محاضرة 1

هندسة البرمجيات (Software Engineering)

هندسة البرمجيات Software Engineering هي علم يهتم بتطوير وتصميم البرمجيات عالية الجودة. تهتم هندسة البرمجيات بتلوين البرنامج منذ مراحله الأولى أثناء تحليل المشكلة وحتى التصميم وكتابة البرنامج من القيام بتجربته واختباره وتثبيته على الأجهزة والقيام بعملية صيانته.

هندسة البرمجيات هي فرع من المعرفة الهندسية Engineering Discapline يختص بكل أشكال انتاج البرمجيات All Aspects of Software Production بدءاً من الخواطر المبكرة لخصائص النظام حتى صيانته بعد استخدامه.

* تاريخ هندسة البرمجيات

استخدمت هندسة البرمجيات كمفهوم نظري من حين لآخر في أواخر الخمسينات وبداية الستينات من القرن الماضي. أما الاستخدام الرسمي الأول لهذا المصطلح فكان في مؤتمر عقد من قبل الجمعية العلمية في منظمة حلف شمال الأطلسي 1968 حول البرمجيات وقد أخذ هذا المصطلح بالانتشار منذ ذلك الحين ولاقى اهتماماً متزايداً في دوائر مختلفة. عقد المؤتمر لمعالجة ما يعرف "أزمة البرمجيات" والتي ظهرت بسبب عدم استخدام منهجية في التفكير (Software Development process) عند بناء أكثر مما هو ممكن لها وبعد تحمل التأخر في الوقت وتجاوز الميزانية كانت البرمجيات ذات كفاءة ضعيفة في إنجاز الوظائف المطلوبة وخلل في الفعالية كذلك بعدم تلبية كافة المتطلبات بالشكل الكامل أو الصحيح.

البرمجية (Software) شئ غير ملموس - وهي سلسلة من الأوامر أو التعليمات التي تطلب من الحاسوب إجراء عمليات معينة مثل عرض المعلومات أو إجراء الحسابات أو تخزين البيانات.

هذه البرمجيات هي نتاجة الدمج من الجسد في النظام الحاسوبي وهي في توسع دائم وازدياد في التعقيد والمتطلبات والمهام التي تقوم بتنفيذها - أما هندسة البرمجيات فهي فرع من فروع الهندسة يقوم على مجموعة أسس وقواعد تهدف الى تصميم وتطوير البرامج وصيانتها بعد استخدامها.

لا يكفي لإيجاد البرمجية المتكاملة والكبيرة عمل شخص واحد وانما يتطلب ذلك فريقاً من المهندسين الجيدين. وقد كان ضرورياً ايجاد علم يعنى بهندسة البرمجيات لوضع الأسس والمعايير التي تضمن هذه الجهة من المتطلبات بحيث يصبح بالامكان تمييز البرنامج الجيد من غيره.

* الفرق بين هندسة البرمجيات وعلم الحاسب.

علم الحاسب يهتم بالنظريات والأسس والمفاهيم بينما تهتم هندسة البرمجيات بالجوانب العملية لتطوير وانتاج البرمجيات لبناء قاعدة.

* الفرق بين هندسة البرمجيات وهندسة النظم

هندسة النظم تهتم بكل اشكال تطوير النظم المبنية على الكمبيوتر (Computer-based system Development) بما فيها العتاد والبرامج وعمليات الهندسة وتعد هندسة البرمجيات جزءاً من هذه العملية.

ويهتم مهندس النظم بمواصفات النظم Specification

والتكامل فيما بينها (Architectural Design) التصميم

• Deployment وتوزعها (integration)

تكاليف هندسة البرمجيات Cost of SE

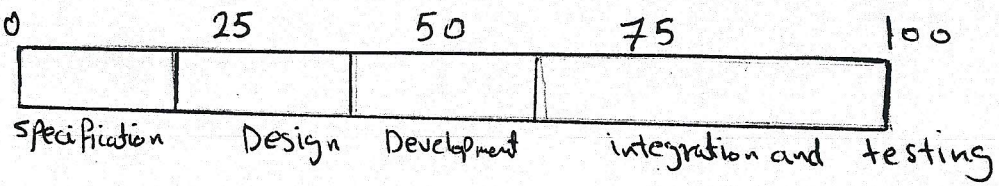
حوالى ٦٠٪ من التكاليف هى تكاليف تطوير Development Cost

وحوالى أربعين بالمائة ٤٠٪ هى تكاليف فحص واختبار Testing Costs

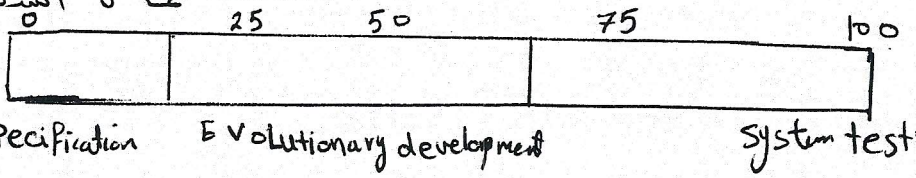
* تختلف تقديرات التكلفة اعتماداً على نوع النظام الذى يجرى تطويره ومتطلبات هذا النظام، وخصائصه، والأداء فيه، والاعتمادية Reliability

كما تعتمد التكاليف على النموذج المستخدم فى التطوير Development Model

ليست هناك إجابة شافية عن توزيع التكاليف على عمليات البرمجيات التى تعتمد على العمليات ونوع البرمجيات التى يتم تطويرها، وقد تكون بالشكل التالى كنوع من التقدير المبدئى الذى يلاحظ فيه ان تجميع واختبار البرمجيات يستحوذ على جزء كبير من التكلفة الذى يصل إلى ٤٠٪ وقد يصل فى بعض الأنظمة الحرجة إلى حوالى ٥٠٪ من إجمالى تكلفة النظام



عند تطوير برمجيات قد لا يظهر خط فاصل واضح بين مراحل المواصفات والتصميم والتطوير لذلك قد تمثل التكلفة كما فى الشكل التالى:



تنخفض فى هذه الحالة تكلفة تحديد المواصفات لأنه ينتج فقط المستوى العالى من تحديد المواصفات High-level specification قبل التطوير، وتنفذ عمليات تحديد المواصفات Specification، والتصميم Design، والتنفيذ Implementation، والتجميع integration، والاختبار testing على التوازى داخل الخطة التطويرية.

SE methods وتقنيات هندسة البرمجيات

طرق هندسة البرمجيات هي اقتران نموذج أو منهجية هيكلية Structured Approaches لتطوير البرمجيات تحتوي على نماذج النظام System Models والملاحظات Notations والقواعد Rules ونصائح التصميم Design Advice والخطوط العامة للعمليات process Guidance

وصف النموذج Model descriptions هي أوصاف النماذج الرسومية التي يجب انتاجها، والقواعد هي القيود التي تطبق على نماذج النظام، والخطوط العامة للعمليات هي تحديد الأنشطة التي يتم اتباعها، ومن التوصيات اتباع نصائح ممارسة التصميم الجيد.

طرق هندسة البرمجيات هي منهجية لتطوير البرمجيات بغرض تسهيل انتاج برمجيات عالية الجودة بطريقة اقتصادية Cost Effective، كانت الطرق الأولى التي نشأت من التحليل الهيكلي

Structured Analysis (DeMarco, 1978) وطريقة JSD (Jackson, 1983) في السبعينات، وحاولت هذه الطرق تعريف الكيانات الوظيفية لنظام، ومازالت هذه الطرق الموجهة بالوظائف (الوظائفية الخ) Function Oriented Methods مستخدمة على نطاق واسع.

في الثمانينات والتسعينات من القرن الماضي 1980s 1990s استخدمت هذه الطرق بالطرق الكائنية الخ object-Oriented Methods مثل الطرق المقدمة بواسطة كل من Booch and Rumbaugh، وتحسن هذه المنهجيات المختلفة في منهجية واحدة موحدة single unified Approach مبنية على لغة النمذجة الموحدة UML (Unified Modeling language) بين كل الطرق على فكرة نماذج تطوير نظام التي يمكن تمثيلها بالرسم Graphically، وباستخدام هذه النماذج كمواصفات نظام Specification

او تصميم (Design) يجب ان تحتوي الطرق على عدد من
المكونات المختلفة، ولا توجد طريقة نموذجية، والطرق المختلفة
مناطق مختلفة لتطبيقها، وكمثال فإن الطرق الكائنية الهدف
Interactive Systems مرتبطة في النظم التفاعلية object-Oriented Methods
للتفاعل ليست كذلك في نظم متطلبات الوقت الحقيقي (الحرجة)
(Stringent Real-Time Requirements)

* خصائص البرمجيات الجيدة Attributes of good software

المنتجات البرمجية عدد من الخصائص المرتبطة بها، والتي تقيس جودة
هذه البرمجيات وهي:-

» قابلية الصيانة
Maintainability تعني ان البرمجيات يجب ان تقبل
احتياجات التغيير

» الاعتمادية Dependability تعني ان تكون البرمجيات جديرة
بالثقة وموثوق بها

» الكفاءة:-
Efficiency تعني الا يكون هناك إهدار لموارد
النظام Resources

» قابلية الاستخدام Usability ان تكون مفهوم وقابل للاستخدام
من قبل المستخدمين لها هي مصممة له.

التحديات التي تواجه هندسة البرمجيات

١- تحدي التسليم Delivery مشكلة تقليدية يعاني منها إنتاج البرمجيات
وهناك ضغط متناسب على زيادة سرعة تسليم البرمجيات

٢- الثقة: تقنيات متطورة لا تثبت ان البرمجيات يمكن الوثوق بها من قبل
المستخدمين

٣- النظم الموروثة Legacy القديمة ذات قيمة ويجب صيانتها وتحديثها
مع تفاوت التكاليف العالية في نفس الوقت يجب الاستمرار في تقديم
الخدمات والأعمال