بسم الله الرحمن الرحيم

جامعة السّودان المفتوحة

برنامج الحاسوب

هندسة البرمجيات (2)

رمز المقرر ورقمه: حسب**4083**

تأليف: أ.د. السيد محمود عبد الحميد الربيعي

التحكيم العلمي: د. محمد الحسن مصطفى

تصميم تعليمي: عبد الباسط محمد شريف محمد

التدقيق اللغوي: أ. الهدي عبد الله محمد محمد أحمد

التصميم الفني: منى عثمان أحمد النقة

منشورات جامعة السودان المفتوحة، الطبعة الأولى 2009

مقدمة المقرر

الحمد لله الذي خلق الإنسان، علم البيان، وله الحمد أن رفع الـذين أوتـوا العلـم درجات .

والصلاة والسلام على رسول الله المبعوث معلماً للناس أجمعين، ورضي الله عـن صحابته أجمعين، ومن تبعهم بإحسان إلى يوم الدين، وبعد.

عزيزي الدارس ،،

بين يديك كتاب " هندسة البرمجيات 2 " .

تتطور هندسة البرمجيات بسرعة كبيرة للغاية، فمع تطور علوم الحاسوب فقـد ظهـرت أساليب حديثة لتطوير وبناء نظم المعلومات. والهدف من هذه الأسـاليب الحديثـة هـو اختصار الوقت الذي تتطلبه عملية بناء نظم المعلومات، وتيسير عمليات كتابة البـرامج ومراجعتها والتأكد من تأديتها للوظائف المتوقعة منها.

ومن أبرز اتجاهات هذا التطور ظهور أدوات وبرمجيات هندسـة البرمجيـات بمساعدة الحاسوب أو ما يعرف اختصارا بأدوات "كيس" (CASE Tools). والهـدف الأساسي من أدوات كيس هو أتمتة عمليات توليد البرمجيات وتطوير التطبيقات. ويشمل ذلك أتمته جميع المراحل والتي تشمل(تحديد الاحتياجات، التحليـل، التصـميم، كتابـة البرامج، اختبار البرامج وصيانتها).

واستخدام أدوات كيس لا يعني الاستغناء عن أي مرحلة من مراحل تطوير نظم المعلومات، بل هو وسيلة لاستخدام الحاسوب في تطوير تطبيقـات الحاسـوب بهـدف اختصار زمن التطوير وتيسير عمليات توليد وثائق النظام وتنظـيم عمليـات الفحـص والصيانة.

ويتكون الكتاب من ستة وحدات توزيعهم **كالتالي:**

الوحدة الأولى: تحت عنوان" الأدوات المساعدة في البرمجة CASE Tools "، وفيها تم تعريف الأدوات المساعدة في البرمجة، كما تم عرض أهم مميزات وعيـوب اسـتخدام

أدوات كيس. الوحدة تجيب عن التساؤل: هل يمكن للبرمجيات أن تساعد في كتابة برامج أفضل؟ و قدمنا في هذه الوحدة أمثلة للأدوات وهي: اليونكس – الإنترلسب – السمالتوك. وتناولنا فيها أنظمة تطوير البرامج، ودور البيئة المادية في تطوير البرمجيات، وأخيراً عرضنا مثالاً تطبيقياً على استخدام CASE Tools.

الوحدة الثانية: تحت عنوان " التوصيل Delivery، والتنصيب Installation، والتوثيق Documentation للبرمجيات "، وفيها تم شرح سيناريوهات توصيل البرنامج اعتماداً على طبيعة العلاقة بين العميل والمؤسسة التي تقوم بتطوير Software. ثم تناولنا التنصيب Installation للبرنامج، والتوثيق حيث تناولنا التوثيق الداخلي، والتوثيق الخارجي، كما تناولنا في هذه الوحدة الأسس المنطقية للتصميم، وأيضاً عملية التثبيت، تدريب المستخدم وكتيبات التشغيل، التوثيق الإلكتروني، وأيضاً مستويات القراءة. الوحدة تناولت كذلك وجهة نظر المدير المنفذ في التسليم، التنصيب والتوثيق، ثم أخيراً التسليم والتنصيب والتوثيق لمشروع برنامج رئيسي .

الوحدة الثالثة:– تحت عنوان " صيانة البرمجيات " وفيها تم التطرق لمفهوم صيانة البرمجيات حيث قدمنا في الوحدة عدداً من التعريفات الخاصة بصيانة البرمجيات، كما تم استعراض عدد من العوامل التي توجب صيانة البرمجيات. وتناولنا في الوحدة الصيانة التصحيحية للبرمجيات، حيث تم بيان الإجراءات التي يجب أن تتخذ عندما يتم تحديد المشكلة، كذلك تناولنا الصيانة المهيأة للبرامج، والصيانة الوقائية للبرامج ومشكلة عام 2000، وكذلك العديد من السيناريوهات والتي ربما تحدث مشاكل في المستقبل. شرحنا في الوحدة كيف تقرأ المتطلبات، التصميم، وشفرة المصدر. كما تم توضيح وجهة نظر المديرين في صيانة البرمجيات. كذلك الصيانة لمعظم مشروعات البرامج .

الوحدة الرابعة: تحت عنوان " إدارة المشاريع البرمجية " وفيها تم بيان الفرق بين هندسة البرامج والأنماط الهندسية الأخرى. كذلك الصفات المشتركة للإدارة لكل من

المشاريع الهندسية ومشاريع تطوير البرمجيات، وتناولنا بالوحدة إدارة الموارد البشرية الأنشطة والمهام المتعلقة بها، تخطيط المشروع البرمجي، جدولة المشروع، وإدارة الكوارث.

الوحدة الخامسة: تحت عنوان "الهندسة العكسية"، أي محاولة استدعاء مواصفات المستوى الأعلى عن طريق تحليل المنتج. وفيها تم تقديم بعض التعريفات والمفاهيم الهامة المرتبطة بالهندسة العكسية والتي من أهمها الهندسة الأمامية، وبيان الفرق بين الهندسة العكسية وباقي فروع الهندسة، كذلك مميزات الهندسة العكسية. كما تم توضيح أسباب القيام بالهندسة العكسية للبرمجيات، والاستخدامات المختلفة للهندسة العكسية. كما تم اسعراض الهندسة العكسية للبرمجيات والمعدات حيث نتناول أدوات الهندسة العكسية للبرمجيات، والهندسة العكسية للمعدات (الأجهزة)، وأساليب الهندسة العكسية للأجهزة. أيضاً تم تناول المواضيع القانونية المتعلقة بالهندسة العكسية. كذلك أنظمة الحماية المختلفة ضد الهندسة العكسية. كما تم شرح بعض الأمثلة على استخدام الهندسة العكسية.

الوحدة السادسة: تحت عنوان "هندسة البرمجيات المشاكل والاحتمالات"، وفيها تناولنا تصنيف البرمجيات، تكلفة إنتاج البرامج حيث استعرضنا أمثلة عن التكاليف : إنتاجية المبرمج، وتنبؤات تكاليف البرنامج، والمعدات مقابل تكاليف البرامج، وأثر الحاسبات الشخصية حزم البرامج وسائل تطوير التطبيق، وثورة تقنية المعلومات، وكيف تكون التكلفة، وإعادة استخدام البرمجيات.

ولقد أعدت جامعة السودان المفتوحة برنامج البكالوريوس في علوم الحاسوب وتقانة المعلومات بغاية تلبية الحاجة المتنامية لمتخصصين في مثل هذا المجال المتطور، وقد اشتمل هذا البرنامج على منهجين خاصين لهندسة البرمجيات (الأول: هندسة البرمجيات

(1)، والثاني: هندسة البرمجيات (2)، يكمل أحدهما الآخر، كلفنا بإعدادهما باللغة العربية ليكون أمام خريجي هذا البرنامج فرص عمل.

والمنهج(الكتاب) الذي بين يديك الآن تحت عنوان هندسة البرمجيات (2) وهو منهج متقدم إلى حد كبير في هندسة البرمجيات بأنماطها المختلفة وهو مكمل للمنهج(الكتاب) الأول والذي تم نشره تحت عنوان هندسة البرمجيات (1) والذي يعتبر متطلباً أساسياً ولا يمكن للطالب دراسة الكتاب الثاني دون دراسة محتويات الكتاب الأول.

وقد راعينا في صياغة هذا الكتاب أن يكون في صورة وحدات منفصلة متخصصة كل وحده متكاملة تناقش فرعية معينة على نفس الطريقة التي اتبعناها في صياغة الجزء الأول من الكتاب.

والمتطلبات الأساسية لدراسة هذا الكتاب هو أن يكون لدي القارئ معرفة أساسية مسبقة بأساسيات هندسة البرمجيات، وكذلك الدراسة المتأنية للكتاب الأول(هندسة البرمجيات (1)) ويهدف هذا الكتاب في مجمله إلى تزويد الطالب بالمعرفة اللازمة والتهيئة لمستقبل مهني في مجالات هندسة البرمجيات المتقدمة.

كل وحدة من هذه الوحدات تحمل عنواناً جامعاً يعبر عن موضوعاتها، ولكل منها مقدمة وخلاصة، كما زودت كل وحدة بعدد من أسئلة التقويم الذاتي، وأحياناً بعدد من التدريبات، وكذلك تشتمل كل وحدة على مسرد للمصطلحات، كما ذيلنا وحدات هذا المقرر بقائمة للمصادر والمراجع المتنوعة التي تم الاستعانة بها في بناء الوحدات ليلجأ إليها الدارس إذا ما احتاج إلى تفاصيل أكثر. كما ذيّلنا بعض الوحدات بملحق يحتوي على مزيد من التفاصيل المتعلقة بموضوع الوحدة .

الأهداف العامة للمقرر

عزيزي الدارس، بعد فراغك من دراسة هذا المقرر يؤمل أن تكون لك القدرة على:

- **تعداد** أهم مميزات وسلبيات استخدام أدوات كيس.
- **تبيين** كيفية التسليم والتنصيب والتوثيق لمشروع برنامج رئيسي.
- **شرح** أهم أنماط الصيانة وحيثيات كل منها .
- **المشاركة** في تخطيط أي مشروع برمجي ووضع هيكلية خطة تنفيذه.
- **وصف** محاولة استدعاء مواصفات المستوى الأعلى عن طريق تحليل المنتج.
- **تلخيص** التحديات الرئيسية المتعلقة بعملية تطوير البرمجيات والتي تشمل:
  - تلبية حاجات المستخدم .
  - تكلفة إنتاج البرامج .
  - أمثلة على التكاليف .
  - الالتزام بآخر موعد .
  - أداء البرنامج .
  - قابلية التنقل .
  - الصيانة .
  - الموثوقية .
  - تفاعل الحاسوب مع البشر .

محتويات المقرر

وختاماً أرجو أن نكون قد وفقنا في تقديم إضافة جديدة لمكتبة جامعة السودان المفتوحة من خلال هذا الكتاب، وأسأل الله تعالى أن يكون هذا العمل خالصا لوجهه الكريم، وأن يكون في ميزان حسناتنا، وأن ينتفع به الجميع في وطننا، ويغفر لنا ما سهونا عنه وعجزنا عن إدراكه. ونرحب بكل نقد بناء يساهم في تطوير هذا العمل إلى الصورة الأفضل.

والله نسأل الأجر والثواب، وآخر دعوانا أن الحمد لله رب العالمين إنه نعم المولى ونعم النصير.


الوحدة الأولى
الأدوات المساعدة في البرمجة
CASE Tools

محتويات الوحدة

المقدمة

تمهيد

عزيزي الدارس،

مرحباً بك في الوحدة الأولى من مقرر **"هندسة البرمجيات 2"**، والتي تبحث في الأدوات المساعدة في البرمجة CASE Tools. القسم الأول من الوحدة يقدم تعريفاً للأدوات المساعدة في البرمجة القسم الثاني من الوحدة يعدد أهم مميزات وعيوب استخدام أدوات كيس. أما القسم الثالث من الوحدة فيجيب عن التساؤل التالي : هل يمكن للبرمجيات أن تساعد في كتابة برامج أفضل من خلال الفرعيات التالية: لا مجال للخطأ – الحلول العلمية – العودة إلى الكود– مقارنة الإصدارات – اختبار واقتراح التصويبات – البرمجة بالنماذج – معرف للنموذج– العامل البشري وكتابة الكود – أدوات لغرض البرمجة – أدوات غير متعلقة بالبرمجة. القسم الرابع من الوحدة يقدم أمثلة للأدوات وهي: اليونكس – الإنترلسب – السمالتوك. والقسم الخامس من الوحدة يتناول أنظمة تطوير البرامج القسم السادس يتناول دور البيئة المادية في تطوير البرمجيات . أما القسم السابع من الوحدة فنعرض فيه مثالاً تطبيقياً على استخدام CASE Tools.

أهداف الوحدة

عزيزي الدارس، بعد فراغك من دراسة هذه الوحدة ينبغي أن تكون قادراً على أن:

- **تعدد** ميزات وعيوب استخدام أدوات كيس.
- **تجيب** على السؤال التالي: هل يمكن للبرمجيات أن تساعد في كتابة برامج أفضل من خلال دراسته للنقاط التالية؟ لا مجال للخطأ، الحلول العملية، العودة إلى الكود، مقارنة الإصدارات، اختبار واقتراح التصويبات، البرمجة بالنماذج ، معرف للنموذج، العامل البشري وكتابة الكود، أدوات لغرض البرمجة:، أدوات غير متعلقة بالبرمجة:
- **تشرح** السمات الأساسية لكل من:
  - اليونكس .
  - الإنترلسب .
  - السمالتوك .
- **توضح** أنظمة تطوير البرامج .
- **تلخص** أدوات تطوير البرمجيات .
- **تشرح** دور البيئة المادية في تطوير البرمجيات.
- **تعطي** مثالاً تطبيقياً على استخدام CASE Tools .

1.تعريف الأدوات المساعدة في البرمجة CASE Tools

CASE tools= Computer aided software engineering

أو أدوات هندسة البرمجيات باستخدام الحاسوب ، وهي مجموعة برمجيات تأخذ منك تصميمك النظري للبرمجيات وتحوله إلى شكل flow charts and design diagrams ، ومن ثم تساهم بصورة شبه آلية في إنتاج الكود الموافق للتصميم، وبالتالي تساهم في تقليل زمن دورة تطوير البرمجيات software development cycle وتزيد من قابلية إعادة استخدام البرمجيات software reusability. من أشهر هذه البرمجيات Rational Rose و Microsoft Visio، كما أن العديد من بيئات التطوير أصبحت تتنافس في تقديم ميزات هندسة البرمجيات باستخدام الحاسوب ،منها مثلا studio.Net Visual و Borland JBuilder X.

ويمكن تعريفها كذلك على أنها برمجيات البرمجة Programming Software وهي البرمجيات المساعدة في عملية إنتاج البرمجيات Software Development نفسها؛ و من أمثلتها برمجيات إدارة قواعد البيانات Database Management Systems DBMS وبرمجيات بيئة البرمجة المتكاملة Integrated Development Environment IDE، و التي تحتوي بدورها على مولد المترجمات Compiler و محرر اللغة البرمجية Editor و الأدوات البرمجية Tools المختلفة التي يحتاجها المبرمج أثناء عمله.

ومع تطور علوم الحاسوب فقد ظهرت أساليب حديثة لتطوير وبناء نظم المعلومات. والهدف من هذه الأساليب الحديثة هو اختصار الوقت الذي تتطلبه عملية بناء نظم المعلومات، وتيسير عمليات كتابة البرامج ومراجعتها والتأكد من تأديتها للوظائف المتوقعة منها.

ومن أبرز اتجاهات هذا التطور هو ظهور أدوات وبرمجيات هندسة البرمجيات بمساعدة الحاسوب أو ما يعرف اختصارا بأدوات "كيس" (CASE Tools). والهدف

الأساس من أدوات كيس هو أتمتة عمليات توليد البرمجيات وتطوير التطبيقات. ويشمل ذلك أتمته جميع المراحل والتي تشمل(تحديد الاحتياجات، التحليل، التصميم، كتابة البرامج ، اختبار البرامج وصيانتها) .

إذن فإن استخدام أدوات كيس لا يعني الاستغناء عن أي مرحلة من مراحل تطوير نظم المعلومات، بل هو وسيلة لاستخدام الحاسوب في تطوير تطبيقات الحاسوب بهدف اختصار زمن التطوير وتيسير عمليات توليد وثائق النظام وتنظيم عمليات الفحص والصيانة.

أسئلة تقويم ذاتي

1) ما التعريفات المختلفة للأدوات المساعدة في البرمجة؟
2) اضرب أمثلة للأدوات المساعدة في البرمجة؟

2. مميزات وسلبيات استخدام أدوات كيس

1.2 مزايا استخدام أدوات كيس

ويمكن تلخيص أهم مزايا استخدام أدوات كيس في التالي :

أ) سرعة إنتاج البرامج

تتضمن أدوات كيس في العادة مكتبةً كبيرة من البرامج الجاهزة التي تؤدي وظائف مختلفة. ومن واقع نتاج تحليل النظام وتحديد متطلبات النظام والعمليات المطلوبة فيه يمكن لأدوات كيس أن تنتج نسخة أولية من البرامج المطلوبة بسرعة. هذه البرامج يمكن أن تكون الأساس الذي تبنى حوله النظم التطبيقية المطلوبة.

ب) خفض تكاليف التطوير

لاختصار زمن التطوير وسرعة إنتاج البرامج المطلوبة، يمكن أن نتوقع خفض التكلفة بما يشمله من تكلفة كتابة البرمجيات وتعديلها وفحصها وكذلك تكلفة إنتاج الوثائق وغيرها.

ت) الالتزام بمعايير ثابتة للتطوير

إن استخدام أدوات كيس يضمن الالتزام بمعايير واضحة وثابتة لجميع مراحل التطوير. وذلك لأن هذه الأدوات تتطلب توفر معلومات محددة منظمة ومرتبة وفق هياكل خاصة، ولا يمكن لهذه الأدوات أن تنتج البرامج والتطبيقات المطلوبة إلا بعد توفر المدخلات المطلوبة.

ث) تيسير تطوير الأنظمة الكبيرة

ونظرا لأن أدوات كيس يمكنها توليد البرامج بطريقة هيكلية منظمة، فإنه يسهل بواسطتها التعامل مع الأنظمة الكبيرة المعقدة. وهنا يمكن بواسطة هذه الأدوات تقسيم النظام الكبير إلى وحدات مترابطة يمكن توزيعها على أكثر من مبرمج، وعلى أن يجري تكاملها وربطها في مراحل لاحقة.

ج) تكامل التوثيق

بينت التجارب أن توثيق النظام يكون في العادة أضعف جوانب تطوير الأنظمة. وعندما يتم تعديل البرامج وتحديثها، فإنه يغفل غالبا تحديث الوثائق حتى تعكس التغيرات التي حدثت على البرامج . ولكن باستخدام أدوات كيس فإن الوثائق الفنية يمكن توليدها آليا وهي جزء أساسي من المخرجات. وعند تعديل البرامج يمكن دائما توليد وثائق فنية حديثة.

2.2 سلبيات استخدام أدوات كيس

وفي مقابل المزايا التي توفرها أدوات كيس ، فإن هناك عدداً من السلبيات التي يجب عدم إغفالها عند اتخاذ القرار المتعلق باستخدام هذه الأدوات من عدمه. ويمكن تلخيص **أبرز سلبيات أدوات كيس في التالي:**

- صعوبة تعلم استخدام هذه الأدوات واحتياجه لفترات تدريب طويلة.
- قلة أعداد المختصين المتمرسين في تطوير التطبيقات باستخدام أدوات كيس.
- صعوبة تطوير بعض التطبيقات الخاصة غير التقليدية بواسطة أدوات كيس.
- ارتفاع التكلفة المبدئية لاستخدام أدوات كيس نسبيا بسبب حداثة هذه البرمجيات وقلة عدد الشركات التي تنتجها. ولكن عند استخدام هذه الأدوات في تطوير العديد من الأنظمة فإن التكلفة إجمالا ستنخفض.

أسئلة تقويم ذاتي

1) ما أهم ميزات استخدام أدوات كيس؟
2) ما أهم السلبيات التي يجب عدم إغفالها عند اتخاذ القرار المتعلق باستخدام أدوات كيس من عدمه؟

3. هل يمكن للبرمجيات أن تساعد في كتابة برامج أفضل؟

الحضارة البشرية الحالية تقوم على البرمجيات، فالبرمجيات متواجدة في كل شيء وليس فقط على الحاسبات، فهي بالأجهزة المنزلية والسيارات والطائرات والمصاعد والتليفونات ولعب الأطفال، وفي مالا يحصى من أجزاء الماكينات والمعدات الصناعية بأنواعها. في مجتمع كهذا يعتمد على البرمجيات إلى هذا القدر، فإن عواقب

الأخطاء في البرمجيات التي بدأت تكثر في شكل فيروسات وجراثيم برمجية في الآونة الأخيرة قد تؤدي إلى كوارث كبيرة، مثل تحطم صاروخ، انهيار شبكة التليفون، توقف نظام المراقبة الجوية عن العمل، وغيرها.

في عام 2002 أجرى المعهد القومي الأمريكي للمعايير والتكنولوجيا دراسة توقعت أن أخطاء البرمجيات قد تكلف الاقتصاد الأمريكي 60 مليار دولار سنويا، أو ما يقرب من 6 في الألف من إجمالي الناتج القومي. ومما زاد الأمر صعوبة، أن نظم المعلومات القائمة على البرمجيات أصبحت متداخلة ومتصلة معا بطريقة غير مسبوقة، كما نجد أن تطبيقاتها تتصرف بتعقيدات كبيرة.

ومتابعة أخطاء البرمجيات بالطريقة التقليدية هي كتابة سطور من لغة البرمجة ثم اختبارها على جهاز الحاسب ومعالجة أي أخطاء تنتج أثناء تجربة البرنامج أصبحت أقل فاعلية بكثير في ظل التعقيدات الحالية في تداخلات البرمجيات. «لقد اصطدمنا بحائط» هكذا علق بلاك ستون رئيس العلماء بشركة بورلاند المنتجة للغات البرمجة.

وللرد على التساؤل السابق دعنا نناقش الفرعيات التالية:

1.3 لا مجال للخطأ

المبرمجون يقضون أوقاتهم في تصحيح معالجة أخطاء البرامج القائمة أكثر من الوقت الذي يقضونه في كتابة برامج جديدة، وطبقا لإحصائيات المعهد القومي الأمريكي للمعايير والتكنولوجيا فإن 80% من تكلفة تطوير البرمجيات لمشروع نظم معلومات تكون في تحديد ومعالجة عيوب البرامج.

ومن هنا تظهر أهمية البرامج التي يمكنها تحليل مفردات البرنامج المكتوب واختباره آليا ووضع أسس تأكيد الجودة. والهدف هو الوصول إلى طريقة لضمان جودة البرمجيات أثناء صناعتها كما يحدث في صناعة السيارات، فالمتابعة الآلية للجودة هي معيار الثقة في البرمجيات. ويمكن اعتباره هو استخدام البرمجيات لتحسين صناعة البرمجيات.

وبرنامج اكتشاف الأخطاء هو أقرب ما يكون قريب جدا من المبرمج، لأنه كلما كان الاكتشاف مبكرا للخطأ كانت تكلفة المعالجة أقل. وهناك مثل دارج في أوساط البرمجيات تشرحه دجينانا كامبرا المدير التكنولوجي لمؤسسة كلوكورك الكندية أن الخطأ في البرنامج يتكلف دولارا واحدا لمعالجته على جهاز المبرمج، ومائة دولار إذا ظهر في إطار برمجيات أخرى وآلاف الدولارات إذا تم اكتشافه بعد انتشار استخدامه، وهناك حالات تكون تكلفة المعالجة أكبر من ذلك بكثير مثل الخطأ في برنامج شبكة التليفونات أو نظام المراقبة الجوية الذي قد يتكلف الملايين إذا كان يستخدم فعليا.

إن استخدام برنامج ما لفحص برنامج آخر واكتشاف أخطائه هو أمر ليس باليسير، وقام خبراء البرمجيات لعقود طويلة مضت في محاولة وضع أسس واختبارات لتحليل البرمجيات والتأكد من سلامتها وقيامها بالدور المنوط بها، ولكن هناك <u>عقبتان تواجهان هذا الأمر:</u>

- **<u>الأولى</u>:** أن هذه الاختبارات مرهقة وغير عملية، فمثلا ثلاثة سطور من البرمجة تحتاج إلى صفحة كاملة من الجبر والمعادلات الرياضية للتأكد من صحتها، والآن يحتوي البرنامج الواحد على ملايين السطور.
- **<u>والثانية</u>:** أن هذه الاختبارات غير قابلة للبرمجة، حيث إنها يدوية وتحتاج إلى عمالة مكثفة.

أما جريدي بوش وهو من رواد تقنيات تطوير البرمجيات المدير العلمي لمؤسسة راشونال الشهيرة في مجال تطوير تقنيات تطبيقات البرمجيات، وهي الآن تتبع شركة أي بي إم فإنه يقول "إن الطرق التقليدية لتحليل ومعالجة البرمجيات لم تحقق وعودها الكبيرة المفترض منها. ويتابع: لقد رأينا بعض التحسينات في التقنيات والتكنولوجيا، ولكني لم ألاحظ أي طفرة منذ الستينيات.
وهذا لا يعني أنها غير مفيدة ولكنها مجدية فقط في المهمات الحساسة مثل مركبات الفضاء وشبكات الاتصالات والنظم العسكرية".

2.3 الحلول العملية

وبالرغم من أن الأبحاث لم توفق في تحقيق الوعود التي تعهدت بها في الستينات، إلا أنها نجحت في الوصول إلى تقنيات مفيدة، فهناك العديد من المؤسسات التكنولوجية تنتج برمجيات تساعدالمبرمجين على ضمان نجاح برامجهم. وحجر الزاوية في الموضوع هو ربط نظم تطوير البرمجيات بتقنيات تحليل البرامج وتصويب أخطائها. ويؤكد بعض الخبراء أنه لا يوجد حل مثالي وحيد، ولكن لابد من تقبل الحلول العملية، لأن لكل بيئة برمجية الوسيلة الملائمة لتحليل برامجها وتصويب أخطائها.

الأمل هو انتشار استخدام تقنيات الاختبار والتحليل حتى تصبح جزءا لا يتجزأ من أدوات البرمجة التي لا غنى عنها للمبرمج، وتكون النتيجة برمجيات بأخطاء أقل وتحسن في نوعيتها.

ومن التقنيات المعروفة في مجال تحديد الأخطاء في البرمجيات كتابة توصيف رياضي للمطلوب من البرنامج تنفيذه، ومقارنة الوصف الرياضي بالبرنامج المكتوب. ولكن الوصف الرياضي من hالصعب التعامل معه تماما مثل البرنامج المكتوب، وتبقى الاستثناءات في البرامج الصغيرة ذات الوظائف المحددة، فهي أسهل للمتابعة والتحليل من البرمجيات المعقد والمتشعبة.

شركة مايكروسوفت تستخدم نظام تحليل البرامج والبحث عن الأخطاء المعروف باسم static driver verifier للتعامل مع برمجيات تعريف الكروت.

هذه البرامج تتكون مما يزيد عن مائة ألف سطر من الشفرات والرموز، ولها وظيفة محددة هي الربط بين جهاز الكمبيوتر والأجهزة الطرفية، مثل كارت الشاشة أو وحدة التخزين. أما المحاكاة الرياضية لبرمجيات نظم التشغيل أو متصفح الإنترنت فهي من الناحية العملية أمر شبه مستحيل. إحدى الوسائل المتعارف عليها للوصول إلى المطلوب هي ترك النمذجة الرياضية لهذه البرمجيات ومحاولة توصيف الأخطاء، وهو أمر أسهل نسبيا.

ومثال لذلك الخطأ الشائع المعروف باسم buffer overflow والذي يحدث عندما يحاول الكمبيوتر تخزين بيانات في الذاكرة الاحتياطية أكثر من سعتها، مما يؤدي إلى ضياع البيانات المخزنة، ويعتبر خطأ في تنفيذ البرنامج. وأحد طرق معالجته تكون بتمشيط كود البرنامج للبحث عن أماكن الكتابة في الذاكرة الاحتياطية والتأكد من سعتها، وإذا لم ينجح في تحديد مكان الخطأ فإنه يسجل أن هناك خطأ.

3.3 العودة إلى الكود

البحث عن أخطاء البرمجيات داخل سطور الكود يسمى «التحليل الإستاتيكي» لأنه يتم والبرنامج لا يعمل، وهذا مفيد في حالة البحث عن الثغرات الأمنية، والكود المضطرب والسطور التي لم تعد تستخدم في البرنامج، وكل هذا لا يحتاج إلى معرفة بوظائف البرنامج أو الغرض منه، ويحتاج إلى أداة بحث عامة، ولقد طورت شركة مايكروسوفت أداة للبحث الإستاتيكي عن الأخطاء في البرامج فور كتابتها مما يؤدي إلى تقلصها، وتسمى PRE fast وهو جزء من الإستراتيجية التي تتبناها مايكروسوفت تحت شعار الثقة في البرمجيات لزيادة الاعتماد على منتجاتها، وتم استخدامه أثناء تطوير بيئة الويندوز اكس بي، وبعد ذلك تم تعميم استخدامها داخل الشركة وتخطط لاستخدامها على نطاق واسع.

4.3 مقارنة الإصدارات

أخطاء البرمجيات يمكن أيضا اكتشافها بمقارنة الإصدارات الحديثة من البرمجيات بمثيلتها القديمة التي كانت تعمل، مثال ذلك برنامج التصحيح الإملائي في أوفيس مايكروسوفت، عندما يتم تطوير إصدار جديد له، يتم اختباره للتأكد من أنه مازال يعمل بالطريقة التقليدية للبرنامج.

هذه الطريقة في اختبار البرمجيات تتم داخل مايكروسوفت وعلى نطاق واسع في أماكن أخرى في العالم. ويقول سريفستافا الخبير الدولي أن الصعوبة تكمن في تحديد الكود المطلوب اختباره، لأن البرنامج يحتوي على عشرات الوظائف والأكواد، ومن

أجل هذا فقد اقترح نظام تحليل جديد يسمى سكوت Scout يقارن البرنامج بشكله الثنائي Binary مع الشكل الثنائي للإصدار القديم، وهذا يظهر نقاط الاختلاف ويحدد الوظائف المطلوب اختبارها. وتقنية سكوت لها فائدة أخرى أنها تحدد عدد الوظائف المطلوب اختبارها في الإصدار الجديد، فإذا كان التغيير الجديد يحتاج إلى اختبارات كثيرة، فهذا دليل على خطورة التعديل، ومن الأفضل تجنبه. وكذلك يمكنه ترتيب أولويات الاختبارات، فيتم اختبار الأهم فالمهم، وهذا يؤثر في معالجة الاختراقات الأمنية لنظم التشغيل حيث سرعة المعالجة عامل فعال.

5.3 اختبار واقتراح التصويبات

شركة أجيتار بولاية كاليفورنيا تؤمن بخطوة أعمق، اقترحت نظام اختبار يسمى Agitator يختبر البرنامج ويحدد الأخطاء ويقترح التصويبات اللازمة، وهذا يوفر وقت المبرمج من الفحص اليدوي وصيانة البرنامج، وهذا شعار ألبرتو سافويا الخبير المحنك بجوجل وسن سيستيمز ومؤسس شركة اجيتار، ووعد بكشف المزيد من التفاصيل قريبا.

6.3 البرمجة بالنماذج

أحد الطرق لاكتشاف أخطاء البرمجيات هي تحليل الكود التفصيلي وتكوين نموذج لفكرة البرنامج ووظائفه، وتتم مقارنة هذا النموذج بالنموذج الجديد للإصدار الجديد للتأكد من تطابقهما، وخلو الإصدار الجديد من أخطاء جديدة وتحديد الأخطاء القديمة الموجودة في الإصدار السابق للبرنامج.

وكما يمكن استنتاج وتكوين النموذج الرياضي من كود البرنامج يمكن العكس أيضا، وهو كتابة كود برامج من النموذج الرياضي وذلك إلى حد ما. فنظم النمذجة مثل Rational Rose تسمح بتصميم البرمجيات طبقا لنموذج فكري ورياضي بدون الحاجة لكتابة أي كود بيد المبرمجين.

وذلك ينتج هيكلاً كودياً يمكن للمبرمج أن يضيف عليه ما يريد من الكود تحقيق متطلباته، وأشهر هذه النماذج هو language unified modeling (UML) الذي ابتكره د/بوش واعتبرها مسودة للبرمجيات. وهو يتفوق على كل لغات البرمجة ونظم التشغيل والتفاصيل الفنية المعقدة ويسمح باختبار البرنامج طبقا لمتطلبات التصميم، وأصبحت الآن جزءا لا يتجزأ من نظم البرمجيات.

هذا لا يعني بأي حال من الأحوال أن النموذج الرياضي للبرنامج هو شق نظري بحت، ويوضح د/بوش أن كتابة النموذج الرياضي ثم كتابة الكود أمر لن ينجح أبداً ولكن النموذج الرياضي والكود هما وجهتا نظر لنفس الشيء، فالنماذج ما هي إلا وسائل لمساعدة المبرمجين على فهم البرامج القائمة وكتابة البرامج الجديدة.

وخلافا للصراع القائم بين نظريات البرمجة التي تختلف بين تكوين البرنامج من أعلى لأسفل أو تكوين النموذج ثم كتابة الكود، فإن د/بوش يؤيد البداية المحدودة للنموذج والكود واختبارهما، ثم تطور الكود تدريجيا مع تحديث متواز للنموذج.

واستخدام النماذج الرياضية للبرمجيات يعزز من إمكانية التحليل والتصويب الآلي للبرمجيات، وخاصة تجانس الكود والنموذج بحيث يكون الكود المضاف إلى البرنامج.

7.3 معرف للنموذج

ولسنوات عديدة حاول الباحثون بناء نظم وبرمجيات بالطرق التقليدية من نماذج تفصيلية، ولكن هذه المحاولات لم تخرج من إطار الطموحات الفكرية. لأنه في الحقيقة معظم المبرمجين يعملون مع الأكواد والبرامج القائمة فعلا، وهي عادة ما تفتقر لوجود نماذج موضحة لها، وهذا ما يجب على أدوات النمذجة القيام به "عمل نماذج للبرمجيات القائمة". والنماذج لها فوائد كثيرة، منها التأكد من أن المبرمجين لا يقومون بتعديل البرمجيات بما يغير من طبيعتها والهدف من إنشائها، هذا بالإضافة إلى مكافحة الأخطاء واكتشافها، وذلك بمقارنة النموذج الناتج من الكود مع النموذج الأصلي للبرنامج، وتطلق السيدة كامبارا على هذه الطريقة «حماية النموذج من التآكل»، وتؤكد على حتمية وجود

هذه الأدوات داخل مطورات البرمجيات نفسها، فمطالبة المبرمجين بتغيير أسلوب عملهم بين عشية وضحاها أمر غير عملي، وتطالب بأن تكون أدوات المساعدة في البرمجة سهلة الاستخدام ومريحة للمبرمج حتى لا ينفر منها.

8.3 العامل البشري وكتابة الكود

كل ما سبق يصب في موضوع مهم ويعتبر أساسيا في تحسين نوعية البرمجيات، وهو أثره على المبرمجين من حيث تحويل انتباههم عن الهدف الرئيس وهو البرمجة. فأدوات الاختبار الآلية وتصويب الأخطاء تعطي إشارات لمطوري البرمجيات لتوضيح المشاكل التي قد تتواجد في البرامج، فرئيس فريق العمل يمكنه حينئذ تحديد أجزاء البرنامج التي بها عيوب ومعالجتها. وهذا يجعل البرنامج أسهل في التتبع والأخطاءأي أن المشاكل تظهر بسرعة، وتؤدي أدوات الاختبار الآلية أيضا إلى مقارنة أداء المبرمجين بطريقة غير مقصودة، حيث يمكن للمديرين تحديد نسبة الأخطاء عند كل مبرمج، فذلك مخيف بالنسبة لبعض المبرمجين، ولكنه أمر حتمي أن تحدث هذه المقارنات.

وهناك خلاف في أوساط البرمجيات في مدى ترحيب المبرمجين بمثل هذه الأدوات التي تكشف أخطاء البرمجة فور كتابتها، فمنهم من يتوقع قبولها واستخدامها وآخرون يؤكدون أن المبرمجين سيرفضونها، فهناك من هو متفائل بشرط أن تستخدم هذه الأدوات بطريقة صحيحة لتحسين مستوى المبرمجين وليس لعقابهم، (عندما تكتشف خطأ ما وتدرب المبرمج لعدم تكراره مرة أخرى فهذا شيء جيد). وفي أحد الأبحاث التي راقبت خمسين من المبرمجين لمدة أربع وعشرين ساعة، وجد أن 30% فقط من الوقت استخدم في كتابة الكود والباقي في التحدث مع الزملاء. فتطوير البرمجيات هي عمل جماعي. والأدوات التي تتعمق في الكود وتسبح خلاله لاكتشاف الأخطاء وتحدد اتجاهات العمل فذلك يساعد في تحسين العمل الجماعي.

وهناك دائما المعارضون، فهم يحذرون من مغبة الثقة الزائدة في الأرقام والاقتراحات الناتجة من أدوات متابعة البرمجيات، ويعترفون أنها تعطي انطباعات داخلية عن الكود. ولكن هل تستخدم هذه الأرقام في قرارات سليمة أم غير ذلك؟ ويقترحون أدوات إضافية تحدد المبرمج الذي عدل وظيفة ما داخل البرامج واقتراح التطورات اللازمة لتطوير البرمجيات. إن التخاطب والاتصال بالآخرين هي رغبة بشرية تلقائية وكذلك عمل الأخطاء في البرمجيات، فلاستخدام البرمجيات في صناعة البرمجيات لابد من مراعاة البعد البشري والنفسي للمبرمجين.

والأدوات البرمجية يمكن تعريفها كأي جزء من البرمجيات والتي تهدف أو تضبط عملية تطوير البرمجيات، **إن الهدف العام لمثل هذه الأدوات هو:**

- الحد من العناء الشخصي المطلوب لتطوير البرمجيات(زيادة الإنتاج والحد من تكاليف البرمجيات).
- تحسين جودة البرمجيات(على سبيل المثال، الاعتمادية وإمكانية إعادة الاستخدام).

وتوجد مجموعة ضخمة من الأدوات البرمجية مثل المترجم والمرتبطة بلغة برمجة معينة، بعض الأدوات مثل معالج الكلمات تعتبر أدوات لأغراض عامة، بعض الأدوات الأخرى مرتبطة بمنهجية معينة مثلا أدوات توفر الدعم لرسم مخططات الطبقة في التصميم الموجه بالكائنات، في الواقع لا توجد منهجيات مكتملة حاليا بدون قائمة من الأدوات التي تدعمها.

في الحاضر يوجد انتشار هائل للأدوات، وللتوضيح، فإن مهندس البرمجيات الذي يقوم بتغيير وظيفته يمكن أن يستمر في استخدام نفس لغة البرمجة ولكن حتما يتعين عليه تعلم نظام تطوير برامج جديدة، إن التنوع في الأدوات حاليا وما يتوقع أيضا في المستقبل يؤدي إلى ضغط في عملية توحيد الأدوات خلال تصنيع البرمجيات، وهذا ما لم يتم إنجازه حتى الآن.

والأدوات في حالة نمو في مداها وتنوعها لذلك فإن هذا التجمع من الأدوات تـــم توفيره من أجل هندسة البرمجيات بمساعدة الحاسب، ومعظم هذه الأدوات تعمل تحـــت أنظمة التشغيل السائدة (يونكس –ويندوز– وماكنتوش) وتقدم واجهة مســـتخدم رســـومية تدعم النوافذ والقوائم المنسدلة واختيار الماوس من أجل عمليات أكثر مرونة.

وفي هذه الوحدة سنعرض أنواع الأدوات المتوفرة حاليا وسنبحث أيضا في الاقتراحـــات التي طرحت عن البيئة المادية والأفضل لمهام تطوير البرمجيات.

9.3 أدوات لغرض البرمجة

تخيل أن عليك الحصول على أية أداة برمجية تريدها، ماذا يمكن أن تكون هـــذه الأداة؟ سنعمل على تدوين قائمة بالمتطلبات وفي الجزء اللاحق سنعرض بعض الأمثلة لأدوات فعلية.

سنبدأ قائمتنا بهذه الأدوات المرتبطة بطور البرمجة في تطوير البرمجيات:

- المترجم.
- محرر نصوص.
- الرابط.
- الناقل.

بعض الأدوات متعلقة بلغات برمجة خاصة مستخدمة فعليا، prettifier يعمل على تهيئة نص البرنامج إلى الصورة القياسية، بتخطيط معين بحيث يمكن قراءتـــه، علـــى ســـبيل المثال الصيغة القياسية if…then…else cross reference يعرض أسماء البيانـــات والدوال والإجراءات المستخدمة في البرنامج متضمنة معلومات تبين أين يـــتم الإعـــلان عن أي منها وأين يتم استخدامه.

Debugger: يسمح للمبرمج بإنشاء نقاط فاصلة في البرنامج ويعرض قـــيم المتغيـــرات كما ينفذها البرنامج.

نحتاج أيضا إلى إضافة library تحتوي على الإجراءات الضرورية، لذلك نستطيع القيام بأفضل عملية إعادة استخدام للبرمجيات التي تمت كتابتها مسبقا، هذه المكتبة تنقسم إلى المقاطع الأساسية (إجراءات رياضية، التفاعل مع الملفات، إجراءات رسومية، برمجيات الاتصال... الخ)، الأدوات يمكنها عرض بناء البرنامج بصورة رسومية كالمخططات مثلا، هذا التمثيل البديل يمكنه مساعدة المبرمج لتصور البرنامج بطريقة أخرى تختلف عن الطريقة التي يتم إنشاء البرنامج بها، هذا يمكنه المساعدة في عملية اكتشاف الأخطاء والاختبار أو في تحسين بناء البرنامج.

نحتاج أيضا إلى file system وهو database لتخزين الكود المصدري للبرنامج، ويزود نظام الملفات بإمكانيات دائمة لإنشاء ملف جديد، عرض أو طباعة محتوى الملف، حذف الملف، عرض جميع أسماء الملفات وملفات المجموعة سوية في مجلد جذري. file compare tool يمكن استخدامها لفحص أي اختلاف بين النتائج الحقيقية والمتوقعة.

10.3 أدوات غير متعلقة بالبرمجة

إن الكثير من المعلومات المرتبطة في المشروع مثل المواصفات ودليل المستخدم وجدول الاختبار وتقارير الأخطاء ستظهر لك باللغة الطبيعية، لذا سيكون معالج النصوص في هذه الحالة من المستلزمات الأساسية.

إن العديد من منهجيات التصميم تستخدم المخططات لوصف التصميم، حيث توجد أدوات يمكنها المساعدة في تحرير وصيانة هذه المخططات، ومن أبسط تلك الأدوات المحرر والذي يسمح بالإنشاء والتحرير والإضافة، وهناك أدوات أكثر تطورا تقوم بإجراء فحص بسيط على المخططات، أما الأدوات المتخصصة فمنها ما يقوم تلقائيا بإنشاء كود من مخطط موجود أصلا. وفي هذه الوحدة نريد مناقشة الأدوات وشرح منهجية كل منها على حدة.

عندما يتم بناء كتلة كبيرة من البرمجيات من العديد من الوحدات الأساسية سيكون من الصعب الاحتفاظ بسجل يصف بالضبط أي إصدار من تلك الوحدات يمكنه المساهمة بإصدار معين من الحزمة الكاملة، إن حزم إدارة التشكيلات تقوم بحفظ سجلات تحدد أي الإصدارات من أي إجراء يدخل في تكوين نظام معين. في المشاريع الكبيرة توجد أدوات يمكنها المساعدة في التتبع وحفظ الأثر وحفظ المجدولات وآخر مواعيد التنفيذ، برامج الحسابات لها دور في هذا المجال.

4. أمثلة للأدوات

في هذا الجزء سنعرض ثلاثة أمثلة مهمة لقوائم الأدوات، اليونكس والذي أصبح معروفا الآن، لان المبرمجين متحمسين للغة أوامره وتفنيذ أنظمته وبساطة مفاهيمه. السمالتوك وهي بيئة برمجة تحيط ب 80 لغة برمجة، وبالمثل الانترلسب يطوق نظام اللسب والذي يعكس العديد من العناصر المبتكرة في لغة اللسب.

1.4 اليونكس

هو نظام تشغيل للأغراض العامة متوفر بشكل واسع على الحواسب الشخصية والخادمات والحواسب الكبيرة (Mainframe) .

وعن طريق اليونكس يمكنها التزويد بـــ:

- تنفيذ الأنظمة باستخدام البناء الشجري للمجلدات الجذرية.
- لغة أوامر نصية تعتمد على (أفعال متبوعة بعوامل).
- الوسائل الممكنة لكتابة برامج بلغة الأوامر.
- تعدد الروابط.
- قائمة من البرامج المساعدة تسمى(Filters) مرشحات، على سبيل المثال أداة نسخ الملف.
- إمكانيات ووسائل تدعى (Pipes) أنابيب لربط المرشحات مع بعضها.

وهذه الإمكانيات توجد بالعديد من أنظمة التشغيل وكان اليونكس أول نظام مزود بها.

واليونكس يتكون من نواة صغيرة محمية بواسطة قائمة غنية بالبرامج المساعدة والمرشحات، مثال على أحد المرشحات برنامج يعرض قائمة من الملفات داخل ملف جذري معين، وهو مبني حول مفاهيم بسيطة ومحدودة ، واحدة من الأفكار الأساسية لليونكس هي أن البرمجيات المحتواة أصلا داخل اليونكس يجب أن تكون مبنية من عناصر صغيرة لأغراض عامة والتي يتم تطويرها بشكل منفصل، هذه العناصر يمكن استخدامها على حده ولكنها أيضا تستخدم مجتمعه كضروريات لتفي بالمتطلبات الجديدة، في يونكس عناصر البرمجيات منفردة تسمى مرشحات والمرشح هو برنامج يقوم بإدخال تيار من المعلومات المتسلسلة ويعالجها ويخرجها كتيار متسلسل آخر، **واليك أمثلة أخرى على المرشحات المزودة بواسطة اليونكس:**

- عد أرقام الأسطر أو الحروف في ملف.
- نسخ ملف.
- طباعة ملف بتنسيق محدد.
- إرسال ملف إلى الطابعة.
- طباعة جميع الأسطر في ملف والتي تحتوي على طراز نصي معين.
- تحليل ملف.

والمرشحات متحدة تقوم بأخذ المخرجات من جهة وتزويدها بأخرى كمدخلات، وسيل البيانات الذي يتدفق من مرشح لآخر يعرف كأنبوب. هذا الارتباط من المرشحات والأنابيب يتم تنفيذه بواسطة تايونكس تحت تحكم أوامر اللغة، مثال لاتحاد المرشحات باعتبار الأوامر: ls1

عندما تقوم بطباعة أمر من أوامر اليونكس كهذا، فأنت في الحقيقة تستدعي فلترا بنفس الاسم، اسم الفلتر ls1 يعرض على شاشة أسماء الملفات في المجلد الجذري الحالي مرة لكل سطر، أداة أخرى تسمى wc تقوم بإدخال الملف وتعطي رقم السطر والكلمة والمحرف لذلك الملف، ومخرج هذه الأداة يمكن التحكم به بواسطة عوامل إضافية،

والعامل 1- يحدد السطر الذي يجب عده لذلك الأمر: **wc-1 file** يخبر كم عدد الأسطر التي يحتويها الملف file.

أغلب أوامر اليونكس تقدم خيارات للعوامل والتي تقوم بتعديل المخرجات، بوضع هاتين الأداتين مجتمعتين يكون أمر اليونكس:

ls | wc-1

ينقل المخرج من المرشح ls كمدخل للفلتر wc، وعلى هذا فإن المخرج الأخير هو عدد الملفات في الملف الجذري الحالي.

علامة الشريط الرأسي في أوامر اليونكس تدل على أن تدفق المخرجات من أحد المرشحات سيكون موجها إلى مرشح أخر كمدخل وليس إلى ملفه الافتراضي، هذه العلامة تشير أيضا إلى أن هذين المرشحين سيتم تنفيذهما في الوقت نفسه، في المعالج العادي(غير المتعدد) هذه العملية ستتم بمشاركة المعالج المتوفر بين هذين المرشحين، نظام اليونكس يراعي ضرورة المزامنة والحد من التضارب، هذا التنفيذ المتوازي يقلل من وقت الاستجابة وحجم ملفات الحد من التضارب الوسيطة.

رأينا من قبل أن اليونكس يزود بعدد من الإمكانيات الفعالة كالمرشحات ولكنها محدودة، ويزود بوسيلة توحد المرشحات عندما تكون هناك برمجية جديدة تتطلب ذلك، وعليه فلدينا **ثلاث خيارات:**

- استخدام مرشح موجود.
- دمج مرشحات موجودة باستخدام الأنابيب.
- إنشاء مرشح جديد ودمجه مع آخر موجود باستخدام الأنابيب أيضا.

ودمج المرشحات بشبه إنشاء البرامج إلى حد ما ولكن على مستوى أعلى، مرشحات اليونكس هي بدائيات اللغة والأنابيب تزود هياكل التحكم بدمج هذه المرشحات لتنتج إمكانيات أكثر قوة وفاعلية.

يراعى في المرشحات أن تكون قصيرة وبسيطة. 90 من مرشحات اليونكس القياسية (باستثناء المترجمات) أقل من 1200 سطراً (حوالي 20 صفحة) من جمل لغة

البرمجة عالية المستوى، كل فلتر عادة تتم كتابته على حدة بلغة البرمجة C والتي تعتبر اللغة النواة في اليونكس، الفلتر يقرأ من أنبوب المدخلات وينقلها إلى أنبوب المخرجات، يقوم بفتح أنبوب المخرجات كما لو قام بفتح ملف، ثم يقرأ البيانات بالتسلسل من الأنبوب(أو الملف) حتى نهاية البيانات، يقوم المرشح أيضا بإرسال مخرجاته كما لو كان يرسلها إلى ملف تسلسلي.

كل أدوات اليونكس صممت لتؤدي مهمة واحدة محددة بالإضافة إلى دعمها العديد من الخيارات، وبالإضافة إلى مرشحات اليونكس للأغراض العامة، يوجد ما يسمى lint و make و sccs ، الأولى تحاول الكشف عن سمات البرامج المكتوبة بلغة C (اللغة البرمجية المعتمدة في يونكس) والتي أشبه ما تكون بأخطاء أو عدم توافق أو استهلاك زائد، وتكشف أيضا عن أشياء أخرى مثل الجمل التي لا يمكن الوصول لها وحلقات مستمرة لم تكن مدخله مسبقا، ومتغيرات تم تعريفها ولكن لم تستخدم، تعريف غير مسموح به لبعض الدوال وكذلك الاستهلاك.

make تستخدم للمشاريع التي يدخل فيها أكثر من مبرمج، حيث تمكن المبرمج من الاحتفاظ بمعلومات يمكنه الرجوع إليها تحدد أي الملفات يحتاج إلى إعادة ترجمته بعد أجراء التعديلات على بعض الأجزاء من الكود، وتقوم أيضا بصيانة الإصدارات التي يتم تحديثها باستمرار من البرامج المركبة من العديد من الملفات.

sccs (source code control system) تستخدم لأغراض مشابهة فهي تعمل كأمان للملفات حيث تقوم باستعادة نسخ معينة، وإدارة التغييرات التي يمكن أن تحدث وتسجيل متى وأين وكيف تم ذلك التغيير ومن قام به؟ ***باختصار إن السمات الأساسية ليونكس والتي تعتبر كأساس للأدوات البرمجية هي:***

- عدد المقسمات الصغيرة والعالية، عناصر تستخدم لأغراض عامة تسمى مرشحات أمكن تزويد النظام بها.
- البرمجيات مبنية على اتحاد المرشحات مع مرشحات أخرى باستخدام الأنابيب.

- الوسيلة المعتمدة للتعامل في يونكس تستند إلى افتراضيات جوهرية تعمل على ربط البرامج خلال التدفقات المتسلسلة وهي وسيلة مرنة.

2.4 الإنترلسب

هي بيئة برمجية معتمدة على لغة اللسب، والهدف من الإنترلسب أن تكون معينة ومساعدة لمجتمع المستخدمين للوصول إلى المقصود منها، هؤلاء المبرمجون مشغولون عمليا في تمثيل تطبيقات الذكاء الاصطناعي على سبيل المثال مبرهن النظريات، وموجد حل المشكلات، فهم اللغة، وأنظمة الرؤيا.

البرامج كهذه تعد برامج تجريبية لأن المشكلات التي يحاولون حلها تم تعريفها بصورة غير كافية، من الصعب التحديد بشكل متقدم ما هي المشكلة بالضبط. الانترلسب تم تصميمه خصيصا لدعم هذا اللون التجريبي من تطوير البرمجيات أو "النمو المركب" كما أصبح معروفا، وأهم التأثيرات في تطوير الإنترلسب تتمثل بتصنيع آله لتقوم بذلك العمل كما يجب أن يكون في وسط تكون فيه مصادر الحاسب أرخص من المصادر البشرية وتكاليف الحاسب يجب أن تكون متوقعة، هذا كان له الدور الأساسي في مجموعة الأدوات المتطورة والعامة، نتيجة لذلك إن إتقان جميع الإمكانيات ليس بالأمر السهل وهذا اعتراف بأن الانترلسب هي بالدرجة الأولى للمبرمجين الخبراء.

تقريبا جميع أنظمة الانترلسب مكتوبة بلغة اللسب، تطوير الأدوات البرمجية المتقدمة تم بمساعدة الحقيقة القائلة (أنه من السهل معالجة برامج لسب أخرى كهياكل للبيانات والتي يمكن تمثيلها وتحويلها أيضا)، ونظرا لأن اللسب هي أنظمة ولغة تطبيقات فإنها تسمح للمستخدم أن يفهم بسهوله وان يعدل على الكود المصدري للأدوات التي تم التزويد بها كما لو كان يكتبها بنفسه.

برنامج المستخدم عبارة عن تراكيب للبيانات وهذه التراكيب هي التي يحدث لها تغيير أثناء عملية التحرير (هذه ميزة وهو ما يسمى بالنظام الذاتي)، محرر اللسب يراعي أن تكون اللغة حساسة، وهذا ما يجعلهم أكثر انتباها لبناء اللغة التي يحررونها،

المحرر (EMACS) على سبيل المثال يضبط آليا أقواس الفتح والإغلاق ويكونها بنفسه، وتسمح للمبرمج بالتنقل خلال تمثيل تركيب البيانات لبرنامج المستخدم، جاعلا التغيرات ضرورية على أية حال، والانترلسب يستخدم مجموعة من أدوات البرامج تسمى (حزمة الملفات) لضبط عملية حفظ المجموعات الضرورية للأنظمة الكبيرة والمحتوية على العديد من الملفات المصدرية وترجمتها بشكل متساوي، حزمة الملفات تبدأ كوسيلة سهلة الاستدعاء من قبل المستخدم لتشكيل مهام محددة، أكثر الإمكانيات تمت إضافتها (منصوصة باقتراحات المستخدم) وأخيرا الأداة تكاملت مع النظام بطريقة معينة بحيث يمكن استدعاؤها تلقائيا تحت أي ظرف، وبالرغم من ذلك الكود المصدري لهذه الأدوات مازال قابلاً للوصول من أي مستخدم عام، لذلك إذا تمت أي عملية غير متوقعة، فإنه يجب على المستخدم تمديد الحزمة لتتسع للكود بطريقة مباشرة.

Masterscope هي أداة تفاعلية لتحليل برامج المستخدم، تحدد أي دالة تم استدعاؤها عندما ترجع المتغيرات بقيمة معينة وما إلى ذلك، وتعمل أيضا على صيانة قاعدة بيانات النتائج لذلك التحليل والذي يمكن المستخدم من أجراء استعلام سهل ومباشر، أو طلب المحرر لاستخدام المرشد، وذلك ليقود دورة التحرير بطريقة معينة.

الوسيلة (DWIM (Do What I Mean) يتم استدعاؤها عندما يكتشف النظام خطأ ما، إنه يخمن ما الذي كان يعنيه المستخدم، ويشتمل على مصحح لأخطاء الكتابة والذي تتم مناداته تلقائيا من العديد من أجزاء نظام الانترلسب، DWIM يخبر حزمة الملفات بأي تغيير، لذلك نفس التصحيح سيتم في أي ملفات أخرى ذات صلة عندما يكون ذلك ضروريا، بالرغم من أن الهدف هو أن يشعر المستخدم أنه كما لو كان في محادثة حية مع النظام فان DWIM في اغلب الأحيان يقوم بعمل بعض التغييرات بدون طلب موافقة المستخدم، ليس كل هذه التغيرات هي تصحيح هجائي، على سبيل المثال إسقاط بعض العوامل لدوال معينة والتي قد لا تؤدي إلى أخطاء وخيمة، لذلك DWIM ستجبرها على اخذ قيم افتراضية واضحة، مثل أخر شي عملت عليه هذه الدالة، إجمالا

أن الوصف في الحقيقة لم يصل إلا للقشور فقط DWIM بشكل عام تعتبر واحدة من أكثر الإمكانيات المسجلة والجديرة بالاهتمام في الإنترلسب.

programmer's assistant مساعد المبرمج يقدم مدى فاعلية وجدوى الأدوات التي تمت مناقشتها أعلاه، ويقوم بصيانة تاريخ الدورة النشطة والذي يمكن أن يشار إليه بشكل واضح بواسطة المبرمج، مساعد المبرمج يمكن مناداته بإعادة تشغيل متتابعة من العمليات ربما بشيء من التعديلات، على نحو مماثل فإن التأثيرات التي حدثت بسبب متتابعة العمليات يمكن أن تكون غير فاعلة. هذا قد يكون مفيدا لاستعادة المعلومات التي تم فقدها خلال أخطاء الطباعة، ولكن بالإضافة إلى ذلك الأمر undo يمكن أن يستخدم للتجربة مع تنفيذ البرنامج بعمل بعض التغييرات وتشغيل البرنامج والتراجع عن التعديلات وتشغيل البرنامج مرة أخرى والتراجع عن عملية التراجع وما إلى ذلك، النظام أيضا يمكنه تذكر ما الذي قام المستخدم بطباعته، لذلك فإن الإجراءات الأخيرة يمكن إعادة استخدامها عند الضرورة.

وصفت الإنترلسب بأنها سهلة الاستخدام ومرنة وكذلك بمتانتها مقارنة بالبيئات الأخرى، حيث إنها قابلة للتوسع والتطوير (يتم إنجاز ذلك بكتابة الكود المصدري للأدوات بلغة اللسب وجعلها ممكنة للجميع)، وكذلك بتكاملها، الإنترلسب متكاملة بحيث انه يمكن مناداة أداة مع أخرى بدون فقد إطار النداء الأصلي، وكل أداة يمكنها استخدام أداة أخرى وبأي عمق ، بينما يتم الحفظ لإمكانية العودة خلال مستويات النداء حتى الوصول إلى السياق الأصلي.

الإنترلسب هي مثال للنظام الذي ظهر مبكرا ولكنه يعرض ميزات اشتهرت الآن في العديد من بيئات تطوير البرامج الحديثة ولكنها أتت قبل فترة طويلة من وقتها، إنها جديرة بالدراسة لتكامل إمكانياتها وكونها قابلة للتوسيع والتطوير.

والإنترلسب الآن أصبحت مستخدمة أكثر وأكثر في التطبيقات خارج مجال الذكاء الاصطناعي وبالتحديد في محطات العمل الجاد أو (آليات اللسب) والتي تتضمن نوافذ للعرض عالية الوضوح، كل منها يقابل مهمة مختلفة أو سياق معين، في المستقبل

وبناء البيئات في الانترلسب قد يشتمل على إمكانيات للتعامل مع العرض الرسومي والذي قد يتمثل في برامج أو هياكل بيانات يمكن الوصول لها مباشرة بتحكم المؤشر، بناء البرمجيات وتطويرها فيما بعد سيصبح مهمة رسومية أو تحرير مصور بدلا من التمثيل النصي للبرامج الأساسية، لذلك يجب أن تزود بواجهة متعددة الاستعمالات ومرنة وقوية.

3.4 السمالتوك

بيئة السمالتوك البرمجية صممت إلى حد كبير كنظام برمجي تمهيدي موجه الأهداف للمبرمجين خاصة، ويمكن استخدامها كنموذج للأدوات. من هذا النوع البيئات البرمجية مثل C++ ، Java وغيرها من اللغات موجهة الأهداف تمت صياغتها بعد نظام السمالتوك.

والسمالتوك تزود بقائمة متكاملة من الأدوات لتطوير برامج السمالتوك، وهي مشابهة لبيئة الإنترلسب في أوجه عديدة، السمالتوك نفسها تدعم العديد من الأدوات والتي تربطها بنظام التشغيل، على سبيل المثال المترجم ونظام الملفات ومكتشف الأخطاء ومحرر النصوص والمدقق الإملائي وأدوات الطباعة المساعدة كلها تم تزويد نظام السمالتوك بها،بيئات البرمجة المتكاملة مثل السمالتوك تسمح للمستخدم بتنفيذ العديد من المهام خلال بيئة التطوير، حيث يستطيع المبرمج التنقل سريعا وبسهولة من مهمة تطوير إلى أخرى وبدون فقدان أي معلومات، إن أي مهمة تمت مقاطعتها يمكن معاودة استئنافها في أي وقت من آخر نقطة توقف لها.

وأدوات السمالتوك مكتوبة ضمنيا في السمالتوك. والمكتبة الواسعة من طبقات النظام متوفرة للمبرمجين في شكل مصدري، لذلك المبرمجون لديهم الإمكانية في البحث خلال الكود والتعديل على الأدوات عند الضرورة لملاءمة المتطلبات الشخصية، وتوجد طريقة عملية لتعديل ما تم تغييره، وذلك بإنشاء طبقة جديدة موروثة من طبقات السمالتوك، مثل اللغات الموجهة. الأهداف الأخرى فإن السمالتوك يدعم "البرمجة

بواسطة التوسيع" بدلا من " البرمجة بواسطة الإنشاء"، والمبرمجون حقيقة ينشئون أنظمة التطبيقات بواسطة توسيع بيئة السمالتوك الموجودة.

❐ ***ومن الأدوات الأساسية التي تساعد مبرمجي السمالتوك هي :***

أ) المستكشفات

وهي أدوات تساعد على التنقل وتمكن المبرمج من الحث سريعا من خلال الكود المصدري الموجود في مكتبة الطبقة، ولسرعة الوصول نجد أن الطبقات والدوال منظمة في فئات، فباستخدام واحد من المستكشفات لعرض دالة معينة يمكن للمبرمج الوصول، وذلك باختيار فئة الطبقة المناسبة ثم الطبقة ثم فئة الدالة وأخيرا اسم الدالة مع نافذة مناسبة من المستكشفات بحيث يعرض مصدر الدالة في نافذة أخرى. إن المستكشف الهرمي للطبقات يسمح للمكتبة أن تعرض ضمن التسلسل الهرمي للطبقات(شجرة الوراثة) بدلا من فئات الطبقات، وهذا يزود بعرض أكثر ملاءمة للدوال والمتغيرات الموروثة من قبل الطبقة، فقط مبرمجي السمالتوك لديهم عدد من المستكشفات يمكن فتحها في نفس الوقت.

والمستكشفات لا تستخدم بسهولة كأدوات للتنقل،إذا تمت إضافة دوال وطبقات جديدة إلى النظام في نطاق المستكشف فالوضع الطبيعي للتطوير يكون بترجمة الإضافة (دالة جديدة) إلى النظام ثم اختبارها مباشرة بإعطاء قيمة لتعبير مناسب لتحقق التفاعل مع النظام، في البرامج الكبيرة من غير السهل أن تكون قادرا على استخراج معلومات لها مرجعية مباشرة وذلك لأنواع متعددة، **والأمثلة التالية توضح ذلك:**

- **تمثيل الدوال:** عند إضافة دالة جديدة من المهم أن تكون قادرا على عرض التمثيل للدالة والتي تشكل نفس المهمة في طبقة أخرى، هذا يعطي المبرمج غالبا بعض الرؤية لكيفية تمثيل دالة جديدة.
- **مستخدمي الدوال:** عند تعديل دالة معينة من المهم أن تكون قادرا على عرض الدوال الأخرى التي استخدمت تلك الدالة، هذا قد يعطي بعض الرؤية لعواقب تلك التغييرات.

• **المرجعيات لبعض القيم:** عند تعديل تمثيل الطبقة ربما بإعادة تسمية المتغيرات، جميع المرجعيات للاسم القديم يجب استبدالها بأسماء جديدة، من المهم أن يكون لديك أداة لجمع كل الدوال التي يجب تعديلها لنفس السبب، ومن المهم أيضا أن تعرض جميع ما يعود بقيمة لمتغيرات الطبقة وأسماء الطبقة.

ب) أداة مراقب السمالتوك

تسمح للمبرمج بالوصول داخل الكائن وفحص وتعديل أجزاء مكوناته، إن المراقبات المتخصصة مفيدة في عرض المشاهدات المختلفة للكائن(أو مجموعة من الكائنات ذات الصلة) للمبرمج، على سبيل المثال المراقب العادي مبني على قاموس يبين أن التمثيل تم باستخدام hash table ولكن هذا يعتبر مستوى خاطئاً للمعلومات بالنسبة لأكثر مستعرضي القواميس، أما المراقب المتخصص للقواميس فتزود بالعرض أينما أمكن وكذلك التعديل والإضافة والحذف لزوج من قيم المفاتيح.

إن أهمية مكتشف الأخطاء كما هي في السمالتوك لا يمكن تجاهلها، إنها تسمح للمبرمجين بعرض الدوال المضمنة في الحسابات الحالية وإدراج النقاط الفاصلة وفحص قيم المتغيرات وتعديل الدالة لتصحيح الأخطاء، بسبب طبيعة التوزيع للكود الموجه الأهداف (من أجل الروابط الديناميكية). والحقيقة التي تنص على أن الكود لا يمكن تحليله على وجه ثابت، فهو الأكثر أهمية من اللغات التقليدية لكونه قادرا على أداء خطوة خلال عملية الحسابات، كما هو الحال في عملية اكتشاف أسباب الأخطاء، فإن هذه الإمكانية غالبا ما تستخدم كأداة تتبع لاكتشاف كيفية عمل الكود في النظام الحالي.

السمالتوك أيضا تزود بأدوات إدارة التغيرات فهو يقوم تلقائيا بتتبع جميع التغيرات التي حدثت في النظام ويحفظها في سجل التغيرات، هذا السجل يمكن أن يستخدم للاستعادة من نظام تالف أو للعودة إلى نسخة سابقة من الدوال أو الطبقات، من الممكن عرض تلك التغيرات بطريقة مرشحة، على سبيل المثال عرض التغيرات التي حدثت لطبقة أو دالة معينة.

ت) أدوات لفريق المبرمجين

في هذا المجال السمالتوك مثل العديد من اللغات الأخرى الموجهة بالكائنات، إن السمالتوك مبتكرة منذ استخدامها من أجل نماذج المشاريع والتي تمثل عادة بواسطة اثنين أو ثلاثة مبرمجين لتوحيد اتجاه لغات تطوير النزعة الأولى حيث كانت معتمدة على الأدوات التي يزود بها اليونكس أو بعض أنظمة التشغيل الأساسية الأخرى، على أية حال هذه الأدوات تم تصميمها لدعم تطوير التطبيقات التقليدية، -ما هي أفضل وسيلة للتعامل مع مشكلات التطوير بواسطة عدة مبرمجين في الإعدادات الموجهة بالكائنات؟- ، للفرق الكبيرة من المبرمجين فان البيئات البرمجية الموزعة معتمدة على استخدام مكتبة الطبقات المشتركة، لذا فإن توصيلها بشبكة قد تكون أفضل وسيلة للتعامل مع تلك المشكلة، الأدوات مطلوبة لمشاركة المصدر ومشاركة كود الكائن وذلك للتزويد بالتحكم على ما يمكن رؤيته من الكود وكذلك للتزويد بحق الوصول للكود (كالعرض والتعديل) بطريقة صحيحة، وتحكم النسخة وأخيرا السماح لحزم البرامج بالانطلاق، وبالرغم من أن هذه الأدوات تعمل بشكل معاكس للمعنى الحقيقي المقصود للسمالتوك كبيئة برمجية تمهيدية للمجموعات الصغيرة لكنها ضرورية للسمالتوك وأنظمة أخرى موجهة بالكائنات عند استخدامها لمشاريع هندسة البرمجيات الكبيرة.

أسئلة تقويم ذاتي

1)ما أهم قوائم الأدوات:

ا) اليونكس؟

ب) السمالتوك؟

ت) الإنترليسب؟

اشرح السمات الأساسية لكل أداة.

وما أهم التطبيقات التي تناسب كل أداه؟

5. أنظمة تطوير البرامج

لم نقم باختيار وصف واحد من آخر أنظمة تطوير البرامج المتكاملة والمتوفرة بأحد مخازن البرمجيات الأساسية، ولكن بدلا من ذلك تناولنا دراسة أكثر عمومية وأكاديمية لبعض أدوات ومستلزمات البرمجة.

نظام تطوير البرامج الحديث يزود المستخدم **بواجهة مستخدم رسومية GUI** بعرض عدد من النوافذ، النافذة الرئيسية تعرض الكود لإجراء تحت التطوير حيث يهيئ الكود تلقائيا بفراغ في بداية السطر وكذلك بفراغات أخرى بيضاء، وقد تستخدم الألوان للتمييز بين الكلمات المحجوزة والتعليقات، وقد تدفع المبرمج أيضا لإضافة اقتراحات في مجال حساس تعد كمفاتيح نصية للمبرمج، على سبيل المثال المبرمجون المصرح لهم فقط يمكنهم الارتباط باجراء مكتبة معينة. النافذة الثانية تعرض تمثيلاً رسومياً للبرنامج وذلك بعرض وظيفة كل إجراء مصحوبا بصورة كبيرة. نوافذ أخرى تعرض معلومات تم اختيارها عن المكتبات المتوفرة، وهذا إما أن يكون بصورة نصية أو رسومية. وعندما تتم مناداة مكتشف الأخطاء فإن النقاط الفاصلة تصبح بارزة خلال نص البرنامج، والمؤشر يضع علامة للتعليمة الحالية تحت التنفيذ.

1.5 أدوات تطوير البرمجيات

إن فكرة أدوات تطوير البرمجيات وجدت لبناء البرمجيات بدون الحاجة إلى تدوين جمل لغات البرمجة التقليدية، بدلا من ذلك فإن المطور يصمم النظام بالتفاعل مع الكمبيوتر وتحديد تخطيطات الشاشة ومعالجة المعلومات.

إن مساحة التطبيقات لهذه الأدوات عادة ما تكون أنظمة معلومات أو في الحواسيب الشخصية المرتبطة بشبكة، مثال لحزمة من هذا النوع MS Access و"نظام إدارة قاعدة البيانات" و VB و Spreadsheet برامج الحسابات الإحصائية.

لتوضيح كيفية استخدام حزمة إدارة قواعد البيانات لنفترض إننا نريد تطوير موجه هاتف شخصي، باعتبار مدخلات البيانات وهي أسماء الأشخاص وأرقام هواتفهم،

المصمم عليه البدء بتصميم تخطيط الشاشة المستخدمة لإدخال المعلومات، سيتم تنفيذ التصميم بالتفاعل مع استخدام القوائم المزودة لهذا الغرض، في هذه الحالة ربما نريد تحديد أن هناك اسم مدخل في صندوق النص ورقم في الآخر، عند اكتمال التخزين فإن الحزمة تقوم بتخزين تخطيط الشاشة للاستخدام اللاحق، أيضا يتم تسجيل معلومات عن الطريقة التي تم بها تخزين معلومات الاسم والرقم على القرص، لدينا الآن بناء لبرنامج مدخلات البيانات والذي يسمح للمستخدم بإدخال الاسم ورقم الهاتف.

وبالرغم من أن الغرض من النظام هو استخراج رقم الهاتف فقد قام المصمم ببناء مخطط لشاشة معينة تسمح للمستخدم بإدخال الاسم لإعلام الحزمة بالبحث في قاعدة البيانات عن المعلومات المطابقة وعرضها، يمكن للمصمم تصميم شاشات بشكل مماثل لذلك المستخدم كما يمكنه أيضا استخدام الحزمة للتنبيه (عند تحرير أو تعديل) المعلومات الحالية في حال محاولة شخص ما تغيير رقم هاتفه، وكذلك لإضافة سجل جديد أو حذف موجود، إن أكثر الحزم تسمح بالبحث الاختياري عن المعلومات. على سبيل المثال في نظام المعلومات الشخصية فإنه يمكن البحث عن جميع السجلات والتي لها صفة معينة (كالأشخاص الذكور والأكبر من 60 عاما)، يمكن تصميم التقارير باستخدام حزمة التخطيطات مثلا تقارير محددة لمعلومات تم اختيارها مسبقا من قاعدة البيانات ورسم المدخلات (عند الضرورة) باستخدام مخططات مختلفة ومتعددة للشاشات، الحسابات أيضا والتي تجرى على عناصر البيانات يمكن تحديدها، على سبيل المثال حقل ما في تقرير معين تم إنتاجه بواسطة حقلين آخرين في قاعدة البيانات.

والحزم من هذا النوع تستخدم المماثلات الأساسية لنظم المعلومات. مثل هذه التطبيقات كل منها يتطلب حق الوصول إلى ملفات البيانات، لذلك فإن برمجيات الوصول للملفات أصبحت شائعة ومضمنة في الحزمة نفسها، كل ذلك خاص للتطبيقات التي تشتمل على مخطط محدد ومحتوى لقاعدة البيانات ومخطط لشاشة المدخلات وآخر لشاشة المخرجات.

بالنسبة لفجوال بيسك فإنه الآن يسمح للمطور بإنشاء برامج (تطبيقات) والتي تعرض بواجهة نوافذ مايكروسوفت ذات الاستخدام الواسع، الحزم تقوم بعمل كبير خلال إنشاء النوافذ، وصناديق الرسائل والقوائم وكذلك الكشف عن نقرة الماوس. كمثال لاستخدام الفجوال بيسك دعنا نأخذ في الاعتبار بناء الآلة الحاسبة، إن الخطوة الأولى هي تصميم واجهة المستخدم باستخدام صندوق الأدوات كما بالشكل المبين 1-1.

فجوال بيسك تزود بصندوق أدوات يدعم تصميم واجهة عملية تستخدم القوائم والماوس، يقوم المبرمج باختيار زر مثلا من قائمة الأدوات ويقوم بإسقاطه في مكان مناسب خلال النافذة والوظيفة المطلوبة يمكن إنجازها بكتابة كود للتفاعل بالنقر على ذلك الزر ، في فجوال بيسك فإن أي إجراء تتم مناداته تلقائيا لأي حدث ممكن يقع في واجهة المستخدم.

شكل 1-1 واجهة المستخدم لبرنامج الآلة الحاسبة

بالنسبة لبرنامج الآلة الحاسبة نحتاج إلى بعض المتغيرات العامة، إن كلمة Dim في فجوال بيسك تقدم تعريفاً لمتغير ما.

```
Dim newNumber As Boolean
Dim first As Boolean
Dim ans As Integer
Dim op As String
```

وهذا يحتاج إلى إسناد قيمة عند بدء البرنامج بالعمل، والإجراء المسمى Form_Load (إذا كان مزودا من قبل المبرمج) تتم مناداته عندما تبدأ النافذة بالتحميل، عموما في فجوال الإجراء دائما يكون مسبوقا بكلمة Sub.

```
Sub Form_Load( )
newNumber = True
First = True
op = "="
ans = 0
End Sub
```

الآن قمنا بكتابة إجراء تتم مناداته عند النقر على أي زر من أزرار الأرقام في الحاسبة

```
Sub zero_button_Click
setDigit (0)
End Sub
Sub one _button_Click
setDigit (1)
End Sub

Sub two_button_Click
setDigit (2)
End Sub
```

وهكذا لأي زر من أزرار الأرقام، وهذا ينادي الإجراء

```
Sub setDigit(digit As Integer)
if newNumber Then
        display = digit
Else
        display = display & digit
End If
newNumber = False
```

```
first = False
End Sub
```

الآن نقوم بكتابة الكود الذي تتم مناداته عند النقر على أحد أزرار العمليات الحسابية

```
Sub add_Click( )
calculate
op = "+"
End Sub

Sub subtract_Click( )
If first Then
        display = "-"
        first = False
Else
        calculate
        op = "-"
End If
End Sub

Sub equal_Click( )
calculate
op = "="
End Sub
```

وأخيرا الإجراء الذي ينفذ على الآلة الحاسبة

```
Sub calculate( )
If op = "+" Then ans = ans + display
If op = "-" Then ans = ans – display
If op = "=" Then ans = display
display = ans
newNumber = True
End Sub
```

بهذا نكمل الكود الخاص بالآلة الحاسبة والقاري يمكنه التخيل أن هناك وراثـــات بسيطة فقط لتمثيل زر المسح وزر الإضافة لجعل الآلة تؤدي عملية الضـــرب والقســـمة

وأي عملية أخرى. والنقطة التي يجب إدراكها في هذا المثال هي انه لا توجد عملية تكويد مطلوبة لإنشاء واجهة مستخدم. والقليل من عملية التكويد مطلوبة للتعامل مع الإحداثيات في أي واجهة مستخدم.

والفجوال بيسك لغة ذات متانة عالية تتضمن المصفوفات والسلاسل الحرفية والإجراءات بعواملها والتعامل مع الملفات والرسومات بالإضافة إلى عناصر موجهة بالكائنات يتم تضمينها تدريجيا في الإصدارات الجديدة، لذلك فإنه من السهل الممكن تمثيل أي برمجية مطلوبة، ومتانة الفجوال بيسك تكمن في أن واجهة المستخدم يمكن بناؤها بسهولة مقارنة بمئات الأسطر من الكود التي يمكن أن نحتاجها لبناء نفس الشيء في C++. وجميع إمكانيات أنواع النوافذ المألوفة ممكنة للمستخدم كشريط التمرير وأزرار الخيارات وصندوق الخيارات والقوائم .... الخ، هذه ما يجعله أكثر جاذبية بالنسبة للنمذجة. ومن الأشياء الأخرى المحببة في المستويات المتقدمة من النظام هي دعم البرمجة الموجهة بالإحداثيات، نجد أن البرمجيات مكتوبة بحسب استجابتها للإحداثيات الصادرة من المستخدم كنقر احد أزرار الماوس.

وعيب الفجوال بيسك هي أنها تدعم نوعاً محدداً من الواجهات (شكل واجهة نوافذ مايكروسوفت) ويعمل فقط تحت أنظمة تشغيل مايكروسوفت وهذا ربما مالا يطلبة المستخدم، وهناك أداة أخرى هي برمجة التطبيقات الحسابية والتي تزود بطريقة أخرى لعملية اختصار وسائل التعامل التقليدية لتطوير البرمجيات في التطبيقات التي تتم معالجة البيانات فيها والتي يمكن تصويرها بشكل جدول.

أسئلة تقويم ذاتي

1) ما أهم الميزات والعيوب الأساسية للغة الفيجوال بيسك؟

6. البيئة المادية

العديد من العناصر إلى جانب الأدوات تدعم عملية تطوير البرمجيات، وواحدة من هذه المخططات المكتبية هي ما سنبحث فيه الآن، شاع اعتبار مطوري البرمجيات كعاملين في مكاتب والعديد من الناس أيضا يعملون في مخططات مفتوحة من المكاتب، ولكن على الأرجح فان تطوير البرمجيات يتطلب الحصة الأكبر من التركيز العالي والمهدد بالإزعاج والتشويش.

لسنوات مضت تمت مواجهة الاي بي ام بمهمة تصميم سكن لـ200 شخص مضمنة في تطوير البرمجيات، استعدادا لذلك الاي بي ام مكلفة بدراسة المتطلبات المكتبية لهؤلاء الأشخاص. نتائج الدراسة اعتمدت على كيفية قضاء المبرمجين لأوقاتهم في العمل:

- 30% يعملون لوحدهم.
- 50% يعملون مع الآخرين.
- 20% أخرى.

دراسة التصميم أنتجت الملاحظات التالية عن احتياجات مطوري البرمجيات:

- مساحة العمل الخاصة والشخصية والتي تسمح بتركيز عال، شاشات تستحوذ الانتباه وتشجع على عدم مقاطعة العمل.
- مكان لتخزين المخططات والأوراق.
- قرب غرف الجلوس.
- سهولة الوصول للمكتبات وخدمات الطعام.
- أثاث خاص.

المصممون تابعوا تحديد صالة البرمجة والمحتوية على قطاعات من المكاتب الفردية كل منها عشر أقدام بعشر أقدام، بجانبها غرف للجلوس، الأثاث في كل غرفة واف للشروط بحيث يلائم الحواسب والمكان المخصص لها. في دراسة أخرى، ديماركو

وليستر قاما بالتحري حول التأثيرات لعناصر متنوعة على جودة البرمجيات، واحدة من الآثار التي درسوها كانت الإزعاج، ووجدوا أن هؤلاء المطورين الذين قرروا بأن مكان عملهم كان هادئا إلى حد مقبول قد أنتجوا برمجيات تحتوي عيوبا قليلة قد تؤثر في مستوى الأداء، ***ديماركو وليستر يقترحان عدداً من الأمور القياسية للتحكم بالإزعاج وإحلال التركيز:***

- مكان كاف لكل عامل.
- مكان خاص (مكاتب فردية أو أقسام).
- إلغاء صوت الهاتف (عند الضرورة بورق مناديل) أو يمكن تحويل المكالمات كطريقة أخرى.
- استخدام البريد الإلكتروني يكون أفضل من الهاتف للتخلص من مشكلة المقاطعة.
- في مركز القيادة في واشنطن مايكروسوفت تزود مطوري البرمجيات ببيئة عمل محببة، كل مبرمج لديه مكتبه الخاص غالبا مع منظر خلاب لحديقة خارجية وغرف للجلوس وكافيتيريا يسهل الوصول إليها.

أسئلة تقويم ذاتي

1) "تلعب البيئة المادية دورا كبيرا إضافة لأدوات كيس في تطوير البرمجيات" اشرح هذه العبارة.

7. مثال تطبيقي على استخدام CASE Tools

تطبيق أدوات البرمجيات المساعدة في التصميم للارتقاء بمهارات التعليم و التدريب لدى طلاب الهندسة الكهربية بإحدى الجامعات العربية

الفترة:2005/7/15 إلى 2006/7/15

يعتبر هذا المشروع من المشاريع الصغيرة لمدة عام واحد والتي تمول من قبل إدارة صندوق تطوير التعليم العالي بالبلد المعني.

مكان التنفيذ: كلية الهندسة .

ملخص المشروع:

يهدف المشروع إلى الارتقاء بمهارات التصميم لدى طلاب و مهندسي الهندسة الكهربية من خلال استخدام أساليب و أدوات تصميم حديثة في العملية التعليمية.

وبالرغم من صغر المشروع من حيث الفترة الزمنية المحددة لتنفيذه (اثنا عشر شهرا) إلا أنه ترك أثرا ايجابياً على عملية تعليم الهندسة الكهربية من حيث جودة التعليم و قدرات الخريج، حيث إنه من خلال هذا المشروع تم إجراء تكامل بين تعليم التصميم المبني على أدوات الحاسوب المساعدة وأساليب التعليم التقليدية نحو تغطية احتياجات مهندسي المستقبل. منهجية العمل تعتمد على تحليل دقيق لأساليب التدريس الحالية و مقارنات وافية بالأساليب التعليمية المتبعة في المؤسسات التعليمية المتقدمة. وقد تم ذلك بطرق كثيرة مثل إجراء استطلاعات لرأى المعنيين بالتطوير من خبراء و مهندسين و أرباب عمل و غيرهم عن أهمية تطبيق الأساليب الحديثة في تنمية و تطوير إمكانيات و قدرات خريجي قسم الهندسة الكهربية. تبع ذلك تصميم و تنفيذ مواد وأدوات تعليمية حديثة يتم استخدامها في التعليم و التدريب ثم تقييمها.

أهداف المشروع

الأهداف الأساسية للمشروع هي:

1- تطوير العملية التعليمية لأربعة برامج من خلال المشروع باستخدام CAD tools

2- تطوير العملية التعليمية عن طريق إعداد محتويات إليكترونية لثمانية مقررات

3- إنشاء (مركز التصميم الحديث) لتدريب و تعليم مصممي أنظمة الكهرباء و الحاسبات .

4- تنمية المهارات التصميمية لدى الطلاب و المهندسين.

5- نشر الوعي بأهمية أدوات التصميم الحديث و كيفية استخدامها.

6- تنفيذ إجراءات الجودة لمخرجات المشروع.

7- نشر الوعي بهدف و مخرجات المشروع.

8- إنشاء بيئة تعليم إليكتروني بكلية الهندسة و تجربة تفعيل و تقييم بيئة التدريس الإليكتروني للمقررات المطورة من خلال المشروع.

خطوات تنفيذ المشروع:

1- تحديد فكرة و أهداف المشروع و صياغتها في شكل مقترح.

2- كتابة مقترح المشروع و تقديمه لهيئة تطوير التعليم في المواعيد المحددة.

وقد تم البدء في إجراءات تنفيذه كما يلي:

1- إعداد النموذج التصوري لتطوير العملية التعليمية.

2- تحديد البرامج الأربعة التي سوف يتم بها التطوير.

3- تحديد الأجهزة والبرامج المطلوبة فى عملية التطوير.

4- عمل إحصائيات حول مهارات التصميم المطلوبة و مقارنتها بالنموذج التصوري.

5- وضع خطة تطوير لثمانية مواد دراسية و أربعة برامج تدريب.

6- تفعيل تجربة التعلم عن بعد للمواد المطورة، و ذلك بإنشاء موقع خاص بالمشروع، تم فيها تحميل المواد التي تم تطويرها من خلال المشروع.

7- توعية الطلاب و المهندسين بأهمية تطبيق فكرة التعلم عن بعد و ضرورة إتقان استخدام البرامج الحديثة للتصميم الحديث.

8- تطبيق البرامج الحديثة في العملية التعليمية.

9- عمل إحصائيات حول مدى استفادة الطلاب من التطوير الذي تم عن طريق المشروع.

10- تنفيذ دورات تدريبية خاصة بالطلاب و المهندسين باستخدام البرامج الحديثة.

11- توكيد جودة مخرجات المشروع و ذلك بالاستفادة من خبرات مراجعين من أساتذة الجامعة.

12- نشر فكرة المشروع و التعريف بأهدافه و مدى الاستفادة منها من خلال إقامة ورش العمل داخل و خارج مقر المشروع و مشاركة المؤسسات الصناعية .

الخطة المستقبلية للمشروع:

1. استمرارية تدريس المقررات التي تم تطويرها خلال فترة المشروع للطلاب بعد نهاية فترة المشروع و العمل على تطوير مواد دراسية أخرى.
2. تطبيق عملية التطوير باستخدام التصميم الحديث بمعاونة الحاسوب الآلي (CAD tools) لعدد آخر من المقررات الدراسية.
3. تنفيذ برامج تدريب دراسية متقدمة للطلاب مدفوعة الأجر.
4. تنفيذ برامج تدريب هندسية متخصصة للمهندسين مدفوعة الأجر.
5. إقامة ندوات ومؤتمرات وورش عمل محلية و دولية.
6. استمرارية تطوير موقع إلكتروني يحتوي على المقررات الدراسية التي يتم تطويرها على الموقع الخاص بالجامعة لتكون متاحة للطلاب و المهندسين.
7. التدريب على استخدام الحاسوب في الأعمال الهندسية المختلفة (تحليل – تصميم – مراقبة و اختبار....الخ)
8. تقديم خدمات بحثية لأعضاء هيئة التدريس بالجامعة و طلاب الدراسات العليا وأقسام البحث والتطوير في المؤسسات الصناعية .

الصعوبات التي تواجه المشروع:

المشاكل	طرق التغلب عليها
أولاً مشاكل فنية:	
- سفر أحد الأعضاء.	تكليف أعضاء آخرين باستكمال برنامج الشبكات.
- الاتصال بالإنترنت من مركز التصميم الحديث.	إنشاء شبكة محلية بالمركز و استخدام جهاز الخادم الخاص بفرع آخر للجامعة لوضع الموقع الخاص بالمشروع عليه.
- انخفاض مستوى مهارات الحاسب الأساسية المطلوبة لدى الطلاب.	تنفيذ تدريب للطلاب في مهارات الحاسب الأساسية.
- ضآلة حجم التعاون مع المؤسسات الصناعية سابقا.	تنفيذ عدد كبير من الزيارات لبناء علاقات مع مؤسسات صناعية.
ثانيا: مشاكل إدارية:	
- بدلات الانتقال تحتاج الى مراجعة.	
- التأخر في تنفيذ بعض أنشطة المشروع.	مد المشروع 3 شهور أخرى.
ثالثاً المشاكل المالية:	
- تأخر وصول التحويلات المالية	تعديل الخطة الزمنية للمشروع.
- تأخر توريد الأجهزة الخاصة بالمشروع.	تعديل الخطة الزمنية للمشروع.

الخلاصة :

*وضحت الوحدة الهدف الأساسي من أدوات كيس (CASE Tools) وهو أتمتة عمليات توليد البرمجيات وتطوير التطبيقات ، ويشمل ذلك أتمتة جميع المراحل والتي تشمل (تحديد الاحتياجات – التحليل – التصميم – كتابة البرنامج – اختبار البرامج وصيانتها).

* عرضت الوحدة مزايا استخدام أدوات كيس: سرعة إنتاج البرامج – خفض تكاليف التطوير – الالتزام بمعايير ثابتة للتطوير – تيسير تطوير الأنظمة الكبيرة – تكاليف التوثيق.

* لخصت الوحدة أبرز سلبيات أدوات كيس في : صعوبة تعلم هذه الأدوات – قلة أعداد المختصين المتمرسين في أدوات كيس – صعوبة تطوير بعض التطبيقات الخاصة – ارتفاع التكلفة.

* ناقشت الوحدة التساؤل الآتي: هل يمكن للبرمجيات أن تساعد في كتابة برامج أفضل؟ وذلك من خلال الفرعيات التالية: لا مجال للخطأ – الحلول العملية – العودة إلى الكود – مقارنة الإصدارات – اختبار واقتراح التصويبات – البرمجة بالنماذج– معرف للنموذج – العامل البشري وكتابة الكود – أدوات لغرض البرمجة – أدوات غير متعلقة بالبرمجة .

* استعرضت الوحدة ثلاثة أمثلة هامة لقوائم الأدوات هي:

اليونكس : وهو نظام تشغيل للأغراض العامة متوفر بشكل واسع على الحواسب الشخصية والخادمات والحواسب الكبيرة .

الانترلسب : وهي بيئة برمجية معتمدة على لغة اللسب، الهدف من الانترلسب أن تكون معينة ومساعدة لمجتمع المستخدمين للوصول إلى المقصود منها .

السمالتوك : بيئة السمالتوك البرمجية صممت إلى حد كبير كنظام برمجي تمهيدي موجه الأهداف للمبرمجين خاصة، ويمكن استخدامها كنموذج للأدوات من هذا النوع السمالتوك يدعم "البرمجة بواسطة التوسيع" بدلا من " البرمجة بواسطة الإنشاء" .

*وضحت الوحدة إن فكرة أدوات تطوير البرمجيات وجدت لبناء البرمجيات بدون الحاجة إلى تدوين جمل لغات البرمجة التقليدية، بدلا من ذلك المطور يصمم النظام بالتفاعل مع الكمبيوتر وتحديد تخطيطات الشاشة ومعالجة المعلومات.

*وضحت الوحدة أن التخطيطات المادية للمساحات المكتبية لها تأثير على أداء مطوري البرمجيات.

لمحة مسبقة عن الوحدة التالية

عزيزي الدارس، الوحدة التالية تأتي بعنوان " التوصيل، والتنصيب ، والتوثيق للبرمجيات " وتشرح الوحدة سيناريوهات توصيل البرنامج، وتنصيب البرامج كما تشرح أشكال الوثائق المعروفة . آمل أن تجدها وحدة مفيدة.

مسرد المصطلحات

الأدوات المساعدة في البرمجة CASE Tools

CASE tools= Computer aided software engineering

أو أدوات هندسة البرمجيات باستخدام الحاسوب، وهي مجموعة برمجيات تأخذ منك تصميمك النظري للبرمجيات وتحوله إلى شكل flow charts and design diagrams ، ومن ثم تساهم بصورة شبه آلية في إنتاج الكود الموافق للتصميم، وبالتالي تساهم في تقليل زمن دورة تطوير البرمجيات software development cycle وتزيد من قابلية إعادة استخدام البرمجيات software reusability.

ويمكن تعريفها كذلك على أنها برمجيات البرمجة Programming Software: وهي البرمجيات المساعدة في عملية إنتاج البرمجيات Software Development نفسها.

التحليل الإستاتيكي

هو البحث عن أخطاء البرمجيات داخل سطور الكود ويسمى «التحليل الاستاتيكي» لأنه يتم والبرنامج لا يعمل .

PRE fast

هي أداة للبحث الإستاتيكي عن الأخطاء في البرامج فور كتابتها مما يؤدي إلى تقلصها، طورت من قبل شركة مايكروسوفت .

اليونكس

هو نظام تشغيل للأغراض العامة متوفر بشكل واسع على الحواسب الشخصية والخادمات والحواسب الكبيرة (Mainframe) .

الإنترلسب

هي بيئة برمجية معتمدة على لغة اللسب، الهدف من الانترلسب أن تكون معينة ومساعدة لمجتمع المستخدمين للوصول إلى المقصود منها، وهؤلاء المبرمجون مشغولون عمليا في تمثيل تطبيقات الذكاء الاصطناعي، على سبيل المثال مبرهن النظريات، وموجد حل المشكلات، وفهم اللغة، وأنظمة الرؤيا.

Masterscope

هي أداة تفاعلية لتحليل برامج المستخدم .

السمالتوك

بيئة السمالتوك البرمجية صممت إلى حد كبير كنظام برمجي تمهيدي موجه الأهداف للمبرمجين خاصة، ويمكن استخدامها كنموذج للأدوات. من هذا النوع البيئات البرمجية مثل C++ ، Java وغيرها من اللغات موجهة الأهداف التي تمت صياغتها بعد نظام السمالتوك.

المستكشفات

وهي أدوات تساعد على التنقل وتمكن المبرمج من الحث سريعا من خلال الكود المصدري الموجود في مكتبة الطبقة .

أداة مراقب السمالتوك

تسمح للمبرمج بالوصول داخل الكائن وفحص وتعديل أجزاء مكوناته.

المراجع

[5] Ian Somerville ، "Software Engineering"، Addison Wesley، 2001.

[6] Ronald J. Leach،، "Introduction to Software Engineering"، CRC Press، 1999.

[7] Douglas Bell ، "Software Engineering A Programming Approach"، 3rd Edition، Addison Wesley.

[8] **Shari Pfleeger، "Software Engineering - Theory and Practice"، 2nd Edition.**

References

- Software Development Tools for Petascale Computing Workshop 2007
- Kernighan، Brian W.; Plauger، P. J. (1976)، *Software Tools*، Addison-Wesley، pp. 352، ISBN 020103669X

Retrieved from "http://en.wikipedia.org/wiki/Programming_tool"

www.svu.edu.eg/specialunits/acadeet/index.html Home page:

أمثلة لأدوات كيس

Programming tool

A **programming tool** or **software development tool** is a program or application that software developers use to create، debug، maintain، or otherwise support other programs and applications. The term usually refers to relatively simple programs that can be combined together to accomplish a task، much as one might use multiple hand tools to fix a physical object.

Categories

Software development tools can be roughly divided into the following categories:

- performance analysis tools
- debugging tools
- static analysis and formal verification tools
- correctness checking tools
- memory usage tools
- application build tools
- integrated development environment
-

List of tools

Software tools come in many forms:

- Bug Databases: gnats، Bugzilla، Trac، Atlassian Jira، LibreSource، SharpForge
- Build Tools: Make، automake، Apache Ant، SCons، Rake، Flowtracer، cmake، qmake
- Code coverage: C++test،GCT، Insure++، Jtest، CCover
- Code Sharing Sites: Freshmeat، Krugle، Sourceforge، ByteMyCode. See also Code search engines.
- Compilation and linking tools: GNU toolchain، gcc، Microsoft Visual Studio، CodeWarrior، Xcode، ICC
- Debuggers: gdb، GNU Binutils، valgrind. Debugging tools also are used in the process of debugging code، and can also be used to create code that is more compliant to standards and portable than if they were not used.
- Disassemblers: Generally reverse-engineering tools.
- Documentation generators: Doxygen، help2man، POD، Javadoc، Pydoc/Epydoc، asciidoc
- Formal methods: Mathematically-based techniques for specification، development and verification
- GUI interface generators: Qt Designer، Cocoa InterfaceBuilder، Windows Forms Visual Studio
- Library interface generators: Swig
- Integration Tools: OESIS
- Memory Use/Leaks/Corruptions Detection: Aard، dmalloc، Electric Fence، duma، Insure++. Memory leak detection: In the C programming language for instance، memory leaks are

not as easily detected - software tools called memory debuggers are often used to find memory leaks enabling the programmer to find these problems much more efficiently than inspection alone.

- Parser generators: Lex، Yacc، Parsec
- Performance analysis or profiling
- Refactoring Browser
- Revision control: Bazaar، Bitkeeper، Bonsai، ClearCase، CVS، Git، GNU arch، Mercurial، Monotone، Perforce، PVCS، RCS، SCM، SCCS، SourceSafe، SVN، LibreSource Synchronizer
- Scripting languages: Awk، Perl، Python، REXX، Ruby، Shell، Tcl
- Search: grep، find
- Source-Code Clones/Duplications Finding: CCFinderX
- Source code generation tools
- Static code analysis: C++test، Jtest، lint، Splint، PMD، Findbugs، .TEST
- Text editors: emacs، vi، vim
-

IDEs

Integrated development environments (IDEs) combine the features of many tools into one complete package. They are usually simpler and make it easier to do simple tasks، such as searching for content only in files in a particular project. IDEs are often used for development of enterprise-level applications.
Some examples of IDEs are:

- Delphi
- C++ Builder
- Microsoft Visual Studio
- GNAT Programming Studio
- Xcode
- IBM Rational Application Developer

- Eclipse
- NetBeans
- IntelliJ IDEA
- WinDev
- Code::Blocks
- List of performance analysis tool
- Computer-aided software engineering tools
- Software development kit
- Configuration System
- Toolkits for User Innovation
- Software engineering and list of software engineering topics
- Software systems
- Computer science


الوحدة الثانية
التوصيل Delivery والتنصيب
Installation
والتوثيق Documentation
للبرمجيات

محتويات الوحدة

المقدمة

تمهيد

عزيزي الدارس،،

مرحباً بك في الوحدة الثانية من مقرر **"هندسة البرمجيات 2"**، والتي تحمل عنوان " التوصيل Delivery ، والتنصيب Installation ، والتوثيق Documentation للبرمجيات ".

تحتوي الوحدة على خمسة أقسام القسم الأول يتناول التوصيل للبرمجيات حيث يشرح القسم سيناريوهات توصيل البرنامج اعتماداً على طبيعة العلاقة بين العميل والمؤسسة التي تقوم بتطوير Software . والقسم الثاني من الوحدة يتناول التنصيب Installation للبرنامج . والقسم الثالث يبحث في التوثيق حيث يتناول التوثيق الداخلي ، والتوثيق الخارجي ، كما يتناول القسم الأسس المنطقية للتصميم ، وأيضاً عملية التثبيت ، وتدريب المستخدم وكتيبات التشغيل ، وكذلك فيتناول القسم التوثيق الالكتروني ، وأيضاً مستويات القرأة . القسم الرابع يتناول وجهة نظر المدير المنفذ في التسليم و التنصيب والتوثيق، القسم الخامس من الوحدة يتناول التسليم والتنصيب والتوثيق لمشروع برنامج رئيس.

أهداف الوحدة

عزيزي الدارس، بعد فراغك من دراسة هذه الوحدة ينبغي أن تكون قادراً على أن:

- **تشرح** سيناريوهات توصيل البرنامج "Software"
- **توضح** أشكال الوثائق المعروفة .
- **تذكر** نماذج المساعدة الإلكترونية للبرنامج .
- **تبين** أهمية تدريب المستخدم وأهمية كتيبات التشغيل.
- **تبين** مستويات القراءة المختلفة.
- **توضح** وجهة نظر المدير المنفذ في التسليم، والتنصيب والتوثيق.
- **تشرح** عمليات التسليم والتنصيب والتوثيق لمشروع برنامج رئيس.

1. التوصيل Delivery

عزيزي الدارس، لا يدخل البرنامج "Software" مرحلة الاستخدام الفعلي إلا إذا أصبح ملائماً للاستخدام. و لنفترض أن نظام التشغيل "System Software " تم تصميمه وتطويره واختباره طبقاً لمتطلبات النظام فإنه بعد ذلك يجب أن يتم جعله ملائماً للمستخدمين.

❑ ***يتم توصيل البرنامج "Software" طبقاً لواحد من السيناريوهات التالية اعتماداً على طبيعة العلاقة بين العميل والمؤسسة التي تقوم بتطوير "Software":***

❶ إذا كان هناك مستخدم واحد أو عدد قليل من المستخدمين فإن البرنامج " Software" يتم تسليمه على شكل شريط مغناطيسي "ديسكات" أو اسطوانات "سي دي" مع الالتزام التام بمتطلبات المشروع.

❷ إذا كان يوجد عدد كبير من المستخدمين للبرنامج مع القيام ببيع البرنامج " Software" في المحلات التجارية بواسطة آلية البيع بالتجزئة فإن البرنامج "Software" غالباً يتم توصيله في أقراص مرنة "Diskettes" أو اسطوانات "CD" طبقاً للشكل الذي تطلبه المحلات التجارية ويكون فريق توصيل البرامج "Software" مسؤولاً عن نسخه لضمان توزيعه بالشكل المناسب كما يقوم فريق التوصيل بعملية التنصيب.

❸ إذا كان يوجد عدد كبير من المستخدمين للبرنامج"Software" يتم توصيله إلكترونياً باستخدام البرنامج الناقل "FTP" أو البرامج المشابهة الأخرى فلا يوجد هناك داعٍ لعمل نسخ متعددة من البرنامج"Software" وفي هذه الحالة فإن مسؤولية فريق التوصيل هو التأكد من أن البرنامج يمكن أن يتم الحصول عليه بشكل مناسب.

⇐ وسوف نناقش الآن كلاً من هذه السيناريوهات:

i. سيناريو التوصيل الأول

يصف نظام التشغيل "System Software" للعميل الواحد ويحتاج هذا العميل للتدريب على استخدام هذا النظام وإذا لم يتم توصيل البرنامج "Software" لهذا العميل طبقاً للمواصفات المطلوبة في العقد المبرم بينهما فإن العميل لا يلتزم بدفع قيمة المبالغ المنصوص عليها في العقد، ومن ثم فإن العميل سوف يُخبر مؤسسة تطوير البرنامج بأن عملية التوصيل لم تكن ناجحة.

- **والمشاكل التي تواجه عملية تنصيب البرنامج في هذا السيناريو هي:**
  - فشل الأداة المادية والتي يتم فيها تثبيت البرنامج.
  - وضع غير صحيح في الأدلة مما ينتج عنه خطأ في الأسماء.
  - استخدام غير صحيح للأسماء الموجودة في الأدلة الفرعية مما يجعل من الأنظمة الفرعية أنظمة غير ملائمة.
  - تصريحات دخول غير صحيحة مما يصعب الدخول إلى الملفات.
  - تهيئة غير صحيحة للبرنامج مثل استخدام أدوات تبادل غير سليمة لضغط أو فك الملفات.
  - توثيق غير سليم لتعليمات التوصيل.

ويجب أن توضع كل المشاكل الرئيسية المتوقعة في الحسبان قبل الشروع في عملية التوصل، وكذلك التأكد من نجاح اختبار خطوات التنصيب بنجاح.

ii. **والسيناريو الثاني المناسب** للتوصيل ملائم، والذي يباع في المتاجر أو بواسطة البريد الإلكتروني، ويشمل جميع النقاط السابقة بالإضافة إلى أن المؤسسة التي تقوم بإنتاج البرنامج يجب أن تأخذ في الحسبان النقص والخلل الموجود في العلاقة بين المطور والمستهلك ومن ثم فإن المطور لا يمتلك المعرفة الكافية ببيئات العتاد "Hardware" الخاصة بالعملاء أو أنظمة التشغيل التي تم تركيبها بالفعل في

حاسبات العميل ويمكن أن يتسبب النقص في إدارة بيئة المستخدم في إحداث مشاكل خطيرة وربما يعاني المستخدم من صعوبة شديدة في تنصيب البرنامج لأن بيئاتها تختلف عن البيئة التي تم فيها تطوير البرنامج.

iii. **أما سيناريو التوصيل الثالث:** فيتم فيه جعل البرنامج ملائماً لاستخدام العميل إلكترونياً عن طريق وسيلة ما مثل الإنترنت. وكما في السيناريو الثاني بالإضافة إلى جميع المشاكل التي تم وضعها سابقاً في هذا القسم والتي توجد مشاكل مشابهة لها أيضاً نتيجة التنوّع في أنظمة التشغيل وبيئات التطبيق.

ويوجد مشكلتان جديدتان تقعان في السيناريو الثالث للتوصيل، فتشمل تحصيل الأموال المستحقة عن البرنامج وحماية الحاسب المضيف والذي يستخدم كخادم "Server" للبرنامج ويتم مناقشة الدفع الإلكتروني للبرنامج بواسطة الخبراء عبر التجارة الإلكترونية. يحتوي الخادم"Server" على البرنامج وسوف يحتوي على بعض المنافذ باستخدام برامج نقل الملفات مثل: "FTP" أو بعض البرامج الأخرى الناقلة للملفات، ويجب تحديد أي السيناريوهات هو الأفضل والأسهل في الاستخدام.

ولهذا السبب فان الخادم"Server" سريع التأثر لذا فإنه يجب اتخاذ القرار المناسب في الوقت المناسب بخصوص العلاقة بين حجم المخاطر المتوقعة في النظام ككل وكيفية إيجاد المدخل الملائم. واستخدام مثل هذه الوسائل الأمنية مثل مقاومة الفيروسات أو ما يطلق عليها الجدر النارية (Firewalls) ينصح به علي نطاق واسع ويمكنك أن تطلع على أحد الكتب الخاصة بأمن الحاسبات والمتاحة على شبكة الإنترنت للحصول على مزيد من المعلومات بشأن هذا الموضوع.

ولتحديد سيناريو التوصيل الملائم للتطبيق يحب علينا أن نقوم بتحديد التهيئة الملائمة واستخدامات ضغط وفك ضغط الملفات. ومن الضروري تقليل حجم الملفات لإمكانية وضعها في فلوبي دسك أو الأشرطة الممغنطة أو السي دي"CD". كما أن تقليل المرور عبر الانترنت يعد فكرة جيدة. وسوف نقوم بوضع قائمة توضح بعض الوسائل التي تستخدم عند اختيار التهيئة المناسبة عند توصيل البرنامج. ويحتوي جدول

1-7 على قائمة مختصرة من الكود الرئيس وغيره من الملفات النصية، ولكننا لم نأخذ في الاعتبار التهيئة والمستويات الخاصة مثل "JPEG"و "MPEG"و "TIF"و GIF" لتخزين الرسوم أو المواد غير النصية الأخرى.

نشاط

اختر بعض البرامج التي تستخدمها حديثاً، وصف كيف تم تسليم هذا البرنامج .

جدول 2-1 : بعض الوسائل التي تستخدم في ضغط وفك ضغط أو تخزين ملفات البرامج.

نظام التشغيل Operating System	الاستخدام Utility
يونيكس UNIX	tar
يونيكس UNIX	bar
يونيكس UNIX	cpio
العديد من البرامج MANY	Gzip
العديد من البرامج MANY	Gunzip
مايكروسوفت ويندوز MS- DOS،Microsoft Windows	Pkzip
ماكنتوش Macintosh	Binhex

نصائح عامة:

- من الضروري دائماً إعطاء المستخدم أو المنصّب فكرة عن حال الملفات حتى يمكن تغيير التصريحات.
- بيئة المستخدم مفيدة مؤقتاً لتسهيل البرمجة.
- من السهل إلى حد ما إنشاء ملف أو أرشيف وتحويله وهو أفضل من طلب المستخدم إرسال ملفات متعددة.

وكملاحظة نهائية فإن استخدام الأرشيفات والأشكال المضغوطة يمكن أن يقلل من حرية حركة الشبكة، لذا فإن من الواجب أن يكون ذلك محل اعتبار لدى المستخدمين.

أسئلة تقويم ذاتي

1/ اشرح سيناريوهات توصيل البرنامج "Software".

2. التنصيب Installation

تنصيب البرنامج لا يأخذ وقتاً طويلاً، مثلاً تنصيب نظام التشغيل غالباً ما يكون عملية متسلسلة (bootstrapping) جزء من نظام التشغيل (mini-kernel) يحمّل أولاَ وبعد ذلك يستخدم هذا الجزء في تنصيب الجزء الثاني (the entire kernel and the "root file system") المتعلق بالنواة وأنظمة الملفات. كما أن الجزء الثاني يستخدم لتنصيب الأجزاء الإضافية مثل (complete/usr file system) وما إلى ذلك. وبسبب القضايا التي أشير إليها سابقاً في هذه الوحده والمتعلقة بالفشل الكبير خلال التسليم فإن تنصيب البرنامج الناجح يعتمد غالباً على معرفة دقيقة للبرنامج الذي تم تنصيبه في النظام ويجب ملاحظة التناقضات خاصة عند عمل الترقيات للبرامج الرئيسية.

إن أفضل تنصيب للبرنامج يشمل تنصيب سجل الأداء log الذي يساعد على حل مشاكل التوقف. نحن لا نوضح الفكرة نفسها ، فإذا لم تكن كالجزء المحدد ، فإنها

تستعمل مرافق كالإعدادات المنصبة في نظام ميكروسوفت ويندوز NT،98،95 ، أيضا الـ VISE المنصبة في نظام ماكنتوش Macintosh . هذه المرافق في كثير من الأحوال تُحدَّث مع تهيئة ملفات النصوص.

نشاط

قم بتنصيب برنامج على نظام كمبيوتر، وحلل عملية التنصيب، واكتب كل اختيار نفذته استجابة لما أعطيت خلال التنصيب .

3. التوثيق Documentation

حتى أن أفضل برنامج ذو واجهة مستخدم بديهية يحتاج إلى عمل وثائق. **ومـــن أشـــكال الوثائق المعروفة:**

- وثائق داخلية (ضمنية داخل ملفات المصدر).
- وثائق خارجية في شكل متطلبات ووثائق التصميم.
- وثائق إضافية تشرح سبب اتخاذ قرارات التصميم مثل منطقية التصميم.
- الكتيبات ، مثل المستخدم ، العمليات، أو كتيبات التثبيت.
- المساعدة المباشرة وهذه جزء من البرنامج نفسه.

و كل هذه الأشكال من الوثائق لها مكان محدد في عملية تطوير البرنامج. حيث إن هـــذه الأشكال مختلفة جدا، وسوف نناقش كلاً منها في جزء منفصل.

ولا يوجد موضوع ينطبق على أكثر من نوع من الوثائق. إن كثيراً من البرامج طويلـــة المدى لها إصدارات مختلفة. بل توجد هناك مشكلة بالنسبة لكتاب الوثائق وهـــي كيفيـــة ترتيب تلك المواد في هذه الحالات. فالاختيارات إما أن تكون بين وثيقة أساسية مفـــردة وذات إصدارات متعددة توضح التغييرات في آخر إصدار ، و لها وثائق جديدة كلية لكل إصدار. فالطريقة الأولى أقل تكلفة في الطباعة ، ولكن ممكن أن تشكل مشكلة في الفهم. والاختيار الآخر يجعل الوثائق أسهل في الفهم لكن من السهل أن تملئ كل رف متاح في المكتب.

1.3 التوثيق الداخلي Internal Documentation

إن مصطلح " التوثيق الداخلي" يشير إلى الوثائق الموجودة داخل ملفات المصدر (Source Code) نفسها. و ربما يقتصر التوثيق الداخلي على التعليقات الموجودة داخل ملفات المصدر نفسها. و من المعتاد، أنه يوجد ملفات للقراءة (Readme File). من الطبيعي، أن مقاييس كتابة الكود (Coding) لمنظمة لها أولوية أكثر من أي جمل عامة أخرى تتعلق بالوثائق الداخلية في هذا الكتاب، و بالرغم من ذلك، ***من الممكن عمل بعض الجمل العامة عن إصدارات الوثائق الداخلية:***

① إن معظم المنظمات، وخاصة ما إذا كانت تصمم برامج لإعادة استخدامها أو لصيانتها، تتطلب كميات كبيرة من الوثائق.

② ليس من العملي أو المرغوب فيه عمل توثيق لكل جملة (Statement) بالبرنامج بتعليق (Comment).

③ إن تعريفات مثل تعريف التسمية للملفات من الممكن أن تساعد في عملية التوثيق.

إن النقطة الأولى من الممكن أن تسبب بعض الارتباك لمهندسي البرامج الجدد. إن العديد من مقاييس البرامج تتطلب ملفات مقدمة لكل ملف مصدر (source code). وبالتأكيد كل جزء من الملف المصدر (source code) أو كل فرع كبير للتحكم يتطلب تعليقاً إضافياً.

أما النقطة الثانية، هي عن الوثائق التي لا يحتاج كل سطر فيها إلى تعليق. و عامة، الوثائق غير ذات معنى غير مفيدة.

```
  I  = I +1; /* add one to the variable I */
Temp = x;    /* assign value of x to temp */
X = y;         /* assign value of y to x */
Y = temp;   /* assign value of temp to y */
```

و هناك طريقة أفضل للتعليق على هذا الكود مثل:

```
I = I + 1;
/* swap x and y */
temp = x;
x = y;
y = temp;
```

أما النقطة الثالثة، هي تأثير عملية تسمية الملفات في الوثائق. حتى فحص عابر للدليل وأسماء الملفات غالباً ما يكون فعالاً لمحو مصدر العديد من البرامج.

و هناك تعليق أخير يتناقله المبرمجون: " **لو أن الوثائق وملفات المصدر لم تتفق، فكلاهما خطأ**". ضع هذا في الاعتبار عند عمل وثائق للكود المصدر (source code). فنحن نرى الوثائق من منظور متكامل لعملية تطوير البرامج. إن محاولة عمل وثيقة للكود عن طريق كتابة تعليق في آخر لحظة، تبدو من أهم الأسباب في مشاكل عدم التكامل. إلى جانب، أن عمل توثيق للكود بينما تكتبه يجعل تطويرك له أكثر كفاءة لأنك تستطيع معرفة آخر شيء عملته.

2.3 التوثيق الخارجي External Documentation

يشمل التوثيق الخارجي متطلبات ووثائق التصميم التي لم تعطى بشكل عام لمستخدمي النظام وأدلة التدريب العامة. يجب أن تنتج كل المتطلبات ووثائق التصميم وتكون متاحة للمديرين والمراقبين الفنيين لمشاريع تطوير البرنامج. يمكن أن تكون للوثيقة صيغة كتابية وصيغة الكترونية. وإذا ما استخدمت النسخة الإلكترونية فإن الوثائق يمكن أن تخزن في شكل من أشكال برنامج معالج للكتابة والذي ينتج مثل هذه الوثائق. أو في شكل عرضي موجه مثل المخطوطات السريعة أو شكل الوثيقة المحمول.

إن غالبية هذه الوثائق الخارجية متفق عليها بواسطة مجموعة تعريفه باسم مكتب النشر الفني وهو مسؤول عن خلق وتشكيل عمليات التوثيق الخارجي والتي ربما بدأها الآخرون والبرهنة على عملية التوثيق وفحصها كي تكون متسقة فنياً. والأعضاء

الأكثر أهمية في فريق النشر الفني هم الكتاب الفنيون. فلديهم خبرة في تنظيم الوثائق ووضعها في مختصر إنجليزي قياسي. المهم أن يكون هناك كتاب فنيون يعملون في توثيق منتجات البرنامج من البداية إلى تاريخ إصدار المنتج كي يقوموا بأي عملية إعادة تنظيم للتوثيق إذا استدعى الأمر.

إن التوثيق الخارجي لنظام برنامج هو بطبيعته منفصل عن الرمز المصدري الموجود في نفس النظام. لهذا توجد مشكلة رئيسية في الاتساق بين أشكال مختلفة من التوثيق. هناك ميزة كامنة في أدوات "كيس" وهي القدرة على تكوين آراء مشتركة عن البرنامج وعملية توثيقه.

نشاط

افحص مغلف برنامج متاح تجارياً ، أي أنواع التوثيق كان متاحاً مع البرنامج؟

3.3 الأسس المنطقية للتصميم Design Rationales

هناك العديد من أنظمة البرامج الأكثر انتشاراً لها أسس تصميم تحاول أن تشرح الفكرة وراء عديد من القرارات المتخذة في تصميم الأنظمة. أحياناً تساعد هذه الأسس على إيضاح أنواع الغموض الناتج بعد استخدام النظام.

4.3 عملية التثبيت، تدريب المستخدم وكتيبات التشغيل

User Training and operations Manuals،Installation

إن الهدف من كتيب التثبيت يجب أن يكون واضحاً، فأي معلومات خاصة بالوضع النسبي للنظام يجب أن تكون متضمنة في هذا الكتيب.

إن كتيب المستخدم مخصص لاستخدام النظام، وبشكل عام فإن مثل هذه الكتيبات تكن مختصرة وموضحة لقدر من الأوامر البسيطة التي يمكن استخدامها لبيان معظم وظائف النظام. ويمكن كتابة وظيفة إضافية في كتيب ثان أطول. فالعديد من

أنظمة البرمجيات تأتي مع كتيب بسيط لتشغيلها وكتيب أكثر تفصيلاً لإنجاز العمليات اليومية.

عندما يكون البرنامج معقدا بدرجة كبيرة أو يتحكم في أنظمة أنظمة أمان حاسمة تكون هناك ضرورة وجود كتيبات منفصلة للتدريب وتشغيل البرنامج. ولسنا بحاجة لأن نكرر أنه يجب تجريب كتيبات التدريب والتشغيل بواسطة الناس العاديين ومن الأفضل أن يجرب بواسطة المستخدمين أنفسهم. كذالك سيكون هناك مراقب فنيُ من أطراف العقد مسئول عن تأكيد أن الكتيبات بها قدر كاف من المعلومات.

5.3 التوثيق الإلكتروني On-line Documentation

يشير مصطلح التوثيق الإلكتروني إلى المساعدة التي قدمت للمستخدمين كجزء من البرنامج نفسه. و كذلك فإن القرارات الخاصة بالتوثيق الإلكتروني يجب أن تؤخذ قبل التوثيق النهائي للنظام البرمجي.

إن منشأ طريقة التوثيق الإلكتروني يجب أن يختار بين تطوير نظام تطبيق نوعي مساعد وبين استخدام هدف عام واحد مع المعلومات المحددة لتطبيق معين مشار إلية بالأهمية. لقد لاحظنا أن القدرة على استخدام الهدف العام ونظام المساعدة الإلكتروني يقدم ميزة تنافسية هائلة لصناع البرامج المكتبية مثل ميكروسوفت أوفيس. ووضع كلاريس ووركس أو لوتس سمارت مجموعة من وحدات معالجة نظام الكتابة، وألواح الورق الممتدة، وأنظمة قاعدة البيانات وبرامج البريد الإلكتروني.

❐ تكون المساعدة الإلكترونية للبرنامج بأحد هذين النموذجين

On-Line help for software can be one of two forms

❶ مجموعة صفحات من الوثائق (ترتبط غالباً في نماذج النصوص الكبيرة) يمكن أن تظهر إما من القائمة المرئية دائماً أو من استخدام مفاتيح معينة.

❷ عامل ذكي أو ممتاز يتفاعل مع المستخدم يمكن أن يحدث أوتوماتيكياً عند حدوث مجموعة معينة من الأعمال.

في الحالة الأولى تكمن الصعوبة الأساسية في التأكد من أن حالة المستخدم بالنسبة للحساب والتوثيق هي نفسها بعد استخدام المساعدة. وعند الوصول لقائمة المساعدة يتوقف عمل المستخدم ويقوم بحفظه على الأقل في مكان مؤقت ثم يستكمله عند الانتهاء من استخدام قائمة المساعدة.

ونفس ذلك يكون في حالة العامل الذكي. وهنا يوجد تطبيق إضافي وهو: إن العامل هو بمراقبة تقدم المستخدم إضافة إلى مساعدته. وهذا يضاعف من تعقيد وتصميم البرنامج.

وفي كلتا الحالتين يعتبر أهم جزء في ميكانيكية المساعدة أنها تدل المستخدم أو المستخدمة عمّا يريد أو يحتاج إلى معرفته.

6.3 مستويات القراءة Reading Levels

كيف تستطيع إن تقول أن عملية التوثيق كافية؟ إن أهم شيء هو أن تجرب عملية التوثيق على أناس لم يستخدموها من قبل. والفكرة هي أن يسترجع المجرب للبرنامج ما قرأه أو رآه، وليس ما عرفة عن تصميم النظام.

إن المستوى العام هو قياس جيد لاستخدام التوثيق. وبشكل عام فإن مستويات القراءة المنخفضة أفضل خاصة إذا قرأ التوثيق شخص من غير مطوري البرنامج. وربما يكفي مستوى القراءة الأعلى لتوثيق نظام جهاز المستخدم الشخصي.

لاحظ أنه مع ذلك يمكن أن تكون هناك حاجات إضافية حتى بالنسبة للمستخدم الأكثر تطوراً. على سبيل المثال هناك حاجة عامة إلى معرفة مفصلة عن مجال التطبيق كي تساعد على فهم الكثير من التفاصيل الفنية لنظام قابل لإعادة الاستخدام إذا أريد استخدامه في أي مكان.

وتعتبر اختبارات "الكنكيد " ومستويات القراءة الأخرى مفيدة في هذا السياق. فهي متاحة في كثير من المصادر ومنطمرة في حزمة برامج كبرنامج "مقعد عمل الكاتب" والمتاح بشكل عام في نظام ألـ " أي تي أند تي " والـ "في يونيكس" وبعض أشكال الـ"يونيكس" الأخرى.

إن طريقة " مقعد عمل الكاتب" تقوم بما يقوم به كل نظام كامل وحديث لمعالجة النصوص. يقوم بكشف الأخطاء الإملائية و الكلمات المكررة و الترقيم والبناء النحوي.

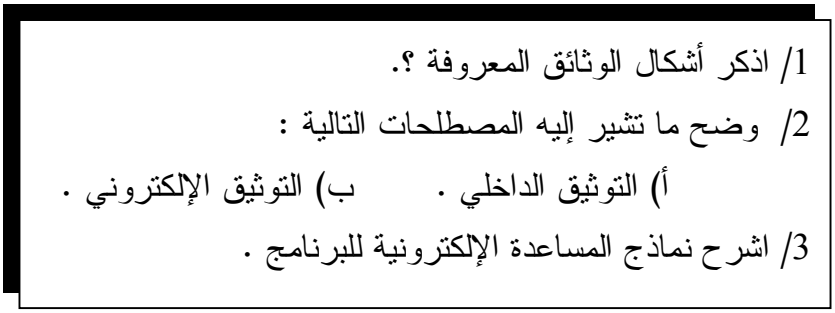
إضافة إلى أن هذا النظام له سمات لا توجد في أنظمة معالجة النصوص التجارية. وتشمل المقاييس المحسوبة بواسطة طريقة "مقعد عمل الكاتب" بوجه عام على: مستوى القراءة و عملية تقييم القيم المتصلة ببعض الوثائق المتفق عليها كأمثلة للكتابة الواضحة. وتشير طريقة " مقعد عمل الكاتب" إلى تعقيد الجملة العادية مثل الكثير من الجمل المركبة والأخرى شديدة التعقيد، والجمل السهلة الموضحة التي تجعل الوثيقة متقطعة. وتلوح هذه الطريقة أيضاً بالوثائق التي تحوي كمية كبيرة من الكلام الغير مباشر.

قامت شركة "هاريز" بتنفيذ مشروع حديث خاص بمستويات القراءة وكان ذلك في وزارة الدفاع الأمريكية. كانت وجهة المشروع نحو تقليل الفجوة بين متوسط القراءة للمجندين ومتوسط قراءة كتيبات الدلائل الفنية. و انقسم حلهم لهذه المشكلة إلى شقين: تحسين مستوى قراءة المجندين بواسطة تدريب متخصص، و تقليل مستوى قراءة كتيبات الدلائل بواسطة طرق لغوية معينة لخفض مستوى القراءة.

ويجب أن تكون حريصاً عندما تستخدم إجراءً منفرداً من مستوى القراءة كعامل محدد لإمكانية قراءة الوثيقة. فهناك الكثير من إجراءات تحديد إمكانية القراءة مثل معرفة

القارئ للموضوع العام، المفردات الفنية، متوسط طول الكلمات، عدد الجمل البسيطة والجمل المركبة ونسبة الصفات والظروف.

أسئلة تقويم ذاتي



1/ اذكر أشكال الوثائق المعروفة ؟.

2/ وضح ما تشير إليه المصطلحات التالية :

أ) التوثيق الداخلي . ب) التوثيق الإلكتروني .

3/ اشرح نماذج المساعدة الإلكترونية للبرنامج .

4. وجهة نظر المدير المنفذ في التسليم ، التنصيب والتوثيق

A Manager's View of Delivery Installation and Documentation

في كثير من الأحيان يكون منظور المدير حول عملية النقل والتثبيت هو نفس منظوره حول فعاليات دورة حياة البرمجيات، وهو تجنب المخاطرة. ومع ذلك فإن هناك عاملاً إضافياً يمكن أن يحدث في هذه المراحل: الضعف والمفاجآت التي ممكن حدوثها في هذه النقطة هي غير ساره بالمرة. بالتأكيد يتوقع المدير أن عملية النقل والتثبيت فحصت كليةً كي يسهل اكتشاف أي مشكلة. ولقد لاحظنا أن العديد من منظمات تطوير البرمجيات تستخدم أفراد غير مثقلين فنياً عند تجريب مواد عملية التثبيت. والهدف من ذلك هو التأكد من أنه لم يُحدِثْ المطورون كثيرو التعامل مع البرنامج أية فرضيات غير منصوص عليها في الضمان.

5. التسليم والتنصيب والتوثيق لمشروع برنامج رئيس

في هذه المرحلة أول خطوة للمشروع الرئيس للبرمجيات هي نقل البرنامج. دائماً لابد من استخدام متطلبات البرنامج لكي نحدد البيئة التي سيتم النقل بواسطتها.

كيف نحدد البيئة المثلى للنقل؟ الإجابة بسيطة – اقرأ متطلبات النظام. متطلبات النظام لديها العديد من المدخلات المناسبة لعملية النقل.

من الواضح أن المتطلبات غامضة بشأن عملية النقل. نحن نعتقد، مهما يكن بأن النظام لابد من نقله على قرص مرن واحد 1.44 Floppy disk وألا يتم استخدام أداة ضغط (ملفات). بما أنه لا توجد إشارة إلى المجمع (المصنف) تكون مسؤولية المطور لكي يحدد البيئة والمجمعات التي من المتوقع أن يستخدمها المستهلك. عدم تقديم المشورة للمستهلك في هذه المرحلة وحده قد يؤدي إلى كارثة.

كما فعلنا في كل حدث من الأحداث الأخرى الهامة، لابد من عمل مراجعة حالة من منظور إدارة المشروع حول: هل كنا في قمة الجدول (وهذا بعيد الاحتمال)، أو كنا في مؤخرة الجدول (وهذا محتمل)؟ أو كنا تقريباً متواجدين؟. هل حدثت هناك أي مفاجئات غير سارة أو كانت هناك أية أجزاء من النظام أكثر صعوبة مما توقعنا؟. هل تطلب أي جزء من النظام عناية زائدة، ربما مصادر إضافية؟ هل تقدّمُ العلم التطبيقي أو ضغط السوق أي جزء من النظام القديم؟

في الواقع قد فعلنا الكثير. إن عملية نقل المشروع تعد فرصة ممتازة لتحليل جاد. فالدروس الموجودة في هذا المشروع يجب أن يتناولها مديري المشاريع الأخرى بجانب من النقاش وكذلك المديرو رفيعو المستوى. إذا كان هناك أمل في تحسين طريقة معالجة النظام للتطور فإن البيانات الخاصة بتكلفة المشروع وجدول الأعمال والنوعية يجب أن تدخل في قاعدة بيانات كي تخضع لطريقة أبعد في التحليل.

جدول 2-2

جزء يوضح تتبع المتطلبات اللازمة للمشروع الرئيس للبرمجيات

الرقم	المتطلبات	التصميم	الرمز	التجريب
1	انتل			
2	ويندوز 95			
3	ويندوز 95 يو أي			
4	متسق مع الإكسل 4.0			
5	نظام حجم واحد فقط			
6	نظام ا ميجابيت.			
7	مساحة خالية على القرص مقدارها 1 ميجابيت			
8	قرص مرن 1.44 ميجابيت			
9	يتضمن التثبيت			
10	لا توجد صلاحية لإزالة ضغط(الملفات)			

الخلاصة

* بينت الوحدة إن البرنامج غير نافع ما لم يسلم للمستهلك ويمكن تحميله على كمبيوتره.كما بينت الوحدة أن تسليم البرنامج مختلف. فهو يعتمد على ما إذا سيتم تسليمه لمستهلك واحد أو عدد من المستهلكين يستخدمون الوسائط المتعددة أو على وسط توزيع مثل الإنترنت.

* وتناولت الوحدة بالشرح سيناريوهات توصيل البرنامج "Software" :

أ) إذا كان هناك مستخدم واحد أو عدد قليل من المستخدمين فإن البرنامج يتم تسليمه على شكل شريط مغناطيسي "ديسكات" أو اسطوانات "سي دي" مع الالتزام التام بمتطلبات المشروع.

ب) إذا كان يوجد عدد كبير من المستخدمين للبرنامج مع القيام ببيع البرنامج في المحلات التجارية بواسطة آلية البيع بالتجزئة فإن البرنامج غالباً يتم توصيله في أقراص مرنة "Diskettes" أو اسطوانات "CD" طبقاً للشكل الذي تطلبه المحلات التجارية ويكون فريق توصيل البرامج مسئولاً عن نسخه لضمان توزيعه بالشكل المناسب كما يقوم فريق التوصيل بعملية التنصيب.

ت) إذا كان يوجد عدد كبير من المستخدمين للبرنامج يتم توصيله إلكترونياً باستخدام البرنامج الناقل "FTP" أو البرامج المشابهة الأخرى فلا يوجد هناك داعٍ لعمل نسخ متعددة من البرنامج وفي هذه الحالة فإن مسؤولية فريق التوصيل هو التأكد من أن البرنامج يمكن أن يتم الحصول عليه بشكل مناسب.

* بينت الوحدة إنه بغض النظر عن طريقة التسليم فإنه يجب تلافي المشكلات التالية قبل التسليم: فشل الأداة المادية والتي يتم فيها تثبيت البرنامج، وضع غير صحيح في الأدلة مما ينتج عنه خطأ في الأسماء، استخدام غير صحيح للأسماء الموجودة في الأدلة الفرعية مما يجعل من الأنظمة الفرعية أنظمة غير ملائمة، تصريحات دخول غير

صحيحة مما يصعب الدخول إلى الملفات، تهيئة غير صحيحة للبرنامج مثل استخدام أدوات تبادل غير سليمة لضغط أو فك الملفات، توثيق غير سليم لتعليمات التوصيل.

* بينت الوحدة إنه عندما يتم تسليم البرنامج بالصورة الصحيحة يجب تنصيبه، وعادة خطوات التنصيب تكون بتتبع طرق التنصيب وفقا لما ورد بكتيب التنصيب وخياراته.

* بينت الوحدة إن تنصيب البرنامج الناجح يعتمد غالباً على معرفة دقيقة للبرنامج الذي تم تنصيبه في النظام ويجب ملاحظة التناقضات خاصة عند عمل الترقيات للبرامج الرئيسية.

* بينت الوحدة أشكال الوثائق المعروفة وهي : وثائق داخلية (ضمنية داخل ملفات المصدر) ووثائق خارجية في شكل متطلبات ووثائق التصميم، وثائق إضافية تشرح سبب اتخاذ قرارات التصميم مثل منطقية التصميم، الكتيبات ، مثل المستخدم ، العمليات، أو كتيبات التثبيت والمساعدة المباشرة وهذه جزء من البرنامج نفسه.

لمحة مسبقة عن الوحدة التالية

عزيزي الدارس، الوحدة التالية تأتي بعنوان" صيانة البرمجيات " الوحدة تتطرق لمفهوم صيانة البرمجيات. وفيها تجد عدد من التعريفات الخاصة بصيانة البرمجيات ، العوامل التي توجب صيانة البرمجيات .الوحدة ستبحث في أهم أنماط الصيانة وحيثيات كل منها.آمل أن تجدها وحدة مفيدة.

مسرد المصطلحات

- **التوثيق الداخلي Internal Documentation**

يشير إلى الوثائق الموجودة داخل ملفات المصدر (Source Code) نفسها. و ربما يقتصر التوثيق الداخلي على التعليقات الموجودة داخل ملفات المصدر نفسها.

- **التوثيق الخارجي External Documentation**

يشمل التوثيق الخارجي متطلبات ووثائق التصميم التي لم تُعط بشكل عام لمستخدمي النظام وأدلة التدريب العامة.

- **التوثيق الإلكتروني On-line Documentation**

يشير مصطلح التوثيق الإلكتروني إلى المساعدة التي قدمت للمستخدمين كجزء من البرنامج نفسه.

المراجع :

- www.csse.monash.edu.au/.../html/text.html
- www.ee.unb.ca/.../IntroToSoftwareEng.htm
- http://elearning.tvm.tcs.co.in/SMaintenance/SMaintenance/sm_contents.htm#top
- www.cs.iit.edu/~icsm2004/
- http://hebb.cis.uoguelph.ca
- http://bcs.org
- www.programs-transformation.org
- www.dur.ac.uk/csm/
- www.info@nidam.net


الوحدة الثالثة
صيانة البرمجيات
Software Maintenance

محتويات الوحدة

المقدمة

تمهيد

عزيزي الدارس،

مرحباً بك في الوحدة الثالثة من مقرر " **هندسة البرمجيات 2** "، والتي تحمل العنوان " **صيانة البرمجيات** ". القسم الأول من الوحدة يتطرق لمفهوم صيانة البرمجيات وفيه تقدم الوحدة عدد من التعريفات الخاصة بصيانة البرمجيات. كما يستعرض القسم عدداً من العوامل التي توجب صيانة البرمجيات .القسم الثاني من الوحدة يتناول الصيانة التصحيحية للبرمجيات حيث نتعرف على الإجراءات التي يجب أن تتخذ عندما يتم تحديد المشكلة . القسم الثالث من الوحدة يبحث في الصيانة المهيأة للبرامج . القسم الرابع من الوحدة يتناول الصيانة الوقائية للبرامج ومشكلة عام 2000 ، وكذلك العديد من السيناريوهات والتي ربما تحدث في المستقبل مشاكل . القسم الخامس يتناول كيف تقرأ المتطلبات ، التصميم ، وشفرة المصدر . كما توضح الوحدة وجهة نظر المديرين في صيانة البرمجيات، وكذلك الصيانة لمعظم مشروعات البرامج .

أهداف الوحدة

عزيزي الدارس، بعد فراغك من دراسة هذه الوحدة ينبغي أن تكون قـــادراً على أن:

- **تعرِّف** مصطلح صيانة البرمجيات .
- **تشرح** أهم أنماط الصيانة وحيثيات كل منها .
- **تكتب** محتويات نموذج طلب صيانة البرامج.
- **تكتب** محتويات نموذج استجابة لطلب صيانة البرامج.
- **تكتب** محتويات النموذج المزدوج لكل من طلب الصيانة والاستجابة.
- **يقرأ** المتطلبات ، التصميم ، و شَفْرَة مصدر؟
- **توضح** وجهة نظر المدراء في صيانة البرمجيات.
- **تذكر** السيناريوهات التي ربما تحدث في المستقبل فيما يرتبط بصيانة البرامج.
- **تلخص** التفاصيل المحتواه في الرسومات والإحصائيات المشتمله في الملحق1 عن الصيانة والرؤى المختلفة بخصوصها.

توطئة

عزيزي الدارس، سوف نبحث في هذه الوحدة عن أهم مرحلة في مراحل هندسة البرمجيات والتي تتم بعد تسليم البرمجية وهي الصيانة. وقد تندهش من معرفة أنه للعديد من النظم الطويلة الأمد تشكل الصيانة ما يزيد على 75% من إجمالي تكلفة البرمجية طيلة دورة حياتها.

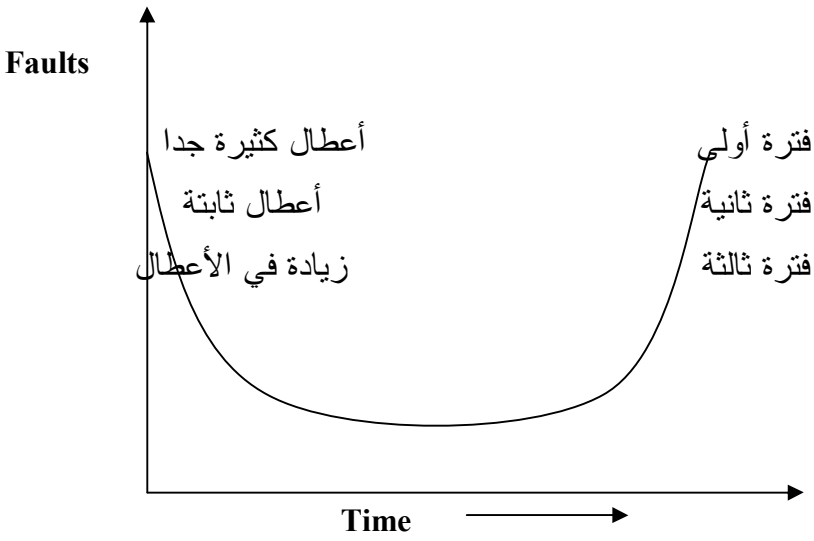
1. مفهوم صيانة البرمجيات

للوهلة الأولى يبدو من الغرابة استخدام المصطلح (صيانة) على البرمجية. فالبرمجية لا تتلف بنفس طريقة تلف مكونات الحاسب الآلي أو كما في الأجهزة الأخرى. فالوسيلة المخزن فيها البرمجيات تتغير بمرور الزمن لأن الجسيمات المغناطيسية تتلف سطح الديسك (القرص) أو الشريط أو حتى القرص المدمج. لذا فإن إداري النظم الحريصين يحتاطون بعمل نسخ احتياطية لكافة البرمجيات الضرورية والهامة (أو لملفات المستخدم) ، والأفضل حفظها في أماكن آمنة لتفادي مشاكل كالحريق والفيضان.

ومن الجدير بالذكر بأن البرمجيات لا تتآكل أو تتلف بسبب الغبار الذي يصل إلى منتصف الوصلة الكهربائية، كما أن التغيرات في الطاقة الكهربائية الموصولة بالحواسيب غير محببة (110 فولت 60 سايكل تيار متردد في الولايات المتحدة ، و220 فولت في المملكة المتحدة) . وبالطبع إن الدفق بالطاقة أو انخفاض الفولطية بسبب مرافق إنتاج الطاقة الكهربائية قد يؤدي إلى إتلاف البرمجيات، إلا أن حيثيات هذه المشاكل غير واضحة تماما.

ويستخدم إداريو نظم الحاسوب مجموعة من أجهزة ضبط الطاقة ومصادر طاقة منظمة ومصادر طاقة مستديمة لضمان المستوى المناسب للطاقة. فهذه المشكلة مرتبطة كما هو واضح بالأجهزة ولا تفرض أي مشكلة محددة على مهندسي البرمجيات.

وصلاحية نظم الأجهزة تتبع التوزيع المبين في الشكل 3-1



(شكل 3-1)

شكل 3-1 يبين توزيع صلاحية نظم الأجهزة. ويطلق عليه منحنى حوض الحمام للأعطال الأجهزة الكهربائية والميكانيكية.

والعدد المرتفع نسبياً للأعطال في وقت مبكر من استعمال الأجهزة يرجع بالأساس إلى ثلاثة عوامل وهي: المكونات التالفة – أخطاء في تركيب هذه المكونات – وسوء الاستعمال من قبل مشغل الحاسوب غيرالمتمرس ، فالعدد المتزايد من الأعطال في الفترة الثالثة من الرسم البياني يمثل الأعطال التي تحدث عند نهاية العمر الافتراضي للأجهزة وإلى الأعطال في المكونات المركبة فعلياً في تلك الأجهزة.

وصيانة الأجهزة تختلف عن اختبار الأجهزة. فمثلاً تتوقع بأن تكون السيارة الجديدة التي اشتريتها حديثا قد جمعت على يد خبراء وباستخدام قطع عالية الجودة، وبها نظم فرعية معينة (مثل تلك الخاصة بالنظم الكهربائية والمقود، والفرامل وغيرها) وبأنه تم تجربيها للسلامة. كما يجري على السيارة تدقيق آخر أثناء إخراجها من خط التصنع للناقل وبعدها يتم قيادتها من الناقل إلى معرض الموزع أو البائع. فكل هذه تعد أنشطة اختبار

للسيارة. وفي حال حاجة السيارة الجديدة إلى إصلاح فراملها أو المقود خلال فترة الضمان، فإنك على الأرجح تعتبرها من النوعية السيئة وبالتأكيد لن تنصح أحد أصدقائك بشراء نفس الموديل من تلك السيارة.

وكما أنك تهتم جداً بنوعية البلاستيك المصنوع منها تابلوه السيارة أو الصدام وخاصة عندما يحدث اصطدام خلفي. ومن ناحية أخرى إذا ما واجهتك نفس المشاكل (التي لا ترتبط بالسلامة) بعد قطع السيارة لمسافة مائة ألف ميل فإنك ستعتبر على الأرجح إصلاح هذه المشاكل كنوع من الصيانة العادية. وقد تفضل حتى عدم إصلاح هذه المشاكل لشعورك بأن العمر الزمني المتبقي للسيارة ما يزال قصيراً بحيث أن تكاليف الصيانة لا شيء يذكر.

فالملاحظة والتقييم الشخصي هنا هو الأرجح . فلو افترضنا بأن متوسط تكلفة السيارة الجديدة بالدولار الأمريكي هو 20 ألف دولار فستكون التكلفة الشهرية لعمر افتراضي للسيارة خمس سنوات يقارب 400 دولار شهرياً ، وعلى افتراض بوجود دفعة أولى بسيطة أو عدم وجودها، وكذلك استيفاء فائدة منخفضة جداً، لذا فإن قراري مثلاً يستند على تقدير قيمة الإصلاحات واستخدام السيارة ، فإذا ما كانت الإصلاحات بهدف أن تكون السيارة قادرة على العمل بأمان لمدة شهرين إضافيين ، فإنني أتوقع بأن لا تتجاوز تكلفة الإصلاحات عن 800 دولار أمريكي، وما عدا هذه النقطة عليّ التساؤل إذا ما كانت السيارة تستحق ذلك أم لا. ومن ثم أقوم بعمل التحليل الغير رسمي للتكلفة/ والمنفعة لتحديد إذا ما يتوجب صيانة السيارة أو إذا يجب التبرع بها لبرنامج إصلاح السيارات لإحدى المدارس الصناعية (كي أحصل على خصم في ضريبة الدخل) أو تباع بعدها للموزع أو ترمى في النفايات.

فأعمال الصيانة الصحيحة للأجهزة الحاسوبية يشتمل على المحافظة على نظافتها، والاهتمام على وجه الخصوص بالقطع المتحركة واستبدال المكونات التالفة، والجدولة العامة لاستبدال المكونات القديمة والتالفة أو على وشك التلف. كما يلزم أجراء تحليل

التكلفة / المنفعة لتحديد إذا ما كانت جهود الصيانة مكلفة وضمن أهداف المنشأة. ومـــن المدهش أن هذه الطريقة أسلوبا جيداً لدراسة واعتبار صيانة البرمجيات.

وهناك العديد من التعريفات الخاصة بصيانة البرمجيات نسرد أهمها:

- تشمل على عمليات تكويد البرامج بعد تسليمها للمستخدم لتصحيح أخطاء التكويد بغرض تحسين أداء البرمجية وللوصول للأداء المثل لها.
- هي إحدى مراحل عمليات هندسة البرمجيات وتشمل تحسين أداء البرمجية ووصولا للأداء الأمثل بعد تسليمها للمستخدم إضافة لمعالجة مشاكلها التي تطرأ أثناء الاستخدام.

وإجمالا توصف صيانة البرمجيات بأنها العملية النظامية لتغير البرمجيات المستخدمة فعلا وذلك لمنع حدوث أعطال في النظام ولتحسين الأداء. وتشتمل عملية صيانة البرمجيات على المحافظة على جعل شاشات البرمجيات سهلة ومعياريه لتفاعل المستخدم مع الاهتمام بالأعطال واستبدال المكونات التابعة والجدولة العامة لاستبدال المكونات القديمة المتهالكة والتالفة أو التي على وشك التلف. كذلك يتوجب الأخذ في الاعتبار العمر الافتراضي المقدر المتبقي للبرمجيات لتحديد إذا ما كانت جهود الصيانة تستحق التكلفة المضافة.

وبعد هذه المقدمة البسيطة للأسباب الموجبة لصيانة البرمجيات، سوف نناقش بالتفصـــيل الفرعيات المتعلقة بها

❐ هنالك عدة عوامل توجب صيانة البرمجيات وهي :

(There are several factors that require software to be maintained)

أ) تغير أنظمة الحاسبات (Computer Platforms) أو انها قد أصبحت متهالكة قديمة.

ب) تغير أنظمة التشغيل (Operating Systems) أو قد أصبحت ملغاة.

ت) تغير المترجمات (Compilers) أو أنها قد أصبحت ملغاة.

ث) تغير معايير اللغة (Language Standards) أو قد أصبحت قديمة.

ج) تغيير معايير الاتصالات (Communication Standards) أو أنها أصبحت قديمة.
ح) تغيير واجهات التداخل (Graphical User Interfaces) أو أنها قد أصبحت قديمة.
خ) تغير مجموعة البرمجيات الخاصة بالتطبيقات (Software Packages) أو أنها أصبحت قديمة.
د) التغيرات والتحسينات مع التطبيقات والأنظمة الأخرى.
ر) قد يكون في البرمجيات عيوب أصبحت بائنة فقط بعد استخدام البرمجية من قبل المستخدم ويتوجب إصلاح وتصحيح هذه الأعطال.
ز) العملاء بحاجة إلى مواصفات جديدة.
س) حاجة البرمجيات إلى تحديث كي تتوافق وتصبح منافسة في السوق.
ش) يجب منع الأخطاء الموجودة في البرمجيات الحالية أن تتكرر في النسخ الجديدة من البرمجية.

وتصنف هذه العوامل على أنها تنتمي إلى عدة فئات مميزة. فالبنود من 1 حتى 9 تصنف على أنها **صيانة تكيفيه** لأنها بهدف جعل البرمجية تتكيف مع التكنولوجية الجديدة. أما العوامل من 10 إلى 11 تصنف على أنها **صيانة تصحيحية** لأنها تتجه نحو جعل البرمجية أكثر دقة وسليمة بمعنى وجود أقل عدد من الأخطاء بها. وقد يصنف البند 12 على أنه **صيانة وقائية** لأنها تهدف إلى الحد من احتمالية حصول الأخطاء المستقبلية في البرمجيات.

ولكن هنالك خيط مشترك بينهما ، فخطوة أساسية في كافة أنواع الصيانة للبرمجيات أن الصيانة هي عملية تفهم نظم البرمجيات الواجب صيانتها ، فقبل شروع فني الصيانة في تغيير البرمجية لإصلاح الخلل أو لإضافة ميزة عليها؛ عليه أن يفهم البرمجية ككل والمنظومات التي بحاجة للتعديل. وقد دلت العديد من التجارب أن فهم متطلبات العميل والتصميمات يأخذ حوالي نصف كافة جهود الصيانة.

وسوف يتم الحديث عن الطرق المختلفة لصيانة البرمجيات في الجزء المتبقي من هذا الفصل والتي تنصب على تحديد ماهية الخلل ثم كيفية تصحيحه.

❐ **فالعديد من مشاكل البرمجيات تقع ضمن واحدٍ أو أكثر من النقاط الآتية:**

(Many software problems can be traded to one or more of the following:)

- عدم وجود الالتزام بمعايير تطوير البرمجيات. وتشمل هذه المشاكل واجهات التداخل الضعيفة ما بين المكونات مثل المرور الخاطئ للمتغيرات بين الدوال.
- وجود أخطاء منطقية في مكونات البرنامج مثل حلقات التكرار (Loops) ، التفرعات أو حالات البرنامج غير المتوافقة.
- وجود أجزاء برمجية كبيرة وعدم اللجوء إلى المكونات الفرعية (الدوال- البريمجات الصغيرة).
- عدم وجود التوافق ما بين الاحتياجات، التصميم والتكويد وتوثيق النظام.

وغالبا ما يندهش الطلبة بمقدار الصعوبة في فهمهم أما للكود التجاري أو للأكواد للجهات الرسمية. وقد استخدمت فعليا كود المصدر من العديد من المؤسسات التي تطور البرمجيات وفي الفصول الدراسية التي أدرسها وكان التجاوب متشابها. فقد وثق الكود بأسلوب سيئ ويصعب فهمه. وفي الواقع فإن الطلبة يواجهون مشكلة عامة وهي دراسة كافة أو أجزاء من النظم التي تكون كبيرة كي تفهم من قبلهم.

أسئلة تقويم ذاتي

1/ ارسم منحنى يبين توزيع صلاحية نظم الأجهزة والبرمجيات مع الزمن.

2/ اسرد التعريفات المختلفة لصيانة البرمجيات واستنتج تعريفا جامعا لها.

3 ما هي أهم العوامل التي توجب صيانة البرمجيات؟

4/ اسرد أهم مشاكل البرمجيات كما وردت في سياق الوحدة.

5/ ما هي أهم أنماط الصيانة ؟

2. الصيانة التصحيحية للبرمجيات

Corrective Software Maintenance

إن أول خطوة في الصيانة التصحيحة للبرمجيات هو تحديد نوعية ما يراد صيانته ففي الصيانة التصحيحة للبرمجيات يبدأ العمل عادة في التعرف أولا على المشكلة الموجودة في البرمجيات المستخدمة. أما تحديد المنظومة المعنية المتسببة بالمشكلة فإنه جانبة صعبة. والطريق الأساسي يقوم على أسلوب وطريقة التجربة والخطأ. ويعتمد على معلومات فني الصيانة للجهاز ومعلوماته عن علم الحاسب الآلي وهندسة البرامج. وعندما يتم تحديد المشكلة يجب اتخاذ القرار عما سيتم وكيف سيتم حل المشكلة. وهـذه عادة تحتاج لإجراءات عديدة نوجزها **كما يلي:**

- ملاحظة وجود مشكلة.
- توثيق المشكلة.
- تحديد أهمية المشكلة.
- تحديد أولوية المشاكل لتحديد أولويات وأهمية إصلاحها.
- تحديد ماهية المشاكل التي لا يمكن إصلاحها نتيجة نقص الموارد.

- حل المشكلة.
- اختبار النظام لمعرفة إذا كان حل هذه المشكلة سيؤدي إلى مشاكل في أجزاء أخرى من البرنامج.
- توثيق الحل كجزء من شفرة المصدر.
- توثيق الحل في أشكال أخرى من الوثائق إذا كانت التغيرات التي أجريت على التصميم الأصلي للنظام.
- تحديث قاعدة البيانات للمعلومات حول أخطاء البرنامج.

وهذا يبدو أنه عمل كبير. وصيانة البرامج تحتاج إلى كميات كبيرة من العمل المكتبي لتحديث الوثائق. ورغم ذلك فإن مطوري أنظمة البرامج في الصناعة أو في الحكومة نادراً ما يمكثون في نفس المشروع لمدة كبيرة. ومع التغيير الكبير في الأشخاص فإن الوثائق المكتوبة التي تبحث عن أي تغييرات يجب أن تحفظ من أجل الصيانة المستقبلية للنظام.

مسؤول الدعم الفني يصف توثيق المشكلة بالكامل ويرسلها للشخص المسؤول عن التسليم الجديد للأنظمة ونلاحظ أن معظم وثائق الكمبيوتر يمكن تحسينها بتوظيف كاتب تقني.

والمشغل النموذجي لا يملك أي قدرة تقنية في أي حدث أو في وظيفته . والشخص المسئول عن الدعم الفني مسئول عن الحصول على المعلومات من المشغل وتنظيمها في شكل يمكن أن يستخدم في عملية الصيانة.

وهدف الشخص الذي يسجل مشكلة البرنامج هو أن يكون قادراً على ملء النموذج بالتفاصيل كما هو مبين في شكل (3–2) . وهذا النموذج غالباً يسمى **نموذج طلب صيانة برنامج** أو **نموذج تسجيل صيانة برنامج** . ونلاحظ أن هذا النموذج به أماكن للوصف الكامل للمشكلة ثم تقدير للأهمية النسبية للمشكلة. ومدير البرنامج العاقل سيفضل أن يضع بصفة عامة أولويات لعمل فنيي الصيانة بناء على تقدير الأهمية النسبية للمشاكل التي يواجهها البرنامج. والتقدير الحقيقي لصعوبة إصلاح العيب سوف

يكون مفيداً في تحديد الأولويات. وبعد حل المشكلة سيتم ملء نموذج آخر، وهذا النموذج يوضح الاستجابة لنموذج صيانة البرنامج، وهذا يسمى **نموذج الاستجابة لصيانة البرنامج**.

وشكل (3-3) يصور نموذج استجابة لإصلاح وصيانة البرنامج.

وكثير من المنظمات تجمع بين هذين النموذجين (طلب الإصلاح وطلب الاستجابة لصيانة البرنامج) كما هو مبين في شكل (3 – 4). لاحظ أن نموذج الاستجابة سوف يوضح قائمة بالتغييرات المتأثرة والوقت المطلوب لحل المشكلة.

لاحظ أهمية التوثيق الزائد لهذه النماذج. والقدرة على إرفاق الصور الجامدة لبقايا الشاشة في البرنامج أو نظام التشغيل. فإن صورة للشاشة يمكن أن تكفي وواضح أن صورة رقمية ستكون الأفضل.

وربما تكون مهمتها بكثرة الأوراق المطلوبة في هذه النماذج، ولماذا كل هذه المعلومات الزائدة، ومن يحتاجها ؟ أنه نموذج فعلي سيوضح الحاجة لهذه الأنشطة.

افترض أنك كنت تتفحص نظام برامج كان من النوع الجيد لأنه استخدم بدون تعديل في قلب كثير من أنظمة ومنتجات البرامج في تطبيق حرج وآمن . أولاً بسبب تصميمه وثانياً بسبب ارتباطه بالمعايير الضرورية .هذا البرنامج لديه نسبة خطأ قليلة جداً (Faults Per Kloc) أكثر مما يفعل معظم البرامج الأخرى في المنظمة. على كل حال فإن الجودة العالية تكون كافية حيث إن هذا القلب المستخدم قد استخدم في تطبيقات عديدة وكان من الأهمية أن تفحص لتحديد أين يمكن أن تقع أخطاء البرنامج.

<u>رقم طلب الصيانة:</u> .....

النظام..... الإصدار..... التاريخ .....

أرقام طلبات الصيانة المتعلقة .....البيئة (environment) .....

مكونات الحاسب الصلبة .....

نظام التشغيل .....الإصدار .....

برنامج النظام المرتبط GUI .....

برنامج التطبيق المرتبط .....

وصف المشكلة.....

خطورة أو حدة المشكلة: (Severity of Problem)

حرجة.....عاجلة..... مهم.....

روتينية.....

التصنيف(Classification) : جسم صلب.....برنامج..... وثائق .....

الشخص الذي يسجل التقرير.....الشخص المتحقق من التقرير

..... تاريخ التحقق من المشكلة.....المنظمة المسؤولة عن حل المسألة.....

شكل (3 – 2) نموذج طلب صيانة للبرنامج

رقم إصلاح الصيانة: (Maintenance Repair Number).....

الاستجابة لطلب الصيانة رقم .....

أرقام طلبات الصيانة المرتبطة .....

النظام..... الإصدار..... التاريخ .....

البيئة :(Environment)

المكونات الصلبة للحاسب .....

نظام التشغيل .....الإصدار .....

واجهة التداخل الرسومية GUI .....

برنامج التطبيق المرتبط.....

حدة المشكلة: (Severity of Problem)

حرجة..... عاجلة..... مهم..... روتينية.....

التصنيف : جسم صلب .....برنامج..... وثائق .....

الشخص المعد للتقرير.....

الشخص المتحقق من التقرير.....تاريخ.....

الشخص الذي حل المشكلة.....

حل المشكلة:.....

.....

.....

.....

.....

.....

.....

الشخص المسؤول عن التحقق من الحل.....

التغييرات التي تأثرت .....

ألوقت المأخوذ لحل المشكلة.....

شكل (3 –3) نموذج استجابة نموذجي لطلب صيانة.

رقم طلب الصيانة .....

أرقام طلبات الصيانة ذات الصلة (المرتبطة).....

النظام..... الإصدار..... التاريخ .....

البيئة : (Environment)

المكونات الصلبة للحاسب:-.....

نظام التشغيل .....الإصدار .....

واجهة الاستخدام الرسومية المرتبطة GUI .....

برنامج التطبيق المرتبط .....

وصف المشكلة.....

.....

.....

.....

حدة المشكلة: (Severity of the Problem)

حرجة..... عاجلة..... مهمة.....روتينية.....

التصنيف: جسم صلب.....برنامج..... وثائق.....

اسم الشخص المعد للتقرير .....

الشخص المحقق للتقرير .....

الشخص المسؤول عن حل المشكلة.....

الشخص المتحقق من الحل .....

تحليل المشكلة.....

حل المشكلة ....

.....

المتغيرات المتأثرة.....

الوقت المتاح لحل المشكلة.....

شكل (3 – 4) تقرير مزدوج يشمل نموذج طلب الصيانة مع نموذج استجابة لطلب الصيانة

أسئلة تقويم ذاتي

1/ عدد الإجراءآت التي يجب أن تتخذ عندما يتم تحديد المشكلة.

2/ ما أهم محتويات كل من:

أ) نموذج طلب صيانة برنامج؟

ب) طلب استجابة لصيانة برنامج؟

3. الصيانة المهيأة للبرامج

Adaptive Software Maintenance

عزيزي الدارس، الصيانة المهيأة (المكيفة) هي عملية تغير البرامج لتقابل الاتجاهات الجديدة في السوق والتغيرات المتوقعة في المكونات الصلبة والأنظمة وحزم تطبيقات التشغيل والمواصفات الحديثة التي تتوفر نتيجة المنافسة كلها مهمة في تحديد الحاجة إلى الصيانة المكيفة للبرامج. خذ على سبيل المثال معالج كلمات ميكروسوفت والتي سارت خلال إصدارات كثيرة منذ إنشائها. بعض هذه التغيرات شملت واجهة الاستخدام. تغيرات أخرى وفرت سمات إضافية وأشكال لملفات شائعة وفي كثير من المرات فإن الوثائق تفتح في إصدار واحد من الإصدارات الأخيرة من الوورد وهذه ربما لا يمكن قراءتها أو أن بها أشكال صفحات متغيرة بالإصدار القديم لنفس البرنامج. وهذه المشكلة لا تقتصر على المايكروسوفت ولكنها موجودة في الكوريل وورد برفكت وبين الآخرين.

لبعض المستخدمين ، فإن مزايا شكل الملف الشائع الذي يمكن فيه قراءة الوثائق على كل من PC والماكنتوش كما تم محاولتها بوجود أحجام ملفات أكبر تستعمل للوثائق . وللآخرين فإن الصفات المتوفرة بواسطة مكروس يمكن أن تكون أقل أهمية من احتياجات الأمان الإضافية لعلاج مكروس الفيروسات والتي يمكن أن تهاجر من الـ PC إلى الماكنتوش والعكس بالعكس وكما يحدث في الصيانة التصحيحية فإن اختبار البرنامج ضروري ومن المفيد من هذه النقطة أن نوضح بعض المشاكل التي لا يمكن توقعها واكتشافها حتى مع أكثر الاختبارات تعقيدا التي تجري على البرامج.

ولسوء الحظ فإن وسائل الفحص القياسية لا يمكنها بصفة عامة اكتشاف الخطأ مثل حالات السباق والتي يمكن أن تحدث نادراً . ومصطلح حالة السباق يشير إلى موقف، حيث إن نتائج الحساب (أو حتى استمرار تشغيل النظام) يعتمد على الأمر الذي تمر فيه عمليات تنفيذ حالية، وتمر إلى وحدة المعالجة المركزية CPU .

ونموذج من حالة السباق يمكن توضيحه في الشكل (3 – 5) وقيمة المتغير X الذي يقع في نهاية الحساب غير المتزامن يعتمد على الأمر الذي تنفذ فيه العملية الحالية.

عملية (1)	عملية (2)
X = 1	X = 1
Y = 7	Y = 7
X = X + Y	X = X – Y

الشكل (3–5) يبين حالة سباق مع نتائج مختلفة للحساب يعتمد على الأمر والتعليمات التي ينفذها الـ CPU .

وهناك مشكلة مهمة وفعلية يمكن أن تتسبب في مشاكل لفنيي الصيانة في البرامج الحالية وخاصة أنظمة الوقت الحقيقي. إن فنيي الصيانة لهذه الأنظمة يجب أن يهتموا بالوظائف التي تتطلب نداءات أنظمة التشغيل ولكنها ذرية بنفسها (التعليمات الذرية . هي التي لا يمكن أن تضطرب بأي حادث خارجي، ولذا فإن هذه التعليمات سوف تستكمل بمجرد أن يبدأ تنفيذها) .

وضع جداول للعمليات على أحد CPU أو أكثر لا يمكن توقعه لذا فإن حالة السباق ونداءات غير الذرية يمكن أن تكون مدمرة .

وهاتان المشكلتان يمكن أن يحدثان أنظمة البرامج التي تنفذ بالتزامن مع عمليات أخرى وكلا من نداءات الوظائف غير الذرية ، وحالات السباق تكون خطيرة جداً لأنظمة الوقت الحقيقي وأنظمة الأمان الحرج لأن النظام لا يضمن أن يعمل بنجاح ولأن نظام أداء دورة الوقت لا يمكن التنبؤ بها . وحتى الأسوأ أن الاختبار لا يمكن أن يوفر ضماناً لجودة النظام الضرورية .

في حالة والظائف غير الذرية المستعملة لتزامن الدخول إلى مصادر النظام فإن حتى أعقد الاختبارات لنظام البرنامج مع العمليات المنفذة الحالية سوف تكون غير كافية لأن ميل النظام للفشل أثناء الاختبار تكون ضئيلة .

شكل (3 – 6) يصور هذه القضية .

Process 1	Process 2
X = 1 ;	X = 1;
Y = 7;	Y = 7;
IF (X = = 1) AND (Y = = 7)	**X = X - Y**
ACCESS shared resource;	Y = X;
X = X + Y;	
Y = 1;	

الموارد المشتركة (Shared Resource)

شكل 3–6 :الوصول إلى الموارد المشتركة متحكم بالوظائف .

وتحدث المشكلة لأن التزامن يعتمد على قيم المتغيرين X و Y ، الذي يختبر في خطوتين مميزتين. ومع ذلك ، لو وصلت العمليات إلى المعالجة على نحو ما ونفذ الأمر الثالث من العملية الثانية قبل الأمر الثالث من العملية الأولى ، فشرط التزامن ربما قد لا يعقد ويمنح صلاحيات الوصول إلى الموارد المشتركة بطريقة خطأ.

والنماذج نفسها يمكن تخزينها بطريقة إلكترونية مع دخول آلي للبيانات في قاعدة البيانات . نموذجياً فإن حزمة البرامج التي تدعم أنشطة الصيانة تكون نماذج آلية، وتخزين المعلومات في قاعدة البيانات سوف تتوافر لفنيي الصيانة . وفي كثير من بيئات البرامج فإن نقص النماذج والقدرة على تكوينها يشكل أكبر مشكلة. وأسهل حل هو أن تدخل بيانات الصيانة في شكل النص مباشرة إلى صحيفة قاعدة البيانات .

مشكلة أخرى تزعج صائنو البرامج خاصة. العديد من مهندسي البرامج ، لا سيما أولئك الذين ذو خبرة بدائية مع الحاسبات ذات الذاكرات المادية الكبيرة ، لا يفكرون في أنظمتهم عند أي حدود. ومع ذلك فالذاكرة يمكن أن تستنفذ ، لا سيما إن كان هناك منتجات COTS، والتي تتطلب ذاكرة غير محددة.

مشكلة صعبة على حد سواء للقائمين بالصيانة هي استخدام أعداد مفرطة من موارد النظام المحددة. على سبيل المثال ، حزمتان من البرامج المختلفة ربما كلا كتب

البيانات إلى نفس المقبس (والتي ربما قد تنفذ كـ Winsoock يعتمد على نظام التشغيل) ولشبكات الحاسب ربما تسبب تعارضاً مفاجئاً.

مشكلة أخرى أكثر تقيداً ، عدد الملفات التي يمكن أن تفتح في نفس الوقت للمستخدمين مداها من 8 في البرامج التي تستخدم إصدارات قديمة من سي إلى 20 لـ ANSI للـ سي القياسية إلى 64 في نفس نظام اليونكس (وهذا ما يسمى البرامج المحدودة والتي يمكن أن تزيد بواسطة مدير النظام) إلى أقصى حد مسموح به (وهذا ما يسمى الهارد المحدود وهو غالبا 100 في أنظمة اليونكس)

ملحوظة

يجب أن تكون واعيا إلى أن الوضع معقد سيما مع صيانة البرامج التفاعلية مع العديد من منتجات COTS التي بنائها الداخلي غير معروف.

وواضح جداً أن القائمون بأعمال الصيانة يجب أن يكون ذو مهارات عديدة وخبرة بالبرامج البسيطة.

أسئلة تقويم ذاتي

1/ وضح مفهوم الصيانة المهيأة (المكيفة) للبرامج ؟.

2/ أشرح معنى المصطلحات التالية :

أ) حالة السباق . ب) التعليمات الذرية .

3/ أعطي نموذج من حالة السباق ؟.

4. الصيانة الوقائية للبرامج ومشكلة عام 2000

Preventive Software Maintenance and the Year 2000 Problem

الصيانة الوقائية للبرامج هي وقاية البرامج الموجودة من المشاكل قبل أن تقع المشكلة. وهذا النوع من الصيانة يميل إلى أن يكون ذو أولوية منخفضة بالنسبة للمديرين وذلك بِسبَبِ الطلبات الأخرى على الموارد والحاجة لتطور برنامجا جديداً.

ومن الأفضل لتوضيح الصيانة الوقائية إعطاء مثال توضيي – مشكلة عام 2000.

مقدار البرامج الضخمة التي ربما قد تأثرت بمشكلة عام 2000 (Y2K) قد سبب كابوس للصيانة، لا سيما بسبب نقص الموظفين المؤهلين. وقد بدأت بعض الشركات أن تستعمل ما يسمى بـ " نهج سيد لعبة الشطرنج " وذلك بتعيين أربعة أو خمسة مبرمجين حديث السن في غرفة وطلب منهم أن يختبروا الشفرة المصدرية لمشكلة التاريخ، ومعهم خبيرا في نفس الغرفة. وللمبرمجين الحديث السن السلطة أو الصلاحية في تصحيح الروتين للمشاكل، ولكن المشاكل المعقدة تحول أو تعطى للخبير. (هذا العنصر " سيد لعبة الشطرنج " يستخدم في وصف هذا الخبير لأنه هو أو هي التي تقوم بعمل لعب سيد لعبة الشطرنج ، عدد كبير من الألعاب آنيا).

☐ وهناك العديد من السيناريوهات والتي ربما تحدث في المستقبل مشاكل مثل مشكلة عام 2000

These are many scenarios that may cause future problems similar to the Y2K problem :

- استهلاك تسعة أرقام – لرقم الأمن الاجتماعيّ بواسطة عام 2010.
- استهلاك رمز المنطقة للتليفون في الوِلايات المتَّحدة بواسطة عام 2020.
- انتقال من الأسكي إلى مجموعة حروف دوليّة مثل Unicode .

- الاستخدام الزائد لمساحة المخزن المستخدمة في التخزين للتاريخ في نظام تشغيل اليونكس في عام 2038. (الوقت والتاريخ في اليونكس يقاس بالثواني من بعض التواريخ الاختيارية في 1970م) .

كل هذه السيناريوهات تمثل مشاكل والتي يجب أن تعالج. وحتى تقع هذه المشاكل في المستقبل ، تكون الصيانة الوقائية مهمة جداً. البرامج المعتمدة مع هذه المشاكل المحتومة سوف لا يكون لها منفعة تنافسية عندما تظهر هذه المشاكل. ومع الصيانة لا بد أن يأخذ في الاعتبار الاختبارات ، ولا سيما اختبار الارتداد (Regression Test) الذي سوف يكون ضروري ومهم جدا.

أسئلة تقويم ذاتي

1/ وضح مفهوم الصيانة الوقائية للبرامج مع إعطاء مثال توضيحي؟.

5. كيف تقرأ المتطلبات ، التصميم ، و شَفْرَة مصدر

How to Read the Requirements , Designs , and the Source Code

إذا أعطيت مشكلة الصيانة إلى القائم بأعمال الصيانة، يجب أن تكون الشفرة التي كتب بها البرنامج (الشفرة المصدرية) يمكن تعديلها (التعديل بها) . و مع ذلك برامج الأنظمة تتكون من ملايين من الأسطر (أسطر الأوامر) . كل سطر موجود في مكان حيث تحدث المشكلة. طبعاً، ليست كل المشاكل في وجهات المستخدم ، على سبيل المثال ، الحاجة إلى الفحص . ولكن من أين نبدأ ؟

صيانة البرامج تحتاج إلى توفر كل المعلومات المتاحة : متطلبات ، تصميم ، كتيبات ووثائق أخرى وبالتأكيد المصدر نفسه. والهدف هو فهم التغيرات في البرمجيات قبل حدوث التغيير. الجميع الذين كتبوا البرمجيات على علم بالتغيير في أحد المواضع

في البرامج والذي يؤثر على بعض في المواضع الأخرى. هكذا يكون الحاجة إلى طريقة آلية ضروري.

دع القول المأثور في ذاكرتك : لو شفرة المصدر وأي من المتطلبات والتصميم أو الوثائق لا تتوافق هذا هو الخطأ. في هذه الحالة ، الشفرة المصدرية هي المرشد الأكيد لأنها تعكس النظام في العمل الحالي. والشفرة المصدرية المتاحة دائماً تقرأ كجزء من عملية الصيانة.

دعونا تَصَوَّرْ في هذه النقطة التي نَعْتَبِرُ فقط شَفْرَة مصدر. ما ذا يجب إن نفعل في البداية ؟

الجواب مؤلف من عدة فرعيات. وقد نبدأ بالنظر في تقارير الصيانة للأوضاع المرتبطة للبحث عن أي مقترحات لمعرفة المشكلة. وقد نجعل أيضاً استعمال للأدوات المساعدة (CASE) في تحليل شفرة المصدر ونعرض بناء البرنامج.

وبعد أن حصلنا على المعلومات الكافية للفت انتباهنا إلى العدد المحدد من الوحدات النمطية يجب أن يتم فحصهم وبدقة كما تم شرحه سابقا.

أسئلة تقويم ذاتي

1/ أشرح كيف تقرأ المتطلبات ، التصميم ، و شَفْرَة مصدر ؟.

❑ وجهة نظر المدراء في صيانة البرمجيات

A Manager's Perspective on Software Maintenance

بالرغم من أن ذلك ربما يكون مدهشاً للطلاب، صيانة البرامج هي واحدة من معظم العناصر المكلفة في دورة حياة البرامج. شئ واحد يفضله المدراء وهو نقص التكلفة. يمكنك أيضاً أن تساعد المدير بواسطة اقتراح بعض الطرق لتقليل تكاليف الصيانة.

قواعد البيانات لطلبات الصيانة يمكن أن تكون مفيدة جداً . كمثال ، النظام الموضح في 2-3 يحتوي على وحدة نمطية معينة تسبب 44% لإخفاقات البرامج التي يمكن تتبعها.

هذا النوع من التحليل لجهد الصيانة هام إلى أبعد حد للمدراء على جميع المستويات. لمعظم المنظمات التجارية ، الإيرادات القليلة في معظم المنظمات التجارية تأتي خلال الصيانة ما لم يخطط إطلاقاً للبرامج الجديدة . هكذا ، ترى أن الصيانة ينظر إليها كتكاليف يجب تخفيضها كلما كان ممكن .

❑ الصيانة لمعظم مشروعات البرامج

Maintenance of the Major Software Project

واحدة من وجهات نظر صيانة البرمجيات تقترح أن تكون البرامج تحت الصيانة عند التغيير في المتطلبات أو الحاجات لاستخدام الإنترنت. سوف نتجاهل التغيرات في الاحتياجات السابقة ونعوض عنها ببعض التغيرات الممكنة بعد التسليم.

النوع الأول للصيانة يعتبر صيانة تصحيحية . هل يسلك البرنامج مثلما يجب ؟ وبعد أن اعتبرنا التصحيح، ويجب أن نضع في الحسبان واجهة التداخل وإمكانيات التحسين. طبعاً ، أداء التنفيذ يظهر إذا ما طلب زيادة زمن الاستجابة.

الخلاصة

* قدمت الوحدة أهم التعريفات الخاصة بصيانة البرمجيات :

– تشمل على عمليات تكويد البرامج بعد تسليمها للمستخدم لتصحيح أخطاء التكويد بغرض تحسين أداء البرمجية وللوصول للأداء المثل لها.

– هي إحدى مراحل عمليات هندسة البرمجيات وتشمل تحسين أداء البرمجية ووصولا للأداء الأمثل بعد تسليمها للمستخدم إضافة لمعالجة مشاكلها التي تطرأ أثناء الاستخدام.

"وإجمالا توصف صيانة البرمجيات بأنها العملية النظامية لتغير البرمجيات المستخدمة فعلا وذلك لمنع حدوث أعطال في النظام ولتحسين الأداء".

* بينت الوحدة أن أعمال الصيانة الصحيحة للأجهزة الحاسوبية يشتمل على المحافظة على نظافتها، والاهتمام على وجه الخصوص بالقطع المتحركة واستبدال المكونات التالفة، والجدولة العامة لاستبدال المكونات القديمة والتالفة أو على وشك التلف. كما يلزم أجراء تحليل التكلفة / المنفعة لتحديد إذا ما كانت جهود الصيانة مكلفة وضمن أهداف المنشأة.

* أوردت الوحدة عدة عوامل توجب صيانة البرمجيات وصنفت الوحدة بنود هذه العوامل على أنها تنتمي إلى عدة فئات مميزة أو **ثلاثة أنواع رئيسية لصيانة البرمجيات** هي كالتالي : صيانة تكيفيه – صيانة تصحيحية – صيانة وقائية .

وبينت الوحدة إن أول خطوة في الصيانة التصحيحة للبرمجيات هو تحديد نوعية ما يراد صيانته حيث يبدأ العمل عادة في التعرف أولا على المشكلة الموجودة في البرمجيات المستخدمة.

* لخصت الوحدة الاجراءات التي تلي تحديد المشكلة والمتعلقة باتخاذ القرار عما سيتم وكيف سيتم حل المشكلة وهذه الإجراءات هي : ملاحظة وجود مشكلة – توثيق المشكلة تحديد أهمية المشكلة – تحديد أولوية المشاكل لتحديد أولويات وأهمية إصلاحها– تحديد ماهية المشاكل التي لا يمكن إصلاحها نتيجة نقص الموارد – حل المشكلة– اختبار

النظام لمعرفة إذا كان حل هذه المشكلة سيؤدي إلى مشاكل في أجزاء أخرى من البرنامج. توثيق الحل كجزء من شفرة المصدر- توثيق الحل في أشكال أخرى من الوثائق إذا كانت التغيرات التي أجريت على التصميم الأصلي للنظام - تحديث قاعدة البيانات للمعلومات حول أخطاء البرنامج.

وبينت الوحدة إن صيانة البرامج يحتاج إلى كميات كبيرة من العمل المكتبي لتحديث الوثائق.

قدمت الوحدة نماذج توثيق المشكلة :

أ) نموذج طلب صيانة برنامج أو نموذج تسجيل صيانة برنامج : وهذا النموذج به أماكن للوصف الكامل للمشكلة ثم تقدير للأهمية النسبية للمشكلة.

ب) نموذج الاستجابة لصيانة البرنامج : ويتم ملء هذا النموذج بعد حل المشكلة وهذا النموذج يوضح الاستجابة لنموذج صيانة البرنامج .

وذكرت الوحدة أن كثير من المنظمات تجمع بين هذين النموذجين (طلب الإصلاح وطلب الاستجابة لصيانة البرنامج)

* بينت الوحدة أن الصيانة المهيأة (المكيفة) هي عملية تغير البرامج لتقابل الاتجاهات الجديدة في السوق والتغيرات المتوقعة في المكونات الصلبة والأنظمة وحزم تطبيقات التشغيل والمواصفات الحديثة التي تتوفر نتيجة المنافسة كلها مهمة في تحديد الحاجة إلى الصيانة المكيفة للبرامج.

* بينت الوحدة أن الصيانة الوقائية للبرامج هي وقاية البرامج الموجودة من المشاكل قبل أن تقع المشكلة.

ملحوظة مهمة

تفاصيل أكثر عن الصيانة والتطوير، الفروق الجوهرية المختلفة بينها ، وسرد لأنواع أخرى من الصيانة، توزيع جهد الصيانة وتكاليفها بالنسبة لدورة الحياة ، وأهداف الصيانة وموقعها بالنسبة لهندسة البرمجيات، وبعض الآراء للمتخصصين عن صيانة البرمجيات موجودة بالملحق المرفق بالوحدة.

لمحة مسبقة عن الوحدة التالية

الوحدة التالية تبحث في " **إدارة المشاريع البرمجية** " الوحدة توضح الفرق بين هندسة البرامج والأنماط الهندسية الأخرى . كذلك الصفات المشتركة للإدارة لكل من المشاريع الهندسية ومشاريع تطوير البرمجيات ، كما تبحث الوحدة في إدارة الموارد البشرية الأنشطة والمهام المتعلقة بها ، تخطيط المشروع البرمجي ، جدولة المشروع ، إدارة الكوارث

أسئلة مراجعة على الوحدة

1) ارسم منحنى يبين توزيع صلاحية نظم الأجهزة والبرمجيات مع الزمن.
2) اسرد التعريفات المختلفة لصيانة البرمجيات واستنتج تعريفا جامعا لها.
3) ما هي أهم العوامل التي توجب صيانة البرمجيات؟.
4) اسرد أهم مشاكل البرمجيات كما وردت في سياق الوحدة.
5) ما هي أهم أنماط الصيانة وما هي حيثيات كل منها؟.
6) ما ي أهم محتويات كل من:-
   أ) نموذج طلب صيانة برنامج.
   ب) طلب استجابة لصيانة برنامج.
7) أسرد على الأقل عشر حزم برامج والتي يجب أن تتغير لتستوعب الزيادة في عدد الحروف المستخدمة لأرقام الأمن الاجتماعي.
8) اسرد على الأقل عشرة حزم برامج والتي يجب أن تتغير لتستوعب التغيير في عدد الأرقام المستخدمة في أرقام التليفون.
9) اسرد عشرة حزم برامج على الأقل والتي يجب أن تتغير لتستوعب التغيير في التاريخ في اليونكس وحساب الوقت . كن متأكداً بأننا سنناقش الوسيلة المستخدمة.

10) اعتبر تطبيقاً محببا للحاسبات الشخصية. صف التغييرات الرئيسية بين الإصدارات المختلفة. اسردها مرتبة على حسب مقدار التكاليف.

11) افحص الحوار الموجود والمتعلق بالمتطلبات لمشروع البرامج الكبيرة والتي ناقشناها خلال مقدمة في هندسة البرمجيات. واستنادا إلى ذلك الحوار، ما التغييرات التي تتوقعها والتي يجب الترتيب لها لجعل البرامج مفيدة لفترة أطول.

المراجع

المراجع الإنجليزية :

[1] Ian Somerville , "Software Engineering", Addison Wesley, 2001.

[2] Ronald J. Leach,, "Introduction to Software Engineering", CRC Press, 1999.

[3] Douglas Bell , "Software Engineering A Programming Approach", 3rd Edition, Addison Wesley.

[4] **Shari Pfleeger, "Software Engineering - Theory and Practice", 2nd Edition.**

ملحق 1

تفاصيل أكثر عن صيانة البرمجيات

سعة انتشار البرامج :

- دخل استعمال البرامج إلى كل مناحي الحياة .
- في بيوتنا وأدواتنا وألعابنا وسياراتنا .
- أو حتى الطائرات والسكك الحديدية والمصانع النووية والأجهزة الطبية .
- في السابق ، لم تسجل شكاوى حول مشاكل البرامج .
- وكان ذلك بسبب أن مستخدمي البرامج عادة هم أناس متخصصون .
- لكن ذلك لم يعد واقعا الآن ، حيث يستخدم هذه البرامج كل شرائح المجتمع.

أزمة البرامج :

- من كل أربعة أنظمة مشاريع يخفق واحد في التسليم.
- 20 % فقط من مبيعات البرامج تعمل بشكل طبيعي.
- تستغرق الأنظمة الكبيرة من 3 إلى 5 سنوات لتطويرها.
- ولهذا ، الكثير من المشاريع تلغى قبل التسليم.
- تمثل صيانة الحاسب أكبر سعر تكلفة فيما يتعلق بالحاسب لدى أكثر الشركات.

دراسات وإحصائيات :

- في حزيران 1994 م نشرت مجموعة IBM الاستشارية استطلاعا أجرته على 24 شركة رائدة في مجال تطوير الأنظمة المنشورة، وقد ذكر ما يلي :
  - 55 % زادت فيها تكلفة تطوير البرنامج.
  - 68% أخذت وقتا أطول من المتوقع لإتمامها.
  - 88% استلزمت إعادة التصميم جوهريا.

- وفي دراسة حديثة أجريت في الولايات المتحدة ، بواسطة مجموعة Standish على 8.380 مشروع من القطاعين الحكومي والخاص أظهرت **ما يلي :**
  - 31% من المشاريع ألغيت قبل إكمالها.
  - 53% منها زادت كلفتها الفعلية بنسبة 198% عن التكلفة الأصلية المقدرة لها ، 42% منها فقط حقق السمات والوظائف المقترحة.
  - 9% فقط من المشاريع أنجزت في الوقت المحدد وعلى الميزانية.

ما هي صيانة البرمجيات

الصيانة والتطوير هل هناك فرق ؟

عادة ..تجرى عملية الصيانة عادة على السيارة أو المنزل لمعالجة مشاكل عديدة ، إما لاستبدال الفرامل أو لإصلاح سقف المنزل وما إلى ذلك .

بينما إذا أردنا أن نبني امتداداً للمنزل أو نغير سطح السيارة جرت العادة على تسميتها (تحسينات) بدلا من الصيانة.

ولكن ..صيانة البرمجيات برامج لا تهتم فقط بتصحيح الأخطاء ، بل هي تهتم أيضا بتطوير البرامج ليُصبح أداؤها أفضل ، كما أن صيانة البرمجيات هي إحدى مراحل تطوير النظم.

تعريفات :

- تعريف IEEE 91 : صيانة البرمجيات هي عملية التعديل على نظام البرنامج أو مكوناته بعد تسليمه ونشره ، وذلك إما لتصحيح العيوب ، أو لتطوير الأداء أو بعض الخصائص ، أو حتى لتكييف البرنامج مع التغيرات في بيئة التشغيل .
- هي كل عمليات البرمجة (كتابة الشفرة) بعد الانتشار الأوليّ للبرنامج ، بهدف تصحيح أخطاء الكود،أو تحسين البرنامج أو زيادة إتقان عمل وظائف البرنامج .
- صيانة البرمجيات هي إحدى مراحل (عملية تطوير البرمجيات)، وتشير هذه المرحلة إلى عملية تتبع انتشار البرنامج في الساحة ، بينما تتضمن : التعديل على

البرنامج بهدف تصحيح العيوب والنقائص التي وُجِدت أثناء استعمال العملاء، وإضافة وظائف جديدة للبرنامج لتحسين تطبيقه واستعماله .

- هي المرحلة الأطول في حياة النظام البرمجي لكي يبقى النظام قادراً على مواكبة التطورات والمعدات الحديثة، جزء من هذه المرحلة يكون في تصحيح الأخطاء، والجزء الآخر يكون في التطوير وإضافة تقنيات جديدة.

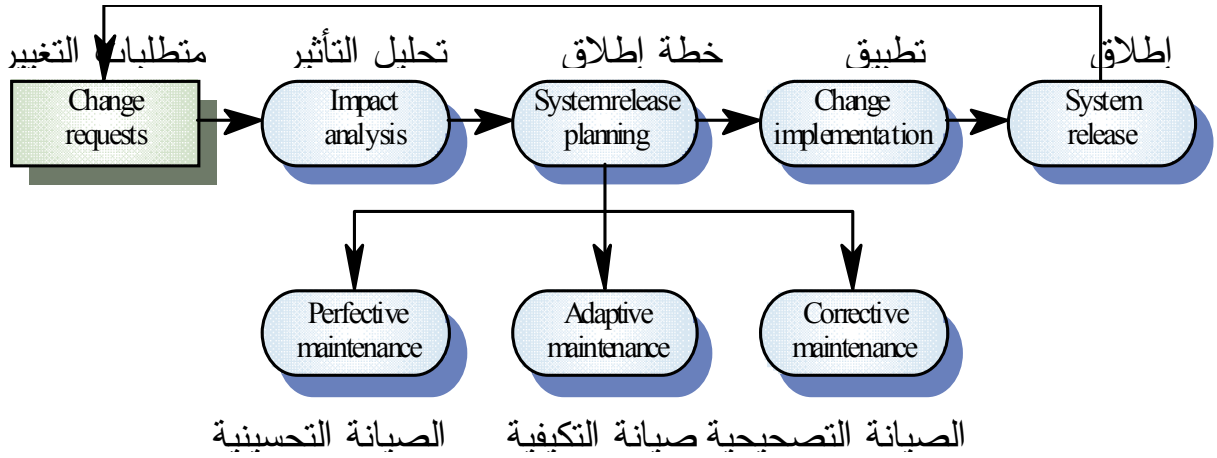
وبالجمع بين التعريفات السابقة يمكن تلخيص **ما يلي :**

- صيانة البرمجيات هي إحدى مراحل تطوير البرمجيات .
- وتجرى للبرنامج بعد إصداره وتسليمه للعميل.
- وتتم فيها عمليات البرمجة وكتابة كود للبرامج المراد صيانتها
- بهدف :
  - التصحيح العيوب .
  - تطوير الوظائف ، وإضافة تقنيات جديدة .
  - تكييف البرنامج مع مختلف الأجهزة و بيئات التشغيل.

البرامج مثلها مثل أي منتج ، يصدر نموذجياً مع بعض العيوب والنقائص المعلومة ، ثم يكون له العديد من الإصدارات تهدف المنظمة من إصدارها لتحديد منفعة وقيمة البرنامج في مستويات جودة معينة لها أهمية ووزن كبير جدا لا يقارن بتأثير العيوب والنقائص المعلومة مسبقا.

- صيانة البرمجيات تشتمل على عدد من التقنيات المحددة . إحداها تسمى تقنية " التقسيم الساكن " (static slicing) وهي التقنية المستعملة في تعريف كل الشفرات التي يمكن التعديل على بعض متغيراتها ، وهذه التقنية مفيدة في إعادة تحليل كود البرنامج.

أنواع صيانة البرمجيات



أنواع الصيانة المختلفة :

أ- صيانة تصحيحية :

- تقوم بتمييز وإزالة العيوب.
- وتصحيح أخطاء فعلية.

ب- صيانة تكيفية :

- النشاط الذي يتم فيه تكييف النظام لكي يشتغل في بيئة مختلفة مثل: (جهاز مختلف – نظام التشغيل).

ت- صيانة تحسينية :

النشاط الذي يضيف قدرات جديدة وتحسين وتطوير الوظائف الموجودة وتحسينات عامة. وهي تأخذ أكثر الجهود المبذولة للصيانة.

ث- الصيانة الوقائية :

النشاط الذي يغير البرنامج لتطوير وتحسين الصيانة المستقبلية أو الاعتمادية أو التزويد بأفكار وأساسات جيدة للتطوير المستقبلي. وهي لا تزال نادرة نسبياً.

ج- صيانة طارئة :

- تعتبر صيانة تصحيحية غير محددة.

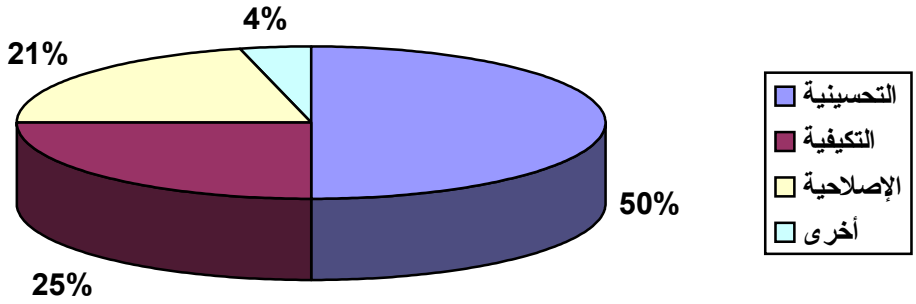
ح- صيانة تامة :

- يتم فيها تحسين الأداء و الموثوقية وقابلية الصيانة.
- أيضا يتم فيها تجديد التوثيق.

توزيع جهد أنشطة الصيانة :

(بناء على دراسة لـــ487 شركة لتطوير البرامج)

- التحسينية 50%
- التكيفية 25%
- الإصلاحية 21%
- أخرى 4%



تكاليف الصيانة

مهما كانت الصيانة المطلوبة (تصحيح أو تحسين أو تكييف) يجب أن تكون مفهومـــة علـــى أكمل وجه من قِبل القائم بعملية الصيانة ،بما في ذلك تأثير الصيانة من حيث التكلفـــة والجهـــد لتنفيذها.

هذا يعني أن مستوى فهم برمجيات النظام القائم يجب أن تُنجز حتى يُتمكن من تحليلها فعلياً.

وتكلفة هذا الفهم ثبت أنها مسئولة عن الجزء الأكبر من تكاليف برامج الصـــيانة وبـــالأخص تكلفة فهم الشفرة الحالية للنظام.

تكلفة الصيانة بالنسبة لتكاليف دورة حياة البرامج :

- منظمات برامج الحاسب الآلي المثالية تنفق من 40-70% من ميزانيتها في الصيانة.
- رغم عدم وجود اتفاق حقيقي على التكاليف الفعلية ، ولكن توجد بيانات كافية للدلالة علـــى أن المحافظة لا تستهلك جزءاً كبيراً من تكاليف البرامج.
- في مرحلة دورة الحياة 1\4-1\3 مجموع التكلفة يعزى إلى برامج التنميـــة وأن حـــوالي 67% منها مستهلكة في التشغيل والصيانة.

- رواتب المبرمجين اليوم تستهلك الجزء الأكبر من ميزانية البرامج، وقد يرجع سبب ارتفاع تكاليف الصيانة إلى أن الكثير من البرامج أُنشئت بدون الاستفادة من أفضل تقنيات التصميم و الترميم .

تكاليف صيانة الأنظمة المتعددة الاستخدام تعتبر عالية وذلك بسبب عاملين هامين هما:-

(1) زيادة عدد طلبات التغيير بالنسبة لعدد أنواع المستخدمين.

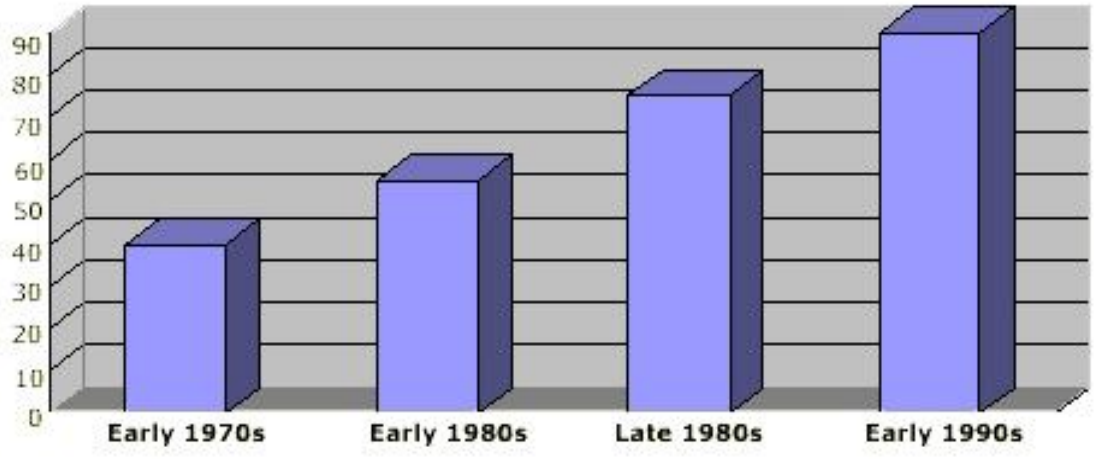
(2) تكلفة الصيانة تتضمن جهود كبيره خارج مساحة التطبيق الرئيسية.

مثلاً : طلب تغيير وحدة واحدة يتطلب التفاعل مع كل المديرين للوحدات الأخرى لتنظيم التغيير الكلي، وهذه التكلفة عالميا أعلى بكثير من الكلفة الأساسية للتغيير المطلوب.

الإنفاق على الصيانة التصحيحية و غير التصحيحية:

الصيانة التحسينية هي أغلى من الصيانة التصحيحية ،فإنها تقتضي إعادة تصميم العمل إلى حد كبير أكثر من أي تصحيح للشفرة.

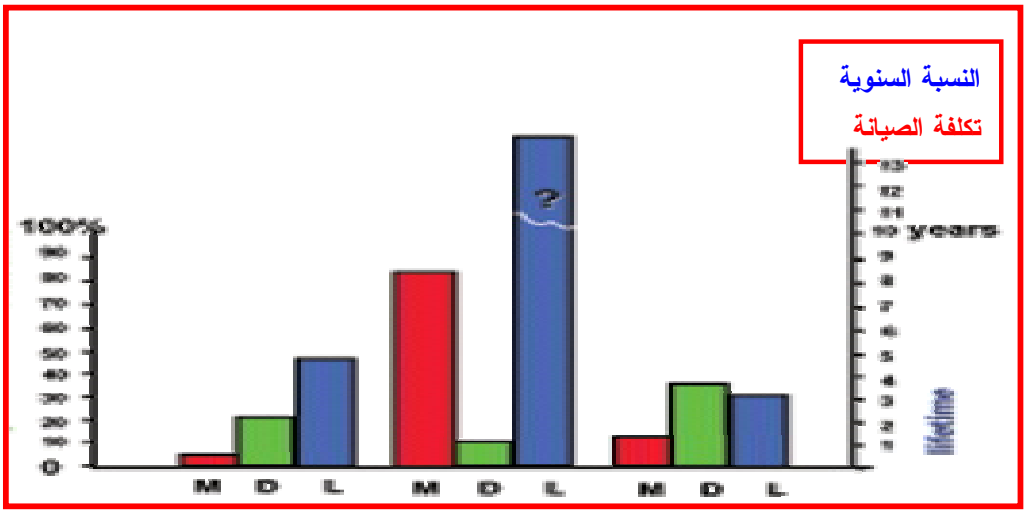
إذن معظم تكاليف الصيانة تنفق على التحسينات وهي صيانة مكلفة وذلك لان متطلبات التغيير و البيئات غالبية تكاليفها يدفعها المستعملون .



دورة حياة البرامج المخصصة لتكاليف الصيانة

الصيانة	لينتز & سوانسون 1980	الكرة 1987	ديكلافا 1990	ابران 1990
التصحيحية	%22	%17	%16	%21
غير التصحيحية	%78	%83	%84	%79

نسبة تكاليف الصيانة مقارنة بالعتاد و الأجهزة ذاتية الحركة:



العتاد البرمجيات الأجهزة ذاتية الحركة

مع ملاحظة أنه : قد تكون تكلفة صيانة الأجهزة أعلى عندما تكون التغييرات متتالية لعمليـــات التطوير التي قد تصل إلى مرتين في السنة.

أهداف صيانة البرمجيات

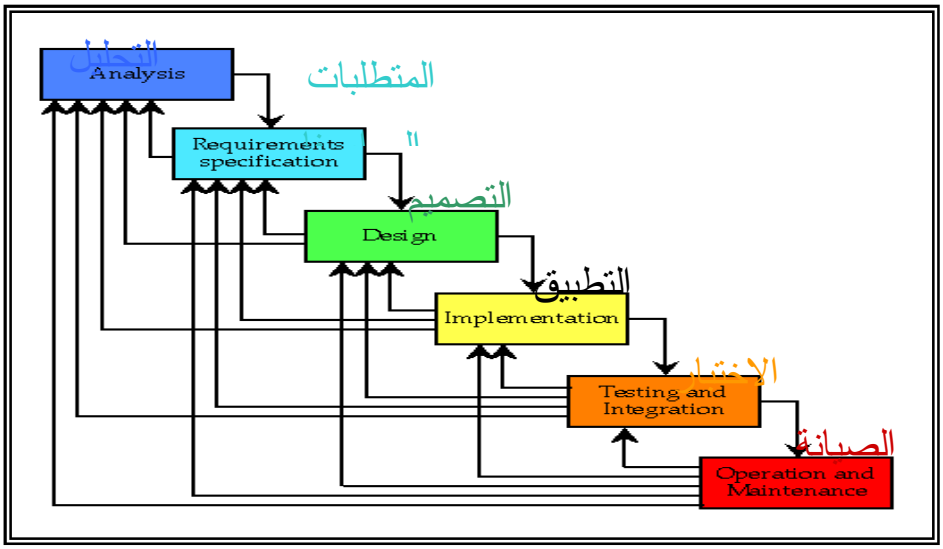
CMMI

CMMI Processes	معهد مهندسي الكهرباء والإلكترونيات / تقييم الأثر البيئي (IEEE/EIA 12207) أهداف صيانة البرمجيات
TS,RSKM	تحديد تأثير التنظيم والتشغيل والوصلات على النظام الحالي.
RD,REQM, RSKM	تحديد وتحديث دورة حياة البيانات.
TS,PI,VER,VAL,CM, PPQA,RSKM	تطوير مكونات النظام بالوثائق والتجارب التي تثبت خطورة متطلبات النظام.
VER,VAL,RSKM	ضمان إيفاد لأنظمة جديدة أو نسخ معدلة بحيث لا تؤثر على العمليات الجارية.
PI,CM,PPQA	نظام النقل وتطوير البرمجيات لبيئة المستخدم.
TS,CM	الحفاظ على القدرة على استئناف المعالجة مع النسخة السابقة.

الصيانة وهندسة البرمجيات

- الصيانة هي المرحلة الأخيرة من مراحل هندسة البرمجيات وقد تستلزم العــودة لما قبلها من مراحل .
- طور صيانة البرمجيات هو جزء واضح (ورئيســي) فــي نمــوذج الشــلال (Waterfall Model) لعملية تطوير البرامج والذي يتكون بالتنقــل الهيكلــي للبرمجة (Structured Programming Movement)
- بينما لا نجد إشارة واضحة إليه في النمــوذج الحلزونــي (Spiral Model) والذي يتكون بالتنقل الشيئي الموجّه (Object Oriented Movement) وإن كان تم التطرق إليه باعتبار حقيقة أن الصيانة تشغل ثلثي تكلفة عمر النظام .

نموذج الشلال Waterfall Model



الاختصارات :

- IEEE: Institute of Electrical and Electronics Engineers
- SM : software maintenance
- CMMI : Capability Maturity Model Integration

صيانة البرمجيات في سطور

أورد الأستاذ خالد عياد الشقروني في أحد مقالته عن صيانة البرمجيات العديد من الحقائق والتي نوجزها في الحقائق التالية:

صناع البرمجيات لا يتحرجون من وجود ثغرات في منتجاتهم، ولا يعدون بإصلاحها عاجلا.

قصور المطورين عن تنفيذ البرامج كما يجب، و قصور أصحاب العمل عن فهم احتياجاتهم وطرحها كما يجب، أمران شائعان.

المنظومة البرمجية كائن حيوي يعيش في بيئة حيوية وهو قابل للتكيف وللنمو كما هو قابل للضمور والتوقف

الصيانة إضافة قدرات جديدة للبرمجية و ليس إصلاحها

عد استكمال تنفيذ المنظومة البرمجية و بدء العمل بها، فإن الأمر يتطلب مرافقة و متابعة مستمرة لوظائف المنظومة ولبياناتها خاصة في السنة الأولى من حياتها. هذه الحاجة تفرضها الطبيعة الخاصة للبرمجيات. والعمليات التي تتم بعد تركيب المنظومة و بدء تشغيلها تندرج تحت ما يسمى "الصيانة" و الدعم الفني وتتضمن التالي:

- **إصلاح الأخطاء و الثغرات البرمجية والمنطقية.**

ربما من المستحيل أن تخلو وحدة برمجية ما من أخطاء أو ثغرات تولدت أثناء تصميمها وتنفيذها. هذه الأخطاء قد تكون برمجية يقع فيها المبرمج عند كتابة تعليماته البرمجية أو ثغرات منطقية تخص علاقات الوظائف ببعضها، أو قصور في مقابلة متغيرات لم يحسب لها. صحيح أن المراجعة المتأنية و الفحص الدقيق و الاختبارات المكثفة تسهم في التقليل من هذه الأخطاء، إلا أن نسبة الأخطاء التي سيتم الكشف عنها نتيجة المراجعة و الفحص والاختبار ستكون منخفضة، وإذا أريد زيادة هذه النسبة للكشف عن معظم الأخطاء و توقع آلاف الاحتمالات الفنية و الوظيفية التي قد تتعرض لها المنظومة؛ فإن الجهد والوقت اللازم سيكونان أضعاف مضاعفة للجهد والوقت الأصليين اللذان استهلكا في تصميم و بناء المنظومة الأمر الذي يبطل جدوها اقتصاديا.

لذلك نرى موردي البرمجيات لا يتحملون إلا مسؤولية محدودة جدا فيما يخص منتجاتهم البرمجية و أدائها، و لا يقدمون أية ضمانات لها، بل إنهم يحملون المستهلك كل التبعات من خسائر أو أضرار قد يتعرض لها وتكون ناتجة مباشرة بسبب استخدامه لمنتجاتهم. وهم عند بيعهم لمنتجاتهم لا يتحرجون من الإشارة لوجود ثغرات في برامجهم كأمر طبيعي، يصفونها للمستهلك دون وعد واضح بإصلاح قريب.إن الأخطاء و الثغرات و ونقاط القصور ستظل كامنة

داخل المنظومة البرمجية تنتظر الوقت المناسب أو الظرف الملائم كي تظهر للعيان. هذا واقع لا مهرب منه، و الحاجة تظل قائمة لاتخاذ التدابير الملائمة للتصدي لها كلما ظهرت.

- **ضبط أداء وحدات المنظومة البرمجية بحسب ما يتكشف عنه التشغيل الفعلي في بيئة المستهلك.**

يسعى المطورون ما أمكنهم لمحاكاة و توقع بيئة العمل الفعلي عند تطوير برمجياتهم واختبارها، و لكن هذه المحاكاة تبقى تشبيهية ولا تناظر الواقع العملي، حتى اختبارات التشغيل التي تتم في بيئة عمل الزبون لا تكفي إلا لبيان ما إذا كانت المنظومة تلبي في وظائفها الرئيسية المتطلبات والمواصفات الموضوعة، لأغراض الاستلام و إجراءات القبول. الحدود والإمكانيات الحقيقية للمنظومة لا تتكشف إلا بعد وضعها في محك العمل الفعلي في بيئة وبيانات وعمليات حقيقية من قبل مستخدمين حقيقيين، وذلك عند مباشرة المنظومة مرحلة الإنتاج و التشغيل الفعلي. لذلك بعد التركيب ومباشرة التشغيل تتكشف الكثير من مشاكل الأداء أو الاعتمادية في بعض جوانب المنظومة، أو حتى وجود جوانب قام المطورون بصياغتها بطريقة تخالف التصور الذي لدى صاحب العمل.

الأمر أيضا يشمل أصحاب العمل مستهلكي المنظومة و الذين قاموا بصياغة احتياجاتهم ومتطلباتهم من أجل تنفيذ المنظومة. فعند التشغيل الفعلي كثيرا ما يكتشف أصحاب العمل وجود متطلبات ضرورية قد تغافلوا عنها أو نسوها، أو أن سياق و تدفق بعض الوظائف يجب أن لا تكون بالصورة التي خططوا لها. أو أنهم قد قدموا مواصفات خاطئة لما يجب أن يكون عليه العمل، و هذا أمر طبيعي، فحتى أصحاب الحاجة لا يستطيعون دائما صياغة حاجتهم وتدوينها أو طرحها بدقة وشمولية. هنا الكثير من عمليات المراجعة و إعادة ترتيب الحسابات

يجب أن تتم، وأن تترجم عن طريق إعادة ضبط المنظومة و تكييفها و تصحيح مساراتها. قصور المطورين عن تصميم و تنفيذ المنظومة كما يجب، و قصور أصحاب العمل عن فهم احتياجاتهم وطرحها كما يجب، أمران شائعان و شديدي الاحتمال، ولن يتكشفا إلا بعد التشغيل الحقيقي للمنظومة، لذلك لا بد من وجود فرصة للتقويم و التصحيح،

- **الاستجابة لما يطرأ من متطلبات تقنية أو وظيفية جديدة أو متبدلة، و ذلك بالتعديل والتبديل و الإضافة.**

التغييرات التقنية :

المنظومة البرمجية ليست معزولة تقنيا عن محيطها الذي يحتضنها، فهي تعيش في بيئة تضم:

أ) مجموع العتاد من أجهزة و ملحقات و كل ما يرتبط مباشرة أو بطريقة غير مباشرة بالمنظومة أو باحدى المكونات البرمجية التي تعتمد عليه المنظومة.

ب) مجموع البرمجيات التي صممت على ضوئها المنظومة أو أحد أجزائها، و تشمل اجمالا: نظام التشغيل، و نظام إدارة قواعد البيانات ومجموع التقنيات والمكتبات البرمجية التي تعتمد عليها المنظومة.

الواقع العملي يظهر أن الجهات تسعى باستمرار لتطوير و تغيير خياراتها التقنية من معدات وبرمجيات لمسايرة التحديثات التي تطرأ على تقنية المعلومات و التي بطريقة أو بأخرى تفرض تفسها و بالحاح على هذه الجهات. أو لكي تتلائم مع برمجيات و نظم أخرى لديها اشتراطاتها التقنية الخاصة.

وحيث أن أية منظومة برمجية تكون مرتبطة مع محيطها من أجهزة و نظم تشغيل وخوادم، فإن أي تغيير في أحد مكونات هذه البيئة قد يؤثر فيها أيضا.

إن تجديد الخيارات التقنية للمنظومة أمر مطلوب لكي تبقى دائما مستعدة للتوافق مع مستجدات التقنية، كما أن هذا التجديد سيكون مفروض عليها إذا ما تجددت أو تغيرت الخيارات التقنية لمحيطها.

التغييرات الوظيفية

إلا أن أهم التغييرات التي تتعرض لها المنظومة و أكثرها تأثيرا و إعاقة هي تلك التي تتعلق بنواحيها الوظيفية و ليست الفنية.

فالمنظومة ليست معزولة عن محيطها الوظيفي، فهي تتعامل مع أمور لها علاقة بالقوانين واللوائح والاجراءات وما هو معتمد من أساليب عمل و دورات مستندية، كما تتعامل وفق رؤية وتصور القائمين على الأعمال في الجهة و خططهم و استراتيجياتهم و نظرتهم لما هو واقع ولما يجب أن يكون. أيضا تتعامل مع طبيعة بشرية تتبدل سلوكياتها و يتطور منظورها تجاه المنظومة و ارتباطها بنواحي العمل.

إذا لم تستجب المنظومة في زمن مناسب لتغير القوانين أو الإجراءات، فهذا سيشكل إعاقة تنفي جانب الاستفادة من إحدى نواحي المنظومة، و إذا ما تراكمت هذه الإعاقات فإن وجود المنظومة أصلا سيكون موضع تساؤل.

إحدى أهم دوافع محثات تطوير نظم العمل والإجراءات هو وجود نظم محوسبة في الجهات تسمح لها بالتفكير والانتقال إلى مستويات أعلى. لذا فإن أية منظومة بعد رسوخها والثقة بها يتوقع لها أن تكون السبب والدافع لتلمس طرق أفضل و دخول مجالات أوسع من تلك التي كانت عليها. و إذا لم تتطور المنظومة لتستجيب لمستويات جديدة في نظم العمل فستكون عائقا أمام الجهة يمنعها من التطور و الارتقاء.

وهذا ما يدفع الجهات إلى استبدال منظوماتها و تغييرها بأخرى، و ما يتبع هذا من مخاطرة قد لا تعيها بعض الجهات فلا تأخذها في حسابها، أو تكون واعية لهذه المخاطر و ترضخ لها مضطرة. و كان ممكن لها أن توفر الكثير لو تعهدت منظوماتها الأصلية بالرعاية و التطوير أولا بأول.

المنظومة البرمجية كائن حيوي يعيش في بيئة حيوية و هو قابل للتكيف وللنمو كما هو قابل للضمور و التوقف.

الصيانة مرحلة من مراحل بناء المنظومة

العناصر السابقة أمر لا بد من وقوعه لأية منظومة برمجية، و لا يمكن تفاديها مهما بذل من مجهود في مرحلة بناء و تطوير المنظومة و مهما تعمقت دراسات تحليل المتطلبات والاحتياجات و مهما تكثفت عمليات الاختبار و التجريب. هذه هي طبيعة البرمجيات و التي تجعل من خصائص بنائها أمرا مميزا تختلف فيه عن باقي المجالات كالصناعات الهندسية و نشاطات البناء و والتشييد. فالمنظومة البرمجية لا تعطي نتائج مأمولة ما لم يتم تعهدها بالرعاية و المتابعة فور التركيب و خلال تشغيلها الفعلي خاصة في بدايات هذا التشغيل، و إذا ما تم تجاهل ذلك بصورة أو بأخرى فإن المنظومة ستكون قاصرة.

لذلك يمكن اعتبار أن اكتمال المنظومة يتم في مرحلتين:

المرحلة الأولى: التنفيذ والتركيب، وتتم فيها عمليات التحليل والتصميم و البناء و الاختبار انتهاء بالتركيب.

المرحلة الثانية: الصيانة، ويتم فيها اصلاح الأخطاء وإزالة الثغرات والضبط والتنقيح والتعديل والتبديل والتحسين واستدراك ما يلزم إضافته بالضرورة.

المرحلة الثانية تقع في دورين، الدور **الأول** في فترات التشغيل الأولى حيث تتكثف عمليات الصيانة حتى تأخذ المنظومة شكلها النهائي وتستقيم في مسارها.

الدور **الثاني** و يتواصل طوال عمر المنظومة للحفاظ على جاهزيتها والاستجابة الفورية لتبني أي تغيير.

إن الأعمال التي تندرج تحت مسمى "صيانة" جزء بسيط منها فقط يتعلق بإصلاح الأخطاء أما الجزء الأكبر فهو يقع ضمن عمليات التحسين و التعديل و التكييف والإضافة، و بتراكمها تكون الصيانة في الواقع العملي هي إضافة قدرات جديدة للبرمجية وليس إصلاحها.

المراجع :

- www.csse.monash.edu.au/.../html/text.html
- www.ee.unb.ca/.../IntroToSoftwareEng.htm
- http://elearning.tvm.tcs.co.in/SMaintenance/SMaintenance/sm_contents.htm#top
- www.cs.iit.edu/~icsm2004/
- http://hebb.cis.uoguelph.ca
- http://bcs.org
- www.programs-transformation.org
- www.dur.ac.uk/csm/
- www.info@nidam.net


الوحدة الرابعة
إدارة المشاريع البرمجية
Software Project Management

محتويات الوحدة

المقدمة

تمهيد

عزيزي الدارس،

مرحباً بك في الوحدة الرابعة من مقرر **"هندسة البرمجيات 2"**، والتي تحمل عنوان " **إدارة المشاريع البرمجية** ". تحتوي الوحدة على خمسة أقسام: القسم الأول من الوحدة نوضح فيه الفرق بين هندسة البرامج والأنماط الهندسية الأخرى . كذلك الصفات المشتركة للإدارة لكل من المشاريع الهندسية ومشاريع تطوير البرمجيات. القسم الثاني من الوحدة يبحث في إدارة الموارد البشرية الأنشطة والمهام المتعلقة بها، حيث يتناول القسم مدير المشروع والنشاطات الإدارية ، كذلك فرق عمل المشاريع البرمجية، وطبيعة فرق عمل المشروع ، وأدوات إدارة المشروع . القسم الثالث من الوحدة يبحث في تخطيط المشروع البرمجي، وفيه يتم تناول أنواع خطط المشروع ، وخطوات تخطيط المشروع ، كذلك هيكلية خطة المشروع، وتنظيم الأنشطة . القسم الرابع يتناول جدولة المشروع حيث يتناول أنشطة عملية جدولة المشروع ، ومخططات الجدولة الزمنية للمشروع . أما القسم الخامس من الوحدة يبحث في إدارة الكوارث، حيث يتناول أنواع الكوارث ، كما يشرح القسم عملية إدارة الكوارث والمراحل التي تتضمنها ، كذلك يتناول القسم التعرف على الكوارث ، وتحليل الكوارث ، والتخطيط للكوارث ، ومراقبة الكوارث .

أهداف الوحدة

عزيزي الدارس، بعد فراغك من دراسة هذه الوحدة ينبغي أن تكون قادراً على أن:

- **تعرِّف** خصائص المنتج البرمجي التي تميزه عن باقي المنتجات الصناعية.
- **تميز** بين إدارة مشاريع البرمجيات والأنواع الأخرى من إدارة المشاريع الهندسية.
- **تقسم** إدارة المشروع البرمجي إلى أنشطة إدارية مختلفة وتعيّن القائمين على إنجاز كل نشاط.
- **تصف** أهم الخصائص الواجب توافرها فيه في مدير المشروع.
- **تعدد** المهام الأساسية لمدير إدارة المشاريع.
- **تختار** أعضاء فريق التطوير طبقاً لمتطلبات المشروع.
- **تصف** بنية فريق التطوير المناسبة لتطوير المشروع.
- **تحدد** تقسيمات ووظائف فرق العمل الثانوية (المساندة) واللازمة لمشاريع هندسة البرمجيات.
- **تشرح** أهمية التخطيط في كل مشاريع البرمجيات.
- **تشارك** في تخطيط أي مشروع برمجي ووضع هيكلية خطة تنفيذه.
- **تصوغ** مهام تطوير المشروع وتعيّن القائمين على إنجاز كل مهمة.
- **تخطط** مخططات الجدولة الزمنية (مخططات القضبان ، ومخططات الأنشطة) واستخدامها بكفاءة لتمثيل الجداول الزمنية للمشاريع البرمجية.
- **تصنف** الكوارث التي من الممكن أن تحدث في المشاريع البرمجية.
- **تختار** الإستراتيجية المناسبة لإدارة وتحليل ومراقبة كوارث المشاريع البرمجية.

توطئة

عزيزي الدارس، كان الفشل في عدد كبير من مشاريع تطوير برامج الحاسبات في عام 1960م وأوائل عام 1970م بمثابة مؤشر للصعوبات التي تواجهها إدارة هذه المشاريع ، فقد كانت من السوء بحيث لا يمكن الاعتماد عليها، كما أنها كانت تكلف أضعاف ما كان متوقعاً منها، بالإضافة إلى أدائها السيئ. ولم تفشل هذه المشاريع بسبب عدم كفاءة المديرين أو المبرمجين بل على العكس من ذلك فقد جذبت هذه المشاريع عدداً كبيراً من الكفاءات وأصحاب المهارات المتميزة ويكمن الخطأ في أسلوب الإدارة المستخدم في تلك المشاريع ، فقد تم تطبيق أساليب الإدارة التي انبثقت عن الأنظمة الهندسية الأخرى والتي فشلت في إثبات كفاءتها وبالتحديد في تطوير برامج الحاسبات ، وتعد الحاجة إلى الإدارة التي تتميز بالكفاءة أحد الفروق الهامة بين البرامج التي تتميز بقدر عاليٍ من الاحتراف وتلك البرامج التجريبية البسيطة.

ونحن نحتاج إلى إدارة جيدة لمشاريع البرامج لأن هندسة البرامج المتخصصة تكون دائماً عرضة لقيود الميزانية واللوائح والتي يتم وضعها بواسطة المؤسسة التي تقوم بتطوير البرامج ، وقد تم ابتكار وظيفة مدير مشروع البرامج لعدم تجاوز تلك القيود أو تلك اللوائح بالإضافة إلى تحقيقها للأهداف المرجوة. ومديرو مشاريع البرامج مسؤلون عن التخطيط والالتزام بالجدول الزمني الموضوع للمشروع، ويقومون بالإشراف على العمل للتأكد من تنفيذه طبقاً للمواصفات المطلوبة كما يتفقدون التقدم في سير العمل للتأكد من إتمام عملية التطوير في الوقت المحدد لذلك وطبقاً للميزانية الموضوعة ، ومع ذلك فإن الإدارة الجيدة لا يمكن أن تضمن نجاح المشروع بينما الإدارة السيئة غالباً تتسبب في فشل المشروع حيث يتم تسليم البرامج متأخرة عن موعدها كما تزيد التكلفة الفعلية لعملية التطوير عن التكلفة المقدرة لذلك مع الفشل في الوفاء بمتطلبات المشروع المتفق عليها.

1. أهم الاختلافات الجوهرية بين هندسة البرامج والأنماط الهندسية الأخرى

يقوم مديري البرامج بنفس المهام التي يقوم بها مديرو المشاريع الهندسية الأخرى وتختلف هندسة البرامج عن الأنماط الهندسية الأخرى في عدة نواحي وهي :

أ) **المنتج غير ملموس :** حيث يتمكن مدير مشروع بناء السفن أو مشروع هندسة مدنية على سبيل المثال من رؤية المنتج وهو يمر بمراحل التطوير المختلفة حيث يمكن رؤية تأثير عدم الالتزام بالجدول الزمني الموضوع على المنتج مما يؤدي إلى عدم انتهاء البناء كلياً أما البرامج فهي أشياء غير ملموسة حيث لا يمكن رؤيتها أو لمسها ولا يتمكن مديري البرامج من رؤية التقدم في المشروع و يعتمدون على الآخرين لإنتاج المستندات المطلوبة.

ب) **لا توجد عمليات قياسية للبرامج :** لا يوجد لدينا الفهم الكافي عن العلاقة بين العملية البرمجية ونوع المنتج ، على عكس الأنظمة الهندسية التي تتمتع بتاريخ طويل حيث يتم اختبار العملية التي تتم فالمعالجة الهندسية لبعض أنواع الأنظمة مثل الجسور يمكن إدراكها بسهولة وقد تطور الإدراك للعملية البرمجية في السنوات القليلة الماضية بشكل كبير.

ت) **مشاريع البرامجِ الكبيرةِ مشاريعَ "وحيدةَ" في أغلب الأحيان :** تختلف مشروعات البرمجيات الكبيرة عن المشاريع السابقة حيث يكتسب مديري هذه المشاريع مقدار كبير من الخبرة السابقة والتي يمكن أن تستخدم لتقليل الالتباس الموجود في الخطط الموضوعة ومن الصعب توقع المشاكل التي يمكن أن تحدث علاوة على أن التغيرات التكنولوجية السريعة في الحاسبات والاتصالات توفر قدر كبير من الخبرة ويمكن الاستفادة من الدروس التي تكتسب من الخبرة حيث يمكن نقلها إلى المشاريع الجديدة .

ث) **تتطلب صناعة البرمجيات جهوداً ذهنية وفكرية :** بعيداً عما يبذله الإنسان من جهود عضلية ، فهي كالشطرنج الذي يتميز عن بقية الألعاب الرياضية ، ونتيجة للتفكير الذي

يلازم المتخصصين فيها ، تعلم صناعة البرمجيات محترفيها ومتخصصيها الصبر. هذا بالإضافة إلى أنه ليس في صناعة البرمجيات وقت محدد للعمل ، إلاّ في حالات اعتبارية إذ تأتي الأفكار البرمجية التي تتبادر إلى أذهان المتخصصين في أوقات مختلفة.

ج) بناء المنتج : لا يستلزم بناء المنتج البرمجي توفير مواد أولية لإنتاج البرمجيات سوى الحاسوب ، وكذلك لا تستلزم قطع غيار كالتي موجودة في الصناعات الميكانيكية والكيماوية والكهربائية.

ح) السلع المنتجة : وهي البرمجيات لا تعد سلعاً استهلاكية كبقية السلع الناتجة من الصناعات الأخرى تندثر بتأثر عوامل الزمن.

خ) التأثير على البيئة : ربما تنفرد صناعة البرمجيات عن سواها بعدم تأثيرها على البيئة من حيث التلوث.

❑ وبرغم الاختلافات السابقة إلا أنه هناك صفات مشتركة للإدارة لكل من المشاريع الهندسية ومشاريع تطوير البرمجيات ومن أهمها :

- يتم تطبيق العديد من تقنيات إدارة المشاريع الهندسية على عملية إدارة مشاريع برامج الحاسب الآلي.
- تعاني الأنظمة الهندسية المبتكرة ، التي تتسم بتعقيداتها الفنية ، من المشاكل ذاتها التي تواجه أنظمة برامج الحاسب الآلي.

وتعد صناعة البرمجيات اليوم العصر الذهبي لعصر الصناعة ، إذ أصبحت واحدة من الصناعات الإستراتيجية المهمة شأنها بذلك شأن الصناعات العملاقة الأخرى الكيماوية أو الكهربائية أو الميكانيكية. وعلى رغم تميزها عن تلك الصناعات بميزات عديدة إلاّ إنها مرتبطة ارتباطاً وثيقاً بتقنية الحاسب ، وهذه التقنية ذات صلة بالصناعات الكهربائية والميكانيكية والإلكترونية.

وبسبب هذه الفروق الجوهرية بين هندسة البرامج والأنماط الهندسية الأخرى فإنه ليس من المفاجئ أن بعض مشاريع البرامج تتأخر وتتجاوز الميزانية الموضوعة لها والجدول الزمني الموضوع لها. وأنظمة البرامج دائماً تكون جديدة ومبتكرة والمشاريع

الهندسية مثل أنظمة النقل الجديدة غالباً تواجه بعض المشاكل المتعلقة بالالتزام بالجدول الزمني الموضوع لها. وقد أصبح في الإمكان الآن تسليم مشاريع البرامج في الموعد المحدد مع الالتزام بالميزانية الموضوعة لها ويعد موضوع إدارة مشاريع البرامج موضوع متشعب ومركب ولا يمكن تغطيته في وحده واحده ومن ثم فإننا في هذه الوحدة سوف نستعرض لوصف تفصيلي لأهم أربعة نشاطات إدارية هامة وهي : إدارة الموارد البشرية ، تخطيط المشروع ، جدولة المشروع ، وإدارة الكوارث.

أسئلة تقويم ذاتي

1/ ما هي أهم الاختلافات الجوهرية بين هندسة البرامج والأنماط الهندسية الأخرى ؟

2/ أذكر أهم الصفات المشتركة للإدارة لكل من المشاريع الهندسية ومشاريع تطوير البرمجيات ؟

2. إدارة الموارد البشرية

Humanity Resources Management

إن إدارة المشاريع بصفة عامة هي نشاط ذي كثافة بشرية وبنيات تنظيمية كثيرة تستلزم تنظيمها لرفع مهاراتها وقدراتها وزيادة فعليتها ، لذا فإن إدارة الموارد البشرية تُعد حجر الزاوية في نجاح المشاريع حيث إنها تشمل العديد من النشطات والمهام التي تساعد على ذلك ، والتي سوف نتناولها فيما يلي.

1.2 مدير المشروع والنشاطات الإدارية

Project Manager and Management Activities

من الصعب تحديد مستوى قياسي أو مواصفات معينة لوظيفة مدير مشروع البرامج ، حيث تختلف هذه الوظيفة بشكل كبير اعتماداً على المؤسسة نفسها أو على المنتج المراد تطويره. ***ولكن بصفة عامة يجب أن يتصف مدير المشروع البرمجي بعدد من الصفات***

الأساسية ، وهي كما ذكرها "روجر بريسمان" نقلاً عن "Edgeman" كالتالي :

أ) **حل المشاكل (Problem Solving)** : يجب أن يتصف المدير الفعال **بقدرته على :**

- تشخيص القضايا التقنية والتنظيمية وثيقة الصلة بالمشروع.
- بناء حلولاً للمشاكل بشكل نظامي ، أو قيادة أفراد من الفريق وتدريبهم على إيجاد حلول.
- تطبيق الخبرات المكتسبة من مشاريع سابقة ، أو ابتكار عمليات برمجة جديدة ، تمكن من ترجمة مواصفات المتطلبات الخاصة بالمشروع إلى منتج نهائي قابل للتشغيل.
- تغيير اتجاه العمل بمرونة كبيرة إذا ما فشلت المحاولات الأولية لحل المشاكل.

ب) الهوية الإدارية (Managerial Identity) : يجب على مدير المشروع الجيد أن يدير أمور المشروع باحترافية وبثقة تولد شعور بالاطمئنان لباقي أفراد الفريق ، مما يجعلهم أن يبرزوا مواهبهم وانتمائهم للمشروع.

ت) الإنجاز (Achievement) : يجب على مدير المشروع أن يكون قادراً على رفع إنتاجية العمل وجعلها مثالية بقدر الإمكان من خلال مكافأة الشخص المبادر والمنجز ، وكذلك يجب أن يتسم بالقدرة على تشجيع أفراد الفريق على الإبداع حتى عندما يجب عليهم أن يعملوا ضمن قيود مفروضة على عملية التطوير ، وأن يبرهن من خلال أفعاله أنه لن يعاقب من يقوم بمجازفات مسيطر عليها.

ث) التأثير وبناء الفريق (Influence and Team Building) : يجب على المدير الفعال أن يكون قادراً على فهم الإشارات التي تصدر من أفراد فريق العمل سواءً كانت شفهية أو غير شفهية ، وأن يتجاوب معها وفقاً لحاجة الأشخاص الذين أرسلوا هذه الإشارات ، ويجب ألا يفقد السيطرة على الأمور في المراحل العصيبة.

❐ ***ويمكن إجمال أهم الأنشطة الإدارية التي يقوم بها المدير فيما يلي :***

① **كتابة وإعداد عرض للمشروع :** تشمل المرحلة الأولى في مشروع البرامج كتابة عرض لتنفيذ المشروع ، بحيث يتضمن وصف أهداف المشروع وكيفية تنفيذه ، وغالباً ما يشمل تقدير التكلفة والجدول الزمني الموضوع لتنفيذ المشروع مع ضرورة وجود المبرر الكافي لإسناد العقد لمؤسسة أو فريق عمل. وتشكل كتابة العرض خطوة في غاية الأهمية حيث يوجد عدد كبير من المؤسسات المختصة في البرامج والتي تعتمد على الكم الهائل من عروضها التي تم قبولها والعقود التي تم إسنادها إليها ويجب أن تتوفر المهارة الكافية عند كتابة العروض ويمكن اكتساب هذه المهارة بالممارسة.

② **تخطيط المشروع :** يهتم تخطيط المشروع بإعداد وجدولة وتقدير تكلفة المشاريع ، وذلك من خلال التعرف على النشاطات التي تتم فيه والأسس التي يعتمد عليها والمنتجات النهائية له ، ويجب أن يتم وضع الخطة اللازمة لجعل عملية التطوير تتمكن من تحقيق الأهداف المرجوة منها ، ويُعد تقدير التكلفة من أهم هذه الأنشطة ، حيث يهتم بتقدير الموارد المالية المطلوبة لإنجاز المشروع.

③ **مراقبة ومتابعة المشاريع :** تُعد عملية مراقبة المشروع نشاط مستمر طوال فترة تنفيذ المشروع ، ويجب أن يحافظ المدير على وتيرة التقدم في تنفيذ المشروع ومقارنة التقدم الفعلي الذي يحدث في المشروع بالتقدم الذي تم التخطيط له وكذلك التكلفة. وبرغم أن معظم المؤسسات لديها آليات ثابتة لمراقبة المشروع فإن المدير الماهر يمكن أن يشكل واجهة جيدة للمشروع عن طريق المناقشات والعلاقات الحميمة مع فريق العمل ، كما يمكن لعملية مراقبة المشروع غير الرسمية أن تساهم في التعرف على المشاكل التي تواجه المشروع وكذلك الكشف عن الصعوبات حال وقوعها ، على سبيل المثال فإن المناقشات اليومية مع طاقم العمل في المشروع ربما تكشف عن وجود الأخطاء بدلاً من انتظار صدور التقارير المتعلقة بوجود أخطاء وعدم الالتزام بالجدول الزمني الموضوع وربما يقوم مدير المشروع بتعيين بعض الخبراء لحل المشكلة ، وخلال تنفيذ المشروع من المعتاد إجراء مراجعة لسير المشروع تتعلق بالنواحي الفنية وتطور سير المشروع

ومدى التزام المشروع بالأهداف الموضوعة من قبل المؤسسة التي تتولى المشروع وربما تمتد فترة التطوير الخاصة بالمشاريع الكبرى للبرامج سنوات طويلة ، وخلال تلك الفترة ربما تتغير الأهداف الموضوعة من قبل المؤسسة وقد يعني هذا التغيير في الأهداف أن البرنامج لم يعد مطلوباً أو أن متطلبات المشروع غير ملائمة وربما تقرر الإدارة وقف تطوير البرامج أو تغيير المشروع ليتلاءم مع أهداف المؤسسة.

④ **اختيار وتقييم الموظفين :** ومن أهم واجبات مديري المشاريع هو اختيار القوى العاملة التي ستعمل في المشروع وغالبا ما يكون الوضع الأمثل عبارة عن عمالة ماهرة لديها خبرة.

ومن أهم العوامل التي يجب أخذها في الاعتبار عند اختيار أعضاء فريق التطوير ما يلي :

أ- خبرة مجال التطبيق **(Application Domain Experience)** : يجب على عضو فريق التطوير أن يكون على خبرة ودراية وفهم لمجال التطبيق.

ب- خبرة منصة العمل **(Platform Experience)** : قد يكون هذا العامل مؤثراً في حالة استخدام لغة برمجة منخفضة المستوى ، ولكنه بصفة عامة ليس عاملاً مؤثراً في الاختيار.

ت- خبرة لغة البرمجة **(Programming Language Experience)** : يكون هذا العامل مؤثراً في عملية الاختيار في حالة تطوير مشروعات قصيرة الأمد ، حيث لا يكون هناك وقت كافي لتعلم لغة برمجية جديدة.

ث- الخلفية التعليمية **(Educational Background)** : من العوامل المؤثرة في عملية الاختيار حيث تعطي مؤشراً على مدى قدرة الشخص على التعلم الذاتي والقدرة على التطوير.

ج- قدرة الاتصال **(Communication Ability)** : من العوامل المهمة بسبب حاجة المشروع إلى الاتصال الشفهي والمكتوب مع باقي أعضاء الفريق والمدرين والعملاء.

ح- قابلية التكيف **(Adaptability)** : ويمكن الحكم عليها من خلال خبرة المتقدم التي حصل عليها ، وهي عامل مهم لأنه يبين مدى قدرة المتقدم على التكيف مع بيئة العمل وقدرته على التعلم.

خ- الهيئة والوضعية **(Attitude)** : يجب أن يملك عضو فريق العمل هيئة ووضعية إيجابية ورغبة في التعلم لتنمية وزيادة مهاراته.

د- الشخصية **(Personality)** : من العوامل المهمة ولكن من الصعب التعرف عليها وخصوصاً عند إجراء عملية الاختيار ، ولكن بصفة عامة يجب أن يكون المتقدم ذات شخصية متوافقة للعمل ضمن فريق.

☞ ***ولكن في بعض الأحوال يكون المديرين عليهم تحفظات في اختيار العمالة بالمواصفات المطلوبة السابقة للعوامل التالية :***

- وجود قصور في ميزانية المشروع بحيث تغطي رواتب العمالة الماهرة.
- طاقم العمالة الذي لديه خبرة ربما لا يكون متاح للمنظمة داخليا أو ربما تعمل هذه العمالة الماهرة في مشروع آخر.
- ربما تلجأ المؤسسة إلى عمالة غير ماهرة بغرض التدريب وإكساب الخبرات لمنسوبيها.

⑤ **كتابة التقارير وإعداد العروض :** غالباً ما يكون مدير المشروع مسئولاً عن إصدار التقارير المتعلقة بالمشروع إلى كلاً من العميل والمؤسسة التي تتولى تنفيذ المشروع على أن يتم كتابة المستندات بشكل موجز ومتناسق وتحتوى على معلومات هامة ، ويجب أن تكون هذه التقارير والمستندات قادرة على توفير كافة المعلومات المتعلقة بالتقدم في سير المشروع وتعد القدرة على التواصل شفوياً وكتابياً بشكل فعال من أهم مهارات مدير المشروع.

2.2 فرق عمل المشاريع البرمجية Software Projects Teams

إن العنصر المحوري في جميع المشاريع البرمجية هو الأشخاص **(Peoples)** ، حيث يمكن تنظيمهم في فرق عمل متعددة البنيات تتوقف على عدة عوامل للمشروع ، وهي كما ذكرها "روجر بريسمان" نقلاً عن "**Mantei**" **كالتالي :**

- صعوبة المشكلة المراد حلها.
- حجم المشروع.

- عمر الفريق (المدة التي سيبقى خلالها أعضاء الفريق معاً).
- درجة تقسيم المشكلة إلى أجزاء.
- الجودة والموثوقية المطلوبة من النظام المراد تطويره.
- مدى صرامة تاريخ التسليم.
- درجة تبادل الآراء اللازمة للمشروع.

ولتخصيص الموارد البشرية لمشروع يحتاج إلى عدد "**n**" من الأشخاص يعملون لمدة معينة تتوفر الخيارات التالية :

❶ يتم تكليف "**n**" فرداً بـــ "**m**" مهمة وظيفية مختلفة (**m = n**) ، في هذه الحالة يحصل قليل من العمل المشترك ؛ ويكون التنسيق مسئولية مدير البرمجيات.

❷ يتم تكليف "**n**" فرداً بـــ "**m**" مهمة وظيفية مختلفة (**m** أصغر من **n**) ، في هذه الحالة يتم تأسيس فرق عمل غير رسمية ؛ ويمكن تعيين رئيس فريق اعتباطي (تتم عملية الاختيار بطريقة عشوائية) ؛ ويكون التنسيق بين هذه الفرق مسئولية مدير البرمجيات.

❸ يتم تكليف "**n**" فرداً في فرق عمل عددها (**t**) ، وفي هذه الحالة يتم تكليف كل فريق عمل بمهمة وظيفية واحدة أو أكثر ، ويكون لكل فريق بنية خاصة محددة معتمدة لجميع الفرق العاملة في المشروع ، ويقوم كل من الفريق ومدير المشروع بضبط عملية التنسيق.

ومن الواضح أن التطوير المنظم لمشاريع هندسة البرمجيات الكبيرة والحديثة يتطلب فرق عمل متكاملة (الخيار رقم 3 السابق) . فمن المفيد في كثير من الأحوال تبني فكرة فرق العمل هذه في بداية أي مشروع تطوير برمجيات ابتداءً بفرق عمل ثانوية متعددة. والحاجة إلى هذه الفرق الثانوية ليست شكلية صورية (ليست تحصيل حاصل فقط) بل هي حاجة ماسة حتى ولو كان الفريق شخص واحد ، و مع ذلك فإن دور أو مهام هذه الفرق مازال يحتاج إلى إنجازه بغض النظر عن تنظيم الفرق المختلفة.

ملحوظة

نلاحظ أن العديد من هذه الفرق غير ظاهرة للعيان خلال مدة إنجاز المشروع. ففي بعض من هذه المشاريع ، واحدة من فرق العمل أو أكثر تتكون من شخص واحد يتولى القيام بعدة مهام موزعة بين عدة مشاريع أخرى ، غالبا ما تكون فرق العمل الثانوية كبيرة جدا تتشكل حسب التوزيع الجغرافي خاصة إذا كانت تعمل في عدة مشاريع. وعادة ما يطلب من أعضاء الفريق تقديم تقرير لشركات مختلفة من خلال مشاريع تطوير البرامج التعاونية.

1.2.2 التقسيم المتعارف عليه لفرق العمل اللازمة لتطوير البرمجيات والأعمال المنوطة بها

- **فريق تحليل النظم (System Analysis Team)** : هذا الفريق مسئول عن إعداد دراسة جدوى المشروع. وتشمل دراسة الجدوى الفرعيات التالية :
  - تحليل التكاليف.
  - تقدير المردود.
  - تقدير الصعوبات الهندسية للمشروع.

وبعد إنتاج دراسة الجدوى الاقتصادية للمشروع يجب على فريق التحليل أن يتواصل مع فريق المتطلبات بتلقي مرئياتهم حول المشروع المقترح فإذا ما كانت عمليات تطوير البرنامج تكرارية، كما هو الحال في موديل التطوير السريع والنماذج اللولبية (الحلزونية) فإن ذلك يتطلب تكرار التفاعل وتلقي المرئيات من فرق العمل الأخرى.

- **فريق التخطيط (Planning Team):** هذا الفريق مسئول عن تطوير المخطط الإداري الكلي للمشروع ليتمشى مع سير العمل في المشروع حسب الخطة الزمنية المقترحة للأنشطة المختلفة.
- **فريق المتطلبات (Requirement Team):** مهام هذا الفريق هي مقابلة العميل وتقديم قائمة تفصيلية لمتطلبات المشروع. وهذا يتطلب عقد لقاءات رسمية وغير رسمية مع العملاء للتوصل إلى الصيغة النهائية لمتطلباتهم من قائمة الطلبات الأولية غير الدقيقة والكاملة. وإذا لم يتوفر عملاء فإنه يجب على فريق المتطلبات الحصول على نفس المعلومات من شخص أو أكثر يتوقع استخدامهم للبرنامج. وإذا لم يتوفر شخص يتوقع استخدامه للبرنامج ، فمن المحتمل وجود عملاء بدلاء في ذلك المكان. و بعد إنتاج (الانتهاء من) طلبات النظام فإنه يجب على هذا الفريق التفاعل (التكامل) مع فريق التصميم بتلقي مرئياتهم. وإذا ما كانت عمليات تطوير البرنامج تكرارية كتسريع النموذج الأصلي للبرنامج والنماذج اللولبية فإن ذلك يتطلب تكرار التفاعل مع ردود الفعل من فريق التصميم.
- **فريق تصميم النظام (System Design Team) :** مهمة هذا الفريق هي العمل لإنتاج تفاصيل تصميم النظام بعد استلام قائمة المتطلبات من فريق المتطلبات. فإذا ما كانت عمليات التطوير تستخدم نموذج الشلال التقليدي ، فإنه يجب على هذا الفريق التزود بمرئيات فريق المتطلبات حول الصعوبات التي قد تواجههم. وبعد الانتهاء من التصميم ينبغي على الفريق التواصل مع فريق الإنجاز (فريق التنفيذ) والأخذ بملاحظاتهم ومرئياتهم.
- **فريق التنفيذ (Implementation Team) :** مهام هذا الفريق تنفيذ البرنامج المصمم عن طريق فريق تصميم النظام. وينبغي على فريق التنفيذ التواصل مع فريق الاختبار والدمج (التكامل) والأخذ بملاحظاتهم ومرئياتهم.
- **فريق الاختبار والدمج (التكامل) (Testing and Integration Team) :** مهمة هذا الفريق صياغة (تحديد) حالات الاختبار لوحدات القياس (مراحل تنفيذ البرنامج) و

الأنظمة المقترحة عن طريق فريق التنفيذ ، وقد يختار هذا الفريق بعض المراحل التنفيذية للنظام لاختبارها بالاتفاق التبادلي مع فريق التنفيذ. وبعد الانتهاء من مخطط الاختبار واختبار مراحل النظام يتولى هذا الفريق مهمة دمج مراحل النظام ضمن نظام العمل. كما ينبغي على فريق الاختبار والدمج الأخذ بملاحظات فريق التنفيذ. فريق الدمج مسئول عن التحكم الداخلي للوثائق **(Internal Control for Documents "ICD")** والذي يصف بدقة العلاقة الداخلية بين مكونات النظام الأساسي.

- **فريق التدريب (Training Team)** : هذا الفريق مسئول عن إنتاج وتطوير مواد ومناهج التدريب.
- **فريق التسليم والتركيب (Delivery and Insulation Team)** : هذا الفريق مسئول عن تسليم وتركيب النظام.
- **فريق الصيانة (Maintenance Team):** هذا الفريق مسئول عن صيانة النظام بعد تسليمه وتركيبه لدى العميل (المستفيد). بعد تسليم وتركيب النظام ينبغي على هذا الفريق التواصل مع فريق التنفيذ والأخذ بمرئياتهم.
- **فريق ضمان الجودة النوعية (Quality Assurance Team):** مهمة هذا الفريق تتلخص في أمرين أولهما وضع وتحديد مواصفات ومعايير متابعة فريق المشروع ومعايرة عمل النظم الجديدة المستحدثة و تحديد معايير أداء النظام المنتج ، وثانيهما تقديم (طرح أو استيعاب) فريق عمل المشروع لهذه المعايير. معايير الممارسة الصناعية (التطبيق الصناعي) لا تعدو سوى معلومات لإحاطة هذا الفريق فقط وليس لمناقشتها أو تقاسمها مع العملاء (الزبائن). هذه المعلومات يمكن أن تنشر وتعلن كحق قانوني ولكن ليس للتشهير والشكوى. هذه المعلومات تقدم لمدير المشروع لاستخدامها للحكم على أداء فريق الجودة النوعية **(QA)**.
- **فريق التقييس (Metrics Team)** : هذا الفريق مسئول عن المحافظة على إحصائيات أداء فرق عمل المشروع. بناء على ما يجمعه من معلومات بطرق رسمية ، بعض هذه الإحصائيات تحتوي على : عدد طلبات الصيانة المقدمة (المطلوبة) ، وعدد

طلبات الصيانة المستلمة ، وعدد الخطوط المشفرة المكتوبة ، وعدد ساعات إنجاز كل مهمة ، و القدر المنتج بواسطة كل جزء (آلة) من الإصدار الجديد للنظام. وينبغي على الفريق التواصل مع فرق عمل المتطلبات ، والتصميم ، والتنفيذ ، والاختبار والتكامل والصيانة ، وذلك لتقييم الجودة والفعالية (كفايته) إضافة إلى مرئيات وملاحظات (ردود فعل) فرق العمل الأخرى.

- **فريق التوثيق (Documentation Team) :** هذا الفريق مسئول عن توثيق المشروع متضمنا طلبات التوثيق الخارجية، التصميم، شفرة المصدر (تشفير المصدر)، وأي وثائق داعمة للمشروع.
- **فريق إدارة النظام (System Administration Team) :** هذا الفريق مسئول عن ضمان متابعة أن عتاد الحاسب والبرمجيات والشبكة الداعمة تقوم بالعمل كما هو مطلوب منها من فرق العمل المختلفة.
- **فريق عمل المشروع (Project Labor Team) :** هذا الفريق عادة ما يتضمن فريق الشبكة الإدارية.
- **فريق إعادة الممارسة والاستخدام (Reuse and Re-engineering Team) :** هذا الفريق مسئول عن انتقاء (اختيار) واستخدام أجزاء وعناصر الأجزاء الجاهزة من البرمجيات والمعدة لأنظمة من قبل. وهذا قد يكون ضروريا إذا ما كان النظام يعتمد على بعض الرموز (الشفرات) القديمة والتي يجب تغييرها (استبدالها) تمشيا مع التقدم التقني الجديد.

هذا بالإضافة إلى فرق عمل مكونة من شخص واحد ، وهي :

- **مُقيِّم العلاقة بين المستخدم للنظام والحاسب الآلي :** هذا الشخص مسئول عن تقييم نوع العلاقة (الارتباط) بين النظام والمستخدم، على الأقل يكون لدى المقيم أو المقيمة نموذج ذهني (عقلي) لتوقع مهارات كل من المستخدمين المبتدئين و ذوي الخبرة للنظام بشكله النهائي.

- **مسئول دعم الأداء :** هذا الشخص مسئول عن ضمان جودة ودقة أداء بيئة دعم النظام ، ويعمل ضمن منظومة عمل أو شبكة إدارية متكاملة.
- **اقتصادي النظام :** هذا الشخص مسئول عن تطوير واستخدام النماذج المناسبة لتقييم تكلفة النظام ، موارد العتاد والبرمجيات والوقت اللازم لإنجاز المشروع.
- **أمين مكتبة المشروع :** هذا الشخص مسئول عن متابعة وحفظ جميع وثائق المشروع.

وهنالك أشخاص آخرين متخصصين (قد يُحتاج إليهم) في مراحل تطوير النظام خاصة في إعادة استخدام النظام.

أسئلة تقويم ذاتي

1/أذكر الصفات الأساسية التي يجب ان يتصف بها مدير المشروع البرمجي ؟

2/ ما هي أهم الأنشطة الإدارية التي يقوم بها مدير المشروع ؟

3/ أذكر أهم العوامل التي يجب أخذها في الاعتبار عند اختيار أعضاء فريق التطوير للمشروع ؟

4/ في بعض الأحوال يكون المديرين عليهم تحفظات في اختيار العمالة بالمواصفات المطلوبة ، اذكر أهم هذه التحفظات.

5/ لخص التصنيف المتعارف عليه لفرق العمل اللازمة لتنفيذ مشاريع هندسة البرمجيات والأعمال المنوطة بكل فريق ؟.

3.2 طبيعة فرق عمل المشروع The Nature of Project Teams

من الواضح أن العديد من المشاريع العالمية تخللها تغيرات وتبديل في العاملين بها خلال فترة إنجازها والتي من أهمها :

① **أعضاء المشروع بحاجة للتغيير :** الكثير من الأفراد بحاجة للعمل في مراحل التشفير والفحص والتكامل أكثر من الحاجة لهم في الطلبات الابتدائية المتعلقة بفاعلية المشروع. غالبا المشروع بحاجة إلى أفراد أقل للصيانة خاصة إذا كان النظام يتوقع أن يكون عمره قصيرا.

② **المشروع قد يستمر لمدة طويلة :** مثلا في مشروع ناسا الوطني لقمر الاستكشاف بالأشعة فوق البنفسجية الحصول على معلومات علمية يستمر لمدة عام ، لذا فإن فريق صيانة برنامج التشغيل الأصلي يتوقع له العمل لمدة عام بعد وصول القمر إلى مداره حول الأرض. عوضا عن بقاء الأقمار العلمية لمدة 19 عاما منتجة معلومات علمية كثيرة وما يتبعها من أبحاث علمية في السنوات الأخيرة مقارنة للسنة الأولى من تشغيله. ومن الواضح أن هنالك تغيير و تبديل في فرق العمل أو الأشخاص خلال هذه المدة.

③ **الأفراد يغيرون أعمالهم :** الكثير من مهندسي النظم يفضل الانضمام إلى التقنية الجديدة ومستوى المشاريع بعد استخدام أوضاع تقنية فاصلة.

④ **خسارة عقود الشركات :** العديد من الهيئات المالية التجارية عمدت إلى الاندماج و إعادة ترتيب وحداتها المهنية التجارية. و في بعض الحالات القصوى تلجأ الشركات لوقف نشاطها التجاري. إذا عملت الشركة في مشروع تعاقدي ثانوي فاشل فإن فريق عمل المشروع سيتغير.

⑤ **الأفراد العاملين عرضه للإحالة للتقاعد أو للتعرض لبعض الأمراض المعقدة.**

⑥ **تعيين أفراد جدد للحصول على أفكار جديدة ومهارات تقنية داخل الهيئة.**

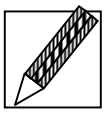
4.2 أدوات إدارة المشروع Project Management Tools

تشمل أدوات إدارة المشروع **التالي :**

أ) أدوات الـــ CASE والتي تساعد في إنجاز المهام التالية :

- إنشاء وتعديل جدول (برنامج) المشروع.
- جعل البرنامج (الجدول) قائما وبالتالي فإن مدير المشاريع المختلفة يمكن أن يفحصها (يراقبها) ويعمد إلى إجراء بعض التغييرات الضرورية.
- دعم المورد المهم الفعال.
- تعريف البنود الحاسمة (المؤثرة) في الجدول (البرنامج) وتلك البنود المبنية عليها.
- جعل الوثائق في متناول جميع الأشخاص المرخص لهم.
- دعم الطرق المختلفة لإدارة المشروع ، لذا فإن الأدوات يمكن أن يتم تطويعها لتؤدي الاحتياجات الإدارية لتنفيذ الأعمال.

تدريب (1)



ولكن **عزيزي الدارس** ماذا يجب عليك أن تتوقع من البرامج الداعمة لإدارة المشروع؟

نشاط

عليك الآن **عزيزي الدارس** البحث في شبكة الإنترنت عن أمثلة لبعض من أدوات هندسة البرامج المساعدة(**CASE)** التجارية النموذجية لإدارة المشاريع ، ويمكن الحصول على معلومات مستفيضة حول هذا الموضوع من المواقع التالية:

http://www.microsoft.com/project/
http://www.projectrak.com/screens
http://www.spr.com/html/knpwlwdgeplan.htm

ب) أدوات الاتصال (Communication Tools) :

إضافة إلى أدوات الـــ **CASE** هنالك أدوات وتقنيات اتصال معينة تعين على إدارة المشروع ، مثل : شبكة الإنترنت **(Internet)** : والتي تُعد أعظم القوى البارزة في مجال الاتصالات الآن من خلال استخدام تقنياتها المختلفة مثل : البريد الإلكتروني ، ومواقع الويب ، والبحث من خلالها ، وعقد المؤتمرات الفيديوية من خلالها ، ..... وخلافه. وكذلك **الشبكات الداخلية (Intranets)** : والمعروفة بالإنترانت والتي لا تعدو سوى أنظمة حاسوبية تعمل ضمن شبكة داخلية مستخدمة مقاييس ومعايير شبكة الإنترنت ، ولكنها محجوبة عن التفاعل الدولي إلا في ظل برمجيات حماية خاصة. هذا بالإضافة إلي أنظمة المؤتمرات المعتمدة على الفيديو ، وأدوات الاتصال التقليدية.

وهكذا فإن الهيئة الإدارية قادرة على تصريف أعمالها من خلال تلك الأدوات مع بعض الحذر تجاه أمن وسائل تطبيقاتها التجارية. وإذا كان من المألوف طلب الوثائق وبرامج (جداول) المشروع من خلال الشبكة الداخلية للهيئة (المنظمة) وأحيانا من خلال شبكة الإنترنت فإن العديد من الأشخاص يمكنهم العمل حول جزء من نفس الوثيقة في نفس الوقت ، ومشاركة الآخرين الراغبين بذلك عند الحاجة.

فحص ومعاينة المتطلبات والتصميم ينبغي أن يكون من خلال وثائق مكتوبة أساسا وعملية الفحص والمعاينة ينبغي أن تتم كالمعتاد. دون النظر إلى إمكانية توفر الوثائق عبر الإنترنت. وبالطبع هنالك بعض المشاكل المرافقة لاستخدام الملفات الإلكترونية **هذه المشاكل يمكن الإقلال منها إذا لم تتمكن من إزالتها كليا بإتباع الإرشادات التالية :**

- تحديد التصميم القياسي المستخدم لحفظ ملفات الرسوم البيانية والمخططات.
- التأكد من أن التصاميم متوافقة (متناغمة) مع المتصفح للتمكن من استخدامها.
- تحديد آلية لاتخاذ الإجراءات الوقائية (المحيطة) للتغذية الاسترجاعية عند ضعف ووهن الوثائق المدرجة على الشبكة.

- تحديد وعلى وجه الخصوص ما إذا كانت الوثائق الورقية ونسختها الإلكترونية ستظل مطلوبة.
- التأكد من أن الوثائق على الشبكة يمكن إخضاعها للفحص التنقيحي والترتيبات الإدارية.
- الأدوات القياسية للترتيب الإداري للنص الأساسي ينبغي استخدامه لترميز المصدر ومتطلبات توثيق النص.
- اتخاذ الإجراءات الوقائية لتسهيل التغذية الإرجاعية بمعنى تسهيل إمكانيات الإرسال بـــ **HTML** مما يجعل تبادل المعلومات أكثر سهولة.
- استخدام البريد الإلكتروني لتنسيق اللقاءات والاجتماعات وتذكير أعضاء فريق المشروع حول تغيير الوثائق.
- تحديد ما إذا كانت غرف المحادثة سوف تستعمل في إدارة المشروع.

لذا فاستخدام الإنترنت والإنترانت كوسيلة اتصال ونشر أمر فعال ، فالعديد من مشاريع تطوير برامج للمركبات الفضائية لدى ناسا كمثال يتم عن طريق إرسال جميع الوثائق المطلوبة ، والتصاميم ، ومحاضر اجتماعات المشروع ، وبرامج (جداول) المشروع ، والبنود الرئيسية ، عبر الإنترنت كوسيلة سهلة للتواصل بين أفراد المشروع. وهذا يقلل من عدد الأرفف المكتظة بالكتب والمراجع والملفات المستخدمة سابقا لحفظ التقارير والملاحظات والأوراق المبعثرة المتضمنة وثائق المشروع. بالإضافة إلى أن قليلا من هذه الوثائق سيتعرض للفقد والضياع.

بالإضافة إلى ما سبق ، هنالك وسيلة هامة تستخدم في بعض النظم الإدارية للمساعدة في تنسيق الاجتماعات واللقاءات وتنقيح ومراجعة المشاريع المختلفة تسمى **مكونات المجموعات (Groupware)**. ومصطلح "مكونات المجموعات" يعني تمكين مجموعة من الأشخاص من دراسة ومعاينة وثيقة واحدة في آن واحد عبر الشبكة. وعلى سبيل المثال: بإمكان مهندسو المشروع الاجتماع ببعضهم البعض في غرفة بها شاشات عرض كبيرة مع تمكن كل منهم من الدخول إلى حاسبه الشخصي أو محطة عمله ، عوضا عن ذلك يمكن استبدال الموقع عبر الشبكة. جميع الحاسبات كل على حده يمكن

أن تصبح شبكة ذات برنامج تشغيل يتحكم في العبور للوثائق فيما يعرف بمكونات المجموعات. ولعل أفضل مثال معروف عن مكونات المجموعات مذكرات اللوتس ومنتجات أنظمة فينتانا، هذا وقد أخذت المايكروسوفت نفس الاتجاه بطرحها للعديد من أدوات التكامل.

ومكونات المجموعات قد تذهب أحيانا أبعد من عبور الإنترنت إلى حد ما وذلك بتمكين أي فرد من أفراد المشروع من تغيير الوثيقة العامة أو قصر وحصر القدرة على تغيير (تبديل) الوثيقة في شخص واحد.

هذا الاستخدام لمكونات المجموعات أيضا أكثر سلاسة من بلوغ مستوى نظام التشغيل ، وذلك لأنه لا حاجة لتشكيل مجموعات جديدة من المستخدمين للسماح لمجموعات محددة من مهندسي برنامج التشغيل بتغيير (تبديل) الوثيقة العامة. ونلاحظ أن استخدام الإنترنت أو الإنترانت الداخلية للمؤتمرات المصورة تلفزيونيا (المتفلزة) تتضمن نفس خصائص ومميزات مكونات المجموعات خاصة إذا كان هنالك آلية محددة لتسجيل الصوت و الصورة في المؤتمر.

أسئلة تقويم ذاتي

1/ "العديد من المشاريع العالمية تتخللها تغيرات وتبديل في العاملين بها خلال فترة إنجازها " ، ما أهم هذه التغيرات ؟.

2/ "هناك مجموعة من الأدوات أو الوسائل التي تساهم على حد كبير في إدارة المشروع" ، ما هي أهم هذه الأدوات وكيف تساهم في إدارة المشروع؟.

3/ ماذا يعني مصطلح مكونات المجموعات(Groupware) ؟.

3. تخطيط المشروع البرمجي Software Project Planning

1.3 أنواع خطط المشروع Types of the Project Plans

تعتمد الإدارة الفعالة لمشروع برامج الحاسب الآلي بشكل كبير على التخطيط الجيد ويجب على مدير المشروع أن يتوقع حدوث عدد من المشاكل وإعداد الحلول المناسبة لهذه المشاكل ، ويتم إعداد الخطة عند بداية المشروع وتستخدم كدليل للمشروع ، ويجب أن تكون الخطة المبدئية للمشروع تحتوى على كافة المعلومات الهامة المتعلقة بالمشروع وتتطور كلما زادت وتيرة العمل في المشروع .

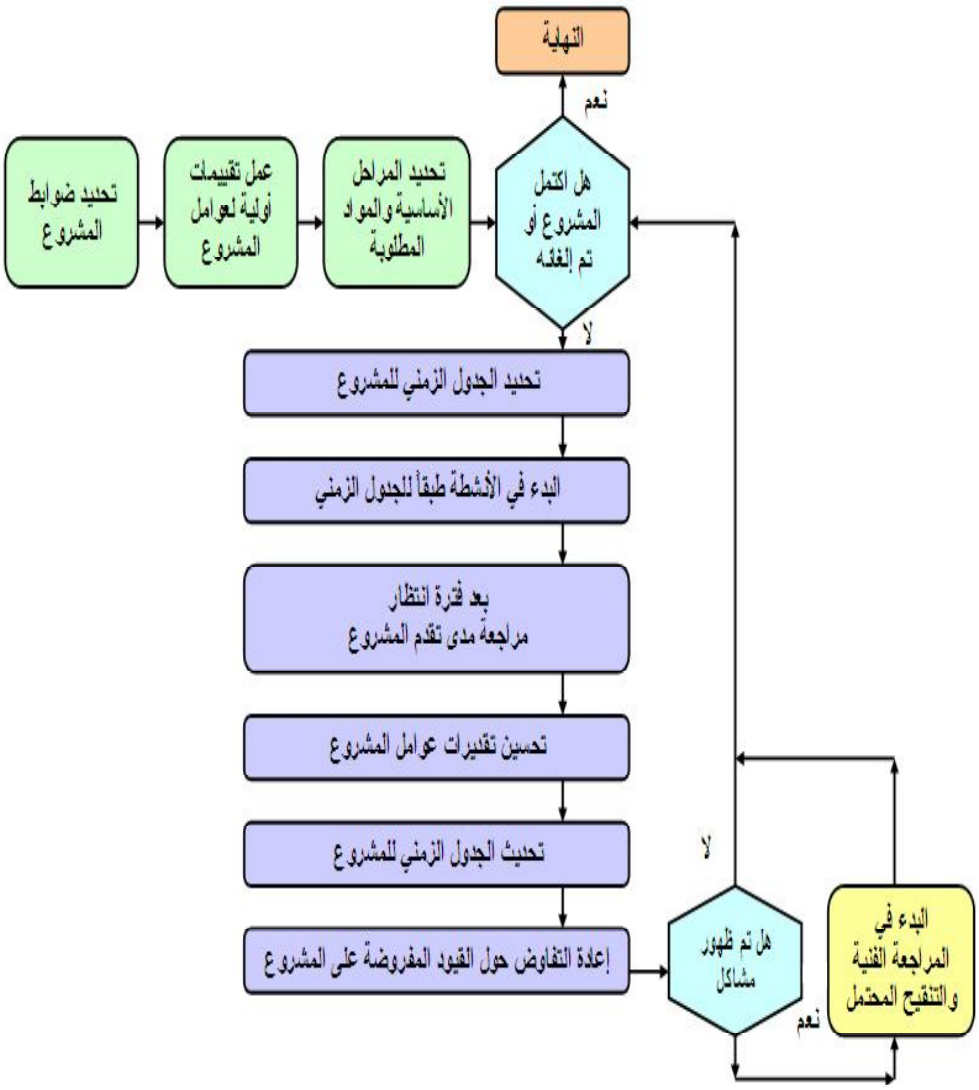
وتخطيط المشروع عملية معلقة يتم الانتهاء منها عند إكمال المشروع نفسه فقط. وتُعد أهداف المشروع أهم عامل يجب أن يؤخذ في الاعتبار عند وضع خطة تنفيذ المشروع ، وعندما تتغير هذه الأهداف يجب أن تتغير بالتالي خطة المشروع. وتبدأ عملية التخطيط مع تقييم الحدود الموضوعة للمشروع (مثل تاريخ التسليم وفريق العمل المتاح والميزانية المتاحة...الخ) والتي تؤثر على المشروع، ويتم تنفيذ هذه البنود بالتزامن مع تقدير أسس المشروع مثل تركيب المشروع وحجمه وتوزيع الوظائف ، وبعد ذلك يتم التعرف على أسس التقدم في المشروع والمنتج النهائي للمشروع . والجدول رقم 4.1 يبين أنواع لخطط المشاريع(**Types of Project Plans**)،ووصف لها.

الخطة (Plan)	الوصف (Description)
خطة الجودة	وتشمل وصف إجراءات ومعايير الجودة التي سيتم استخدامها في المشروع.
خطة التحقق	وتشمل توضيح الأسلوب والمصادر والجدول الزمني المستخدم للتحقق من النظام.
خطة إدارة التهيئة	وتشمل توضيح الإجراءات المتعلقة بإدارة التهيئة والبنية المطلوب استخدامها.
خطة الصيانة	وتشمل التكهن بمتطلبات الصيانة الخاصة بالنظام ، وتكاليف الصيانة والجهود المطلوبة.
خطة تطوير مستوى الموظفين	توضح كيف يتم تطوير مهارات وخبرات أعضاء فريق العمل في المشروع.

جدول رقم 4-1 : تصنيف خطط المشاريع

2.3 خطوات تخطيط المشروع Steps of the Project Planning

الخطوات الموضحة بالشكل 4.1 تبين الحلقة الكاملة لتخطيط المشـــروع ، حيث يبدأ التخطيط بتحديد ضوابط المشروع ، ومن ثم عمل تقييمات أولية لعوامله وتحديد المراحل الأساسية والمواد المراد توريدها لتطويره ، ومن ثم يتم وضع جدول زمني للمشروع والبدء في تنفيذ النشاطات الموجودة في ذلك الجدول ، وبعد فترة من الوقت (غالباً حوالي 2-3 أسابيع) يتم مراجعة التقدم في سير المشروع مع ملاحظة وجود أي خلل في سير المشروع.



شكل 4-1 : شكل توضيحي لخطوات تخطيط المشروع البرمجي

وحيث إن التقييم المبدئي للمشروع يكون تجريبي فإنه يتم إجراء بعض التعديلات على خطة المشروع ويقوم مديري المشروع بتنقيح الافتراضات المطروحة حول المشروع عندما تصبح المعلومات متاحة ويعيدوا تخطيط جدول المشروع وإذا تأخر المشروع يجب عليهم التفاوض مع العميل بشأن شروط المشروع التي تم تحديدها من قبل وإذا فشلت عملية التفاوض ولم يتم الالتزام بالجدول الموضوع ربما يتم تعليق الناحية الفنية للمشروع ، والغرض من ذلك هو إيجاد بعض وسائل التطوير البديلة والتي تتوافق مع الشروط الموضوعة للمشروع وتلبي بنود جدول المشروع ولا يفترض مديري المشروع الذين يتميزون بالكفاءة بأن كل شئ سيسير على ما يرام فغالباً تظهر بعض المشاكل خلال تنفيذ المشروع وتكون الافتراضات والجدول المبدئي تميل إلى التشاؤم أكثر من التفاؤل ويجب أن يوجد هناك خطة طوارئ جنباً إلى جنب مع الخطة الأساسية.

3.3 هيكلية خطة المشروع The Project Plan Structure

توضح خطة المشروع مصادر التمويل المتاحة للمشروع وجدول لتنفيذ العمل والاحتياطيات الواجب تواجدها في حالة تعطل العمل وفي بعض المؤسسات تكون خطة المشروع عبارة عن مستند واحد تشمل جميع أنواع الخطط ، وفي حالات أخرى تتم خطة المشروع بشكل كبير مع عملية التطوير وتعتمد تفاصيل خطة المشروع على نوع المشروع والمؤسسة بينما تشمل معظم الخطط **الأقسام التالية :**

❶ مقدمة (Introduction) :

تحتوى على وصف مختصر لأهداف المشروع والقيود المفروضة على تطويره مثل(الميزانية والوقت ...) والتي تؤثر على إدارة المشروع.

❷ تنظيم المشروع (Project Organization) :

تحتوى على وصف الطريقة التي بواسطتها يتم تنظيم فريق التطوير والأدوار التي يلعبها كل من عضو من أعضاء الفريق.

❸ تحليل الكارثة (Risk Analysis) :

تحتوى على وصف الكوارث المتوقعة في المشروع مع اقتراح استراتيجيات لتقليل حدوث الكوارث.

❹ متطلبات موارد الأجهزة والبرامج :

(Hardware & Software Resource Requirements) :

تقوم بتحديد الأجهزة والبرامج المطلوبة لتنفيذ عملية التطوير مع تقدير سعر الأجهزة عند شراؤها وتحديد الجدول الزمني للتوصيل.

❺ تقسيم العمل (Work Breakdown) :

يصف تقسيم المشروع إلى أنشطة وما تقوم به هذه الأنشطة من إنتاجية وطرق تسليمها.

❻ الجدول الزمني المشروع (Project Schedule) :

يوضح العلاقة الموجودة بين النشاطات المختلفة في المشروع والوقت المطلوب لتنفيذها والتزام أفراد الفريق بأداء العمل المنوط به.

❼ آليات المراقبة والتقارير (Monitoring & Reporting Mechanisms) :

توضح التقارير الإدارية التي يتم إصدارها وآليات متابعة المشروع المستخدمة.

ويجب أن يتم تنقيح خطة المشروع بشكل دوري خلال تنفيذ المشروع كما يتم تغيير بعض أجزاء المشروع مثل جدول المشروع بينما تظل الأجزاء الأخرى ثابتة كما أن تنظيم المستندات يجب أن يتم بعناية شديدة.

4.3 تنظيم الأنشطة (Activities Organization)

يحتاج مديري المشروع للمعلومات الكافية للحكم على سير العمل وتقدير التكلفة أو تحديث جدول المشروع وأسس المشروع ، وحيث إن البرامج منتج غير ملموس فإن هذه المعلومات يجب أن يتم توفيرها على شكل مستندات تصف حالة البرامج التي يتم تطويرها ، وبدون هذه المعلومات فمن المستحيل لمدير المشروع الحكم على سير المشروع إذا كان يتم بشكل مناسب أم لا ، لذا يجب تنظيم أنشطة المشروع **(Project Activities)** لإنتاج مخرجات ملموسة وواضحة لأغراض الإدارة وتقييم تطور المشروع. ويتم ذلك خلال تخطيط المشروع حيث يتم تعيين مجموعة متتالية من المعالم الأساسية للمشروع **(Milestones)** ، حيث يُعرف المَعلَّم الأساسي **(Milestone)** بأنه عبارة عن نقطة النهاية **(End Point)** وغاية لنشاط معين من أنشطة عمليات البرمجيات **(Software Process Activity)** ، وعند كل مَعلَّم أساسي يجب أن يكون هناك مُخرج رسمي **(Formal Output)** مثل : تقرير يقدم للإدارة ، ويجب أن تكون هذه التقارير بسيطة وتعبر عن المنجزات التي تحققت خلال النشاط الخاص بها. من جهة أخرى تُعرف نتائج المشروع القابلة للتسليم **(Deliverables)** بأنها مخرجات المشروع التي تُسلم للعميل في نهاية كل مرحلة أساسية من مراحل المشروع مثل مرحلة تحديد مواصفات المتطلبات **(Requirement Specification)** ، مرحلة التصميم **(Design Phase)** ، .... وخلافه. وتُعد مخرجات المشروع القابلة للتسليم **(Deliverables)** دائماً من المعالم الأساسية **(Milestones)** ، ولكن ليس بالضرورة أن تكون المعالم الأساسية مخرجات قابلة للتسليم ، حيث إنها يمكن أن تكون نتائج داخلية يستخدمها مدير المشروع لمتابعة تقدم المشروع ولكنها لا تُسلم للعميل.

ولتنظيم الأنشطة وتعيين المعالم الأساسية فإن المراحل الأساسية لتطوير المشروع أو عمليات البرمجية لا بد من تقسيمها إلى أنشطة فرعية مع بيان مخرجاتها (المعالم الأساسية) ، مثل الموضح بالشكل رقم 4-2 ، والذي يبين الأنشطة المختلفة ومخرجاتها لمرحلة تحديد مواصفات

المتطلبات ، وتعُد وثيقة متطلبات المستخدم ووثيقة مواصفات متطلبات النظام مخرجات قابلة للتسليم **(Deliverables)**.

أسئلة تقويم ذاتي

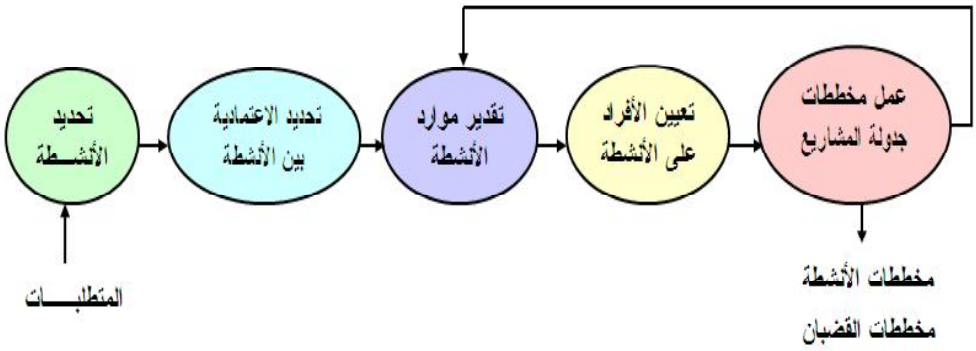
1/ ارسم جدول بين فيه تصنيف لخطط المشاريع ووصف لها ؟.
2/ وضح بالرسم خطوات تخطيط المشروع البرمجي ؟
3/ ما هي أهم الأقسام التي تشملها خطط المشاريع ؟.
4/ أشرح كيف يمكن لمدير المشروع الحكم على سير المشروع إذا كان يتم بشكل مناسب أم لا ؟.

4. جدولة المشروع Project Scheduling

1.4 أنشطة عملية جدولة المشروع

Project Scheduling Process Activities

جدولة المشروع هو هدف رئيس لمديري مشاريع البرامج حيث يقوم مدير المشروع بتقدير الوقت والتمويل اللازم لأداء الأعمال المطلوبة وتنظيمها بشكل منطقي. وتُعد عملية تقييم الجدولة عملية معقدة نظراً لأن المشاريع المختلفة تستخدم طرق تصميم مختلفة ولغات تنفــيذ مختلفة ، وإذا كان المشروع متقدم تقنياً فإن التقديرات المبدئية تكون متفائلة دائماً حتى عندما يحاول المديرون الأخذ في الاعتبار جميع الاحتمالات ، وبهذا الخصوص فإن جدولة البرامج لا تختلف عن جدولة أي نوع آخر من المشروعات مثل المطارات أو الجسور أو حتى السيارات ، حيث يجب تحديث الجداول بشكل مستمر حتى تصبح كافة المعلومات متاحة. وتتضمن عملية جدولة المشروع جميع الأعمال التي تؤدى خلال المشروع وتحديد الوقت اللازم لإكمال المشروع وبعض النشاطات التي يتم تنفيذها في المشروع بالتوازي ، ويجب أن تتضمن جداول المشروع هذه النشاطات المتوازنة وتقوم بتنظيم العمل ، لذا فإنه يجب استخدام القوى العاملة بشكل متوازي ،



شكل 4-3 : شكل توضيحي لعملية جدولة المشاريع

ويجب أن يتم تجنب وضع المشروع في مأزق مثل حدوث تأخير في التنفيذ أو ما شابه ذلك.

ويجب أن تستمر نشاطات المشروع على الأقل أسبوع ويجب أن يتم قضاء وقت أطول في تقدير وتنقيح العمل. ومن المفيد تحديد الفترة الزمنية القصوى التي يتم قضاءها في أداء نشاط معين بحيث لا تزيد عن 8-10 أسابيع. وإذا استغرق أداء نشاط ما أكثر من ذلك يجب أن يتم تقييم هذا النشاط ويجب أن يتم تضمين التوقعات التي تتعلق بالمشاكل المتوقعة ، كما يجب إضافة خطة طوارئ لمواجهة المشاكل غير المتوقعة ، ويعتمد عامل الطوارئ على نوع المشروع ومعايير العملية ذاتها (مثل موعد التسليم ومواصفات المشروع ... الخ) وجودة وخبرة مهندسي البرامج الذين يعملون في المشروع. وبالإعتماد على الخبرات السابقة يمكن إضافة 30% إلى التقدير الأولي للمشاكل غير المتوقعة وكذلك 20% لتغطية الأشياء الأخرى التي لم تؤخذ في الاعتبار. والشكل 4.3 يمثل عملية جدولة المشروع ، حيث يتم تعريف الأنشطة **(Identify Activities)** من خلال المتطلبات ، ومن ثم تعريف اعتمادية هذه الأنشطة على بعضها البعض **(Identify Activities Dependencies)** ، وتقدير موارد الأنشطة الخاصة بكل نشاط **(Estimate Resources for Activities)** ، وتخصيص الأشخاص المطلوبة لتنفيذ الأنشطة **(Allocate People for Activities)** ، ومن ثم إنشاء مخططات المشروع للحصول على المخططات البيانية للأنشطة **(Activity and Bar Charts)**.

❑ ***ويمكن إجمال أهم مشاكل الجدولة الزمنية للمشاريع كالتالي :***

- تقييم مدى صعوبة المشاكل ، وبالتالي فان تكلفة تطوير الحل تكون صعبة.
- الإنتاجية لا تتناسب مع عدد الأفراد الذين يعملون على تنفيذ مهمة.
- إن تعيين أفراد إضافيين على مشروع متأخر عن موعده ، يزيد في تأخيره نظرا لمشكلة الاتصالات بينهم.
- دائما يوضع في الحسبان حدوث الأسوأ : يجب أن تشتمل خطة المشروع على الأمور الطارئة التي قد تحدث فيه.

2.4 مخططات الجدولة الزمنية للمشروع

Project Scheduling Charts

تشمل عملية الجدولة الزمنية للمشروع مجموعة من المخططات وشبكات الأنشطة لتوضيح العلاقات والفترات الزمنية والمسارات الحرجة **(Critical Paths)** للأنشطة المختلفة ، وكذلك توضيح مدى التقدم في سير العمل وواجبات ومهام فريق العمل. ويمكن الاعتماد على برمجيات الإدارة المختلفة **(Software Management Tools)** مثل **(Microsoft Project)** ، و **(SmartDraw:**) www.smartdraw.com لتوليد هذه المخططات. ومن أهم تلك المخططات : مخطط جانت **"Gantt Chart"** ويسمى أيضاً بـ **"مخطط القضبان أو الأعمدة (Bar Chart)"** ، ومخطط بيرت **"Pert Chart"** ويسمى أيضاً بـ **"شبكة الأنشطة (Activates Network)"**.

وقبل أن نبدأ في شرح هذه المخططات ، عليك عزيزي القارئ أن نلقي نظرة على أهم المصطلحات التي سوف نستخدمها خلال هذا الجزء ، وهي موضحة بالجدول رقم 4-2 التالي.

المصطلح	المعنى
الحدث Event	هو الوصول إلى نقطة معينة من الزمن و لا يحتاج إلى بداية ونهاية زمنية.
النشاط Activity	هو مجهود يحتاج إلى نقطة بداية ونهاية وموارد لتنفيذه.
النشاط الوهمي Activity Dummy	النشاط الذي لا يحتاج إلى زمن أو موارد لإتمامه ويستعمل فقط للدلاله على تتابع الأنشطة منطقيا ويرسم بسهم متقطع.
النشاط الحرج Activity Critical	النشاط الذي إذا تم تأخير انتهائه فإنه يتسبب في تأخير المشروع.
المسار الحرج Critical Path	مجموعة من الأنشطة الحرجة، تبدأ من بداية إلى نهاية المشروع.
المشروع Project	عبارة عن مجموعة من الأنشطة والأحداث مرتبة حسب تسلسل منطقي.
شبكة الأعمال Network	عبارة عن مجموعة من الأنشطة والأحداث مرتبة بطريقة منطقية لتسلسل الأنشطة.
زمن البداية المبكر للنشاط Earliest Start	هو الزمن الذي يبدأ فيه النشاط إذا أنجزت جميع الأنشطة السابقة في أوقاتها.
زمن النهاية المبكر Earliest Finish	هو الزمن الذي يمكن أن ينجز فيه النشاط إذا بدأ في وقته المبكر. نهاية مبكرة = بداية مبكرة + وقت النشاط
زمن نهاية متأخر Latest Finish	هو أخر زمن يمكن إتمام النشاط فيه بدون أن يسبب تأخير لأية أنشطة لاحقة.
زمن بداية متأخر Latest Start	هو أخر وقت يمكن أن يبدأ فيه النشاط بشرط عدم تأخير الأنشطة اللاحقة.

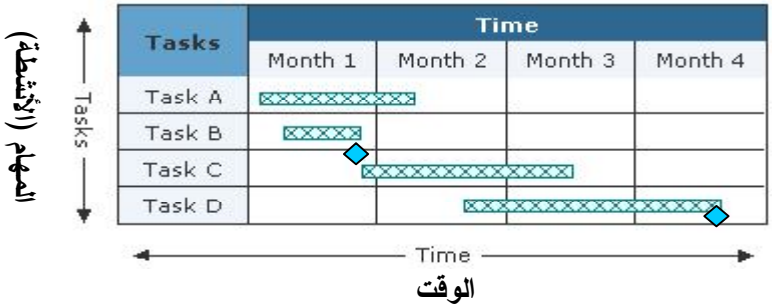
المصطلح	المعنى
	بداية متأخرة = نهاية متأخرة – وقت النشاط
الفائض Slack Time	الفائض في النشاط = زمن بداية متأخر – زمن بداية مبكر

جدول رقم 4-2 : بيان بالمصطلحات الهامة في عملية جدولة المشاريع

1.2.4 مخطط جانت "Gantt Chart"

يعدّ مخطط **"Gantt"** أحد أدوات تخطيط المشاريع ، حيث يقوم بإظهار المهام بشكل بياني ، كالتقويم مثلاً. هذا ، ويُعد أيضاً نمطاً من أنماط خطوط الزمن ، أو الجداول الزمنية التي تأخذ بعين الاعتبار جميع المهام (الأنشطة) المراد إنجازها في المشروع. وقد سميت هذه المخططات كناية للعالم **"Henry Gantt"** الذي طورها في أواخر القرن التاسع عشر ، وتبين هذه المخططات متى تبدأ المهام ، ومتى تنتهي ، كما وتظهر فعاليات المشروع بشكل أشرطة تتناسب أطوالها مع مدة الفعالية ، وترتبط بالإطار الزمني مباشرة. وتحتل كل مهمة في مخطط **"Gantt"** سطراً واحداً ، وتظهر التواريخ في الأعلى كأيام ، وأسابيع ، أو اشهر ، وذلك حسب المدة الكلية للمشروع. ويتم تمثيل الوقت المتوقع لكل مهمة بشريط أفقي تعلم نهايته اليسرى البداية المتوقعة للمهمة ، في حين تعلم نهايته اليمنى تاريخ إتمام المهمة المتوقع ، كما هو موضح بالشكل رقم 4-4.

وتسهل مخططات **"Gantt"** معرفة كيفية تداخل الأنشطة أو حدوثها على التوازي ومعرفة حالة كل نشاط في أية لحظة. وتظهر معالم المشروع **"Project Milestone)** بشكل مثلثات مقلوبة أو معينات ، كما هو موضح بالشكل 4.4 ، ويجب أن يكون لكل مرحلة من المشروع معلم واحد على الأقل. توفر المعالم نقاطاً يمكن مراجعة تقدم



شكل 4-4 : شكل عام توضيحي لمخططات "Gantt"

المشروع عندها. ويمكن لدى الحاجة إجراء تعديلات على جدول المشروع الزمني أو موارده ، وذلك للحفاظ على سير المشروع.

تتمتع مخططات **"Gantt"** بالوضوح وسهولة الفهم. كما أنها أيضا سهلة البناء وتعد أكثر الأدوات شيوعاً بين مدراء المشاريع في كافة المشاريع ، باستثناء تلك المعقدة منها. تولد الحزم البرمجية مثل : **Microsoft Project** مخططات **"Gantt"** معقدة للغاية تُظهر بوضوح العلاقات أو الارتباطات بين الأنشطة المختلفة. ومن أهم تلك العلاقات أو الارتباطات مثلاً : لا يمكن بدأ نشاط قبل انتهاء النشاطات المرتبطة به ، أو لا يمكن بدأ نشاط قبل بدأ النشاط المرتبط به ، ويمكن أيضاً أن تتداخل الأنشطة وتتقدم أو تتأخر عن بعضها البعض بزمن تقدم أو تأخر معين. وفي بعض الأحيان لا يمكن لنشاط أن يبدأ بعد انتهاء الأنشطة المتعلقة به مباشرة ؛ فغالباً ما يحدث هذا بسبب الأنشطة والموارد الخارجية بالنسبة للمشروع ، كالتأخر في تسليم البضائع أو المواد ، ويدعى هذا بزمن التأخير.

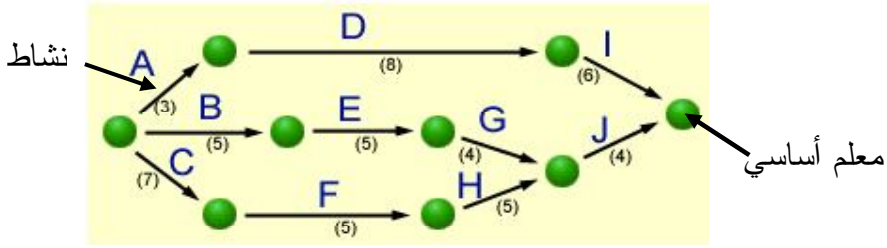
كما يمكن تعقب التقدم بسهولة باستخدام مخططات **"Gantt"** ، وذلك من خلال تحديث مخططات **"Gantt"** مع تقدم المشروع ، حيث يتم ذلك بتظليل الأشرطة لتبلغ الطول

المتناسب مع كمية العمل المنجز ضمن النشاط. ويمكّن هذا من مقارنة التقدم الفعلي مع الجدول الزمني المخطط له.

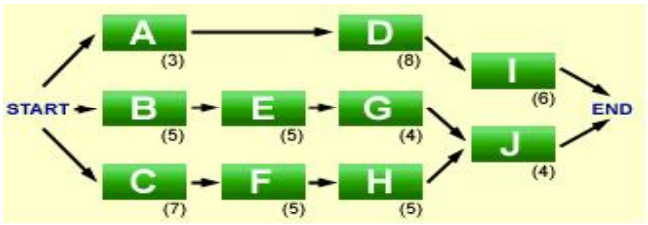
2.2.4 مخطط بيرت "PERT Chart"

تُعد مخططات **"PERT"** البيانية طريقة بديلة لعرض معلومات المشروع. وكلمة **"PERT"** هي اختصار "لتقنية تقييم ومراجعة البرامج" **(Program Evaluation & Review Technique)** ، وتعرض هذه الطريقة المشروع كشبكة من الأنشطة. و تأخذ مخططات **"PERT"** الأنشطة من مخططات **"Gantt"** وتعرضها بشكل مخطط تدفقي ، ويحدد محوره الأفقي الفترات الزمنية. ويتم تمثيل الأنشطة بصناديق تظهر اسم النشاط ، ورقمه ، ومدته الزمنية ، وتاريخي البدء والانتهاء

ويُظهر مخطط **"PERT"** تدفق المشروع والعلاقات والارتباطات بين الأنشطة المختلفة ، كما هو موضح بالشكل رقم 4.5 ، ويسهل تحديد المسارات الحرجة والمهام ذات الزمن البطيء. وقد يكون مخطط **"PERT"** لمشروع كبير ضخماً جداً ، لذا ، فغالباً ما يتم تجزئة المشروع إلى أجزاء أصغر.



شكل (a) 4-5 : شكل توضيحي عام لمخطط "PERT" باستخدام الاصطلاح وضع النشاط فوق السهم (Activity on Arrow Convention).



شكل (b) 4-5 : شكل توضيحي عام لمخطط "PERT" باستخدام الاصطلاح وضـــع النشاط فوق العقدة (Activity on Node Convention).

النشاط Activity	الأنشطة السابقة Predecessors Activities (Dependencies)	الوقت (أسابيع) Time (Weeks)
A	-	3
B	-	5
C	-	7
D	A	8
E	B	5
F	C	5
G	E	4
H	F	5
I	D	6
J	G - H	4

شكل (c) 4-5 : جدول الأنشطة والأزمنة والعلاقات الخاصة بها

❑ ***ويمكن تلخيص عملية تخطيط مخطط "PERT" في الخطوات التالية :***

❶ تحديد الأنشطة **(Activities)** والمعالم الرئيسية **(Milestones)** المطلوبة لإتمام تطوير المشروع ، وكما ذكرنا سابقاً أن المعلم الأساسي هو ناتج نشاط أو مجموعة من الأنشطة ، ويتم وضع علامة مميزة له في نهاية النشاط أو الأنشطة المتعلق بها.

❷ تحديد التتابع المناسب للأنشطة **(Proper Sequences of the Activities)**.

❸رسم مخطط شبكة الأنشطة **(Activities Network)** طبقاً للتتابع المحدد في الخطوة السابقة مستخدماً ــ مثلاً ــ الاصطلاح **(Convention)** المبين بالشكل رقم 4-5(a).

❹ تقدير أزمنة أداء الأنشطة **(Estimation of Activities Times)** : ويحتاج كل نشاط عدد ثلاثة أزمنة لتقدير الزمن المتوقع **(Expected Time)** لإتمام النشاط **وهي :**

① الوقت المتفائل **(Optimistic Time)** : وهو أقل زمن لإتمام النشاط **.**

②الوقت الأكثر احتمالا **(Most Likely Time)**: هو الزمن الأكثر تكرارا لإتمـــام النشاط **.**

③ الوقت المتشائم **(Pessimistic Time)**: هو أطول زمن لإتمام النشاط **.**

بعد تقدير أزمنة كل نشاط ، يُحسب متوسط زمن النشاط بناءا على معادلة أداء النشاط التالية :

$$\text{زمن أداء النشاط} = \frac{[\text{الوقت المتفائل} + 4 \times \text{الزمن الأكثر احتمالا} + \text{الزمن المتشائم}]}{6}$$

❺ تحديد المسار الحرج **(Determine the Critical Path)** : ويتم عن طريـــق تحديـــد زمن إتمام كل تتابع (مسار) من الأنشطة الموجودة بشبكة الأنشطة ، وذلك بجمع أزمنـــة جميع الأنشطة في هذا التتابع (المسار) ، ومن ثم تحديد أطول مسار في شبكة الأنشـــطة (المشروع) ، حيث إن هذا المسار هو المسار الحرج ، بمعنى إنه في حالة تأخير تنفيذ أي نشاط من الأنشطة الموجودة في هذا المسار سوف يؤدي إلى تـــأخير تنفيـــذ المشـــروع ، أمـــا

الأنشطة الموجودة خارج هذا المسار يمكن أن يتم تأخير تنفيذها في حدود معينة بحيث لا يتم أي تأثير على الزمن الكلي لإتمام المشروع.

في حالة عدم وضوح المسار الحرج وخاصة في حالة المشاريع الكبيرة ذات شبكات الأنشطة المعقدة ، يكون من المفيد تحديد الأزمنة التالية لكل نشاط ، **وهي :**

- زمن البداية المبكر **("Earliest Start Time "ES")**.
- زمن النهاية المبكر **("Earliest Finish Time "EF")**.
- زمن البداية المتأخر **("Latest Start Time "LS")**.
- زمن النهاية المتأخر **("Latest Finish Time "LF")**.

ويتم تحديد زمن البداية المبكر وزمن النهاية المبكر من خلال إتباع ما يلي :

(أ) ابدأ من بداية المشروع وتقدم أمام الشبكة.

(ب) حدد أقرب موعد لبدء المشروع بحيث يكون مساوي للصفر.

(ج) احسب زمن النهاية المبكر **(EF)** لكل نشاط من خلال إضافة المدة التي تستغرقها إلى زمن البداية المبكر الخاصة به.

(د) بالنسبة لكل نشاط متسلسل لا يسبقه مباشرة إلا نشاط واحد ، حدد زمن البداية المبكر له **(ES)** بحيث يكون مساوي لزمن النهاية المبكر للنشاط السابق.

(هـ) بالنسبة لكل نشاط متسلسل يسبقه أكثر من نشاط واحد ، حدد زمن البداية المبكر له بحيث يكون مساوياً لأكبر زمن النهاية المبكر للأنشطة السابقة.

(و) دوّن زمن البداية المبكر ، وزمن النهاية المبكر

(ز) كرر الخطوات من (ج) إلى (و) حتى تصل إلى نهاية المشروع ، **مع ملاحظة أنه لا يمكن** تحديد زمن البداية المبكر لنشاط معين إلا بعد تحديد زمن النهاية المبكر لجميع الأنشطة السابقة له.

وبنفس الخطوات السابقة ، ولكن تكون البداية من نهاية شبكة الأنشطة والتقدم يكون للخلف ، يمكن حساب زمن البداية المتأخر **(LS)** ، وزمن النهاية المتأخر **(LF)** ،

كما سنوضحه في المثال التالي.

ولكن كيف يتم حساب الزمن الفـــائض **(Slack Time)** ، أي فتـــرات التـــأخير المسموح بها ، للأنشطة ، و تحديد الأنشطة الحرجة ، وبالتالي المسار الحرج؟.

بالنسبة لكل نشاط يتطابق زمن البداية المبكر **(ES)** مع زمن البداية المتأخر **(LS)** ، أو زمن النهاية المبكر **(EF)** مع زمن النهاية المبكر **(LF)** ، فإن الزمن الفائض للنشاط يساوي صفر ، ويُعد نشاطاً حرجاً. وفيما عدا ذلك ، فإن الزمن الفـــائض للنشـــاط **(ST)** يمكن حسابه من خلال المعادلة التالية :

$$ST = LS - ES \quad or \quad ST = LF - EF \ldots\ldots\ldots\ldots\ldots\ldots(1)$$

وتسلسل الأنشطة الحرجة من بداية إلى نهاية المشروع هو المسار الحرج للمشروع.

❐ ***ويمكن الاستفادة من مخطط "PERT" في متابعة المشروع ، وتحديد فترة تنفيذه من خلال التالي :***

أ- دراسة تتابع الأنشطة ، ومدى أهميته وإمكانية تغيير أسلوب إدارة المشروع.

ب- دراسة أنشطة المسار الحرج ومدى إمكانية تقصير فتراتها.

ت- بحث إمكانية استخدام موارد إضافية لتقصير فترة الأنشطة الحرجة.

ث- بحث إمكانية تغيير الأداء المحدد لنشاط معين لتقصير المدة المقررة له.

ج- أخيراً ، اعتبر مدة المشروع هي زمن النهاية المبكر للنشاط الأخير ، أو نهايـــة المشروع.

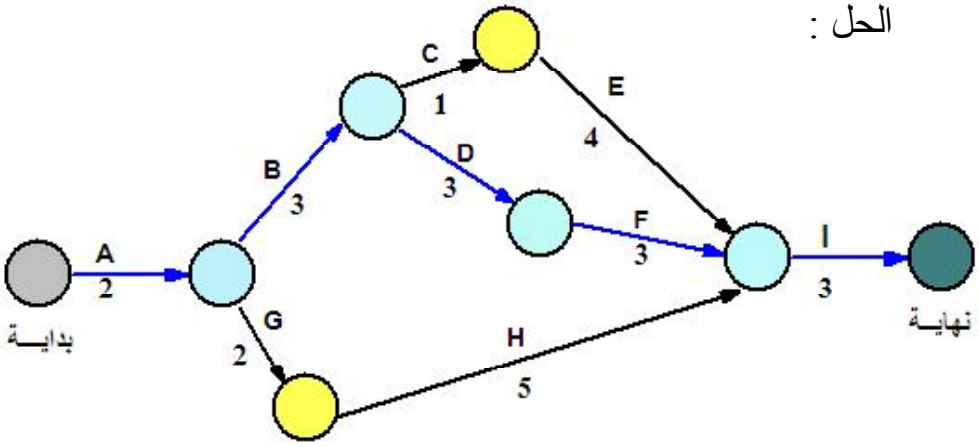
مثال 1: **الجدول التالي يوضح الزمن اللازم لإنجاز عدد من الأنشطة اللازمة لتطوير مشروع ما والعلاقات بينها :**

النشاط	النشاط السابق	الزمن يوم	النشاط	النشاط السابق	الزمن يوم
A	_	2	E	C	4
B	A	3	F	D	3
C	B	1	G	A	2
D	B	3	H	G	5
			I	E F H	3

1. **ارسم مخطط "PERT" مبيناً تتابع الأنشطة.**
2. **حدد المسار الحرج والأنشطة التابعة له.**
3. **ما هو الزمن الفائض للأنشطة "C" ، "D" ، "G" ؟.**
4. **إذا تأخر النشاط "C" بمقدار يوم واحد ، ما تأثير ذلك على الزمن الكلي لانتهاء المشروع ، ولماذا؟.**
5. **قام مدير المشروع بتقليل الزمن الخاص بالنشاط "D" ، والنشاط "F" يوماً واحداً ، فكيف يؤثر ذلك على انتهاء المشروع؟ ولماذا؟.**

عزيزي القارئ فكر قليلاً في حل هذا المثال قبل أن ترى الحل في الصفحة التالية.

الحل :



1- مخطط "Gantt"

2- تحديد المسار الحرج :

- **زمن المسار " بداية ، "A"، "B" ، "D" ، "F" ، "I" ، نهاية " = 14 يوم**
- **زمن المسار " بداية ، "A"، "B" ، "C" ، "E" ، "I" ، نهاية " = 13 يوم**
- **زمن المسار " بداية ، "A"، "G" ، "H" ، "I" ، نهاية " = 12 يوم**

إذن المسار الحرج هو المسار : " بداية ، "A"، "B" ، "D" ، "F" ، "I" ، نهاية "

الأنشطة الحرجة : "A"، "B" ، "D" ، "F" ، "I" .

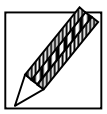
3- الزمن الفائض للأنشطة : "C" ، "D" ، "G" :

- **زمن المسار "C" ، "E" = 5 يوم ، زمن المسار "D" ، "F" = 6 يوم ، وحيث إن بداية ونهاية المسارين واحدة ، إذن يوجد زمن فائض يوم واحد (الفرق بينهما) لصالح النشاط "C" ، أو النشاط "E" مقارنة بالنشاطين "D" ، "F". والنشاط "D" ، والنشاط "F" ليس لهم زمن فائض حيث إنهم من ضمن المسار الحرج.**
- **زمن المسار "B" ، "C" ، "E" = 8 يوم ، زمن المسار "B" ، "D" ، "F" = 9 يوم ، وزمن المسار "G" ، "H" = 7 يوم ، وحيث إن بداية ونهاية هذه المسارات الثلاثة واحدة ، إذن يوجد فائض زمني للأنشطة "G" ، "H" تقدر بيومين.**

4- تأخير النشاط "C" بيوم واحد لن يؤثر على زمن انتهاء المشروع حيث إنه يمتلك يوم فائض زمني كما تم توضيحه سابقاً.

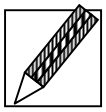
5- تقليل الزمن الخاص بالنشاط "D" ، والنشاط "F" بمقدار يوم لكل منهما سوف يقلل الزمني الكلي للمشروع يوماً واحداً فقط ليصبح (13) يوماً بدلاً من (14) يوماً ، حيث إن هذا الخفض سوف يسبب أن يكون المسار " بداية ، "A"، "B" ، "C" ، "E" ، "I" ، نهاية " = 13 يوم هو المسار الحرج بدلاً من المسار " بداية ، "A"، "B" ، "D" ، "F" ، "I" ، نهاية " الذي سيصبح (12) يوماً فقط.

تدريب (2)



عزيزي الدارس حاول الآن الإجابة عن السؤال الأول ، والسؤال الثالث من المثال السابق عن طريق حساب الزمن الفائض باستخدام المعادلة رقم (1) السابقة.

تدريب (3)



النشاط	النشاط السابق	الزمن يوم	النشاط	النشاط السابق	الزمن يوم
T1	–	8	T7	T1 (M1)	20
T2	–	15	T8	T4 (M5)	25
T3	T1 (M1)	15	T9	T3, T6 (M4)	15
T4	–	10	T10	T5, T7 (M7)	15
T5	T2, T4 (M2)	10	T11	T9 (M6)	7
T6	T1, T2 (M3)	5	T12	T11 (M8)	10

1- طبقاً للجدول السابق ارسم مخطط "PERT" مبيناً تتابع الأنشطة والمعالم الأساسية وتاريخ بدء ونهاية كل نشاط ومعلم أساسي على فرض أن تاريخ بدء المشروع هو 4/7/1999 ، وأوقات العمل (5) أيام في الأسبوع.

2- حدد الزمن الفائض لكل نشاط ومعلم أساسي ، والأنشطة والمعالم الأساسية الحرجة ، ومن ثم المسار الحرج ، وزمن انتهاء المشروع.

3- ارسم مخطط "Gantt" بحيث يحتوي على موعد ونهاية كل نشاط ومعلم أساسي ، وكذلك رسم الزمن الفائض لكل نشاط (في حالة وجوده) بشريط مظلل بعد تاريخ نهاية النشاط أو المعلم الأساسي ، حيث يمثل هذا الفائض مرونة في موعد الانتهاء لهذه الأنشطة أو المعالم الأساسية بحيث لا يتأثر بذلك موعد نهاية المشروع.

4- ارسم مخطط "Gantt" لتخصيص فريق العمل الذي سوف يقوم بإنجاز مهام المشروع.

أسئلة تقويم ذاتي

1/ أشرح كيف تتم عملية الجدولة الزمنية للمشاريع ، وما هي أهم مشاكلها ؟.

2/ بين معاني المصطلحات التالية :

- الحدث Event .
- النشاط Activity .
- النشاط الوهمي Dummy Activity .
- النشاط الحرج Critical Activity .
- المسار الحرج Critical Path .
- المشروع Project .
- شبكة الأعمال Network .
- زمن البداية المبكر للنشاط Earliest Start .
- زمن النهاية المبكر Earliest Finish .
- زمن نهاية متأخر Latest Finish .
- زمن بداية متأخر Latest Start .
- الفائض Slack Time .

3/ ما هي أهم سمات مخططات القضبان وشبكات الأنشطة ؟

4/ لخص خطوات عملية تخطيط مخطط PERT .

5/ ما هو المسار الحرج وكيف يتم استنتاجه ؟

5. إدارة الكوارث Risk Management

من الأهداف الرئيسة لتعيين مدير المشروع هو توقع الكوارث التي ربما تؤثر على جدول المشروع أو جودة البرامج التي يتم تطويرها واتخاذ الإحتياطات اللازمة لتجنب هذه الكوارث ، ويجب أن يتم تضمين نتائج تحليل الكوارث وما يترتب عليه من نتائج بالتوالي في خطة المشروع. وتسمى عملية تحديد الكوارث وعمل الخطط اللازمة لتقليل آثارها على المشروع بـ "إدارة الكوارث". ويمكنك أن تفكر في الكارثة كمجرد احتمال لوقوع بعض الظروف المناوئة لسير المشروع. ويمكن أن تؤدي هذه الكوارث إلى تهديد المشروع أي البرامج التي يتم تطويرها أو المؤسسة التي تقوم بتلك العملية ذاتها.

1.5 أنواع الكوارث Types of Risks

عزيزي **الدارس** ، يمكن تمييز عدة أنواع من الكوارث من **أهمها :**

❶ **كوارث المشروع (Project Risks)** : وهي الكوارث التي تؤثر على الجدول الزمني للمشروع أو على مصادر تمويل المشروع.

❷ **كوارث المنتج (Product Risks)** : وهي الكوارث التي تؤثر على جودة أو أداء المنتج أو البرامج التي يتم تطويرها.

❸ **كوارث الأنشطة التجارية (Business Risks)** : وهي الكوارث التي تؤثر في المؤسسة التي تتولى تطوير أو شراء برامج الحاسب الآلي.

ولا يعد هذا التصنيف تصنيف نهائي فعلى سبيل المثال : عندما يترك مبرمج ما المشروع فإن ذلك يعد :

- كارثة مشروع لأن توريد النظام قد يتأخر.
- كارثة منتج لأن الشخص الذي يتم توظيفه بدلاً من المبرمج الذي ترك العمل ربما لا يتمتع بخبرته لذا فربما يقع عدد كبير من الكوارث.
- كارثة عمل لأن هذه الخبرة غير متاحة في العمل المستقبلي.

وإدارة الكوارث هامة جداً لمشاريع البرامج ، وذلك بسبب بعض العقبات غير المتوقعة التي قد يواجهها المشروع مثل الصعوبات في تقدير الوقت والتمويل اللازمين لتنفيذ مشروع تطوير البرامج ، والاعتماد على المهارات الشخصية ، وتغيير المتطلبات كنتيجة لتغير احتياجات العميل. ويجب على مدير المشروع توقع هذه الكوارث وإدراك تأثيرها على المشروع والمنتج والعمل واتخاذ الإجراءات اللازمة لتجنب هذه الكوارث. ويعتمد نوع الكارثة التي تؤثر على المشروع على المشروع نفسه ، وكذلك على البيئة التنظيمية حيث يتم تطوير البرامج. ويوجد تصنيف للكوارث والذي يُعد تصنيفاً عالمياً كما هو موضح في الجدول رقم 4–3.

الكارثة	نوع الكارثة	الوصف
تحول فريق العمل (Staff Turnover)	المشروع (Project)	أحد أو عدد من أعضاء فريق العمل ذو الخبرة سوف يترك المشروع قبل انتهائه.
تغيير الإدارة (Management Change)	المشروع	حدوث تغييرات في الإدارة التنظيمية.
عدم توفر العتاد (Hardware Unavailability)	المشروع	قد لا يتم تسليم العتاد الضروري للمشروع في موعده المحدد.
تغيير المتطلبات (Requirements Change)	المشروع والمنتج	قد يوجد عدد كبير من التغييرات في المتطلبات أكثر من المتوقع.
تأخير المواصفات (Specification Delay)	المنتج والمشروع	قد لا تتاح مواصفات الواجهات الرئيسية كما هو مخطط في جدول المهام.
الاستخفاف بحجم المشروع (Size Underestimate)	المنتج والمشروع	قد يتجاوز حجم المشروع التقدير الخاص به.
قصور في أداء أدوات هندسة البرمجيات بمساعدة الحاسب CASE Tools Underperformance	المنتج (Product)	عدم قيام أدوات هندسة البرمجيات بمساعدة الحاسب بدعم المشروع كما هو مخطط له.
تغير التقنية (Technology Change)	أعمال (Business)	توفر تقنيات جديدة بديلة لتلك المستخدمة في تطوير المشروع
منافسة المنتج (Product Competition)	أعمال	وجود منتج منافس يتم تسويقه قبل الانتهاء من النظام.

جدول 4-3 : تصنيف كوارث المشاريع البرمجية

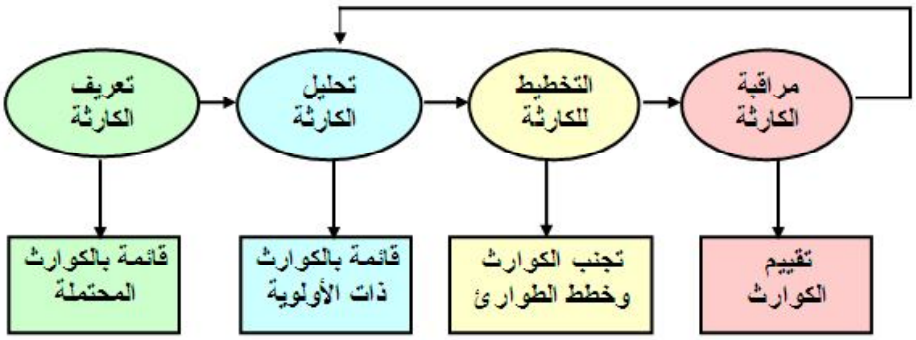
2.5 عملية إدارة الكوارث

(The Risks Management Process)

تتضمن عملية إدارة الكوارث ، كما هو موضح بالشكل رقم 6-4 ، **المراحل التالية :**

- **تعريف الكارثة (Risk Identification)** : تتم بتعريف كوارث المشروع ، والمنتج ، والأعمال الخاصة بالأنشطة التجارية.
- **تحليل الكارثة (Risk Analysis)** : بتقدير وتقييم وترجيح الاحتمالات والنتائج المنطقية لهذه الكارثة.
- **التخطيط للكارثة (Risk Planning)** : يتم بوضع الخطط اللازمة للتغلب على أو التقليل من تأثيرات الكارثة.
- **مراقبة الكارثة (Risk Monitoring)** : وتشمل مراقبة الكارثة عبر مراحل المشروع.

والتخطيط لإدارة الكوارث مثل التخطيط لجميع الفرعيات الأخرى عملية دائمة تستمر خلال المشروع بكامله ، وعندما يتم عمل مجموعة من الخطط المبدئية يتم مراقبة الوضع بدقة شديدة ، وعندما يتوفر مزيد من المعلومات عن الكوارث المحتملة يتم تحليل هذه الكوارث وتحديد أولويات جديدة وربما يتم تعديل أساليب تجنب الكارثة وخطط الطوارئ عندما تتوفر معلومات جديدة عن وجود كوارث. ويجب أن يتم تضمين نتائج عملية إدارة الكارثة في خطة إدارة الكوارث ويجب أن يشمل ذلك مناقشة الكارثة التي يواجهها المشروع وتحليل هذه الكوارث والخطط اللازمة لمواجهة هذه الكوارث وربما تشمل أيضاً بعض نتائج إدارة الكوارث مثل (خطط الطوارئ الخاصة التي يتم تنشيطها عند وقوع الكوارث).



شكل 6-4 : شكل توضيحي لعملية إدارة الكوارث

3.5 التعرف على الكوارث Risk Identification

يُعد التعرف على الكوارث الخطوة الأولى لإدارة الكوارث ، ويتعلق ذلك باكتشاف الكوارث المحتملة في المشروع ، ولا يجب تقدير هذه الكوارث أو تحديد أيها تحتل الأولوية في هذه المرحلة ، وذلك لأن الكوارث ذات العواقب التافهة أو وجود احتمالية ضئيلة لحدوثها لا يتم أخذها في الاعتبار ويتم التعرف على الكوارث كعملية جماعية باستخدام أسلوب الحوار والنقاش أو من خلال خبرة مدير المشروع ، وللمساعدة في ذلك يمكن عمل قائمة تشمل تصنيفات الكوارث وتشمل :

أ) كوارث التقنية (Technology Risks) :

وهي كوارث تنتج عن تقنيات البرامج أو الأجهزة التي تستخدم كجزء من النظام الذي يتم تطويره.

ب) كوارث ناتجة عن البشر (People Risks) :

وهي كوارث تنتج عن البشر الذين يشتركون في فريق عملية التطوير.

ت) كوارث تنظيمية (Organizational Risks) :

وهي كوارث تنتج عن البيئة التنظيمية حيث يتم تطوير البرامج.

ث) كوارث ناتجة عن الأدوات المستخدمة (Tools Risks) :

وهي كوارث تنتج عن أدوات **CASE** وبقية البرامج المساندة التي تستخدم في عملية تطوير النظام.

ج) كوارث المتطلبات (Requirements Risks) :

وهي كوارث تنتج عن تغير متطلبات العميل وعملية إدارة تغير المتطلبات.

ح) كوارث التقدير (Estimation Risks) :

وهي كوارث تنتج عن التقديرات الإدارية لخصائص النظام والموارد المطلوبة لبنائه.

والجدول 4.4 يوضح تصنيف لأنواع الكوارث وفرعياتها المحتملة لكل نوع.

نوع الكارثة	الكارثة المحتملة
التقنية	• **لا يمكن لقاعدة البيانات المستخدمة في النظام من معالجة قدر من العمليات في الثانية حسب المتوقع.** • **مكونات برامج الحاسب والتي يتعين إعادة استخدامها تشتمل على عيوب تعمل على تحديد وظيفتها.**
الأفراد	• **يتعذر تعيين موظفين يتمتعون بالمهارات المطلوبة.** • **يعاني الموظفون الرئيسيون من المرض ، وهم غير متوفرين في الأوقات الحرجة.** • **يلزم تزويد الموظفين ببرامج تدريبية ولكنها غير متاحة.**
الناحية التنظيمية	• **يتم إعادة هيكلة المؤسسة بحيث تكّون إدارة مختلفة مسئولة عن المشروع.** • **تؤدي المشاكل المالية التنظيمية إلى خفض ميزانية المشروع.**
الأدوات	• **إن الشيفرة المتولدة عن أدوات CASE غير كافية.** • **يتعذر دمج أدوات CASE.**
المتطلبات	• **اقتراح إجراء تغييرات على المتطلبات، والتي تستدعي عمل رئيسي وإعادة لتصميم.**

نوع الكارثة	الكارثة المحتملة
	• فشل العملاء في إدراك التأثير الناتج عن إجراء تغييرات على المتطلبات.
التقدير	• عدم تقدير الوقت اللازم لتطوير برنامج الحاسب الآلي بالشكل الصحيح. • عدم تقدير معدل إصلاح العيوب بالشكل الصحيح. • عدم تقدير حجم برنامج الحاسب بالشكل الصحيح.

جدول رقم 4.4 : تصنيف أنواع الكوارث وفرعياتها

4.5 تحليل الكوارث Risk Analysis

يشمل تحليل الكوارث الفرعيات التالية :

- تحليل احتمالية ومدى جدية كل كارثة من الكوارث.
- الاحتمالية قد تكون متدنية جداً ، أو متدنية ، أو متوسطة أو عالية أو عالية جداً.
- تأثيرات الكوارث قد تبدو فاجعة، أو جدية أو يمكن تحملها أو ليست ذات أهمية.

والجدول رقم 4.5 **يبين أنواع الكوارث ودرجة احتمال حدوثها وتأثيراتها المتوقعة.**

الكارثة	الاحتمالية	التأثيرات
• مشاكل تنظيمية مالية تؤدي إلى تقليل ميزانية المشروع.	منخفض	كارثي
• عدم إمكانية استئجار فريق يتمتع بالمهارة اللازمة للمشروع.	مرتفع	كارثي
• مرض فريق العمل الرئيسي في المشروع خلال أوقات حرجة.	متوسط	خطير
• احتواء مكونات البرامج التي سيعاد استخدامها على بعض العيوب.	متوسط	خطير
• وجود تغييرات في المتطلبات تتطلب إعادة التصميم.	متوسط	خطير
• إعادة هيكلة المؤسسة مع وجود إدارات مختلفة مسئولة عن المشروع.	مرتفع	خطير
• قاعدة البيانات المستخدمة لا يمكنها أن تقوم بمهامها المحددة بشكل جيد.	متوسط	خطير
• الوقت المطلوب لتطوير البرامج تجاوز الوقت المحدد لذلك.	مرتفع	خطير
• لا يمكن تكامل أدوات CASE.	مرتفع	يمكن تحملها

الكارثة	الاحتمالية	التأثيرات
• عدم فهم العميل لتأثير تغير المتطلبات.	متوسط	يمكن تحملها
• التدريب المطلوب لفريق العمل غير متوفر.	متوسط	يمكن تحملها
• الخطأ في تقدير الوقت اللازم لتصحيح الخطأ.	متوسط	يمكن تحملها
• سوء تقدير (تخمين) حجم البرمجية.	مرتفع	يمكن تحملها
• الجزئية المنتجة عن طريق أدوات CASE. لا تعمل بكفاءة.	متوسط	غير مؤثرة

جدول رقم 4.5 : أنواع الكوارث ودرجة احتمال حدوثها وتأثيراتها المتوقعة

5.5 التخطيط للكوارث Risk Planning

يشمل التخطيط للكوارث دراسة كل كارثة وتطوير إستراتيجية لإدارتها ويمكن تقسيم هذه الإستراتجيات إلى **ثلاثة فرعيات هي :**

- **إستراتيجية تجنب الكارثة (Avoidance Strategy) :** وتشمل تجنب احتمال إمكانية ارتفاع مستوى الكارثة.
- **إستراتيجية الحد من الكارثة (Minimization Strategy) :** وتشمل التقليل من تأثير الكارثة على المشروع أو المنتج.
- **إستراتيجية خطط الطوارئ (Contingency Plans Strategy) :** وتشمل وضع خطط للتعامل مع أي كارثة محتمل حدوثها ، بحيث يعتد بهذه الخطط للتعامل مع أي كارثة في حالة حدوثها.

والجدول رقم 4.6 يبين إستراتيجيات إدارة الكوارث **(Risk Management Strategies).**

الكارثة	الإستراتيجية

الكارثة	الإستراتيجية
المشاكل المالية التنظيمية	إعداد وثيقة مختصرة للإدارة العليا تبين كيف يساهم المشـــروع بشكل فاعل في تحقيق أهداف النشاط التجاري.
مشاكل التوظيف	تنبيه العملاء لاحتمالية وجود صعوبات واحتمالات التأخير والبحث في مكونات الشراء.
مرض الموظفين	إعادة تنظيم فريق العمل بحيث يكون هنالك تداخل بـــين العمـــل والأفراد ، وبالتالي إدراك الأفراد لمهام بعضهم البعض.
الأجزاء التالفة	استبدال المكونات التالفة بمكونات أخرى مماثلة ذات إعتماديـــة عالية.
التغيرات على المتطلبات	استنتاج المعلومات السابقة ذات العلاقة لتقييم تأثير التغير الـــذي يطرأ على المتطلبات وتفعيل المعلومات المشمولة في التصـــميم وغير الظاهرة.
إعادة الهيكلة التنظيمية	إعداد وثيقة مختصرة للإدارة العليا تبين كيف يساهم المشـــروع بشكل فاعل في تحقيق أهداف النشاط التجاري.
أداء قاعدة البيانات	بحث إمكانية شراء قاعدة بيانات ذات أداء عالي.
وقت التطوير الذي لم يتم تقييمه بالشكل الصحيح	بحث إمكانية شراء مكونات، وبحث إمكانية استخدام مولد لشفرات البرنامج.

جدول رقم 4.6 : إستراتيجيات إدارة الكوارث

6.5 مراقبة الكوارث Risk Monitoring

وتشمل مراقبة الكوارث الفرعيات التالية :

- تقييم كافة الكوارث المعروفة كلاً على حدة بشكل منتظم لتحديد احتمال حدوثها من عدمه.
- كذلك تقييم فيما لو أن تأثيرات الخطر قد طرأ عليها أي تغيرات.
- يجب بحث كافة جوانب الكوارث الهامة في اجتماعات تطور المشروع الإدارية.

والجدول رقم 4.7 يوضح أمثلة للعوامل التي يمكن أن تساعد في مراقبة أنواع الكوارث.

نوع (تصنيف) الكارثة	المؤشرات
التقنية	**تأخير تسليم البرمجيات أو الأجهزة يسبب العديد من مشـــاكل التقنية.**
العاملين	**السلوك الأخلاقي السيئ للعاملين وعدم وجود اتصالات جيـــده بينهم.**
النواحي التنظيمية	**تأخير اتخاذ القرار من الإدارة العليا.**
الأدوات	**عدم اهتمام العاملين لاستخدام الأدوات المتاحة والشكوى مـــن هذه الأدوات وطلب معدات ذات كفاءة أعلى.**
المتطلبات	**التغيرات العديدة للمتطلبات وشكاوى العملاء.**
التقدير	**مثل الفشل في الالتزام بالمواعيد المحددة ، وعدم القدرة على تقديم تقارير شاملة للسلبيات.**

جدول رقم 4.7 : أمثلة للعوامل المساعدة في مراقبة أنواع الكوارث

أسئلة تقويم ذاتي

1/ ما هي أنواع الكوارث التي يمكن أن تحدث أثناء تنفيذ المشروع ؟.
2/ هناك تصنيف عالمي للكوارث ، ارسم جدول بين فيه اسم الكارثة ، نوع الكارثة ، ووصف لها.
3/ أشرح مراحل عملية إدارة الكوارث .
4/ ما هي أهم الفرعيات التي يشملها تحليل الكوارث ؟.
5/ ما هي أهم الفرعيات التي يشملها التخطيط للكوارث ؟.
6/ أرسم جدول وضح فيه استراتيجيات إدارة الكوارث .
7/ ما هي أهم الفرعيات التي تشملها مراقبة الكوارث ؟.
8/ ارسم جدول وضح فيه أمثلة للعوامل التي يمكن أن تساعد في مراقبة أنواع الكوارث .

الخلاصة

*بينت الوحدة أن هندسة البرامج تختلف عن الأنماط الهندسية الأخرى في عدة نـــواحي وهي : المنتج غير ملموس – لا توجد عمليات قياســية للبــرامج – مشـــاريع البـــرامجِ الكبيرةِ مشاريعَ "وحيدةَ" في أغلب الأحيان – تتطلب صناعة البرمجيات جهـــوداً ذهنيـــة وفكرية – بناء المنتج – السلع المنتجة – التأثير على البيئة.

* لخصت الوحدة أهم الأنشطة الإدارية التي يقوم بها المدير فيما يلي: كتابـــة وإعـــداد عرض للمشروع – تخطيط المشروع – مراقبة ومتابعة المشـــاريع – اختيـــار وتقيـــيم الموظفين – كتابة التقارير وإعداد العروض .

*قدمت الوحدة التقسيم المتعارف عليه لفرق العمل اللازمة لتطوير البرمجيات والأعمال المنوطة بها : فريق تحليل النظم – فريق التخطيط – فريق المتطلبات – فريق تصميم النظام – فريق التنفيذ – فريق الاختبار والدمج (التكامل) – فريق التدريب – فريق التسليم والتركيب – فريق الصيانة– فريق ضمان الجودة النوعية – فريق التقييس – فريق التوثيق – فريق إدارة النظام – فريق عمل المشروع – فريق إعادة الممارسة والاستخدام . **هذا بالإضافة إلى فرق عمل مكونة من شخص واحد ، وهي :** مُقيِّم العلاقة بين المستخدم للنظام والحاسب الآلي – مسئول دعم الأداء – اقتصادي النظام – أمين مكتبة المشروع .

* تناولت الوحدة أدوات إدارة المشروع وهي : أدوات CASE – أدوات الاتصال .

* بينت الوحدة أن الإدارة الفعالة لمشروع برامج الحاسب الآلي تعتمد بشكل كبير على التخطيط الجيد ويجب على مدير المشروع أن يتوقع حدوث عدد من المشاكل وإعداد الحلول المناسبة لهذه المشاكل ، كما بينت أن تخطيط المشروع عملية معلقة يتم الانتهاء منها عند إكمال المشروع نفسه فقط. وتُعد أهداف المشروع أهم عامل يجب أن يؤخذ في الاعتبار عند وضع خطة تنفيذ المشروع ، وعندما تتغير هذه الأهداف يجب أن تتغير بالتالي خطة المشروع.كما بينت الوحدة أن معظم الخطط للمشاريع تشمل الأقسام التالية

: مقدمة – تنظيم المشروع تحليل – الكارثة – متطلبات موارد الأجهزة والبرامج – تقسيم العمل – الجدول الزمني المشروع – آليات المراقبة والتقارير .

* ذكرت الوحدة أن جدولة البرامج لا تختلف عن جدولة أي نوع آخر من المشـــروعات حيث يجب تحديث الجداول بشكل مستمر حتى تصبح كافة المعلومات متاحة. وتتضـــمن عملية جدولة المشروع جميع الأعمال التي تؤدى خلال المشروع وتحديد الوقت الـــلازم لإكمال المشروع وبعض النشاطات التي يتم تنفيذها في المشروع بالتوازي ، ويجـــب أن تتضمن جداول المشروع هذه النشاطات المتوازنة وتقوم بتنظيم العمل .

* لخصت الوحدة أهم مشاكل الجدولة الزمنية للمشاريع كالتالي : تقييم مـــدى صـــعوبة المشاكل الإنتاجية لا تتناسب مع عدد الأفراد الذين يعملون على تنفيذ مهمة – إن تعيـــين أفراد إضافيين على مشروع متأخر عن موعده ، يزيـــد فـــي تـــأخيره نظـــرا لمشـــكلة الاتصالات بينهم – دائما يوضع في الحسبان حدوث الأسوأ : يجـــب أن تشـــتمل خطـــة المشروع على الأمور الطارئة التي قد تحدث فيه.

* شرحت الوحدة أهم مخططات الجدولة الزمنية للمشروع وهي: مخطط جانت ويســـمى أيضاً بـــ "مخطط القضبان أو الأعمدة ، ومخطط بيرت ويسمى أيضاً بـــ "شبكة الأنشطة

*ذكرت الوحدة أن عملية تحديد الكوارث وعمل الخطط اللازمة لتقليـــل آثارهـــا علـــى المشروع تسمى بـــ "إدارة الكوارث". كما بينت الوحدة أن عملية إدارة الكوارث تتضمن المراحل التالية: تعريف الكارثة – تحليل الكارثة – التخطيط للكارثة – مراقبة الكارثة .

* بينت الوحدة أن تحليل الكوارث يشمل الفرعيات التالية : تحليل احتمالية ومدى جديـــة كل كارثة من الكوارث – الاحتمالية قد تكون متدنية جداً ، أو متدنية ، أو متوســـطة أو عالية أو عالية جداً – تأثيرات الكوارث قد تبدو فاجعة، أو جدية أو يمكـــن تحملهـــا أو ليست ذات أهمية.

* بينت الوحدة أن التخطيط للكوارث يشمل دراسة كل كارثة وتطوير إستراتيجية لإدارتها ويمكن تقسيم هذه الإستراتجيات إلى ثلاثة فرعيات هي:إستراتيجية تجنب الكارثة –إستراتيجية الحد من الكارثة – إستراتيجية خطط الطوارئ .

* بينت الوحدة أن مراقبة الكوارث تشمل الفرعيات التالية :تقييم كافة الكوارث المعروفة كلاً على حدة بشكل منتظم لتحديد احتمال حدوثها من عدمه – كذلك تقييم فيما لو أن تأثيرات الخطر قد طرأ عليها أي تغيرات – يجب بحث كافة جوانب الكوارث الهامة في اجتماعات تطور المشروع الإدارية.

لمحة مسبقة عن الوحدة التالية

الوحدة التالية تبحث في الهندسة العكسية أي محاولة استدعاء مواصفات المستوى الأعلى عن طريق تحليل المنتج. والوحدة تتناول التعريفات والمفاهيم الهامة المرتبطة بالهندسة العكسية ، كما توضح الفرق بين الهندسة العكسية وبباقي فروع الهندسة ، وتبين كذلك مميزات الهندسة العكسية ، كما توضح أسباب القيام بالهندسة العكسية للبرمجيات .

تستعرض الوحدة كذلك الهندسة العكسية للبرمجيات والمعدات ، والمواضيع القانونية المتعلقة بالهندسة العكسية ، وتستعرض كذلك أنظمة الحماية المختلفة ضد الهندسة العكسية .

مسرد المصطلحات

مكونات المجموعات (Groupware)

وسيلة هامة تستخدم في بعض النظم الإدارية للمساعدة في تنسيق الاجتماعات واللقاءات وتنقيح ومراجعة المشاريع المختلفة . وهو يعني تمكين مجموعة من الأشخاص من دراسة ومعاينة وثيقة واحدة في آن واحد عبر الشبكة.

المعالم الأساسية للمشروع (Milestones)

يُعرف المَعلَّم الأساسي (Milestone) بأنه عبارة عن نقطة النهاية (End Point) وغاية لنشاط معين من أنشطة عمليات البرمجيات (Software Process Activity)

نتائج المشروع القابلة للتسليم (Deliverables)

هي مخرجات المشروع التي تُسلم للعميل في نهاية كل مرحلة أساسية من مراحل المشروع مثل مرحلة تحديد مواصفات المتطلبات (Requirement Specification) ، مرحلة التصميم (Design Phase) ، .... وخلافه.

مخطط جانت "Gantt Chart"

أحد أدوات تخطيط المشاريع ، حيث يقوم بإظهار المهام بشكل بياني ، كالتقويم مثلاً. هذا ، ويُعد أيضاً نمطاً من أنماط خطوط الزمن ، أو الجداول الزمنية التي تأخذ بعين الاعتبار جميع المهام (الأنشطة) المراد إنجازها في المشروع. وقد سميت هذه المخططات كناية للعالم "Henry Gantt" الذي طورها في أواخر القرن التاسع عشر ، وتبين هذه المخططات متى تبدأ المهام ، ومتى تنتهي ، كما وتظهر فعاليات المشروع بشكل أشرطة تتناسب أطوالها مع مدة الفعالية ، وترتبط بالإطار الزمني مباشرة.

مخطط بيرت "PERT Chart"

تُعد مخططات "PERT" البيانية طريقة بديلة لعرض معلومات المشروع. وكلمة "PERT" هي اختصار "لتقنية تقييم ومراجعة البرامج" (Program Evaluation & Review Technique) ، وتعرض هذه الطريقة المشروع كشبكة من الأنشطة.

و تأخذ مخططات "PERT" الأنشطة من مخططات "Gantt" وتعرضها بشكل مخطط تدفقي ، ويحدد محوره الأفقي الفترات الزمنية. ويتم تمثيل الأنشطة بصناديق تظهر اسم النشاط ، ورقمه ، ومدته الزمنية ، وتاريخي البدء والانتهاء .

المسار الحرج (Determine the Critical Path)

يتم تحديده طريق تحديد زمن إتمام كل تتابع (مسار) من الأنشطة الموجودة بشبكة الأنشطة ، وذلك بجمع أزمنة جميع الأنشطة في هذا التتابع (المسار) ، ومن ثم تحديد أطول مسار في شبكة الأنشطة (المشروع) ، حيث إن هذا المسار هو المسار الحرج ، بمعنى إنه في حالة تأخير تنفيذ أي نشاط من الأنشطة الموجودة في هذا المسار سوف يؤدي إلى تأخير تنفيذ المشروع ، أما الأنشطة الموجودة خارج هذا المسار يمكن أن يتم تأخير تنفيذها في حدود معينة بحيث لا يتم أي تأثير على الزمن الكلي لإتمام المشروع.

الزمن الفائض (Slack Time)

هو فترات التأخير المسموح بها ، للأنشطة .

إدارة الكوارث Risk Management

هي عملية تحديد الكوارث وعمل الخطط اللازمة لتقليل آثارها على المشروع .

كوارث المشروع (Project Risks)

هي الكوارث التي تؤثر على الجدول الزمني للمشروع أو على مصادر تمويل المشروع.

كوارث المنتج (Product Risks)

وهي الكوارث التي تؤثر على جودة أو أداء المنتج أو البرامج التي يتم تطويرها.

كوارث الأنشطة التجارية (Business Risks)

هي الكوارث التي تؤثر في المؤسسة التي تتولى تطوير أو شراء برامج الحاسب الآلي .

كوارث التقنية (Technology Risks)

وهي كوارث تنتج عن تقنيات البرامج أو الأجهزة التي تستخدم كجزء من النظام الذي يتم تطويره.

كوارث ناتجة عن البشر (People Risks)

وهي كوارث تنتج عن البشر الذين يشتركون في فريق عملية التطوير.

كوارث تنظيمية (Organizational Risks)

وهي كوارث تنتج عن البيئة التنظيمية حيث يتم تطوير البرامج.

كوارث ناتجة عن الأدوات المستخدمة (Tools Risks)

وهي كوارث تنتج عن أدوات CASE وبقية البرامج المساندة التي تستخدم في عملية تطوير النظام.

كوارث المتطلبات (Requirements Risks)

وهي كوارث تنتج عن تغير متطلبات العميل وعملية إدارة تغير المتطلبات.

كوارث التقدير (Estimation Risks)

وهي كوارث تنتج عن التقديرات الإدارية لخصائص النظام والموارد المطلوبة لبنائه.

المصطلح بالإنجليزية	معناه بالعربية
Ac tivities	الأنشطة
Activity Bar Chart	المخطط الزمني للأنشطة
An Activity Network	شبكة الأنشـــطة
Avoidance Strategy	إستراتيجية تجنب الكارثة
Bar Charts & Activity Network	مخططات القضبان وشبكات الأنشطة
Business Risks	كوارث الأنشطة التجارية
CASE Tools	أدوات هندسة البرمجيات بمساعدة الحاسب
Contingency Plans	خطط الطوارئ
Deliverables	المواد المطلوب توريدها
Delivery and Installation Team	فريق التسليم والتركيب
Documentation Team	فريق التوثيق
Estimation Risks	كوارث التقدير
Groupware	مكونات المجموعات
Hardware & Software Resource Requirements	متطلبات موارد الأجهزة والبرامج
Implementation Team	فريق التنفيذ
Intuitive Introvert	المنفتحة العقلانية
Intranet	الإنترانت
Intuitive Extrovert	إنطوائية عقلانية
Maintenance Team	فريق الصيانة
Management Activities	النشاطات الإدارية
Metrics Team	فريق التقييس
Milestones & Deliverables	أسس (معالم) المشروع القابلة للتسليم
Minimization Strategy	إستراتيجية الحد من الكارثة
Monitoring & Reporting Mechanisms	آليات المراقبة والتقارير
Nature of Project Teams	طبيعة فرق عمل المشروع

المصطلح بالإنجليزية	معناه بالعربية
Organizational Risks	كوارث تنظيمية
People Risks	كوارث ناتجة عن البشر
Planning Team	فريق التخطيط
Product Risks	كوارث المنتج
Project Management	إدارة المشاريع
Project Management Tools	أدوات(وسائل) إدارة المشروع
Project Manager	مدير المشروع
Project Organization	تنظيم المشروع
Project Planning	التخطيط للمشروع
Project Schedule	الجدول الزمني المشروع
Quality Assurance Team	فريق ضمان الجودة النوعية
Rational extrovert	إنطوائية السلوك
Rational Introvert	منفتحة السلوك
Requirements Risks	كوارث المتطلبات
Requirements Team	فريق المتطلبات
Reuse and Reengineering Team	فريق إعادة الاستخدام
Risk Analysis	تحليل الكوارث
Risk Analysis	تحليل الكوارث
Risk Analysis	تحليل الكارثة
Risk identification	تعريف الكوارث
Risk Management	إدارة الكوارث
Risk Management Strategies	إستراتيجيات إدارة الكوارث
Risk Monitoring	مراقبة الكوارث
Risk Monitoring	مراقبة الكوارث
Risk Planning	تخطيط الكوارث
Risks Project	كوارث المشروع

المصطلح بالإنجليزية	معناه بالعربية
Role of Networks in Project Management	دور الشبكات (الداخلية والإنترنت) في إدارة المشروع
Software Management Tools	برمجيات الإدارة المختلفة
Staff Allocation	تعـــيين الموظفـــين
Sub-Teams Needed in Software Engineering Projects	فرق العمل اللازمة لمشاريع هندسة البرمجيات
System Administration Team	فريق إدارة النظام
System Analysis Team	فريق تحليل النظام
System Design Team	فريق تصميم النظام
Task Durations & Dependencies	الفترات الزمنية والعلاقات بين المهام
Technology Risks	كوارث التقنية
Testing and Integration Team	فريق اختبار وتكامل النظام
Tools Risks	كوارث ناتجة عن الأدوات المستخدمة
Training Team	فريق التدريب
Work Breakdown	تقسيم العمل

إجابات التدريبات

تدريب (1)

الإجابة مختلفة (متنوعة) باختلاف خبرة الاستخدام. وهكـذا يفتـرض أن نتوقـع منهـا دعـم (جدولة) المشروع ، وإعداد التقارير ، وإدارة التغيير ، وأي شيء آخـر يناسـب المشـروع.

وعلى أية حال فإن البرمجية ينبغي أن يتم استخدامها في إعداد التقارير والتحكم في العمليات التي من المعتاد (المألوف) استخدامها في النظام الإداري.

تدريب (2)

النشاط	الزمن	النشاط الذي قبله	بداية مبكرة	نهاية مبكرة	بداية متأخرة	نهاية متأخرة	الزمن الفائض	نوع النشاط
A	2	—	0	0+2=2	0	5-3=2	0	حرج
B	3	A	2	2+3=5	2	8-3=5	0	حرج
C	1	B	5	5+1=6	6	11-4=7	1	
D	3	B	5	5+3=8	5	11-3=8	0	حرج
E	4	C	6	6+4=10	7	14-3=11	1	
F	3	D	8	8+3=11	8	14-3=11	0	حرج
G	2	A	2	2+2=4	4	11-5=6	2	
H	5	G	4	4 + 5=9	6	14-3=11	2	
I	3	E, F, H	11	11+3=14	11	14	0	حرج

في الجدول السابق :

- النهاية المبكرة للنشاط = البداية المبكرة للنشاط + زمن النشاط
- البداية المبكرة للنشاط = النهاية المبكرة للنشاط السابق ، وفي حالة وجود أكثر من نشاط سابق مثل : الأنشطة السابقة للنشاط **"I"** هي الأنشطة : **"H"** ،**"F"** ، **"E"** يتم احتساب أكبر زمن نهاية مبكر لهذه الأنشطة ، وهو في المثال السابق يساوي (11).
- النهاية المتأخرة للنشاط الأخير = النهاية المبكرة له = زمن انتهاء المشروع.
- البداية المتأخرة للنشاط = النهاية المتأخرة للنشاط – زمن النشاط

من الجدول السابق :

- الأنشطة الحرجة (الزمن الفائض يساوي صفر) هي : **"A"، "B" ، "D" ، "F" ، "I"** .
- المسار الحرج هو: " **بداية ، "A" ، "B" ، "D" ، "F" ، "I" ، نهاية "= 14 يوم**
- الزمن الفائض للأنشطة **"C" ، "E" = 1 يوم.**
- الزمن الفائض للأنشطة **"G" ، "H"** = 2 يوم.

تدريب (3)

أولاً : قم بإنشاء الجدول التالي كما هو موضح بالمثال السابق.

النشاط	الزمن	النشاط الذي قبله	بداية مبكرة	نهاية مبكرة	بداية متأخرة	نهاية متأخرة	الزمن الفائض	نوع النشاط
T1	8	ـــ	0	8	0	8	0	حرج
T2	15	ـــ	0	15	3	18	3	
T3	15	T1(M1)	8	23	8	23	0	حرج
T4	10	ـــ	0	10	20	30	20	
T5	10	T2,T4(M2)	15	25	30	40	15	
T6	5	T1,T2(M3)	15	20	18	23	3	
T7	20	T1(M1)	8	28	20	40	12	
T8	25	T4(M5)	10	35	30	55	20	
T9	15	T3,T6(M4)	23	38	23	38	0	حرج
T10	15	T5,T7(M7)	28	43	40	55	12	
T11	7	T9(M6)	38	45	38	45	0	حرج
T12	10	T11(M8)	45	55	45	55	0	حرج

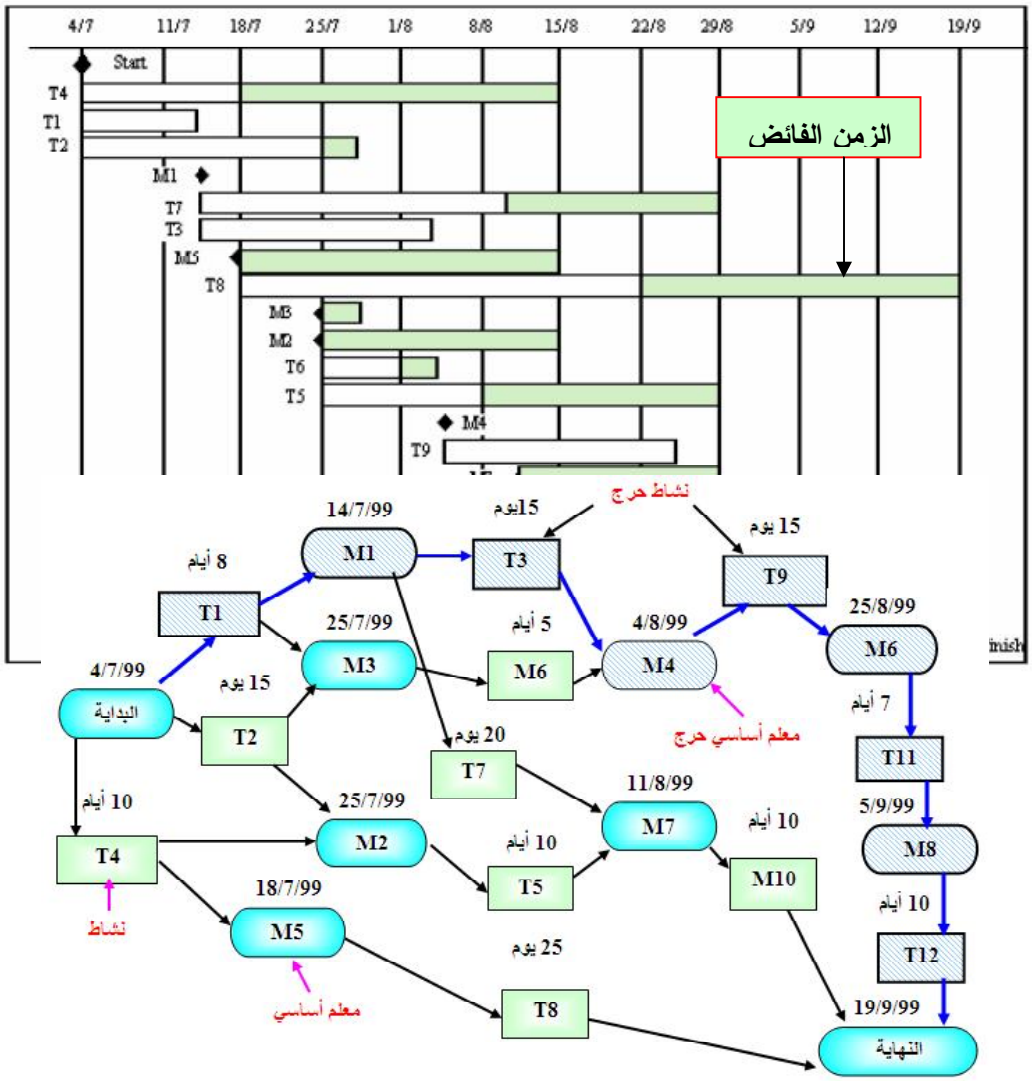
ثانياً : حدد الأنشطة والمعالم الأساسية الحرجة كما هو موضح بالجدول.

ثالثا : حدد المسار الحرج وهو المسار الذي يتضمن الأنشطة والمعالم الأساسية الحرجة وهو :

"البداية" ـ **T1** ـ **T3** ـ **T9** ـ **T11** ـ **T12** ـ "النهاية" ، مروراً بالمعالم الأساسية الحرجة : **M1** ، **M4** ، **M6** ، **M8** .

رابعاً : الزمن الكلي لإنهاء المشروع = مجموع أزمنة الأنشطة الحرجة = 55 يوماً (11 أسبوعاً).

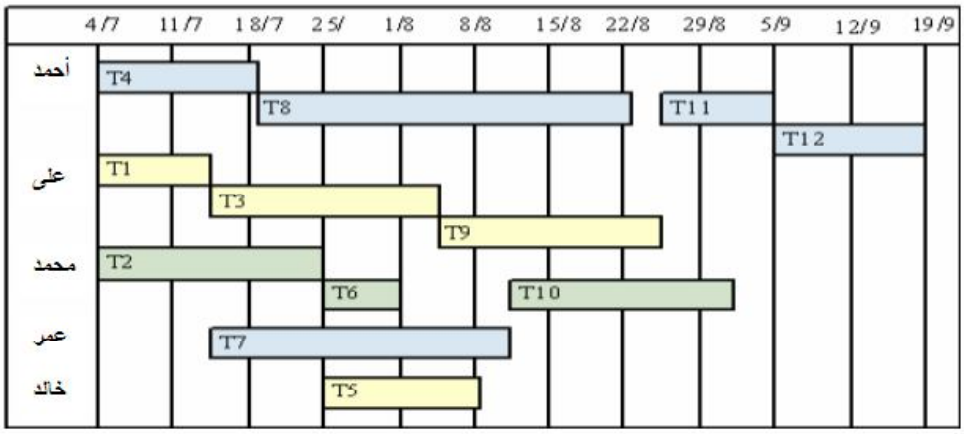
خامساً : بمساعدة الجدول السابق ارسم مخطط **"PERT"** الموضح أدناه.



مخطط **"PERT"**

سادساً : بمساعدة الجدول السابق ارسم مخطط **"Gannt"** الموضح أدناه.

سابعاً : بمساعدة الشكل السابق ارسم مخطط **"Gannt"** التالي لتخصيص فريق العمل ، أي توزيع الموارد على الأنشطة.


4/7
11/7
18/7
25/
1/8
8/8
15/8
22/8
29/8
5/9
12/9
19/9
أحمد
علي
محمد
عمر
خالد
T4
T8
T11
T12
T1
T3
T9
T2
T6
T10
T7
T5

المراجع

أولاً : المراجع العربية:

[1] روجر بريسمان ، "هندسة البرمجيات" ، الدار العربية للعلـــوم ، مركـــز التعريـــب والبرمجة ، الطبعة الأولى ، 1425هـــ – 2004م.

[2] مهندس عبد الحميد بسيوني ، "أساسيات هندسة البرمجيات" ، دار الكتـــب العلميـــة للنشر والتوزيع ، القاهرة ، 2005 م.

ثانياً : المراجع الإنجليزية :

[3] Ian Somerville , "Software Engineering", Addison Wesley, 2001.

[4] Ronald J. Leach,, "Introduction to Software Engineering", CRC Press, 1999.

[5] Douglas Bell , "Software Engineering A Programming Approach", 3rd Edition, Addison Wesley.

[6] **Shari Pfleeger, "Software Engineering - Theory and Practice", 2nd Edition.**

ثالثاً : مواقع على شبكة الإنترنت تم الاستفادة منها :

[7] **http://whatis.techtarget.com/definition/0,,sid9_gci331 ,**
" Gantt chart"

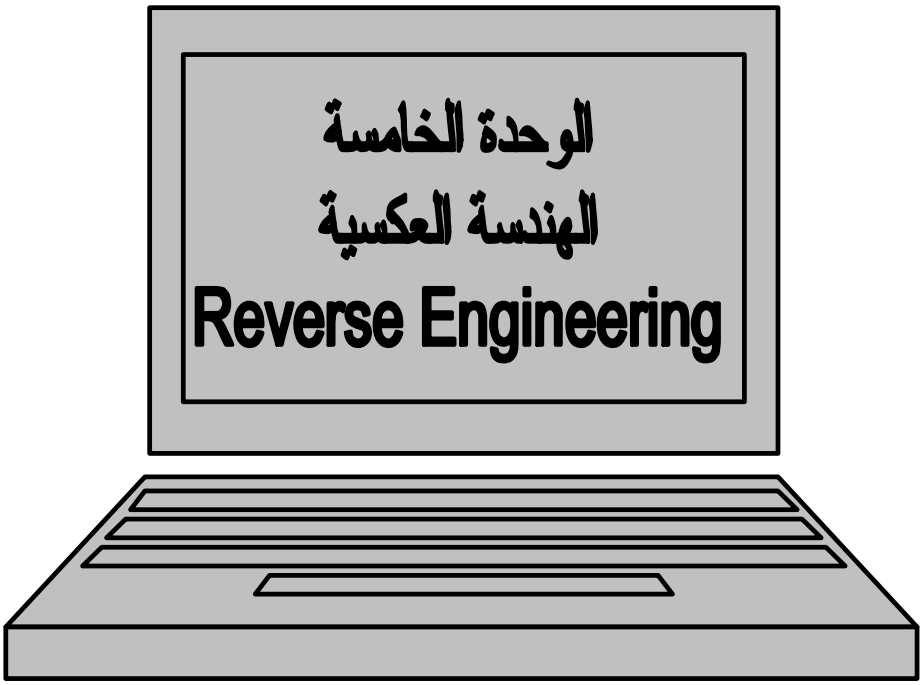
[8] http://www.w3c.org/TR/1999/REC-html401-19991224/loose.dtd
"PERT chart"

[9] http://whatis.techtarget.com/definition/0,,sid9_gci331 ,
" PERT chart "

[10] http://studentweb.tulane.edu/~mtruill/dev-pert.html ,
" PERT, CPM and GANTT"

[11] http://www.mckinnonsc.vic.edu.au/la/it/ipmnotes/ganttpert/pert-tute/pert-tute.htm
" PERT Practice"

[12] http://www.arab-api.org/course8/pdf/ex7.pdf ,
" PERT chart"
[13] http://www.tutorialsandhelp.com/Network%20diagrams.html ,
" Gantt chart "
[14] http://www.itu.org.eg/Doc05/unit%204k.doc ,
" Critical Path method "
[15] http://www.arab-api.org/course8/c8_1.htm ,
" مقدمة في إدارة المشاريع "
[16] http://www.snc.edu/socsci/chair/333/numbers.html ,
" Project Management"


الوحدة الخامسة
الهندسة العكسية
Reverse Engineering

محتويات الوحدة

المقدمة

تمهيد

عزيزي الدارس،

مرحباً بك عزيزي الدارس في الوحدة الخامسة من مقرر "هندسة البرمجيات 2" والتي تبحث في الهندسة العكسية أي محاولة استدعاء مواصفات المستوى الأعلى عن طريق تحليل المنتج. القسم الأول من الوحدة يتناول بعض التعريفات والمفاهيم الهامة المرتبطة بالهندسة العكسية وهنا سنتعرف على الهندسة الأمامية، والفرق بين الهندسة العكسية وبباقي فروع الهندسة، الخلفية التاريخية للهندسة العكسية الفرص التي تقدمها الهندسة العكسية، مميزات الهندسة العكسية. القسم الثاني من الوحدة يوضح أسباب القيام بالهندسة العكسية للبرمجيات. القسم الثالث من الوحدة يبحث في الاستخدامات المختلفة للهندسة العكسية. القسم الرابع من الوحدة يستعرض الهندسة العكسية للبرمجيات والمعدات وهنا نتناول أدوات الهندسة العكسية للبرمجيات، الهندسة العكسية للمعدات (الأجهزة)، أساليب الهندسة العكسية للأجهزة. القسم الخامس من الوحدة يتناول المجموعات التي تستفيد من الهندية العكسية، حيث يقسم القسم المجموعات إلي: المستخدمين – المنافسون – القراصنة. القسم السادس من الوحدة يتناول المواضيع القانونية المتعلقة بالهندسة العكسية، وهنا نستعرض براءات الاختراع، حقوق الطبع والحقوق ذات الصلة بها، برامج الحاسوب، البيانات، حقوق الطبع وحقوق التصميم. القسم السابع من الوحدة يجيب على السؤال : هل تنجح الهندسة العكسية دائماً؟ القسم الثامن من الوحدة يستعرض أنظمة الحماية المختلفة ضد الهندسة العكسية. القسم التاسع والأخير من الوحدة نشرح فيه بعض الأمثلة على استخدام الهندسة العكسية.

أهداف الوحدة

الهدف من هذه الوحدة هو إعطاءك **عزيزي الدارس** مدخل لعلم الهندسة العكسية ، وبحيث تكون مع نهاية دراسة هذه الوحدة قادراً على أن :

- **تذكر** التعريفات والمفاهيم الهامة المرتبطة بالهندسة العكسية.
- **توضح** أسباب القيام بالهندسة العكسية للبرمجيات .
- **تبين** الاستخدامات المختلفة للهندسة العكسية .
- **تشرح** الهندسة العكسية للبرمجيات والمعدات .
- **تذكر** المجموعات التي تستفيد من الهندسة العكسية .
- **تشرح** المواضيع القانونية المتعلقة بالهندسة العكسية.
- **تبين** هل تنجح الهندسة العكسية دائما؟.
- **تشرح** أنظمة الحماية المختلفة ضد الهندسة العكسية .
- **تقدم** أمثلة على الهندسة العكسية .

1. بعض التعريفات والمفاهيم الهامة المرتبطة بالهندسة العكسية

- **الهندسة الأمامية:**

إن الخطوة الأولى في تطوير صورة ذهنية لماهية الهندسة العكسية هي الهندسة الأمامية. إن الهندسة الأمامية هي عملية تحويل مواصفات إلى منتج يقوم بالأداء لغرض معين. تعطي المواصفات الكاملة وصفا لما يقوم به المنتج ويعطي وصفا لمعايير الأداء التي يتعين عليه الوفاء بها. وبتزايد تعقيد أنظمة تقنية المعلومات فإن أهمية مواصفات السلع تزداد هي الأخرى. وما بين المواصفات وما يقوم المنتج بعمله ووصف معايير الأداء التي يتعين عليه الوفاء بها كذلك تكون أهمية المواصفات الجيدة.

- **الهندسة العكسية:**

والهندسة العكسية كما واضح من الاسم هي عكس هذا وبكلمات أخرى هي محاولة استدعاء مواصفات المستوى الأعلى عن طريق تحليل المنتج. وليس من الممكن من الناحية العملية أو حتى من الناحية النظرية استعادة كل شيء في المواصفات الأصلية عن طريق دراسة المنتج.

عند البدء في استخدام الهندسة العكسية في بداية الأمر كانت صعبة للغاية وتأخذ وقتا طويلا. إلا أنها الآن أصبحت أكثر سهولة في جميع الأوقات بسبب تكنولوجيا المعلومات **ولسببين:**

<u>السبب الأول</u>

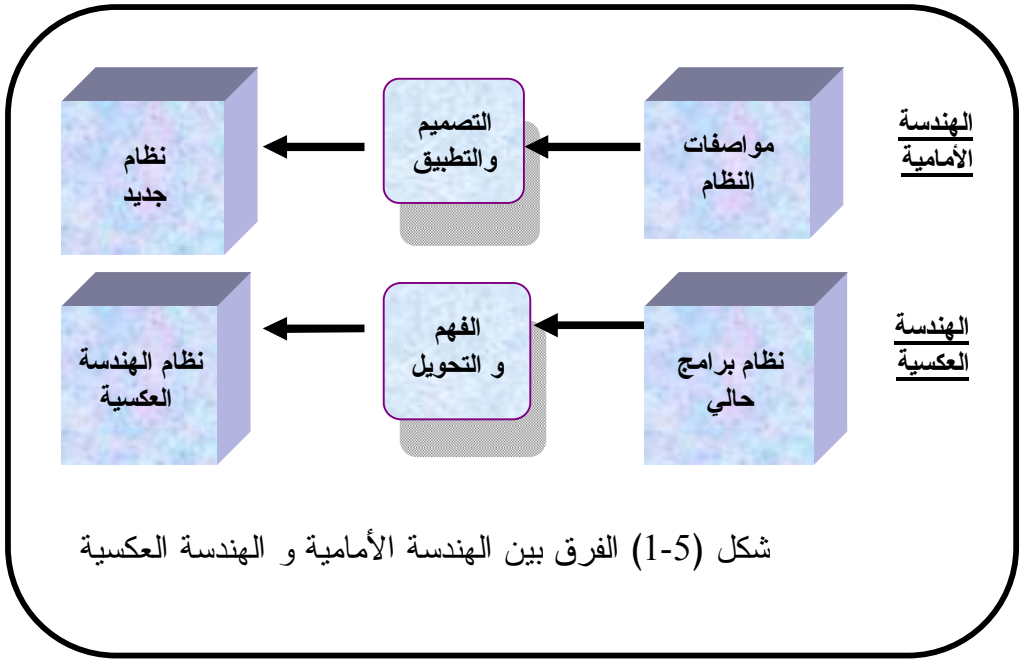
أن الأساليب الهندسية نفسها أصبحت تعتمد على الحاسب الآلي. وأصبح الآن الجزء الأكبر من التصميم يتم عن طريق أجهزة الحاسب الآلي. وعلاوة على ذلك فإن الوحدات التي يمكن تمييزها أو مجموعات عناصر الدائرة في الطبقة الفرعية يمكن

أن توجد في العديد من التصميمات التي ينتجها نفس برنامج جهاز الحاسب الآلي ومن السهولة تمييزها وتفسيرها مما هو الحال مع المنتج المخصص.

السبب الثاني

لقد أصبحت الأساليب الاصطناعية لتمييز الأنماط وتفسيرها متقدمة للغاية بحيث أنها أصبحت تميز التركيبات ضمن المنتج تلقائيا.

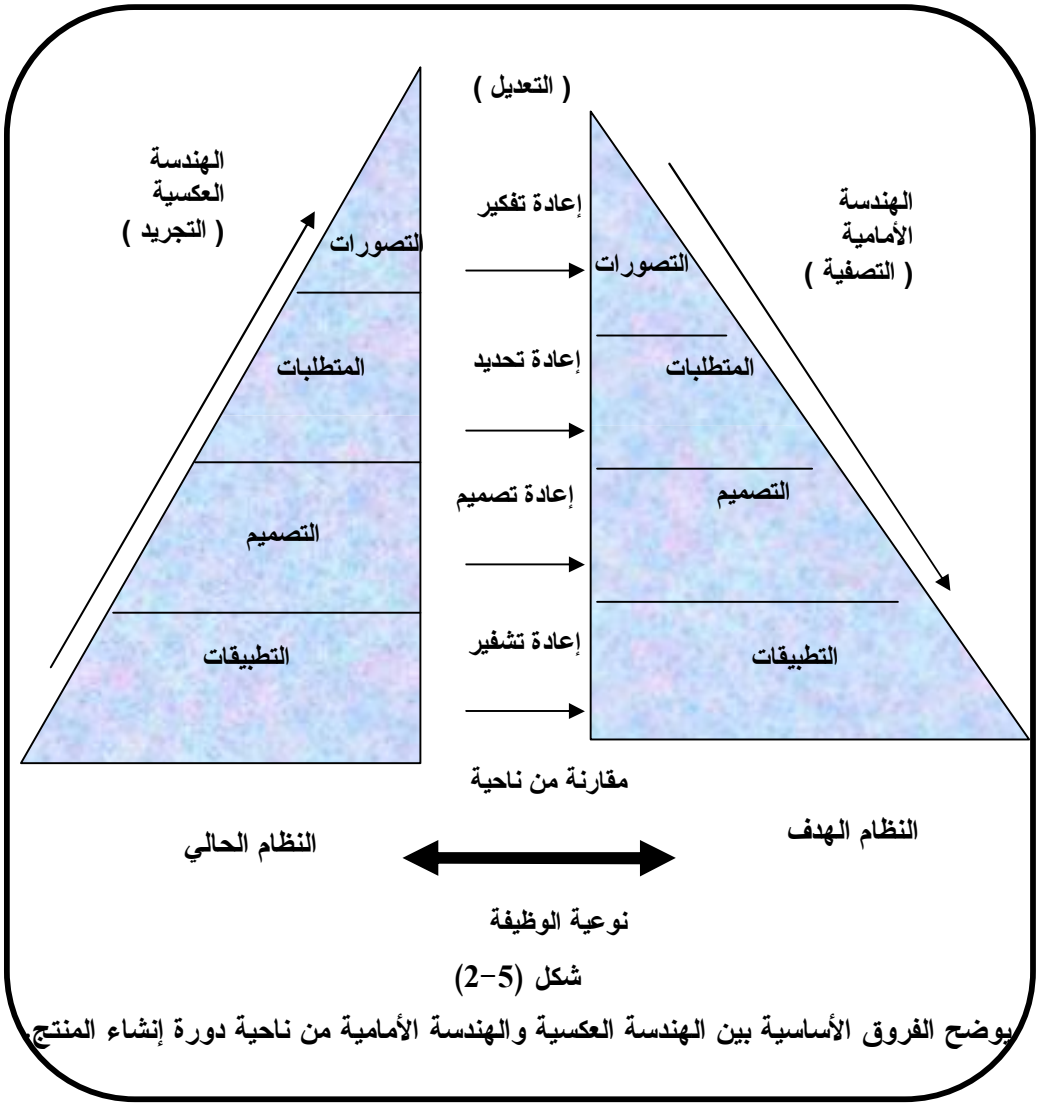
ويمكن أن تُعرَف الهندسة العكسية بأنها عملية تحليل أحد التقنيات تحديدا لاكتشاف كيفية تصميمها أو كيفية عملها. إن الهندسة العكسية كطريقة هي في الغالب جزء مهم من طريقة علمية وتطور تقني. إن عملية تفكيك شيء ما إلى أجزاء وكشف الطريقة التي يعمل بها هي في الغالب طريقة فعالة لمعرفة كيفية تصنيع وتركيب التكنولوجيا أو إدخال التحسينات على هذه التكنولوجيا. والشكل المبين يوضح الفروق الأساسية بين الهندسة الأمامية والهندسة العكسية.



شكل (5-1) الفرق بين الهندسة الأمامية و الهندسة العكسية

• الفرق بين الهندسية العكسية وباقي فروع الهندسة

تختلف الهندسة العكسية عن الفروع الأخرى من الهندسة في أنها تبدأ من المنتج النهائي وتعمل عكسيا لإعادة إنشاء المفاهيم الهندسية عن طريق تحليل تصميم النظام والعلاقات الداخلية التبادلية لمكوناته. والشكل المبين يوضح الفروق الأساسية بين الهندسة العكسية والهندسة الأمامية من ناحية دورة إنشاء المنتج.



شكل (5-2)

يوضح الفروق الأساسية بين الهندسة العكسية والهندسة الأمامية من ناحية دورة إنشاء المنتج

• الخلفية التاريخية للهندسة العكسية

تستخدم الهندسة العكسية في الغالب بواسطة الجيوش لنسخ تكنولوجيا الدول أو أجهزتها أو معلوماتها أو أجزاء منها والتي يتم الحصول عليها بواسطة الجيوش النظامية في ميادين المعارك أو بواسطة العمليات الاستخباراتية. وقد استخدمت الهندسة العكسية في الغالب أثناء الحرب العالمية الثانية والحرب الباردة. وتشمل الأمثلة المعروفة من الحرب العالمية الثانية **ما يلي:**

أ) الجريكانات

لاحظت القوات الأمريكية والبريطانية أن تصميم علب الجازولين الألمانية تصميم ممتاز فقاموا بعمل نسخ معكوسة منها والتي تعرف بالاسم الشعبي المعروف بـ (الجريكانات).

ب) طائرات توبلوف – تي يو 4

تم إجبار عدد من المقاتلات الأمريكية ب – 29 على الهبوط في الأراضي الروسية وقرر السوفيت الذين لم يكن لديهم قاذفات إستراتيجية مماثلة في ذلك الوقت أن يقوموا بنسخ المقاتلة الأمريكية بي – 29. وخلال سنوات قليلة طوروا طائرة تي يو – 4 وهي نسخة تكاد تكون طبق الأصل.

ويمكن إنجاز الهندسة العكسية بطريقة مقبولة إلا أن هناك مخاطر ومحاذير. تعرف الهندسة العكسية عموما بأنها دراسة منتج مكتمل بهدف فهم التكنولوجيا والعملية المستخدمة في تصميمه أو صناعته أو تشغيله. وفي غالب الأحيان تتطلب الهندسة العكسية تفكيك أو حتى تدمير المنتج.

ويتم القيام بالهندسة العكسية في الغالب لأن توثيق جهاز أو أداة محددة مفقود (أو أنه لم تتم كتابته مطلقا) أو أن الشخص الذي قام بتركيبه ترك الشركة. وتبدو الدوائر المتكاملة دائما وكأنها قد شيدت على أنظمة مملوكة وقديمة انتهى زمنها مما يعني أن

الطريقة الوحيدة لتضمين النواحي العملية في تكنولوجيا جديدة هي عكس الرقاقة الإلكترونية الموجودة حاليا وإعادة تصميمها.

❑ ***وهناك من الناحية العملية نوعان رئيسيان من الهندسة العكسية :***

الحالة الأولى:

وفيها يكون رمز المصدر موجودا بالفعل إلا أن المستخدم يكتشف أن هناك توثيق ضعيف للخصائص ذات المستوى الأعلى للبرنامج أو أنها موثقة ولكنها لم تعد ملائمة أو أنها غير قابلة للتطبيق.

الحالة الثانية:

وفيها لا يتوفر رمز المصدر للبرمجيات وبالتالي فإن أي مجهود يبذل لاكتشاف رمز مصدر محتمل يعتبر نوعا من الهندسة العكسية. هذا النوع الثاني من الهندسة هو النوع المألوف لدى الناس.

- **مراحل هندسة المنتجات عكسياً**

وحيث إن عملية الهندسة العكسية قد تأخذ وقتا طويلا كما أنها مكلفة فإن الهندسة العكسية تدرس عموما ما إذا كانت المخاطر المالية لهذه العملية مفضلة على شراء المنتج من الشركة الصانعة الأصلية أو شراء رخصة للمنتج إذا كان ذلك ممكنا.

ولكي تتم هندسة أحد المنتجات عكسيا فإن الباحثين والمهندسين عموما يقومون بإتباع العمليات التالية:

تحديد المنتجات أو المكوّن الذي ستتم هندسته عكسيا.

قياس المنتج رقميا عن طريق أجهزة رقمية ثلاثية الأبعاد أو جهاز مسح بالليزر ثلاثي الأبعاد.

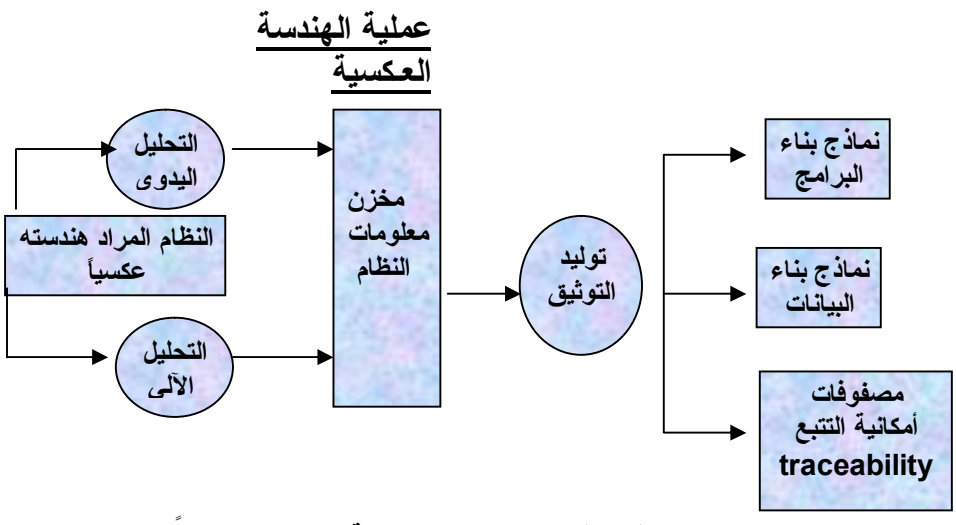
ملاحظة أو مراقبة أو تفكيك المعلومات التي توثق كيفية عمل المنتج الأصلي.

تطبيق البيانات الفنية التي نتجت عن عملية الهندسة العكسية في نسخة مكررة أو معدلة من النسخة الأصلية.

إنشاء منتج جديد (وربما طرحه في الأسواق).

توثيق مواصفات المنتج.

والشكل المبين يوضح المراحل السابقة التي تم سردها.



شكل (5-3) يوضح مراحل هندسة المنتجات عكسياً

• الفرص التي تقدمها الهندسة العكسية

هناك دائما سعي حثيث لتوفير الفرص لتحسين أداء المكونات وعليه فإن الهندسة العكسية تعطي أكثر من مجرد استبدال قطع الغيار.وتشمل الفرص في العادة **ما يلي:**

- تحديث المواد.
- تطبيق التحديثات.
- تحسين تصميم المكونات.
- تحسين قابلية الصيانة.

إن أحد المميزات الرئيسية لتقنية الهندسة العكسية للمستخدم النهائي هو القدرة على تضمين تحديثات التصميم كجزء من العملية. ويمكن للتصميمات التي تم إنتاجها منذ سنوات عديدة أن تستفيد من التغييرات في التصميم أو تحديثات المواد.

- **مميزات الهندسة العكسية**

تتميز الهندسة العكسية بالمميزات **التالية:**

- التوفر السريع للموديلات التي يتم تنفيذها بالتصميم بمساعدة الكمبيوتر (CAD).
- يتم استخدام الموديل الطبيعي كنقطة بداية.
- يكون وقت تطوير المنتج أقصر.
- يكون المنتج تام التطوير عند بداية الإنتاج.

- **الفوائد التي يجنيها العميل من الهندسة العكسية**

ومهما كان الأسلوب المستخدم فإن خدمات الهندسية العكسية توفر الفوائد والمميزات التالية للعملاء:

- تحسين الكفاءة.
- تحسين الأداء.
- تقليل تكاليف التشغيل والصيانة.
- إعطاء نتائج مضمونة.

وفي بعض المواقف يفضل المصممون عمل النماذج من الطين والجبس والخشب أو المطاط ولكن هناك حاجة إلى عمل نموذج بالتصميم بمساعدة الكمبيوتر (CAD) في عملية الإنتاج. ويكون التصميم بمساعدة الكمبيوتر مطلوبا في عمليات الإنتاج الإضافية. إن التصميم بمساعدة الكمبيوتر يأخذ وقتا طويلا ولا يمنح ضمانة بأن تكون الموديلات متماثلة. وتوفر الهندسة العكسية حلا لهذه المشكلة لأن الموديل الطبيعي هو مصدر المعلومات للنموذج المرتكز على الحاسب الآلي.

- **علاقة التصميم بالحاسب الآلي والهندسة العكسية**

لقد زادت شعبية التصميم بمساعدة الحاسب الآلي وبالتالي فقد أصبحت الهندسة العكسية طريقة عملية لإنشاء نماذج ثلاثية الأبعاد لجزء طبيعي موجود للاستخدام في التصميم ثلاثي الأبعاد بمساعدة الحاسب الآلي والبرمجيات الأخرى. وتتضمن عملية الهندسة العكسية قياس شيء وإعادة تشييده كنموذج ثلاثي الأبعاد. ويمكن قياس الشيء باستخدام تكنولوجيا المسح ثلاثية الأبعاد مثل ماسحات الليزر. إن المعلومات التي يتم قياسها تنقصها المعلومات الطوبولوجية وبالتالي تتم معالجتها في شكل قابل للاستخدام مثل الشبك ذو الوجه المثلثات أو نموذج التصميم بمساعدة الحاسب الآلي. وقد حصلت قوات التحالف الغربي على المستندات الفنية لمنتج V2 والتقنيات الأخرى ذات العلاقة به في نهاية الحرب. وقد كان على المهندسين السوفيت والألمان الأسرى إعادة إنتاج المستندات الفنية والخطط الفنية بالعمل من المعدات التي حصلوا عليها وذلك بغرض نسختهم من الصاروخ (R-1) الذي بدأ به السوفيت برنامجهم الصاروخي لفترة ما بعد الحرب والذي قاد إلى صناعة الصاروخ (R-7) وبداية الصراع على الفضاء.

وقد تستخدم منظمات الأعمال التجارية الهندسة العكسية أيضا لتضمين الهندسة الطبيعية في بيئة تطوير المنتجات الرقمية ولعمل سجلات رقمية ثلاثية الأبعاد لمنتجاتهم أو تقييم منتجات المنافسين كما أنها تستخدم لتحليل كيفية عمل المنتجات مثلا وماذا تفعل هذه المنتجات وما هي مكوناتها وما هي تكاليفها التقريبية وكذلك لتمييز الانتهاكات المحتملة لبراءات الاختراع.

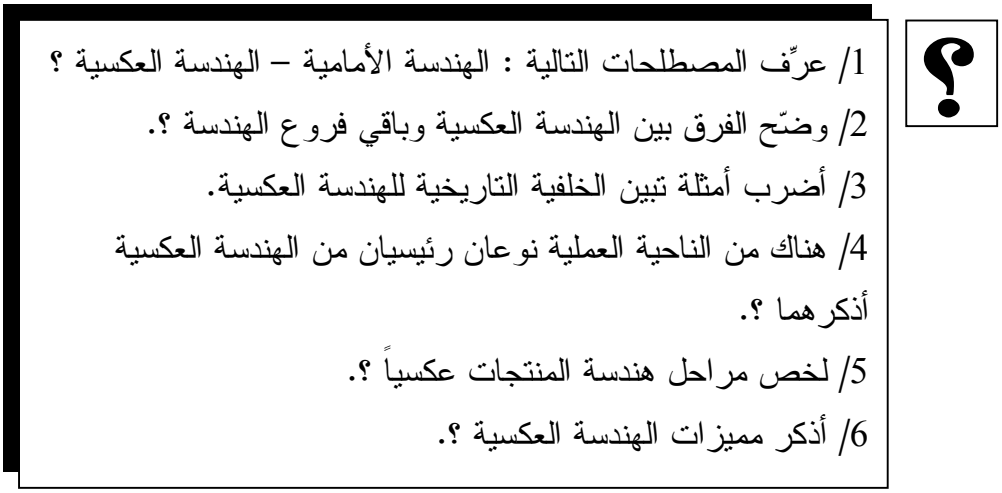
ويمكن للباحثين عن طريق استخدام الهندسة العكسية القيام بالتالي:

- فحص ودراسة قوة الأنظمة .
- تحديد نقاط الضعف في منتجاتهم من حيث الأداء والأمن والتشغيل المتبادل.

تسمح عملية الهندسة العكسية للباحثين فهم كلا من كيفية عمل البرنامج وأيضا ما هي النواحي التي يسهم بها البرنامج في عدم عمله. ويمكن للشركات الصانعة المستقلة

المشاركة في سوق تنافسي مثال التدقيقات الأمنية التي تتيح لمستخدمي البرمجيات تحسين منتجاتهم وشبكاتهم بالكشف عن الأخطاء الأمنية التي تتطلب الهندسة العكسية. وتقوم في الغالب مجموعات مختلفة من المهندسين بأداء كل خطوة بشكل منفصل مستخدمين فقط المستندات لتبادل المعلومات التي تعلموها في كل خطوة. إن هذا لمنع تكرار التكنولوجيا الأصلية التي قد تنتهك حقوق الطبع. إن الهندسة العكسية تنشئ تطبيقا مختلفا بنفس الناحية التشغيلية.

أسئلة تقويم ذاتي



1/ عرِّف المصطلحات التالية : الهندسة الأمامية – الهندسة العكسية ؟
2/ وضّح الفرق بين الهندسة العكسية وباقي فروع الهندسة ؟.
3/ أضرب أمثلة تبين الخلفية التاريخية للهندسة العكسية.
4/ هناك من الناحية العملية نوعان رئيسيان من الهندسة العكسية أذكرهما ؟.
5/ لخص مراحل هندسة المنتجات عكسياً ؟.
6/ أذكر مميزات الهندسة العكسية ؟.

2. أسباب القيام بالهندسة العكسية للبرمجيات

يتم القيام بالهندسة العكسية للبرمجيات **للأسباب التالية:**

- للحصول على رمز المصدر للبرنامج لأن رمز المصدر مفقود.
- لدراسة الكيفية التي يقوم بموجبها البرنامج بأداء بعض العمليات.
- لتحسين أداء البرنامج.
- لتصحيح خطأ في البرنامج عندما لا يكون رمز المصدر موجودا.
- للتعرف على المكونات الضارة في البرنامج مثل الفيروسات.

- ❋ لتكييف برنامج مكتوب للاستخدام مع معالج دقيق من نوع معين لكي يعمل مع معالج دقيق من نوع آخر.
- ❋ التدقيق الأمني.
- ❋ إزالة حماية النسخ.
- ❋ التغلب على موانع الدخول الموجودة في غالب الأحيان في إليكترونيات المستهلكين.
- ❋ تخصيص نظام محدد مضمن (مثل أنظمة إدارة المحركات).
- ❋ القيام بالإصلاحات والتعديلات.
- ❋ تمكين تشغيل مميزات إضافية في البرمجيات ذات التكلفة المنخفضة (مثل مجموعات الرقاقات الإليكترونية في كرت الصور).

- ▪ جانب نفسي للشعور بالرضا عن القيام بعملية الهندسة العكسية لأحد التقنيات.

إن القيام بالهندسة العكسية لمجرد نسخ أو تكرار البرامج يمثل انتهاكا لحقوق الطبع وهي غير قانونية. وفي بعض الأحيان تحدد رخصة البرمجيات إمكانية القيام بالهندسة العكسية على البرنامج.

أسئلة تقويم ذاتي

1/ ما هي أهم أسباب القيام بالهندسة العكسية للبرمجيات ؟.

3. الاستخدامات المختلفة للهندسة العكسية

هناك سوء فهم شائع يعتبر أن الهندسة العكسية تستخدم لغرض سرقة أو نسخ أعمال تخص جهات أخرى. إن الهندسة العكسية لا تستخدم فقط لمعرفة كيفية عمل أحد التقنيات ولكن أيضا لمعرفة الطرق التي لا تعمل بها.

وهناك الكثير من الاستخدامات للهندسة العكسية **من ضمنها:**

- فهم كيفية عمل أحد المنتجات فهما شاملا أكثر من مجرد ملاحظته فقط.
- فحص الأخطاء والقصور في البرامج الحالية وتصحيحها.
- دراسة مبادئ التصميم لأحد المنتجات كجزء من عملية التعليم في علوم الهندسة.
- جعل المنتجات والأنظمة تتوافق مع بعضها حتى تعمل معا أو تتشارك في المعلومات.
- تقييم المنتج الخاص بك لفهم نقاط القصور فيه.
- لتحديد ما إذا كانت جهة أخرى قد نسخت حرفيا عناصر تقنية أخرى.
- عمل التوثيق لتشغيل أحد المنتجات التي لا تستجيب شركتها الصانعة إلى طلبات خدمة العملاء.
- تحويل المنتجات القديمة الطراز إلى منتجات مفيدة بتكييفها إلى أنظمة ومنصات حديثة.

ويكون للشركات الحق في الحصول على المعلومات عن منافسيهم والتي تكون متوفرة للجمهور. إلا أنه يتوجب على هذه الشركات أن لا تحصل على المعلومات بالغش والخداع والتعريف الخطأ عن أنفسهم. إضافةَ إلى ذلك من الضروري أن يتجنبوا كل أنواع النقاشات المباشرة مع منافسيهم حول مواضيع مثل التكاليف والأسعار والعمليات إلا على نطاق ضيق وفي مواقف معينة عندما يكون المنافسون موردين أو عملاء أو كاتحادات تربطها علاقات عمل مشروعة مسموح بها.

وهناك فرق ما بين القيام بالهندسة العكسية بغرض جمع معلومات مفيدة والاستفادة منها وبين استخدام ممارسات غير مقبولة وغير مسموح بها في جمع المعلومات والتي سيكون لها نتائج قانونية وأخلاقية خطيرة.

وفيما يلي بعض الأمثلة لمثل هذه الممارسات:

- تمثيل نفسك تمثيلا غير صحيح على أنك تعمل لجهة أخرى.
- التآمر للحصول على الأسعار أو تحديد الأسواق أو العملاء.
- انتقاد أعمال أحد المنافسين والتقليل من شأنها إلى العملاء والجهات الأخرى.
- محاولة الحصول على المعلومات السرية عن الأعمال التجارية للمنافسين.

أسئلة تقويم ذاتي

1/ "هناك الكثير من الاستخدامات للهندسة العكسية" أشرح هذه العبارة.

4. الهندسة العكسية للبرمجيات والمعدات

عموما تستخدم الهندسة العكسية في:

- تطوير الموديلات المرتكزة على التصميم بمساعدة الكمبيوتر.
- تحرير ملفات التصميم بمساعدة الحاسب الآلي.
- إنشاء ملفات للاستخدام في تقنيات عمل النماذج الأولية الأخرى.

ويمكن استخدام الهندسة العكسية إما على البرمجيات أو الأجهزة.

1.4 الهندسة العكسية للبرمجيات

تتضمن الهندسة العكسية للبرمجيات عكس رمز الماكينة رجوعا إلى رمز المصدر الذي كُتب له باستخدام لغة البرنامج.

- **أدوات الهندسة العكسية للبرمجيات**

من الممكن استخدام عدد من الأدوات لتفكيك البرنامج وفيما يلي بعض الأمثلة لهذه الأدوات:

(أ) جهاز الهيكساديسمال

يقوم هذا الجهاز بطباعة أو عرض الأعداد الثنائية للبرنامج في شكل سداسي الأرقام مما يسهل قراءته أكثر من الشكل الثنائي. وبمعرفة نمط "البت Bit" الذي يمثل تعليمات المعالجة وكذلك تعليمات الأطوال يمكن لمهندس الهندسة العكسية تمييز بعض الأجزاء للبرنامج ليرى كيفية عملها.

(ب) المفكك

وهي برامج تحول رمز الموضوع مرة إلى لغات المستوى العالي مثل لغة C. وتتوفر مجموعة من الأدوات الأخرى التي يمكنها تتبع تحليل تشغيل أحد البرامج وتمثيل دوائر البرنامج وهكذا.

- **ما هو الفرق بين رمز المصدر ورمز الموضوع**

رمز المصدر هو فئة تعليمات لغة جهاز الحاسب الآلي التي يكتبها مبرمجو البرمجيات في الغالب قراءتها. لا يستطيع الحاسب الآلي عموما تشغيل برنامج مكتوب بلغة رمز المصدر. وتتم ترجمة رمز المصدر باستخدام جهاز تجميع في شكل لغة يتضمن تعليمات على الحاسب الآلي يعرف كرمز للموضوع. ويتكون رمز الموضوع من رموز عددية تحدد كل تعليمات الحاسب الآلي التي يجب تنفيذها وكذلك مواقع البيانات في الذاكرة التي تعمل فيها التعليمات.

- **التشغيل المشترك**

يسمح التشغيل المشترك في العادة للتقنيات المختلفة بالعمل مع بعضها البعض عندما تستخدم نفس المدخلات وإنشاء نفس المخرجات. وبالنسبة إلى أجهزة الحاسب الآلي فإن التشغيل المشترك هي قدرة البرامج والأنظمة على العمل في الأنواع المختلفة من البرمجيات والمعدات للاتصال مع بعضها البعض.

إن وجود معايير تعزز التشغيل المشترك بين الأجهزة والأنظمة للتأكد من أن كل المجموعات تطبق المعايير وتفسرها بنفس الطريقة حتى تستطيع التكنولوجيا إعطاء أداء متوافق بغض النظر عن النموذج الفردي. وبالمقارنة مع ذلك فإن عدم وجود

المعايير يعني أن على الأطراف أن تقوم بالهندسة العكسية للتكنولوجيا حتى تستطيع الاتصال مع بعضها البعض. إضافة إلى ذلك فإن مالكي التقنيات غير الحصرية المعيارية يحتفظون بالسيطرة على تحديث وتطوير هذه التقنيات وقد يغيرونها بإرادتهم مما يخل بالتشغيل المشترك مع التقنيات الأخرى.

ما تم سرده بالنسبة للبرمجيات أما بالنسبة للبيانات فتتبع منهجيات إعادة هندسة البيانات Approaches to data re-engineering

يمكن تقسيم منهجيات إعادة هندسة البيانات **كالتالي:**

أ- تنظيف البيانات Data Cleanup: وفيها يستبعد التكرار وتحذف المعلومات الزائدة.

ب- امتداد البيانات Data Extension: وفيها تعاد هندسة البيانات لاستبعاد القيود على معالجتها وقد يتطلب ذلك تغيرات في أطوال الحقول أو تغيير حدود جداول.

ت- هجرة البيانات Data migration: وفيها تنتقل البيانات إلى تحكم إدارة قواعد بيانات حديثة وقد تخزن البيانات في ملفات منفصلة أو قد تدار بواسطة قاعدة بيانات أخرى. ***وهناك العديد من مشاكل هجرة البيانات من أهمها:***

- مشاكل تسمية البيانات Data naming problems .
- مشاكل طول الحقل Field length problems .
- مشاكل تنظيم السجلات Record organization problems .
- مصطلحات عالية التشفير Hard coded literals .
- عدم وجود قاموس بيانات No data Dictionary .

أسئلة تقويم ذاتي

1/ أشرح كيف يمكن تطبيق الهندسة العكسية على كل من البرمجيات والأجهزة.

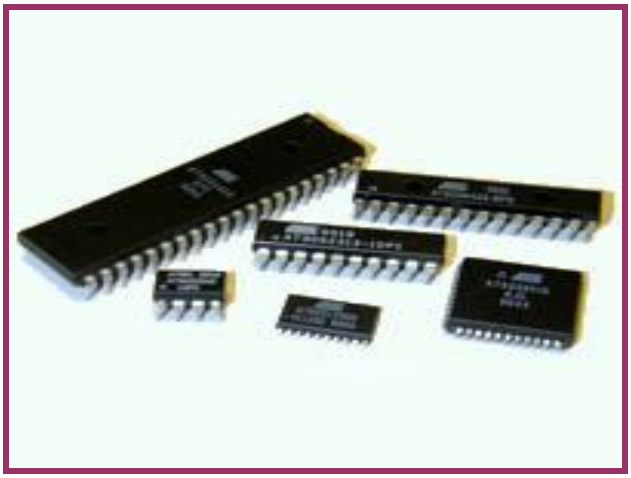
2/ إضرب أمثلة لأدوات الهندسة العكسية للبرمجيات.

3/ ما هو الفرق بين رمز المصدر ورمز الموضوع ؟.

4/ وضح تقسيم منهجيات إعادة هندسة البيانات ؟.

2.4 الهندسة العكسية للمعدات(الأجهزة)

تتضمن الهندسة العكسية للمعدات تفكيك الجهاز أو المعدة لرؤية كيفية عملها. وكمثال إذا أرادت شركة صانعة للمعالجات الدقيقة أن ترى كيف يعمل معالج دقيق في الحاسب الآلي فيمكنهم شراء معالج دقيق خاص بالمنافس وتفكيكه ومن ثم صناعة معالج دقيق مماثل له. إلا أن هذه العملية غير مشروعة في العديد من الدول. وعموما تتطلب الهندسة العكسية قدرا كبيرا من الخبرة وهي عملية مكلفة للغاية.



شكل (5-4)

- **أساليب الهندسة العكسية للأجهزة**

المكونات الهندسية

التحليل الوظيفي

يمكن مراقبة حالات الدخل والخرج للرقاقة على جهاز "الأوسيلسكوب" أو المجاسات المستخدمة للأغراض الخاصة للحصول على صورة من تصرف الرقاقة مع الوقت أو استجابة إلى إشارات الدخل.

لوحات الدوائر الإليكترونية

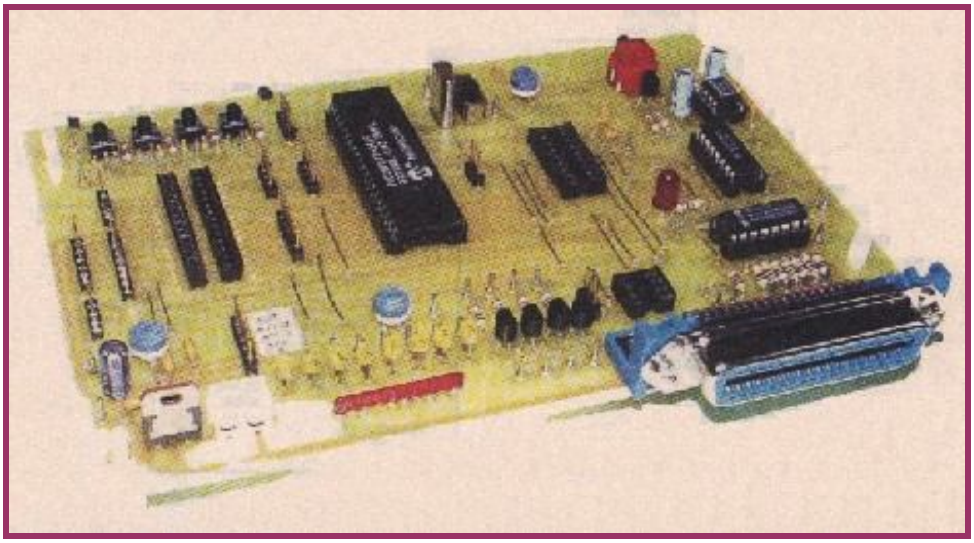
وقد تم استخدام تكنولوجيا البصريات باستخدام الحاسب الآلي على نطاق واسع لمسح لوحات الدوائر الكهربائية لأغراض ضبط الجودة والفحص وبناء على ذلك هناك عدد من الماكينات التي تستخدم لتحليل لوحات الدوائر الإليكترونية المنتجة عن طريق الهندسة العكسية. وهناك العدد من الشركات الموجودة على شبكة الإنترنيت تقدم خدمات مسح لوحات الدوائر الإليكترونية وتوفير قائمة بها.

مكونات الدوائر المتكاملة

إن الهندسة العكسية لمكونات الدوائر المتكاملة هي عمل أكثر صعوبة حيث أن كل شيء أصغر للغاية. إن الخطوة الأولى هي الحصول على المواد المضمنة في المنتج نفسه عن طريق الحفر الكيمائي. إن هذا عمل صعب للغاية في نفسه لأن بعض الشركات الصانعة تقوم بتضمين جزيئات صلبة من مواد صلبة مثل الكاربوندوم أو السافير حتى يقوم الحك الميكانيكي بتدمير الرقاقات الإليكترونية. وبمجرد الوصول إلى سطح الرقاقة فإنه يتم تصوير كل طبقة من المكونات وإزالتها لتكشف عن الطبقة التي تليها. تكشف هذه العملية عن تركيبة الرقاقة الإليكترونية. ومرة أخرى يمكن جعل العملية أكثر صعوبة مثلا بوضع بعض المكونات عموديا على عدد من الطبقات.

وعلى الرغم من أن هذه العمليات تكشف عن سطح الرقاقة الإليكترونية إلا أنها لا تحدد الفولطية في كل نقطة. ولكن إذا لم يتم تدمير الرقاقة فإنه من الممكن استخدام تباين

الميكرسكوب الإليكتروني لمسح الرقاقة الإليكترونية المستخدمة ومراقبة مستوى الفولطية مع الوقت. وبعد "تقشير" الرقاقة يمكن تحليلها باستخدام برمجيات التعرف على الأنماط والفحص البشري للحصول على قائمة ومن ثم الرسم البياني للرقاقة. وتوفر العديد من الشركات الموجودة على الإنترنيت خدمات مسح الخلايا العادية لإنشاء قائمة بشكل تلقائي (أيوصف المستوى الأدنى للدائرة الذي يمكن استخدامه لإنشاء دائرة متكاملة جديدة). والشكل يبين دائرة إلكترونية يتم اكتشاف مكوناتها عن طريق الهندسة العكسية.



شكل (5-5)

أسئلة تقويم ذاتي

1/ كيف تتم الهندسة العكسية على كل من لوحات الدوائر الإليكترونية ومكونات الدوائر المتكاملة ؟.

5. المجموعات التي تستفيد من الهندسة العكسية

يمكن تقسيم المجموعات التي تستفيد من الهندسة العكسية **كالتالي:**

① المستخدمين.

② المنافسون.

③ القراصنة.

❶ المستخدمين

إن إحدى مجموعات المستفيدة من الهندسة العكسية هم مستخدمي المنتجات القديمة الذين يرغبون في الحفاظ عليها ولكنهم يجدون أن المورد الأصلي غير موجود الآن أو أنه قد تخلى عن دعم المنتج. إن معظم المستخدمين سيتطلبون خدمات مقاول من الباطن لإعادة هندسة البرمجيات بدلا عن القيام بذلك بأنفسهم. للاحتفاظ بمنتج قديم فإن من الضروري فهم الكيفية التي يعمل بها.

❷ المنافسون

قد يرغب المنافسون الحقيقيون استخدام أساليب الهندسة العكسية **لسببين:**

<u>السبب الأول:</u>

- قد يرغبون في إنتاج منتج يعمل مع المنتج الذي يتم تحليله (ولكنه لا يتنافس معه)

<u>السبب الثاني:</u>

- قد يرغبون في إنتاج منتج يتنافس مع المنتج الذي يتم تحليله.

أمثلة المنتجات ذات التشغيل المشترك هي التطبيقات التي تتطلب التشغيل المشترك مع أنظمة التشغيل أو التبادلات المضبوطة التي تتطلب التشغيل مع الآخرين والبرامج التي يتم إغلاقها بواسطة أدوات الحماية (الدنقل).

❸ القراصنة

أخيرا هناك القراصنة الذين قد يستخدمون الهندسة العكسية في بعض الحالات. إن القراصنة في العادة لا يحتاجوا إلى فهم الطريقة التي يعمل بها المنتج وذلك بغرض

نسخه، ولكن في بعض الحالات قد يتضمن المنتج مميزات أمنية يحتاج القراصنة إلى التغلب عليها.

إن المنافس الحقيقي سيحاول تقليل استخدامه للوصف المتوسط الذين ينتج عن الهندسة العكسية إلى أقل حد ضروري ممكن وذلك لتجنب النسخ بينما يحاول القرصان استخدام الرمز الناتج عن الهندسة العكسية إلى أقصى مدى ممكن وإضافة الحد الأدنى من المواد التي أنتجها بشكل مستقل لكي يجعل المنتج يشبه المنتج الأصلي إلى أكبر مدى ممكن لكي يقلل الجهد الذي يقوم به.

أسئلة تقويم ذاتي

1/ ما هي أهم المجموعات التي تستفيد من الهندسة العكسية؟.

6. المواضيع القانونية المتعلقة بالهندسة العكسية

إن مفهوم الهندسة العكسية يلاقي شعبية كبيرة في المجتمع الهندسي. ولكن على الرغم من ذلك فإن الموقف التشريعي من الهندسة العكسية للمعدات يجعل الهندسة العكسية للبرمجيات قانونية فقط في مواقف محددة للغاية في أوربا وفي الولايات المتحدة الأمريكية فإنه تتوفر حرية أكثر قليلا للمهندسين المعنيين بالهندسة العكسية إلا أنه يتوجب على هؤلاء المهندسين توخي الحيطة والحذر لتجنب انتهاك القانون.

- **براءات الاختراع**

تباع الكثير من السلع المسجلة ببراءات الاختراع بدون أن تكون مصحوبة بتقييد بموجب رخصة وعليه فإن براءة الاختراع لا تمنع في العادة مشتري السلعة من عمل ما يريد بالمنتج المسجل ببراءة اختراع.

وبالطبع فإن براءة الاختراع نفسها تعطي المهندس معلومات قيّمة عن كيفية تشغيل المنتج المسجل ببراءة الاختراع. ولكن على الرغم من ذلك فإن المنتج الذي تم إنتاجه بواسطة الهندسة العكسية لا يزال في حالة انتهاك للبراءة نفسها.

- **حقوق الطبع والحقوق ذات الصلة بها**

من المقبول على نطاق واسع أن حقوق الطبع لا تحمي "الأفكار" ولكنها تحمي فقط "التعبير" عن هذه الأفكار أي الطريقة التي يتم التعبير بها عن هذه الأفكار. إن المواصفات في المستوى الأعلى هي نفسها أعمال تخضع لحقوق الطبع. وهناك افتراض عام بأن المستوى الأعلى من مواصفات التصميم لا يمكن حمايته بواسطة الحقوق الفكرية إذ أنه يتكون فقط من "أفكار". ويفترض البعض بأن هذا ينطبق أيضا على بعض المستويات الدنيا من حيث تغطيتها للوغريتم وليس للرمز الفعلي.

- **برامج الحاسب الآلي**

في حالة أجهزة الحاسب الآلي فإن الأفكار والمبادئ الأساسية للبرنامج لا تحميها حقوق الطبع وأنه من الممكن أن يشمل اللوغاريتم والمنطق ولغات البرمجة الأفكار والمبادئ. إن تحليل وظيفة البرنامج مسموح به إذا تم القيام به بواسطة مستخدم يحمل رخصة في حالة الاستخدام العادي للبرنامج. ويسمح باستخدام الهندسة العكسية ولكن فقط لغرض واحد هو إنتاج برنامج قابل للتشغيل مع برامج أخرى (وليس متنافسا معها).

ولهذا الغرض وبالإضافة إلى الهندسة العكسية نفسها (أي إنتاج نسخة من المستوى العالي للرمز) فإن الهندسية الأمامية التالية لإنتاج البرنامج القابل للتشغيل مع البرامج الأخرى مسموح بها.

ولكن لا بد للمهندس المعني بالهندسة العكسية أن يمر خلال العديد من الحواجز الصعبة قبل استخدام هذا الحق **وهذه الحواجز هي:**

أ- لا بد أن يحتم الموقف استخدام الهندسة العكسية للحصول على المعلومات.

ب- يتعين أن تكون الهندسة العكسية بموجب رخصة أو مستخدم معتمد.

ت- يجب أن لا تكون المعلومات الضرورية متوفرة بسهولة لهؤلاء الناس.

ث- يمكن فقط إنتاج أجزاء البرنامج اللازمة للتشغيل مع الأجهزة الأخرى.

ج- يمكن استخدام المعلومات الناجمة عن الهندسة العكسية فقط لتحقيق قابلية التشغيل المشترك لبرنامج يتم إنشاؤه بشكل مستقل.

ح- لا يمكن إعطاء المعلومات إلى الآخرين إلا في حالات الضرورة لمثل هذا الغرض.

خ- لا يمكن استخدام المعلومات التي يتم الحصول عليها لعمل برنامج منافس

د- يجب عدم المساس بالاهتمامات المشروعة لصاحب حقوق الطبع للبرنامج.

- **البيانات:**

يمكن حماية قواعد البيانات أو مجموعات البيانات عموما بموجب قانون الحماية الفكرية. ووفقا لقانون الحماية الفكرية السابق في المملكة المتحدة فإن أبسط جداول البيانات كانت تعتبر أعمالا أدبية إلا أنه وفقا للتوجه الأوربي فإن نوع البيانات التي تكوَن مستندات التصميم للبرمجيات والمعدات أو الترابطات تستحق في بعض الحالات النادرة فقط حمايتها بموجب قانون الحماية الفكرية.

- **حقوق الطبع وحقوق التصميم**

لا يعطي قانون حقوق الطبع أي اعتبارات خاصة للهندسة العكسية. إن هذا الموقف سيتغير في نهاية الأمر في المجالات المحددة لبرمجيات الحاسب الآلي وأشباه الموصلات، ولكن في حالات نسخ أي مستند تصميم غير برنامج للحاسب الآلي فإن قوانين حقوق الطبع العادية سوف تنطبق.

- **السرية**

لا تعطي الهندسة العكسية حقا عاما لانتهاك الالتزامات بالحفاظ على سرية المعلومات. إن هذا الموقف يمثل أكثر صعوبةَ عندما يكون هناك منتج موزع على نطاق واسع ولكن بموجب عقد بيع يعلن أنه يشمل فقرة السرية.

أسئلة تقويم ذاتي

1/ أشرح الوضع القانوني للفرعيات التالية فيما يتعلق بالهندسة العكسية : براءة الاختراع- حقوق الطبع- برامج الحاسب الآلي- البيانات- حقوق الطبع وحقوق التصميم- السرية.

7. ولكن هل تنجح الهندسة العكسية دائما؟

- من الممكن في غالب الأحيان التشغيل الآلي لجيل من المواصفات الإنشائية للمنتج. إلا أن معرفة وظيفتها أمر صعب للغاية ولا يزال يتطلب مهارات العنصر البشري. وببساطة قد لا يكون ممكنا إعادة بعض الأجزاء من المواصفات الأصلية.
- وحيث أن الهندسة العكسية لا تزال تحتاج إلى العنصر البشري، فإن الهندسة العكسية تحتاج في أحد المراحل إلى إنتاج وصف كامل لنظام المنتج ليسمح لأحد العناصر البشرية لكي يرى كيف يعمل المنتج ومن ثم يمكن تقسيم المنتج إلى الأجزاء المكونة له.

عليه فإن الهندسة العكسية تتكون بصورة عامة من المراحل التالية:

أ- تحليل المنتج.

ب- إنشاء وصف المستوى المتوسط للمنتج.

ت- التحليل البشري لوصف المنتج لإنتاج المواصفات.

ث- إنشاء منتج جديد باستخدام المواصفات.

إذا كان نفس الشخص يقوم بعملية الهندسة العكسية للمنتج القديم ويصمم المنتج الجديد وهناك أمور متشابهة فمن الصعب تجنب الافتراض أنه قد تم القيام ببعض عمليات النسخ. وعليه فإن الممارسات الأفضل للهندسة العكسية تتضمن تفكيك السلسلة بقدر ما يمكن في مرحلة المواصفات. وتكون المواصفات مجردة وعملية بقدر الإمكان بواسطة مهندسي الهندسة العكسية ومن ثم يتم إرسالها إلى فرقة الغرفة

النظيفة أو الفريق الذي قام بتحليلها والذي سيقوم بعد ذلك بتصـــميم المنـــتج الجديـــد باستخدام أقل قدر من معلومات المستوى المنخفض من المنتج القديم.

أسئلة تقويم ذاتي

1/ هل تنجح الهندسة العكسية دائما؟.

2/ ما هي أهم مراحل الهندسة العكسية ؟.

8. أنظمة الحماية المختلفة ضد الهندسة العكسية

وفيما يلي مجموعة من أهم أنظمة الحماية ضد الهندسة العكسية:

أ) الأرقام المسلسلة ذات الرموز الصعبة.

ب) الحماية بالرقم المسلسل والاسم.

ت) الشاشة العنيدة.

ث) التجارب الوقتية.

ج) الدنقل (حماية الأجهزة).

ح) حماية البرمجيات (فحص الأسطوانة المدمجة، تعطيل الوظيفة إلخ).

(أ) الأرقام المسلسلة ذات الرموز الصعبة

هذا نوع بسيط من أنواع الحماية مقارنة بالأنواع الأخرى. في هذا النوع يتم فقط إدخال الرقم المسلسل وهو نفسه لكل المستخدمين. وعلى الرغم من أن لهذا النوع حماية عاليـــة إلا أن به عيوب أهمها بأنه بمجرد فك شفرة البرنامج فإن كل البرامج الأخـــرى تصـــبح معرضة للفك.

(ب) الحماية بالرقم المسلسل والاسم

في هذا النوع من الحماية يتم إدخال الاسم والرقم المسلسل. بعدها تـــتم مقارنـــة الـــرقم المسلسل مع الرقم المسلسل الأصلي الذي يستخدم نفس اللوغريتم.

(ت) الشاشة العنيدة

في هذا النوع من الحماية تظهر شاشة في كل مرة يبدأ فيها المستخدم تشغيل التطبيق لتذكيره بأن التطبيق يشاطره فيه آخرون. وتظهر رسالة في الشاشة توضح عدد الأيام المتبقية على تسجيل البرمجيات. يمكن إزالة الشاشة العنيدة بسهولة بواسطة المستخدم الذي يملك المعرفة الكافية بتطبيقات ويندوز.

(ث) التجارب الوقتية

i. إلى عدد محدد من الأيام مثلا 30 يوما تبدأ من اليوم الأول للتركيب.
ii. إلى فترة محددة من الوقت (تنتهي في يوم محدد مستقل عن يوم البدء).
ii. إلى عدد محدد من الدقائق و/أو الثواني كل مرة (في بعض ألعاب الكمبيوتر التي تسمح للاعب غير مسجل باللعب لبعض الوقت مثلا 5 دقائق).
iii. إلى عدد محدد من "المرات" مثلا 30 مرة.

(ج) الدنقل (حماية الأجهزة)

يفترض أن يكون هذا النوع هو الصعب في فكه وتتكون هذه الحماية من EPROM متصل بمدخل على جهاز الحاسب الآلي. ويقوم البرنامج المحمي بهذه الطريقة بالتأكد ما إذا كان الشخص مسجلا ومن ثم التأكد ما إذا البرنامج مسجلا. هذا النوع من الحماية ليس مستخدما على نطاق واسع.

(ح) حماية البرمجيات

هناك العديد من الأساليب المستخدمة لحماية البرمجيات والتي تستخدم عادة في ألعاب الحاسب الآلي. ومن أمثلتها الحماية بالقرص المدمج فإذا لم يكن المستخدم يملك أسطوانة محددة لإحدى الألعاب فإنه لن يستطيع اللعب لأنه عندما يقوم المستخدم بتشغيل البرنامج فإن البرنامج يبحث عن الأسطوانة المدمجة وبالتالي لا يستطيع المستخدم اللعب من القرص الصلب مباشرة.

أسئلة تقويم ذاتي

1/ قدم ملخص اشرح فيه أهم أنظمة الحماية ضد الهندسة العكسية؟.

9. أمثلة على استخدام الهندسة العكسية

مثال (1): تفكيك نظام التشغيل أبل (2)

يعتقد بعض الناس أن الهندسة العكسية صعبة للغاية بحيث لا يرغب شخص في القيام بها. وبالتالي فإنهم يقولون بأن منعها ليس موضوعا هاما. إن الهندسة العكسية تأخذ وقتا طويلا ومتعبة ولكنها تعطي الكثير من المعلومات. وكما يستفيد المحامون من دراسة المختصرات المكتوبة بواسطة المحامين الآخرين فإن العديد من المبرمجين يكتسبون مهارات جديدة عن طريق دراسة برامج الأشخاص الآخرين.

تتكون الهندسة العكسية لهذا المثال من ثلاث مراحل:

كتابة صفحة من جدول المتابعة الذي يوضح كيفية استخدام المواقع الخاصة في الذاكرة بواسطة النظام التشغيلي. وتوضح الصفحة التالية جزء مفكك من الذاكرة وما هي الأجزاء المخزنة في هذا الجزء. وعند فهم الوظيفة فهما جيدا تتم طباعة المذكرات.

توضح الصفحة الثالثة التحليل المكتمل تقريبا من الروتنات الفرعية لنظام التشغيل.

وهناك عدة طرق يمكن استخدامها مع الهندسة العكسية للبرمجيات ولكل منها فوائده ومتطلبات خاصة بالمصدر والوقت. ويستخدم أحد الأساليب الاعتيادية مجموعة من الطرق عند فك البرمجيات وفحصها. وأفضل طريقة للجمع بين الطرق المختلفة تعتمد كليا على أهداف من يقوم بالهندسة العكسية. وكمثال فقد يرغب في القيام أولا بعمل مسح سريع على الرمز للتأكد مما إذا كانت هناك بعض العناصر معرضة للخطر. بعدها قد يرغب المهندس القيام بتعقب تفصيلي لكل مسار حتى يمكنه استخدام نقاط التفكيك والأدوات الأخرى لتسريع العملية.

مثال (2): الثغرات الأمنية للأجهزة الالكترونية و Microcontroller

الثغرات الأمنية في الأجهزة الالكترونية تحدث كثيرا وأمثلة على ذلك ما حدث في فرنسا:- تفاصيل أكثر عن الموضوع في الموقع المعنون ب:

http://www.parodie.com/english/smartcard.htm

فقد تمكن الهاكرز من التغيير في برنامج البطاقة الذكية وتم تغيير بعض أكواد التحقق ليدخل صاحب البطاقة لرصيد البنك بالكامل، وتحدث الكثير من الاختراقات في موضوع البطاقات الذكية وسرقة بطاقات الهوية والتزييف بشكل عام.

مثال (3): التشويش على أنظمة الاتصالات

وتوجد أنواع أخرى من الثغرات للتشويش على الاتصالات اللاسلكية والرادارات وغيرها تكتشف بواسطة مجال الهندسة العكسية للأجهزة الالكترونية ،ومثال على ذلك في بعض مواقع الهاكرز تجد شخص يقول انه اكتشف ثغرة في نظام اتصالات معين ويقوم بإرفاق الاستغلال عبارة عن ملف صوت wav هذا الملف يمثل تردد الموجة اللاسلكية للتشويش على نظام معين أو حتى إغلاقة ونشاهد هذا النوع من الثغرات يحدث في أوربا وبواسطة موبايل او لاب توب في سيارة تستطيع اغلاق أو تشويش كل الرادارت التي في الطريق وبنفس المبدأ ولكن بموضوع متقدم أكثر تستخدم هذه الأمور في المجال العسكري للتشويش على رادارات الطائرات.

مثال (4) : استخدام الهندسة العكسية في كسر حماية برامج الجوال.

الخلاصة

في نهاية هذه الوحدة نعرض لك أهم الأفكار التي وردت فيها:

* عرفت الوحدة الهندسة الأمامية بأنها عملية تحويل مواصفات إلى منتج يقوم بالأداء لغرض معين. أما الهندسة العكسية فهي محاولة استدعاء مواصفات المستوى الأعلى عن طريق تحليل المنتج.

* وضحت الوحدة أن استخدام الهندسة العكسية أصبح أكثر سهولة في جميع الأوقات وذلك لسببين: أن الأساليب الهندسية نفسها أصبحت تعتمد على الحاسوب – الأساليب الاصطناعية لتمييز الأنماط وتفسيرها متقدمة للغاية بحيث أنها أصبحت تميز التركيبات ضمن المنتج تلقائيا.

* بينت الوحدة الفرق بين الهندسية العكسية وباقي فروع الهندسة في أنه تختلف الهندسة العكسية عن الفروع الأخرى من الهندسة في أنها تبدأ من المنتج النهائي وتعمل عكسيا لإعادة إنشاء المفاهيم الهندسية عن طريق تحليل تصميم النظام والعلاقات الداخلية التبادلية لمكوناته.

* قدمت الوحدة مميزات الهندسة العكسية: التوفر السريع للموديلات التي يتم تنفيذها بالتصميم بمساعدة الكمبيوتر (CAD)– يتم استخدام الموديل الطبيعي كنقطة بداية– يكون وقت تطوير المنتج أقصر– يكون المنتج تام التطوير عند بداية الإنتاج.

* قدمت الوحدة بعض من الاستخدامات للهندسة العكسية من ضمنها:

فهم كيفية عمل أحد المنتجات فهما شاملا أكثر من مجرد ملاحظته فقط.

فحص الأخطاء والقصور في البرامج الحالية وتصحيحها.

دراسة مبادئ التصميم لأحد المنتجات كجزء من عملية التعليم في علوم الهندسة.

جعل المنتجات والأنظمة تتوافق مع بعضها حتى تعمل معا أو تتشارك في المعلومات.

تقييم المنتج الخاص بك لفهم نقاط القصور فيه.

لتحديد ما إذا كانت جهة أخرى قد نسخت حرفيا عناصر تقنية أخرى.

* عرضت الوحدة بعض من أدوات الهندسة العكسية للبرمجيات: جهاز الهيكساديسمال المفكك.

* قسمت الوحدة منهجيات إعادة هندسة البيانات كالتالي:تنظيف البيانات Data Cleanup- امتداد البيانات Data Extension- هجرة البيانات Data migration.

* عرضت الوحدة المجموعات التي تستفيد من الهندسة العكسية: المستخدمين- المنافسون- القراصنة.

* عددت الوحدة مكونات الهندسة العكسية بصورة عامة في المراحل التالية:تحليل المنتج - إنشاء وصف المستوى المتوسط للمنتج - التحليل البشري لوصف المنتج لإنتاج المواصفات - إنشاء منتج جديد باستخدام المواصفات.

* قدمت الوحدة مجموعة من أنظمة الحماية المختلفة ضد الهندسة العكسية: الأرقام المسلسلة ذات الرموز الصعبة- الحماية بالرقم المسلسل والاسم- الشاشة العنيدة- التجارب الوقتية- الدنقل (حماية الأجهزة)- حماية البرمجيات (فحص الأسطوانة المدمجة، تعطيل الوظيفة إلخ).

لمحة مسبقة عن الوحدة التالية

الوحدة التالية تأتي بعنوان "هندسة البرمجيات المشاكل والاحتمالات" الوحدة توضح تصنيف البرمجيات، كذلك تتناول الوحدة تكلفة إنتاج البرامج حيث تستعرض أمثلة عن التكاليف .

مسرد المصطلحات

الهندسة الأمامية

هي عملية تحويل مواصفات إلى منتج يقوم بالأداء لغرض معين. تعطـــي المواصـــفات الكاملة وصفا لما يقوم به المنتج ويعطي وصفا لمعايير الأداء التي يتعين عليـــه الوفـــاء بها.

الهندسة العكسية

هي محاولة استدعاء مواصفات المستوى الأعلى عن طريق تحليل المنتج.

جهاز الهيكساديسمال

يقوم هذا الجهاز بطباعة أو عرض الأعداد الثنائية للبرنامج في شكل سداسي الأرقام مما يسهل قراءته أكثر من الشكل الثنائي.

المفكك

وهي برامج تحول رمز الموضوع مرة إلى لغات المستوى العالي مثل لغة C.

التشغيل المشترك

هي قدرة البرامج والأنظمة على العمل في الأنواع المختلفة من البرمجيات والمعدات للاتصال مع بعضها البعض.

تنظيف البيانات Data Cleanup

فيها يستبعد التكرار وتحذف المعلومات الزائدة.

امتداد البيانات Data Extension

فيها تعاد هندسة البيانات لاستبعاد القيود على معالجتها وقد يتطلب ذلك تغيـــرات فـــي أطوال الحقول أو تغيير حدود جداول.

هجرة البيانات Data migration

فيها تنتقل البيانات إلى تحكم إدارة قواعد بيانات حديثة وقد تخزن البيانات فـــي ملفـــات منفصلة أو قد تدار بواسطة قاعدة بيانات أخرى.

التحليل الوظيفي

هو مراقبة حالات الدخل والخرج للرقاقة علـــى جهـــاز "الأوسيلســـكوب" أو المجســـات المستخدمة للأغراض الخاصة للحصول على صورة من تصرف الرقاقة مع الوقـــت أو استجابة إلى إشارات الدخل.

المصطلح بالإنجليزية	معناه بالعربية
Forward Engineering	الهندسة الأمامية
Reverse Engineering	الهندسة العكسية
Electronic Chips	الرقائق الإليكترونية
Printed Circuits	الدوائر المطبوعة
Historical Background	الخلفية التاريخية
General Concepts	المفاهيم العامة
Reverse Engineering Success	نجاح الهندسة العكسية
Dongle	مكون صلب للحماية
Approaches to data re-engineering	منهجيات إعادة هندسة البيانات
Data Cleanup	تنظيف البيانات
Data Extension	امتداد البيانات
Data migration	هجرة البيانات
Data naming problems	مشاكل تسمية البيانات
Field length problems	مشاكل طول الحقل
Record organization problems literals	مشاكل تنظيم السجلات
No data Dictionary	عدم وجود قاموس بيانات
Hard coded literals	مصطلحات عالية التشفير

المراجع

أولاً : المراجع العربية :

[1] روجر بريسمان، "هندسة البرمجيات"، الدار العربيـــة للعلـــوم، مركـــز التعريـــب والبرمجة، الطبعة الأولى، 1425هـــ – 2004م.

[2] مهندس عبد الحميد بسيوني، "أساسيات هندسة البرمجيات"، دار الكتب العلمية للنشر والتوزيع، القاهرة، 2005 م.

ثانياً : المراجع الإنجليزية :

[3] Ian Somerville , "Software Engineering", Addison Wesley, 2001.

[4] "Software Re-engineering"
Prepared By:
Dr. Linda H. Rosenberg
Engineering Section head
Software Assurance Technology Center
Unisys Federal Systems

صفحات الانترنت

http://en.wikipedia.org/wiki/Reengineering
http://www.chillingeffects.org/reverse/faq.cgi
/http://www.jaascois.com/research/36601029
/http://www.jaascois.com/research/36601004
http://www.c4arab.com/showthread.php?threadid=35007


الوحدة السادسة
هندسة البرمجيات
المشاكل والاحتمالات

محتويات الوحدة

المقدمة

تمهيد

عزيزي الدارس،،

مرحباً بك عزيزي الدارس في الوحدة الرابعة من مقرر "**هندسة البرمجيات 2**" والتي تبحث في "**هندسة البرمجيات المشاكل والاحتمالات**" القسم الأول من الوحدة يوضح تصنيف البرمجيات حيث صنفت الوحدة البرمجيات إلى نوعان رئيسان : **برامج النظام** – **البرامج التطبيقية** ، القسم الثاني من الوحدة يتناول تكلفة إنتاج البرامج وفيه نستعرض أمثلة عن التكاليف : إنتاجية المبرمج ، تنبؤات تكاليف البرنامج ، المعدات مقابل تكاليف البرامج ، أثر الحاسبات الشخصية ، حزم البرامج وسائل تطوير التطبيق ، ثورة تقنية المعلومات ، كيف تكون التكلفة ، إعادة استخدام البرمجيات .

أهداف الوحدة

عزيزي الدارس، بعد فراغك من دراسة هذه الوحدة ينبغي أن تكون قادراً على أن:

- **تذكر** التقسيمات الرئيسية لأنماط البرمجيات.
- **تلخص** التحديات الرئيسية المتعلقة بعملية تطوير البرمجيات والتي تشمل:
  - تلبية حاجات المستخدم .
  - تكلفة إنتاج البرامج .
  - أمثلة على التكاليف .
  - الالتزام بآخر موعد .
  - أداء البرنامج .
  - قابلية التنقل .
  - الصيانة .
  - الموثوقية .
  - تفاعل الحاسوب مع البشر .
- **تناقش** أزمة السوفت وير ؟
- **تلخص** الأفكار الرئيسية عن كيفية علاج ــ أزمة هندسة السوفت وير ؟

1. تصنيف البرمجيات

هناك أدوات وتقنيات تستعمل كثيرة تستخدم لتطوير البرمجيات. وهـذه الوحـدة تهتمّ بالأسباب التي أدت إلى تعدد حقول دراسة هندسة البرمجيات، وبالمشاكل التي قـد تصادفنا أثناء عملية تطوير البرمجيات.

البرمجيات تحيطنا في كل مكان: في المناطق الصناعية- الاستعمالات المحلية- أنظمة الاتصالات، أنظمة النقل وفي الأعمال. فـالبرامج تـأتي مختلفـة فـي أشـكالها وأحجامها- من البرامج التي في الهواتف المحمولة إلى البرامج التـي تقـوم بتصـميم السيارة الجديدة- أما من ناحية تصنيف البرمجيات، **فإننا نستطيع أن نميّـز** أن **هنـاك نوعان رئيسيان:**

❶ **برامج النظام:** وهي مجموعة البرامج التي تمثل الأدوات التي تساعد فـي بنـاء أو دعم البرامج التطبيقية. أمثلة لذلك: أنظمة التشغيل، قواعد البيانات، بـرامج التشـبيك ، والمترجمات.

❷ **البرامج التطبيقية :** وهي مجموعة البرامج التي تساعد في تأديـة بعـض المهـام المفيدة أو الممتعة بشكل مباشر. أمثلة لذلك: الألعاب، برامج آله الصرافة الاوتوماتيكية (ATMs)، برامج التحكم في توجيه الطائرات، بـرامج البريـد الإلكترونـي، بـرامج معالجات النصوص، وبرامج الجدولة.

ويقع تحت تصنيف البرامج التطبيقية، مجموعة البرامج **التّالية:**

- برامج الألعاب (Games)
- برامج نظم قواعد المعلومات – النظم التي تخزن وتوفر لك كميات ضـخمة من البيانات، على سبيل المثال: نظام حجز مقعد بالخطوط الجوية.
- برامج نظم الوقت الحقيقي (Real Time Systems) – والتي يكون فيهـا استجابة الحاسب الآلي سريعة جدا، مثال ذلك: برنامج التحكم في مراكز الطاقة النووية، برامج توجيه الطائرات.

- النظم المطمورة (المطوقة بإحكام) (Embedded Systems)– والتي يؤدي فيها الحاسب الآلي دورا صغيرا ضمن نظام أوسع، مثال ذلك: برنامج في سنترال تلفوني أو هاتف جوال، والأنظمة المطمورة هي كذلك يمكن تصنيفها على أنها برامج نظم الوقت الحقيقي.
- البرنامج المكتبي (Office Software)– ومن أمثلتها برامج معالجة الكلمات وتنسيق النصوص، البرامج الإحصائية المساعدة، برامج بريد إلكتروني.
- البرنامج العلمي (Scientific Software) – وهي مجموعة البرامج التي تساهم في إجراء العمليات الحسابية، النمذجة (Modeling)، التنبؤ (Prediction)، مثال ذلك: تنبؤات الأحوال الجوية.

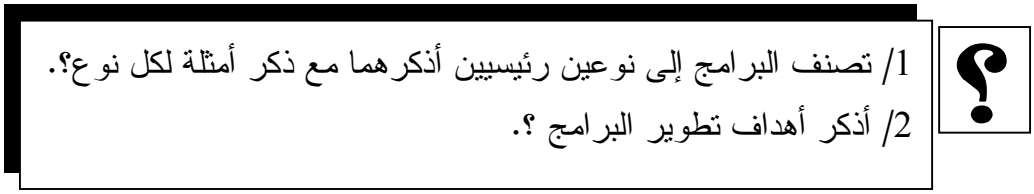
2.التحديات الرئيسية المتعلقة بعملية تطوير البرمجيات

وإنشاء برمجية يمثل مسألة تحدي ومملوء بالتعقيد ويمكن تلخيص المشاكل المتصورة في تطوير البرامج والأهداف التي يهدف إليها تطوير البرامج هي:

- تلبية حاجات المستخدمين .
- قلة تكلفة الإنتاج .
- الأداء العالي .
- إمكانية نقله وتوصيله .
- قلة تكلفة الصيانة .
- الموثوقية العالية .
- التسليم في الوقت المحدد .

وسنبحث الآن في كل واحدة من هذه الأهداف تباعا: وكل واحد من هذه الأهداف يمثل مشكلة، حيث أن الوصول لهندسة البرامج عمل صعب نوعا ما، وبعد ذلك سنبحث في كيفية ارتباط هذه الأهداف بعضها بالآخر.

أسئلة تقويم ذاتي



1/ تصنف البرامج إلى نوعين رئيسيين أذكرهما مع ذكر أمثلة لكل نوع؟.
2/ أذكر أهداف تطوير البرامج ؟.

1.2 تلبية حاجات المستخدم Meeting users' needs

من البديهي أن نلاحظ أن الغرض الأساسي من أي برنامج أن ينفذ ما يريده مستخدموه. هكذا فمن المنطقي أن تكون الخطوة الأولي في تطوير البرامج هي اكتشاف حاجات العميل أو الزبون أو المستخدمين. في الغالب هذه الخطوة تسمي بتحليل المتطلبات أو هندسة المتطلبات.

ويمكن القول من خلال الإحصائيات والرسومات البيانية المنشورة في العديد من الدراسات بأن نسبة كبيرة من النظم (أكثر من 50% منها) لا تلبي حاجات مستخدميها وبالتالي لا تستخدم بالصورة المطلوبة. وكذلك يمكننا الذهاب بعيدا والاستدلال على أن المشكلة الرئيسية لتطوير البرامج تقع في نطاق تحليل المتطلبات، أكثر منه في أي مجالات أخرى. وعملية ضمان تنفيذ البرنامج لما يريده مستخدموه يعرف بالثبوتية (Validation).

2.2 تكلفة إنتاج البرامج

The cost of software production

1.2.2 أمثلة على التكاليف

أولا دعونا نأخذ فكرة عن مقياس تكاليف البرامج في العالم وبالذات في الدول المتقدمة. في الولايات المتحدة تقدر بما يقارب 500 مبليون دولار تصرف نقدا كل عام في إنتاج البرامج. هذا المبلغ يمثل 1% من إجمالي الناتج القومي. الرقم التقديري للعالم

هو 1,000 بليون دولار تصرف كل عام في إنتاج البرامج. هذه الأرقام ارتفعت ومن المتوقع ارتفاعها بما يقارب 15% كل عام. نظام التشغيل الذي طورته (آي بي إم) لواحدة من السلسلة الرئيسية لحساباتها (ويسمى أو س 360) تقدر تكلفته بـ 200 مليون دولار. في الولايات المتحدة، تكاليف البرامج الخاصة ببرنامج الفضاء الذي طلع به الإنسان على القمر بلغت 1 بليون دولار بين عامي 1960 م و 1970م.
هذه الأمثلة تشير إلى أن المبالغ التي تنفقها الدول الصناعية على البرمجيات عالية جدا. وتشير الإحصائيات أن تكلفة إنتاج البرمجيات للتطبيقات المختلفة في الفترة الأخيرة أكبر بكثير مما يصرف على العتاد.

2.2.2 إنتاجية المبرمج Programmer productivity

تكلفة البرامج تحدد بدرجة كبيرة بالرواتب التي تدفع للمبرمجين وإنتاجية المبرمجين. ربما كان من المدهش أن إنتاجية المبرمج متوسط المستوى هي 10-20 جمله في اليوم. وهذا الأداء الضعيف الواضح لا يعكس فقط الوقت الذي استغرق للقيام بالبرمجة ولكن يشمل أيضا الوقت المطلوب لتوضيح مواصفات المشكلة وتصميم البرنامج والتشفير والاختبار والتوثيق. ومن الملفت للنظر أن هذا الرقم في الإنتاجية لا يعتمد على اللغة المستخدمة في البرمجة. الأمر سيان سواء استخدمت لغة جيل ثالث كـ (سي) أو لغة جيل رابع (الفيجوال بيسك). ومع ذلك، فإن هذا الرقم الهزيل يعتمد على كثرة الاختلافات بين المبرمجين وكذلك على نمط البرامج التي تكتب فكتابة برامج التطبيقات أسهل بكثير من كتابة برامج نظم التشغيل. ومعدل كتابة 20 جملة في اليوم مشينة – ولكن هذه نتيجة طبيعية للكسل ورداءة الأدوات أو الأساليب المستخدمة الغير ملائمة؟.

3.2.2 تنبؤات تكاليف البرنامج Predicting software costs

من الصعوبة بمكان على مديري تطويرالبرمجيات التنبؤ بالوقت الذي تستغرقه كتابة البرنامج، ومن ثم التكلفة وتاريخ التسليم.

4.2.2 المعدات مقابل تكاليف البرامج

Hardware versus software costs

التكاليف النسبية للمعدات والبرامج قد تكون مثار جدل كبير. في أزمنة الحاسب الآلي التي ولت كانت المعدات مكلفة جدا والبرامج رخيصة نسبيا، وفي الوقت الحاضر أصبحت المعدات رخيصة جدا والبرامج (الجهد العقلي) مكلفة. وتشير الإحصائيات أنه من عام 1970–1990 كانت تكلفة العتاد عالية جدا وتكلفة البرمجيات قليلة في حين من عام 1990–2010 أصبحت تكلفة العتاد صغيرة مقارنة بتكلفة البرمجيات.

5.2.2 أثر الحاسبات الشخصية

(The Impact of Personal Computer)

ربما أتى التأثير الأكبر على المفاهيم الشائعة لتكاليف البرامج منذ ظهور الحاسب الآلي الصغير والحاسب الآلي الشخصي. الكثير من الناس يشترون الحاسب الآلي الشخصي لبيوتهم ثم يتحققون من التكاليف.

أولا: في حالة شرائك لحاسب آلي شخصي بـ 1,000 دولار لا تتوقع دفع 10,000 دولار للبرامج وهو تكلفة استئجار مبرمج للقيام بكتابته لك. لذا، بالطبع رزمة برنامج لحاسب آلي شخصي سعره 50 دولار تقريبا أو نحو ذلك. من الصعوبة بمكان فهم ذلك الشيء الذي دفعت من أجله 50 دولار يكلف ملايين الدولارات لتطويره.

ثانيا: الكثير من طلاب المدارس يألفون البرامج إما كجزء من دراساتهم أو كهواية. قد يفكر والد بسهولة في استطاعة طفلي تأليف لعبة حاسب آلي في أسبوعين. ما هي الصعوبة في البرمجة؟ لماذا يكون البرنامج مكلفا؟ "

6.2.2 حزم البرامج (Software packages)

هناك رد فعل هام آخر لمدى إتاحة الحاسبات الرخيصة. إذا كنت ترغب في استخدام الحاسب الآلي في المحاسبة أو تخطيط الإنتاج في إمكانك شراء برنامج للقيام بذلك. مثل هذه الحزم من البرامج قد تكون رخيصة الثمن. حتى في إمكان مجمعي الحاسبات الآلية الشخصية شراؤها بسعر أرخص. السبب في السعر المتدني الملفت للنظر هو بالطبع أن منتجي البرامج يبيعون الكثير من النسخ المماثلة – والإنتاج الضخم من الحزم في هذه الأيام قد وصل إلى حد كبير. المشكلة مع الرزم الموضوعة على الرف ولا تستخدم بالطبع هي أنها قد لا تؤدي ما تريده بدقة وربما تلجأ لبرنامج أعد لغرض معين، وبالرغم من ذلك: فإن الحزم الرخيصة تعطي انطباعا في أن إنتاج البرمجيات شيء رخيص.

7.2.2 وسائل تطوير التطبيقApplication development tools

إذا كنت ترغب في كتابة بعض البرامج التطبيقية بسرعة وسهولة، فأن هناك الكثير من وسائل ولغات التطوير المتاحة. ومن أمثلة هذه الوسائل المتاحة لغة الفيجوال بيسك وبرنامج ميكرويوفت أكسس (لإعداد قواعد بيانات). هذه الوسائل تمكن من إنشاء أنماط معينة بسهولة تامة، حتى أولئك الذين هم من غير المبرمجين في إمكانهم تعلم استخدام البرامج الإحصائية (مايكروسفت إكسل). هكذا تم إيجاد مفهوم مفاده أن البرمجة سهلة، وقد يتم الاستغناء عن المبرمجين. والحقيقة التي لا جدال فيها أن كتابة بعض البرامج للاستخدامات الشخصية بسيط وسهل للغاية، ولكن إعداد معظم البرامج أو الحزم المستخدمة تجاريا في غاية التعقيد.

8.2.2 ثورة تقنية المعلومات The IT revolution

تعقيد برامج اليوم يختلف كثيرا على تلك التي كانت في الماضي حيث أصبحت واجهات الاستخدام أساسية ولها أهمية كبيرة وكذلك تطور أنظمة شبكات الحاسب الآلي بالنسبة للعتاد وكذلك وجود البرمجيات الموزعة وباختصار وما تشير إليه الإحصائيات الذي نراه اليوم هو أن البرامج مكلفة بالنسبة لإجمالي الناتج القومي عموما.

9.2.2 كيف تكون التكلفة؟ How is the cost made up?

من المهم جدا معرفة أي مرحلة من مراحل تطوير برنامج تكون مكلفة بالنسبة للأخرى وتشير معظم الإحصائيات التي تمت على العديد من البرمجيات أن:

- مرحلة التحليل والتصميم تشغل ثلث التكاليف.
- مرحلة التكويد سدس التكاليف.
- مرحلة الاختبار نصف التكاليف.

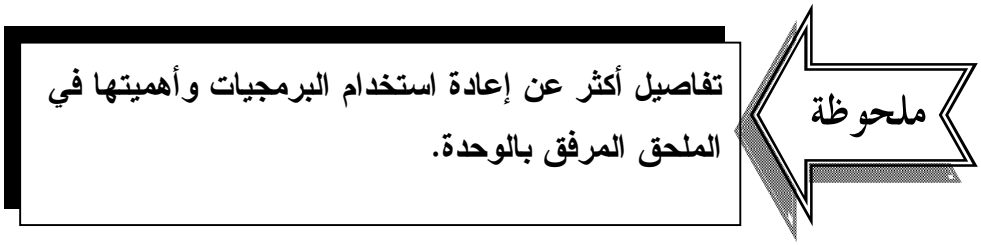
وكذلك تشير الإحصائيات أن:

- نصف عدد الأخطاء تكون في مرحلة التصميم.
- ثلث عدد الأخطاء تكون في مرحلة البرمجة.
- سدس عدد الأخطاء تكون في مرحلة التحليل النحوي.

10.2.2 إعادة استخدام البرمجيات Software re-use

إحدى استجابات التكلفة العالية للبرامج يتمثل في اقتراح بإعادة استخدام البرمجيات الجاهزة أو أجزاء منها ، لكن كما سنرى لاحقا، من الصعوبة بمكان تكييف برنامج ليتناسب مع البرنامج المطلوب. وهناك أسلوب آخر يتمثل في إنشاء عناصر البرنامج كالعناصر الإلكترونية التي يمكن نسخها وإعادة استخدامها من محتويات الرف وربطها معا لتنفيذ العمل المطلوب. هكذا، في حين أن بعض مظاهر التقدم في لغات

البرمجة وتقنيات التصميم تحاول تعقب الطريق، فالمسألة بعيدة عن كونها تقنية ناضـــجة مدروسة.



تفاصيل أكثر عن إعادة استخدام البرمجيات وأهميتها في الملحق المرفق بالوحدة.

أسئلة تقويم ذاتي

1/ ما هي الخطوة الأولى في تطوير البرامج ؟.

2/ ما المقصود بمصطلح الثبوتية (Validation) ؟.

3/ ما هو تقدير إنتاجية المبرمج متوسط المستوى ؟.

4/ أكمل ما يلي :

تشير معظم الإحصائيات التي تمت على العديد من البرمجيات أن:

- مرحلة التحليل والتصميم تشغل ثلث التكاليف.
- مرحلة التكويد .....التكاليف.
- مرحلة الاختبار .....التكاليف.

وكذلك تشير الإحصائيات أن:

- نصف عدد الأخطاء تكون في مرحلة.....؟.
- ثلث عدد الأخطاء تكون في مرحلة .....؟.
- سدس عدد الأخطاء تكون في مرحلة .....؟.

3.2 الالتزام بآخر موعد Meeting deadlines

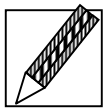
الالتزام بآخر موعد لتسليم البرمجية يمثل صداع دائم في إنتاج البرمجيات، ولقد بينت النتائج أن هذه أسوء مشكلة يواجهها مديرو تطوير البرمجيات. والمشكلة تعود لصعوبة التنبؤ بالوقت الذي يستغرقه تطوير شيء ما لارتباط ذلك بالعديد من المتغيرات المتعلقة بفريق العمل والميزانية المتاحة.

4.2 أداء البرنامج Software performance

أداء البرنامج يدعى في بعض الأحيان بالكفاءة. ويعود هذا المصطلح عندما كانت تكلفة وسرعة البرامج تعني أن كل جهد بذل لاستخدام العتاد – بصفة رئيسية الذاكرة والمعالج. ومنذ عهد قريب جدا حدث تحول ثقافي نتيجة للسرعة المتزايدة والتدني الذي حدث في تكلفة الحاسبات الآلية. في الوقت الحاضر هناك تأكيد أكثر لتلبية المتطلبات ولا يعني هذا تجاهل الأداء – **في الغالب نحن معنيون لتأكيد أن :**

- أن النظام متفاعل ويستجيب خلال وقت قصير.
- لو كانت لعبة مبرمجة تعمل بكفاءة وحركتها تبدو منتظمة.
- العمل على دفعات (Batches) لا يستغرق اثني عشرة ساعة في وقت كانت تكفيه ساعة واحدة.

تدريب (1)



حدد عدد 2 من أنظمة البرمجيات تكون فيهما السرعة عاملا هاما.

5.2 قابلية التنقل Portability

قابلية التنقل تعني نقل البرنامج من طراز حاسب آلي لآخر بأقل جهد وبدون مشاكل. وبظهور التقنيات الحديثة كذاكرات الفلاش والإسطوانات الضوئية الأقراص الصلبة المحمولة أصبحت هذه الخاصية متحققة للغاية.

6.2 الصيانة Maintenance

الصيانة هي مصطلح لأي جهد يبذل ويوجه إلى قطعة من السوفت وير بعد أن تكون قد كتبت ووضعت قيد التشغيل ، وهناك نوعين رئيسيين :

❶ الصيانة العلاجية Remedial maintenance :

وهي الوقت الذي يتم قضاؤه في تصحيح الأخطاء والعيوب في البرنامج.

❷ الصيانة التكيفية Adaptive maintenance :

هي تغيير السوفت وير إما بسبب حاجات المستخدمين قد تغيرت ، أو لأن الحاسوب أو نظام التشغيل أو لغة البرمجة قد تغيرت على سبيل المثال.

والصيانة العلاجية بالطبع هي نتيجة عدم الاختبار الكافي للبرمجية ، ولكن وكما تم شرحة سابقا أن الاختبار الفعال صعب ويستهلك الوقت وأنه حقيقة حتمية ومقبولة في دورة حياة البرمجية. وأنه من الصعبالتنبؤ بالاستخدامات المستقبلية للبرمجيه.

وكذلك الصيانة التكيفية نادراً ما يتم تجنبها أيضا لأن السوفت وير يعتقد أحيانا أنه يمكن تعديله بسهولة. والصيانة تعتبر عادةً شيء مزعج ، من قبل كل من المديرين والمبرمجين الذين يعتبرونها أقل إمتاعا من كتابة برامج جديدة.

وكمثال رئيسي عن مشاكل الصيانة هو " العلة الألفية " حيث كم كبير من السوفت وير تمت كتابته عندما كانت الذاكرة للحاسوب قليلة وباهظة الثمن. ولذلك تم تخزين التواريخ بشكل مقتصد باستخدام فقط الرقمين الأخيرين للعام.

7.2 الموثوقية Reliability

يقال أن قطعة السوفت وير موثوقة إن عملت واستمرت تعمل بدون تعطل وبدون عمل شيء ما غير مرغوب فيه ، ونفترض أن المطور يعلم ما هو المطلوب ولذلك السلوك الغير متوقع ليس مقصوداً ، ومن الشائع أن نتحدث عن العلل في السوفت وير . ولكن من المفيد تعريف بعض المصطلحات الإضافية بوضوح أكثر:

- **الخطأ** (Error): قرار خاطىء يتم اتخاذه خلال تطوير السوفت وير.
- **العيب** (الخلل) (Fault): مشكلة يمكن أن تؤدي بالسوفت وير أن يبتعد عن سلوكه المقصود.
- **الفشل** (Failure) : هو حدث يتم عندما يبتعد السوفت وير عن سلوكه المقصود.

بهذه المصطلحات العيب (الخلل) يكون هو ذاته العلة والخطأ، هو خطأ يتم من قبل كائن بشري خلال إحدى مراحل تطوير السوفت وير ، والخطأ يسبب خللاً أو أكثر ضمن السوفت وير ، أو مواصفاته أو توثيقاته، وبدوره يمكن أن تسبب الخلل الفشل أو أكثر. وهذا الفشل سيظهر بينما يكون النظام قيد الاختبار وبعد أن وضع في الاستخدام وحالات الفشل هي عرض يعاني منه المستخدمون ، بينما حالات الخلل هي مشكلة يجب على المطور أن يتعامل معها. والخلل يمكن أن لا تظهر نفسه لأن الحالات التي تسببه لتجعل السوفت وير يفشل لا تظهر أبداً ، وعلى العكس خلل واحد يمكن أن يسبب عدة حالات فشل مختلفة ومتعددة .

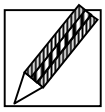
وعمل إزالة العلل ومحاولة ضمان الموثوقية يدعى " التحقق " وهناك علاقة وثيقة ولكن مميزة بين مفهوم الموثوقية ومفهوم تلبية حاجات المستخدمين المذكورة أعلاه. وتحليل المتطلبات يهتم بتحديد الوضوح ماذا يريده المستخدم. "والصلاحية " هي مجموعة من الأساليب التي تحاول أن تضمن أن السوفت وير يلبي المتطلبات ومن ناحية أخرى الموثوقية لها علاقة مع القضية الفنية فيما إذا هناك أية حالات خلل في السوفت وير. وأحد الأساليب لمحاولة ضمان أن البرمجية تعمل بشكل صحيح هو الاختبار ، في

الاختبار أنت تشغل السوفت وير وتتأكد من سلوكه مقابل ما تتوقعه ليحدث، ولكن هنـــاك مشكلة رئيسية بالاختبار بقدر ما تختبر البرمجية ، فلا تستطيع أبداً أن تتأكد بأنـــك قـــد وجدت كل علة أخيرة ، وهذا يقودنا للجزم أن كل برمجية تحتوي على أخطاء.

وبمقارنة أداء السوفت وير مع الهارد وير نجد أن الهارد وير يفشل ثلاثة أضعاف فشل السوفت وير . إن السوفت وير بالنتيجة له سمعة سيئة بالموثوقيـــة ، وهنـــاك تطبيقـــات خاصة للحاسوب تتطلب موثوقية عالية وهذه تعرف بالأنظمة الدقيقة للسلامة " .

وسيتم النظر في الأساليب التي يمكن استخدامها في تطوير الأنظمة كهذه.

تدريب (2)



حدد ثلاثة أمثلة أخرى حول أنظمة البرمجيات تكون حساسة للسلامة .

8.2 تفاعـــل الحاســـوب مـــع البشـــر Human-Computer Interaction

واجهة الاستخدام هي ما يراها المستخدم عند استخدامه للبرمجية وهناك أمثلـــة عديدة من أنظمة الحاسوب ليست سهلة الاستخدام مثل أنظمة التشـــغيل DOS. وهنـــاك بعض الناس يجد من الصعب تحويل خط الهاتف إلى مستخدم آخر ضمن منظمـــة مـــا. وفي الأيام الأخيرة ظهرت عدة واجهات مستخدم رسومية (GUIS) تستخدم " ويندوز" النوافذ بمزايا مثل الأزرار وشريط القلب والتغيير مع أجهزة تأشير مثل " الماوس" الفأرة ورأى عديد من الناس هذه كخطوة كبيرة في تحسين واجهة المستخدم.

❑ أزمة السوفت وير ؟ A software crisis?

ناقشنا عدة مشاكل متصورة بالسوفت وير ومن أهمها :

- إنه يفشل في عمل ما يريد المستخدمون عمله.
- إنه باهظ الثمن.

- ليس دائماً سريعاً كفاية.
- لا يمكن نقله إلى معدات أخرى بسهولة.
- الصيانة مكلفة.
- إنه غير موثوق به.
- إنه متأخر غالباً.
- إنه غير سهل الاستخدام دائماً.

هناك " أزمة " حقيقية مستمرة في تطوير السوفت وير والعلاج الرئيسي كما يقول البعض بجلب طرق علمية لتحل المشكلة ــ ومن هنا ظهرمصطلح هندسة السوفت وير.

إن أحدى العقبات للمحاولة في حل مشكلات السوفت وير هي أنها غالباً تتعارض مع بعضها على سبيل المثال البناء القليل التكلفة يتعارض مع الموثوقية العالية . والأداء العالي يتعارض مع إمكانية الحمل، وبعض الأهداف لا تتعارض مع بعضها البعض لحسن الحظ . مثلاً الصيانة القليلة التكاليف والموثوقية العالية متكاملتان.

❑ علاج ــ هندسة السوفت وير ؟

A remedy- software engineering?

بعض الأفكار الرئيسية هي:

- تأكيد كبير لتنفيذ كل مراحل التطوير بشكل منهجي.
- مساعدة الحاسوب لتطوير السوفت وير ــ أدوات السوفت وير وبيئات تطوير السوفت وير " وهندسة السوفت وير بمساعدة (CASE).
- التأكيد على إيجاد بالضبط ما يحتاجه مستخدمو النظام في الواقع (هندسة وصلاحية المتطلبات).
- مواصفات رسمية لمتطلبات النظام.
- إظهار نسخة أولى من النظام لعملائه (النموذج الأصلي) .
- استخدام لغات برمجة حديثة.
- تأكيد كبير على محاولة ضمان أن السوفت وير خالي من الأخطاء (التحقق).

والأفكار المعروضة غالباً ما تكمل بعضها البعض ، حيث إن المواصفات الرسمية والتحقق والنسخة الأصلية وأساليب أخرى مماثلة تخاطب في الواقع فقط بعض المشاكل التي تتم مواجهتها في تطوير السوفت وير. ولكي يتم عمل البرمجية في الوقت الصحيح وضمن الميزانية ، فإنها يجب أن تخطط لها سلفاً وتدار بفاعلية أثناء تنفيذها . ولهذا الهدف يجب أن يتم استبدال طرائق " أدهوك" (adhoc) بواسطة طريقة منهجية مرتبة.

وأحد المصطلحات المتعلقة بالسوفت وير هو كلمة "الجودة Quality" وأي منتج يؤدي الغرض من إنتاجه يعتبر منتج ذات جودة (مثل الغسالة الكهربائية) . وإذا استمر السوفت وير في تلبية آمال وتوقعات العميل يمكن اعتباره أيضا منتج ذات جودة وبهذا المنظار يمكن تحقيق الجودة فقط إذا ما وجدت معايير فعالة وأساليب وإجراءات لتطبق وترى لكي تستخدم وتراقب بشكل سليم ، لذلك ليس على الطرائق الجيدة أن تطبق فقط ولكن يجب أن تري ليتم تطبيقها. والاستخدام الواسع لنظام إدارة المشروع هو عامل رئيسي في تحقيق سوفت وير عالي الجودة.

خلاصة

*صنفت الوحدة البرمجيات إلى برامج النظام: وهي مجموعة البرامج التي تمثل الأدوات التي تساعد في بناء أو دعم البرامج التطبيقية. أمثلة لذلك: أنظمة التشغيل، قواعد البيانات، برامج التشبيك ، والمترجمات.

البرامج التطبيقية : وهي مجموعة البرامج التي تساعد في تأدية بعض المهام المفيدة أو الممتعة بشكل مباشر. أمثلة لذلك: الألعاب، برامج آله الصرافة الاوتوماتيكية (ATMs)، برامج التحكم في توجيه الطائرات، برامج البريد الإلكتروني، برامج معالجات النصوص، وبرامج الجدولة.

*لخصت الوحدة المشاكل المتصورة في تطوير البرامج والأهداف التي يهدف إليها تطوير البرامج إلى: تلبية حاجات المستخدمين – قلة تكلفة الإنتاج – الأداء العالي – إمكانية نقله وتوصيله – قلة تكلفة الصيانة – الموثوقية العالية – التسليم في الوقت المحدد .

*صنفت الوحدة الصيانة إلى نوعين رئيسيين :–الصيانة العلاجية:وهي الوقت الذي يتم قضاؤه في تصحيح الأخطاء والعيوب في البرنامج – الصيانة التكيفية : هي تغيير السوفت وير إما بسبب حاجات المستخدمين قد تغيرت ، أو لأن الحاسوب أو نظام التشغيل أو لغة البرمجة قد تغيرت على سبيل المثال.

*ناقشت الوحدة عدة مشاكل متصورة بالسوفت وير ومن أهمها : إنه يفشل في عمل ما يريد المستخدمون عمله – إنه باهظ الثمن – ليس دائماً سريعاً كفاية – لا يمكن نقله إلى معدات أخرى بسهولة – الصيانة مكلفة – إنه غير موثوق به – إنه غير سهل الاستخدام دائماً – إنه متأخر غالباً.

مسرد المصطلحات

برامج النظام

وهي مجموعة البرامج التي تمثل الأدوات التي تساعد في بناء أو دعم البرامج التطبيقية. أمثلة لذلك: أنظمة التشغيل، قواعد البيانات، برامج التشبيك ، والمترجمات.

البرامج التطبيقية

وهي مجموعة البرامج التي تساعد في تأدية بعض المهام المفيدة أو الممتعة بشكل مباشر. أمثلة لذلك: الألعاب، برامج آله الصرافة الاوتوماتيكية (ATMs).

برامج نظم الوقت الحقيقي (Real Time Systems)

وهي التي يكون فيها استجابة الحاسب الآلي سريعة جدا، مثال ذلك: برنامج التحكم في مراكز الطاقة النووية، برامج توجيه الطائرات.

النظم المطمورة (المطوقة بإحكام) (Embedded Systems)

وهي التي يؤدي فيها الحاسب الآلي دورا صغيرا ضمن نظام أوسع، مثال ذلك: برنامج في سنترال تلفوني أو هاتف جوال، والأنظمة المطمورة هي كذلك يمكن تصنيفها على أنها برامج نظم الوقت الحقيقي.

الثبوتية (Validation).

هي عملية ضمان تنفيذ البرنامج لما يريده مستخدموه .

قابلية التنقل Portability

قابلية التنقل تعني نقل البرنامج من طراز حاسب آلي لآخر بأقل جهد وبدون مشاكل.

الصيانة Maintenance

الصيانة هي مصطلح لأي جهد يبذل ويوجه إلى قطعة من السوفت وير بعد أن تكون قد كتبت ووضعت قيد التشغيل .

الصيانة العلاجية Remedial maintenance

وهي الوقت الذي يتم قضاؤه في تصحيح الأخطاء والعيوب في البرنامج.

الصيانة التكيفية Adaptive maintenance

هي تغيير السوفت وير إما بسبب حاجات المستخدمين قد تغيرت ، أو لأن الحاسوب أو نظام التشغيل أو لغة البرمجة قد تغيرت .

الخطأ (Error).

قرار خاطىء يتم اتخاذه خلال تطوير السوفت وير.

العيب (الخلل) (Fault)

مشكلة يمكن أن تؤدي بالسوفت وير أن يبتعد عن سلوكه المقصود.

الفشل (Failure)

هو حدث يتم عندما يبتعد السوفت وير عن سلوكه المقصود.

مسرد المصطلحات

المصطلح بالإنجليزية	معناه بالعربية
Real Time Systems	برامج نظم الوقت الحقيقي
Embedded Systems	النظم المطمورة (المطوقة بإحكام)
Scientific Software	البرنامج العلمي
Programmer productivity	إنتاجية المبرمج
Predicting software costs	تنبؤات تكاليف البرنامج
Software packages	حزم البرامج
Batches	دفعات
Portability	قابلية التنقل
Remedial maintenance	الصيانة العلاجية
Adaptive maintenance	الصيانة التكيفية
Reliability	الموثوقية
Human-Computer Interaction	تفاعل الحاسوب مع البشر
A software crisis	أزمة السوفت وير
Quality	الجودة

إجابات التدريبات

تدريب (1)

برنامج التحكم في مراكز الطاقة النووية، برامج توجيه الطائرات.

تدريب (2)

الأمثلة هي :

- التوجيه في الطيران.
- التحكم في الإجراءات والعمليات الدقيقة مثل محطة التوليد.
- التحكم في المعدات الطبية.

المراجع

أولاً : المراجع العربية :

[1] أ. د. السيد محمود الربيعي "هندسة البرمجيات 1" جامعة السودان المفتوحة،1429 هـ.

[2] روجر بريسمان ، "هندسة البرمجيات" ، الدار العربية للعلوم ، مركز التعريب والبرمجة ، الطبعة الأولى ، 1425هـ - 2004م.

ثانياً : المراجع الإنجليزية :

[3] Ian Somerville , "Software Engineering", Addison Wesley, 2001.

[4] Professor EL-Sayed M. El-Rabaie, "Some main aspects for writing specialized microwave circuit design packages", Internal Report, under request.

ملحق 2

التصميم مع إعادة الاستخدام
Design with reuse

إن الهدف من دراسة هذا الجزء هو شرح كيفية إعادة استخدام البرمجية المتواجدة و التي يمكن أن تندمج في عملية تصميم النظام الجديد المطلوب تطويره.

تعتمد عملية التصميم في أغلب الأنظمة الهندسية على إعادة استخدام المكونات ، فالمهندسون الكهربائيون أو الميكانيكيون لا يحددون نموذجًا لكل جزئية من الجزئيات التي يجب أن تصنع ، و لكنهم يعتمدون في تصميمهم على المكونات التي سبق أن جربت و اختبرت من خلال أنظمة أخرى.

وهذه المكونات ليست فقط أجزاءً صغيرة مثل الشفائر والصمامات لكنها تشتمل على أنظمة فرعية رئيسية مثل المحركات والمكثفات والتربينات . ولقد أصبحت الحاجة الآن إلى بحوث مقارنة لتطوير البرامج ، والتي ينبغي أن تعتبر أحد الأصول أو الموجودات، وإعادة استخدام هذه الأصول يعتبر شيء ضروري لزيادة العائد من خلال تكاليف التطوير.

إن الطلب على تخفيض إنتاج البرامج وتكلفة الصيانة وأنظمة التوزيع السريع و النوعية الأجود يمكن تحقيقها فقط عن طريق إعادة استخدام البرمجيات.

ولكي نتمكن من إعادة استخدام البرمجيات، ينبغي أن يؤخذ ذلك في الاعتبار عند تصميم البرامج ومتطلبات العملية الهندسية. إن فرص إعادة الاستخدام ممكنة أثناء البرمجة وذلك عندما تناسب المكونات المتطلبات وعليه فإن نظام إعادة الاستخدام يتطلب عملية تصميم تأخذ بعين الاعتبار كيف يمكن أن يعاد استخدام الأنظمة الموجودة .

إن إعادة الاستخدام المبنية على هندسة البرمجة هو بحث بحاجة إلى التطوير ومحاولة إعادة الاستخدام بالصورة القصوى، فوحدات البرمجة التي تتم إعادة استخدامها ربما تكون ذات أحجام مختلفة، فعلى سبيل المثال:-

1. إعادة استخدام نظام التطبيق Application system reuse:

إن نظام التطبيق الكامل يمكن إعادة استخدامه إما عن طريق نسخه دون إحداث أي تغيير في الأنظمة الأخرى أو عن طريق تطوير عائلات التطبيقات التي يمكن تشغيلها عن طريق خطط مختلفة أو من خلال تخصيصها لتلبية حاجات مستهلكين ذوي متطلبات خاصة.

2. إعادة استخدام المكونات Component reuse:

إن مكونات التطبيق يتراوح حجمها من النظام الفرعي إلى الكائنات المفردة التي يمكن إعادة استخدامها ،على سبيل المثال : تطوير نظام تشغيل النموذج ، والذي تم تطويره كجزء نصي من العملية يمكن أن يستخدم في نظام إدارة قواعد البيانات.

3. إعادة استخدام الوظيفة Function reuse :

مكونات التطبيق والذي تنفذ وظيفة واحدة ، مثل الوظيفة الحسابية (الرياضية) يمكن إعادة استخدامها، وهذا الشكل من إعادة الاستخدام والمبني على معايير مكتبية، قد شاع خلال الأربعين سنة الماضية.

إن نظام تطبيق إعادة الاستخدام قد استخدم بصورة واسعة خلال سنوات كثيرة من خلال شركات البرامج التي تنفذ برمجتها عبر العديد من الآليات تكييفها وفق بيئات مختلفة كما إن إعادة استخدام الوظيفة أسست بمعايير مكتبية ، مثل الصور البيانية والمكتبات الحسابية ، و بالرغم من الاهتمام بمكونات إعادة الاستخدام منذ أوائل الثمانينات إلا أنه في السنوات القليلة الماضية تم قبول هذا النظام كبحث عملي تطبيقي لتطوير أنظمة البرمجة .

ومن الفوائد الواضحة من إعادة استخدام البرامج تقليل التكلفة الكلية للتطوير ، فهناك بعض مكونات البرامج تحتاج إلى التحديد والتصميم والتنفيذ وإعادة معرفة مصداقيتها ، ومع ذلك ... فإن خفض التكلفة قد لا تعتبر الميزة الوحيدة الممكنة من إعادة الاستخدام ، فهناك أيضاً ، عدد من مميزات إعادة الاستخدام غير تقليل التكلفة . و هناك العديد من الفوائد لإعادة استخدام البرمجية كما تم توضيحها في الجدول التالي:-

الفائدة Benefit	الشرح Explantion
زيادة المصداقية / الاعتماد على البرنامج Increased reliability	إن إعادة استخدام مكونات النظام التي سبق استخدامها في أنظمة العمل ، ينبغي أن تكون أكثر مصداقية من المكونات الجديدة و تم اختبار وتجربة هذه المكونات من خلال بيئات مختلفة و متنوعة ، وقد تم اكتشاف وحذف أخطاء التنفيذ و التصميم في الاستخدام الأولي للمكونات و نتيجة لذلك فقد تم تقليل عدد المحاولات الفاشلة عند إعادة استخدام المكونات .
تقليل الخطر الناتج عن العملية Reduced process risk	في حالة ظهور العنصر أو المكون الأساسي، فإن هنالك شك بأن تكاليف إعادة استخدام العنصر هو اقل من تكاليف تطويره ، وهذا عامل مهم لإدارة المشروع، وذلك لأنه يقلل الشكوك في تقدير تكلفة المشروع ، وهذا يمكن أن يكون صحيحاً في حالة عدم إعادة استخدام المكونات الكبيرة نسبياً مثل النظام الفرعي.
الاستخدام الفعال للاختصاصين Effective use of specialists	بمقدور الاختصاصين – بدلاً من التطبيق– تنفيذ نفس العمل على مشاريع مختلفة و تطوير المكونات القابلة للاستخدام و التي يضعونها بطريقة تختصر المعلومات.
تطويع المعايير Standard compliance	إن بعض المعايير مثل المعايير الخاصة بالمستخدم و التي يمكن تنفيذها كمجموعة مكونات المعيار ، مثال ذلك ، المكونات القابلة للاستخدام يمكن تطويرها لتنفيذ قائمة على سطح المستخدم . إن جميع التطبيقات تقدم بنفس القائمة للمستخدمين و

استخدام معيار سطح المستخدم يحسن المصداقية ، لأن المستخدمين يقل احتمال ارتكابهم أخطاءً عندما تقدم بأسطح متشابهة .

التطوير المتسارع Accelerated development

إن طرح النظام في السوق بأسرع وقت ممكن غالباً ما يكون أكثر أهمية من تكاليف التطوير الكلية، و إعادة استخدام المكونات تزيد من سرعة إنتاج النظام وذلك لأن كلاً من التطوير و وقت الصلاحية ينبغي أن تقلل.

فوائد إعادة استخدم البرمجية
Benefits of software reuse

وهناك ثلاث متطلبات ضرورية لتصميم البرمجية و تطويرها عن طريق إعادة الاستخدام :-

1) من الضروري إيجاد مكونات مناسبة قابلة للاستخدام . فالمنظمات و المؤسسات تحتاج إلى قاعدة من المكونات القابلة للاستخدام والموثقة و المصنفة بصورة مناسبة ، ويجب تسهيل البحث عن المكونات في هذا التصنيف إن وجدت.

2) يجب أن تتوفر السرية في مكونات إعادة الاستخدام بحيث تتصف المكونات بالمصداقية و الدقة ، فجميع المكونات في تصنيف المنظمة يجب أن يشهد لها بالتأكيد على وصولها لمعايير نوعية ، وهذا في الجانب العملي غير حقيقي لكون العاملين في المنظمة يتعاملون بصورة غير رسمية حول مصداقية المكونات .

3) المكونات يجب أن يكون لها وثائق مرتبطة بها، وذلك لمساعدة المستخدم لفهمها وتكييفها مع التطبيقات الجديدة ، ويجب أن يشتمل هذا التوثيق على معلومات تدلنا على مكان إعادة المكونات أو على إيجاد أي مشاكل تتعلق بإعادة الاستخدام .

إن الاستخدام الناجح للفيجوال بيسك (Visual Basic) و لغة فيجوال سي بلس بلس (C++) مع المكونات ولغة الجافا (Java) و الجافا بينس (Java Beans) قد بينت أهمية إعادة استخدام المكونات المبنية على البرمجية .

والجدول التالي يبين أهم مشاكل إعادة الاستخدام Problems with reuse

المشكلة Problem	التوضيح Explanation
ارتفاع تكلفة الصيانة Increased maintenance costs	إن تكاليف الصيانة قد ترتفع بسبب عدم توفر شفرة مصدر العنصر ، كما أن إعادة استخدام عناصر النظام ربما تصبح غير منسجمة مع تغيرات النظام بصورة متزايدة.
الافتقار إلى أدوات مساعدة Lack of tool support	إن مجموعة الأدوات التابعة لنظام (CASE) لا تساند نظام إعادة الاستخدام في حالة تطويره و لذلك ، ربما يصعب أو حتى يستحيل علينا دمج هذه الأدوات مع نظام مكتبة المكونات ، فعملية البرمجة افترضت بواسطة هذه الأدوات أن لا تأخذ في الاعتبار إعادة الاستخدام .
أعراض (ليست مكتشفة هنا) Not-invented –here syndrome	يفضل بعض مهندسي البرامج أحياناً إعادة كتابة المكونات ، لاعتقادهم بإمكانية التحسين من مكونات إعادة الاستخدام. إن هذه العملية هي جزءاً من الحقيقة التي تقول : (أن كتابة البرنامج الأصلي هو أكثر تحديًا من إعادة استخدام البرنامج السابق كتابته).
صيانة مكتبة المكونات Maintaining a component library	إن عملية تزويد مكتبة المكونات و تأكيد مطوري البرامج باستخدام هذه المكتبة ، يمكن أن يكون ذا تكلفة عالية . إن تقنياتنا الحالية لتصنيف وتبويب ومراجعة مكونات البرمجية لا زالت غير مكتملة في وقتنا الحالي .

تأقلم وإيجاد المكونات القابلة للاستخدام Finding and adapting Reusable components	إن مكونات البرمجة تُكتشف و تُفهم وأحياناً تعدل من خلال البحث المكتبي، لتعمل في بيئة جديدة فيتحتم على المهندسين أن يكون لديهم ثقة معقولة بإيجاد هذه المكونات قبل تضمينها بصورة روتينية إلى بحث المكونات كجزء من عملية التطوير العادية