؟

- *Does the solution produce results that conform to the data, functions, and features that are required?* Has the software been validated against all stakeholder requirements?

هل ينتج الحل نتائج تتفق مع البيانات والوظائف والميزات المطلوبة؟ هل تم التحقق من صحة البرنامج ضد جميع متطلبات أصحاب المصلحة؟

Hooker's General Principles

المبادئ العامة هوكر

Hooker proposes **seven principles** that focus on software Engineering practice as a whole

هوكر يقترح سبعة مبادئ التي تركز على هندسة البرمجيات الممارسة ككل

- **1: *The Reason It All Exists***

to provide value to its users

1: السبب كل شيء موجود

لتوفير قيمة لمستخدميها

- **2: *KISS (Keep It Simple, Stupid!)***

All design should be as simple as possible, but no simpler .

2: قبلة (يبقيه بسيط، غبي!)

وينبغي أن يكون كل تصميم بسيط قدر الإمكان، ولكن لا أبسط.

- **3: *Maintain the Vision***

A clear vision is essential to the success of a software project

3: الحفاظ على الرؤية

رؤية واضحة أمر ضروري لنجاح مشروع البرمجيات

Hooker's General Principles

المبادئ العامة هوكر

- **4: *What You Produce, Others Will Consume***

Always specify, design, and implement ..knowing someone else will have to understand what you are doing

4: ما كنت تنتج، الآخرين سوف تستهلك

دائما تحديد وتصميم وتنفيذ .. معرفة شخص آخر سوف تضطر إلى فهم ما تقومون به.

- **5: *Be Open to the Future***

A system with a long lifetime has more value, true "industrial-strength" software systems must endure far longer.

5: كن مفتوحا للمستقبل

نظام مع عمر طويل له قيمة أكثر، صحيح "الصناعية-- قوة" نظم البرمجيات يجب أن تحمل أكثر من ذلك بكثير

- **6: *Plan Ahead for Reuse***

Planning ahead for reuse reduces the cost and increases the value of both the reusable components and the systems into which they are incorporated .

6: التخطيط للمستقبل لإعادة الاستخدام

إن التخطيط المسبق لإعادة الاستخدام يقلل من التكلفة ويزيد من قيمة كل من المكونات القابلة لإعادة الاستخدام والنظم التى يتم دمجها فيها.

- ***7: Think!***

Placing clear, complete thought before action almost always produces better results

7: فكر!

وضع واضح، والفكر الكامل قبل العمل تنتج دائما نتائج أفضل دائما