مرجع المحاضرة : كتاب هندسة البرمجيات – تأليف د.أحمد شعبان دسوقي – أ.د.السيد محمود الربيعي

أهداف الوحدة

عزيزي الدارس،

بنهاية دراسة هذه الوحدة ينبغي أنّ تكون قادراً على أن :

- تعرِّف هندسة البرمجيات.
- صف دور هندسة البرمجيات في هندسة وبناء البرمجيات.
- تشرح الفرق بين علم هندسة البرمجيات وعلوم الحاسبات الآلية.
- تصف العلاقة بين هندسة البرمجيات وهندسة النظم..
- تعدد عمليات البرمجيات.
- تعرِّف أدوات هندسة البرمجيات بمساعدة الحاسب الآلي.
- تشرح أهمية أدوات هندسة البرمجيات بمساعدة الحاسب الآلي في بناء البرمجيات.
- تسرد المهام النموذجية لهندسة البرمجيات.
- تعدد مبادئ هندسة البرمجيات.
- تسمي أهم أنواع البرمجيات.
- تشرح التحديات الرئيسة التي تواجه هندسة البرمجيات.
- تصف المسئولية المهنية والأخلاقية لمهندسي البرمجيات والعمل بها.

1. مـا هـي هندسـة البرمجيـات؟ (What is Software Engineering?)

يوجد العديد من التعريفات التي تم اقتراحها من العديد من المهتمين بهندسة البرمجيات ويمكن إجمال هذه التعريفات في أن "**هندسة البرمجيات**" هي **:** علم يهتم ببناء الأنظمة البرمجية الكبيرة والمعقدة بواسطة فريق من مهندسي البرمجيات باتباع طرق ومبادئ هندسية منظمة وصحيحة واستخدام وسائل وأدوات وتقنيات تجعل من تصميم وتطوير وبناء واختبار هذه النظم عملية منظمة يمكن تكرارها في حدود الإمكانيات المتاحة مادياً وبشرياً وزمنياً ، وذلك من خلال دراسة دورة حياة النظام مع إعطاء أشكالاً متعددة لعمليات تصميمه وتطويره وبنائه واختباره بحيث يكون المنتج البرمجي النهائي على درجة عالية من الجودة والموثوقية وقليل التكاليف بقدر الإمكان مع تسليمه للعميل في الوقت المناسب على أن يعمل بكفاءة على أجهزة الحاسب الآلي المختلفة.

وبصفة عامة ، يهتم علم هندسة البرمجيات بجميع نواحي إنتاج نظم البرمجيات التي تلبي متطلبات المستخدمين تحت قيود معينة ، وذلك من خلال تطبيق النظريات والمعرفة بأسلوب فعال وبكفاءة موثوقة، وتدخل الطرق المستخدمة في هندسة البرمجيات في جميع أطوار دورة حياة نظم البرمجيات من المراحل الأولى لتحليل النظام حتى صيانته مروراً بعمليات وضع المواصفات ، والتصميم ، والبناء ، والفحص ، والتشغيل. كما يوظف علم هندسة البرمجيات الطرق والأساليب والعمليات والقياسات الهندسية لتحقيق الأهداف وإتمام الفائدة من الأدوات المعدة لإدارة عمليات تطوير البرمجيات ، مع الاهتمام بالجودة والتحكم في النوعية.

وبالتدقيق في مصطلح " هندسة البرمجيات " نجده يحتوي على شقين :

الشق الأول : هندسي : وفيه يطبق المهندسون النظريات والأساليب والأدوات الملائمة بفعالية للحصول على حلول مثالية للمشاكل التي تواجههم مع التقيد بالقيود التنظيمية والمالية التي تحيط بالمشكلة ، ودائماً ما يحاولون اكتشاف الحلول للمشاكل حتى عندما لا توجد نظريات أو أساليب مطبقة لتدعمهم ، وذلك من خلال تبنيهم أسلوب تصنيفي منظم لعملهم مع التركيز على اختيار الطريقة الأكثر ملائمة لتنفيذ النظام البرمجي طبقاً للمواصفات المطلوبة والطريقة الأكثر ابتكاراً.

الشق الثاني : فهو برمجي : وفيه يتم الاهتمام بجميع مظاهر إنتاج البرمجيات ، حيث إن هندسة البرمجيات لا تهتم فقط بالعملية الفنية لتطوير البرمجيات ولكن تهتم أيضاً بالنشاطات الأخرى مثل إدارة مشروع البرمجيات ، وتطوير الأدوات والأساليب والنظريات لدعم إنتاج البرمجيات.

وبناءاً على ما سبق يتضح أن دور هندسة البرمجيات هو : **تطبيق منهج مرتب وقابل للقياس لعمليات تطوير وتشغيل وصيانة البرمجيات ، أي تطبيق الهندسة على البرمجيات** ، وهو ذات التعريف الذي وضعه معهد المهندسين الكهربائيين والإلكترونيين **(IEEE)** لهندسة البرمجيات ، وذلك من خلال الإجابة على العديد من الأسئلة التي تخص هندسة وتطوير البرمجيات ، والتي من أهمها :

- ما هي الطرق والمبادئ الهندسية المنظمة والصحيحة التي يمكن تطبيقها على بناء البرنامج؟.
- ما هي الوسائل والأدوات والتقنيات التي تجعل من تصميم وتطوير وبناء واختبار البرنامج عملية منظمة يمكن تكرارها في حدود الإمكانيات المتاحة مادياً وبشرياً وزمنياً؟
- كيف نتمكن اقتصاديا من بناء برنامج موثوق يعمل بكفاءة ليس على جهاز حاسب واحد بل على العديد من أجهزة الحاسب المختلفة؟.
- ما هي القياسات والإجراءات التي يجب إجراؤها لضمان جودة المنتج البرمجي؟.
- كيف سيتم دعم البرنامج على المدى الطويل عندما يطلب المستخدم إجراء تعديلات أو تحسينات عليه؟.

- ما هو المنهج الواجب اتباعه لدراسة تقنيات الإدارة اللازمة لتخطيط المشاريع البرمجية وتنظيمها واستخدامه كآلية للتحكم في المشروع بشكل يقود إلى تسليم منتج برمجي عالي الجودة قابل للتشغيل في الوقت المحدد؟.

تدريب(1)

ما هو الفرق بين علم هندسة البرمجيات وعلوم الحاسبات الآلية؟

تدريب (2)

ما هو الفرق بين هندسة البرمجيات وهندسة النظم؟

تدريب (3)

ما هي عمليات البرمجيات؟

2. المهام النموذجية لهندسة البرمجيات

(Typical Software Engineering Tasks) :

هناك مجموعة من المهام النموذجية التي تتعلق بهندسة البرمجيات يجب أن تتبع في تطوير المشاريع البرمجية من قِبل فريق متكامل من مهندسي البرمجيات ، وهذه المهام هي كالتالي :

• تحليل المشكلة.	• تحديد المتطلبات.	• تصميم البرمجية.
• الترميز (عملية البرمجة).	• اختبار وتكاملية الشفرة البرمجية.	
• التثبيت وتوزيع البرمجية.	• توثيق البرمجية.	• صيانة البرمجية.
• التدريب	• تقدير الموارد	• إدارة المشروع

وبنظرة ثاقبة للمهام المذكورة أعلاه نجد أن هندسة البرمجيات تهتم بجميع أطوار دورة حياة تطوير البرمجيات (**Software Life Cycle**) بدءاً من المراحل الأولى لتحليل النظام حتى صيانته، وسوف نتناول أطوار دورة حياة تطوير البرمجيات بالتفصيل لاحقاً ، ولكن سوف تغطي فكرة بسيطة هنا عن أهم الخطوات المتبعة (الأطوار) في تطوير البرمجيات من وجهة نظر هندسة البرمجيات :

- مبدئيا لا بد من التعرف على متطلبات النظام فإذا كان هناك زبون أو عدة زبائن فإنه لا بد من حدوث مقابلة بين الزبائن ومتخصصي دراسة وتحليل المتطلبات في فريق التطوير ، وإن لم يكن هناك زبون محدد ولكن هناك متطلبات لفئة عامة من الزبائن فإن دراسة متطلبات واحتياجات السوق يتم عملها وهذه الطريقة مناسبة لتطوير برمجيات عامة.

- بعد دراسة وتحليل المتطلبات والانتهاء من كتابة وثائق التحليل يتم الانتقال إلى وضع التصميم الأمثل لترجمتها إلى شفرة برمجية قابلة للتنفيذ ، وذلك من خلال المصممين في فريق العمل ولا بد أن يكونوا ملمين إلماماً كاملاً بطرق التصميم المختلفة والفروق المختلفة بينها ولا بد أن يكون لديهم إلمام بخواص النظم الموجودة مثل أنظمة التشغيل ، ونظم المعلومات المختلفة المتاحة ، وواجهات التطبيق المختلفة الرسومية والبرمجيات المساعدة التي يمكن أن تساعد في عملية الترميز (البرمجة).

- بعد الانتهاء من تخطيط وثائق التصميم يتم الانتقال إلى توزيع المهام على المبرمجين في فريق العمل لتحويل تلك المخططات وترجمتها إلى شفرة برمجية قابلة للتنفيذ باستخدام إحدى لغات البرمجة عالية المستوى (**High Level Programming Languages**) ، ومن ثم تبدأ عملية التكامل للنظام وهي عملية تجميع المهام البرمجية بعد ترميزها عن طريق فريق التطوير مع برمجيات جاهزة تخدم في نفس المشروع.

- بعد الانتهاء من مرحلة برمجة وتكامل النظام تبدأ مرحلة اختبار النظام البرمجي من

حيث الأخطاء البرمجية ، وكذلك من حيث مطابقته للمواصفات المطلوبة وجودة الأداء. ومصطلح جودة البرمجية (Quality Assurance) يتعلق بعملية تطوير المنتج البرمجي والطريقة التي تم تطويره بها بحيث تكون وفقا لمعايير مقاييس الجودة والتي يتم وضعها من خلال مجموعة مراقبة الجودة من فريق التطوير بعد تحديد متطلبات المشروع البرمجي مباشرة وقبل البدء في التنفيذ الفعلي للمشروع، وتتم عملية مراقبة الجودة عند الانتهاء من أي جزئية أو مكون من المشروع حيث يتم اختباره وفقاً للمعايير المتفق عليها سابقاً ، وعند وجود أي خلل أو خطأ فعلى الفريق المختص إصلاحه.

- عملية التوثيق ليست مرحلة منفصلة ولكنها تتم بالتوازي مع المراحل السابقة ، وهي لا تشمل عملية التعليق على الشفرة البرمجية فقط ولكنها تشمل العديد من العمليات مثل كتابة ملفات المساعدة ، وكتيب المستخدم ، وكتيب التدريب، وكتيب المرجع ، وكتيبات التثبيت ولا غرابة أن يكون حجم ما يكتب من سطور لتوثيق النظام يزيد على مرتين من عدد سطور النظام نفسه.

- بعد عمليات التحليل ، والتصميم ، والبرمجة ، والاختبار والتوثيق ، لا بد أن توزع هذه البرمجيات وتثبت على أجهزة العملاء.

- ومع بداية إصدار وتوزيع المنتج البرمجي تبدأ مرحلة الصيانة، ومصطلح الصيانة هنا يصف النشاطات التي تتم بعد إصدار المنتج البرمجي والتي تشمل عملية تصحيح الأخطاء ، وتطوير البرمجية لتعمل مع أوساط جديدة وحاسبات جديدة وأنظمة تشغيل جديدة ، وزيادة فاعلية البرمجية، وفي كثير من الحالات تكون الصيانة هي الأكثر كلفة من ناحية المصاريف والوقت في دورة حياة تطوير البرمجية.

وبالطبع فإن كل العمليات السابقة لا بد من إدارتها من خلال إدارة واعية وعلى دراية كاملة بجميع مراحل تطوير مشاريع البرمجيات ، حيث تمثل عملية إدارة المشروع

النشاط المهم والحرج لتطوير البرمجيات.

3.مبـادئ هندسـة البرمجيـات (Software Engineering Principles

لهندسة البرمجيات أهمية كبيرة ، إذ أنها تساعد في زيادة المردود لشركات النظم البرمجية ، ومستخدمي البرمجيات، من خلال التوزيع السليم للموارد المتوفرة ، والتصميم الجيد لهذه البرمجيات مما يقلل من تكاليف وزمن إنتاجها ، مع زيادة وتحسين مستوى جودتها وموثوقيتها ، ويتم ذلك من خلال تطبيق مجموعة من المبادئ الأساسية التي تتميز بها والتي من أهمها ما يلي :

- **الدقة والصورية (Rigor And Formality)** : يمكن الحصول على منتج موثوق ومضبوط الكلفة من خلال الدقة في عملية الإنتاج وتوجد درجات متنوعة للدقة أعلاها الصورية التي تتطلب أن تكون المنتجات مصممة ومقيمة بطريقة رياضية، وبالتالي تقتضي الدقة الصورية ولكن العكس غير صحيح إذ يمكن أن نكون دقيقين بطرق غير صورية (رياضية). يحتاج أي عمل لتنفيذه إلى خطوات وإذا نفذنا هذه الخطوات اعتماداً على خبرات وتجارب ونظريات عندها ستزداد الدقة، فإذا لم تتوفر هذه الاعتمادات (الخبرات والتجارب والنظريات) نعتمد عند ذلك على التمثيل الرياضي وهذه هي الصورية واللغات البرمجية أدوات صورية للتعبير عن وظائف ثم تحول المترجمات اللغة الصورية إلى لغة الآلة.

- **فصل الاهتمامات (Separation of Concern)** : يعتمد هذا المبدأ على اتباع سياسة " فرق تسد" حيث يتم تقسيم المشكلة إلى مشاكل أصغر غير مرتبطة ببعضها البعض أو يكون ارتباطها صغيراً ، ثم التركيز على حل كل مسألة على حدة. ويتم فصل الاهتمامات حسب الطرق التالية :
  - **حسب الزمن** : جدولة الأعمال وفق محور الزمن.

- **حسب وجهات النظر:** مثلاً عند تحليل متطلبات نظام برمجي ، قد يكون من المفيد التركيز على المعطيات التي تتعامل معها البرمجية من جهة والتركيز على الوظائف والتحكمات التي تقوم بها البرمجية من جهة أخرى.
- **حسب الخواص والمواصفات** : يتضمن معالجة المواصفات المطلوبة من البرمجية على حدة، فعلى سبيل المثال إذا كان المطلوب بناء نظام برمجي فعال وصحيح ، فيتم أولاً تصميمه بطريقة متأنية ومنظمة تضمن صحته ثم يتم التغيير في النظام لتحسين فعاليته.
- **حسب الأقسام : وهو عبارة عن** فصل التعامل مع الأقسام المكونة للنظام البرمجي كل على حدة.

▪ **التجزئة (Modularity)** : وهي تقسيم النظام البرمجي المعقد إلى أقسام أبسط تسمى كتلاً ، ويهدف مبدأ التجزئة في هندسة البرمجيات إلى :

- القدرة على تجزيء النظام البرمجي إلى كتل (التجزيء التنازلي (الشجري) للنظام البرمجي).
- القدرة على تركيب النظام البرمجي من كتل (التركيب التصاعدي للنظام).
- القدرة على فهم الكتل كلاً على حدة لتعديلها (الاستقلالية).

يجب أن يتحقق في الكتل :

- تماسك قوي داخلي للكتل (الكتلة مبنية بشكل منطقي لتحقيق هدف محدد).
- ترابط ضعيف بين الكتل (الاستقلالية).

▪ **التجريد (Abstraction)** : وهو تحديد المظاهر المهمة وتجاهل تفاصيلها أثناء دراسة مسالة معينة (فصل الشيء المهم عن الشيء غير المهم في المسألة) ، وهي نظرة نسبية مرتبطة بالهدف الذي نعمل عليه.

▪ **توقع التغيير (Anticipation of Change)** : وهو إحصاء الأماكن التي يمكن أن يتم

تعديل عليها في البرمجيات ، ومن أكثر هذه الأماكن تغييراً :

- **الخوارزميـــات** : يمكـن أن تكـون هنـاك عـدة خوارزميـات تقـوم بـنفس الوظيفـة (الفرز مثلاً) وبالتالي فاستبدال إحدى هذه الخوارزميات بأخرى أفضل منها، هو أمر محتمل ، ولذلك يفضل كتابة الخوارزمية في وحدة برمجية منفصلة.
- **تمثيل المعطيات** : يتغير أداء البرنامج بتغير بنية المعطيات التي يستخدمها.
- **الآلات التجريديـة**: يمكن اعتبار البرامج التي نكتبها يتم تنفيذها على آلات تجريدية تـتفهم التعليمـات والأوامـر المكتوبـة بلغـة عاليـة المسـتوى، أي أن لكـل لغـة عاليـة المستوى آلـة تجريدية تتفهم تعليمات اللغة وتنفذها بشكل تجريدى وبعيد عن العتاد الصلب.
- **الوسط الاجتماعي** : التغير في الوسط الاجتماعي يؤدي إلى تغيير النظام البرمجي المرافق (مثلاً تغير العملة يؤدي إلى تعديل النظم البرمجية البنكية الموجودة) .
- الطرفيات : يتعلق هذا النوع بالبرمجيات التي تحتاج للتعامل مع طرفيات خاصة كأنظمة التحكم. وبالتالي تغيير الطرفيات يتطلب تغيير النظم البرمجية المرافقة.

- **العموميـة (Generality)** : يجب علينا عند طرح مشكلة أو مسألة أن نحاول إيجاد مسألة أعم تحوي المسألة المطروحة، لأنه قد يكون حل المسألة الأعم أسهل من المسألة الأصلية، ويمكن أن يكون الحل العام قابلاً لإعادة الاستخدام. وكما يمكن أن يكون الحل العام موجوداً في المكتبات البرمجية الجاهزة، ويعتمد هذا المبدأ عندما يكون الهدف من إنتاج النظام البرمجي تسويقياً.
- **إعادة الاستخدام (Reuse)** :

يشير هذا المصطلح إلى تطوير برمجية معتمده على استخدام أجزاء جاهزة تم إعدادها وفقا لمعايير الجودة حيث تم اختبارها من قبل مما يزيد من إنتاجية المشروع وتوفير المزيد من الوقت، ويمكن النظر لهذا المصطلح – أيضاً – على أنه استخدام ما هو متاح فعليا لتحقيق ما هو مطلوب.