بسم الله الرحمن الرحيم

مقرر هندسة البرمجيات
١٤٩٤

الوحدة الأولى

مقدمة في هندسة البرمجيات
Introduction To Software Engineering

إعداد: م. هناء قشطة

الفصل الدراسي الأول

٢٠٢١م-٢٠٢٢م

أهداف اللقاء

- مفاهيم عامة في هندسة البرمجيات.
- تطوير هندسة البرمجيات.
- صفات البرمجيات الجيدة.
- دورة حياة نظام البرمجيات ومراحل تطويره.
- أساسيات البرمجة الكينونية.
- مخططات انسياب التحكم.

١. مقدمة

لقد أصبح التعامل مع الحاسوب و برمجياته، و توظيف هذه البرامج و الاستفادة منها ضرورة من ضرورات العصر. و كلما امتلكت الأمة قدرات جيدة في هذا المجال، كلما استطاعت تحقيق التقدم و الازهار في شتى المجالات، و هذا يعني تحقيق أنشطة اقتصادية تؤدي إلى زيادة الإنتاج و قوة الاقتصاد، و لقد أمسى النافس في صناعة البرمجيات شديدًا، و لذلك يجب أن تتبع أساليب متعددة تساعد في التحليل و التصميم الجيدين و اللذين يؤديان في نهاية المطاف إلى منتج يمكن ترويجه بتكاليف قليلة و جودة عالية. من هنا جاء مفهوم هندسة البرمجيات، الذي يعني بناء برمجيات تطبق في بنائها الأساليب العلمية الحديثة والمتطورة.

مجالات الحياة

١. نظم إدارة البنوك.

٢. نظم إدارة المستشفيات.

٣. نظم إدارة الجامعات و المؤسسات التعليمية.

٤. نظام الأفراد و شؤون العاملين.

٥. السيطرة الآلية في الصناعة.

٦. نظم الدفاعات الجوية.

٧. إطلاق الصواريخ عبر القارات.

٢. مفاهيم عامة في هندسة البرمجيات

النظام

هو مجموعة من العناصر و القواعد التي تعمل معًا لأداء مهمة، و في الأغلب يكون له مدخلات، يتم إجراء عليها عمليات و معالجة، للخروج في نهاية المطاف بالمخرجات المرغوبة.

قد يكون النظام

- كبيرًا و معقدًا كنظام منظومة صاروخية متقدمة.
- صغيرًا و بسيطًا كنظام أداء الوضوء.

٢. مفاهيم عامة في هندسة البرمجيات

البرمجة

يمكن أن تكون عملا فرديا ، بينما هندسة البرمجيات عمل فريق.

المبرمج وحده يكتب برنامجا كاملا بينما مهندس البرمجيات يعتني بجزء من نظام متكامل لينضم هذا الجزء إلى أجزاء أخرى يعتني بها مهندسو برمجيات آخرون.

٢. مفاهيم عامة في هندسة البرمجيات

هندسة البرمجيات

هي عملية بناء نظام متعدد الأجزاء بواسطة عدد من المختصين.

- فهندسة البرمجيات يقوم بها فريق، أما البرمجة يمكن أن يقوم بها مبرمج فردي.
- و هندسة البرمجيات تحتوي على مراحل لبناء النظام، يكون لها مرحلة بداية و مرحلة نهاية و المراحل فيما بينهما تشكل دورة حياة النظام.

٣. تطور هندسة البرمجيات

مشكلات صناعة البرمجيات فى أواخر الستينات:

- التكاليف الباهظة المصاحبة لتطوير البرمجيات.
- الجهد الكبير المبذول.
- عدم إمكانية حساب الفترة الزمنية التي يستغرقها المشروع.

هذه الأسباب أدت بصناعة البرمجيات إلى:

- العجز عن تنفيذ المشروع في الوقت المحدد له.
- العجز عن تقليل التكاليف الباهظة.
- العجز عن إنتاج برمجيات ذات جودة عالية.

٣. تطور هندسة البرمجيات

عوامل تبنى مفهوم هندسة البرمجيات :

- ارتفاع تكاليف صناعة البرمجيات نسبيًا مقترنة مع المكونات المادية للحاسوب.
- العجز عن تسليم المشروع في الوقت المحدد له، و عدم القدرة على تحديد الزمن المطلوب.
- عدم القدرة على إنتاج برمجيات ذات جودة عالية.
- الدور المتزايد لصيانة البرمجيات.
- التقدم السريع في التقنيات المادية للحاسوب.
- الطلب المتزايد على البرمجيات.
- التقدم في أساليب بناء البرمجيات.
- الطلب على البرمجيات الكبيرة و المعقدة.

٣. تطور هندسة البرمجيات

أسس و مفاهيم تطوير هندسة البرمجيات

أولاً: الاعتمادية (أو المصداقية) **Reliability**

- **درجة الاعتمادية:** هي صلاحية البرمجيات في حال تطبيقها في بيئة معينة، لفترة زمنية محددة.
- **الصلاحية:** هي خلو البرمجيات من الأخطاء، بحيث تطبق بثقة و عدم الخوف من نتائج خاطئة، و ضمان تشغيل البرمجيات باحتمال قليل جدًا للعطل أو التوقف عن العمل.

٦. مثال ص٩.

٣. تطور هندسة البرمجيات

-**يمكن تحقيق الاعتمادية باتباع ثلاث طرق:**

- تطوير برمجيات خالية من الأخطاء (التصميم، التنفيذ الصحيح، الفحص و التحقق).
- تطوير برمجيات استثنائية (تستطيع أداء العمل بالرغم من وجود الأعطاب، استبدال اللجوء إلى التوقيف بوسيلة أخرى).
- الكشف عن الأخطاء قبل تشغيلها في بيئة العمل (و ذلك باختبارها و فحصها بوسائل الفحص الثابت و الديناميكي).

٣. تطور هندسة البرمجيات

أسس و مفاهيم تطوير هندسة البرمجيات

ثانيًا: سهولة القراءة Readability

- هي سهولة قراءة الجمل البرمجية المكونة للبرمجيات.
- **تتم سهولة القراءة عن طريق:**
  - تجنب الاختصارات غير الواضحة عند كتابة البرنامج.
  - استخدام الأسماء ذات المعنى التي تعكس المسميات (مثال ص١٠).

٣. تطور هندسة البرمجيات

- **فوائد سهولة القراءة:**
  - يمكن اكتشاف الأخطاء في البرنامج بسهولة، و بالتالي تصحيحها بسهولة أكبر.
  - القدرة على تطوير البرنامج و التعديل عليه بيسر.
  - الاستفادة من بعض أجزاء البرنامج في مواضع و برامج أخرى، مما يساعد على زيادة الإنتاجية.

فمثلاً من التقاليد المتعارف عليها استخدام الأفعال في تسمية البرامج الفرعية.

٣. تطور هندسة البرمجيات

أسس و مفاهيم تطوير هندسة البرمجيات

ثالثًا: جودة البرمجيات **Software Quality**

- هي بناء برمجيات تؤدي المهام المطلوبة منها للجهة المستفيدة بكفاءة عالية، و تلتزم بالمقاييس المتبعة، و تحقق خصائص البرمجيات المتعارف عليها بين مراكز صناعة البرمجيات.
- **عوامل تؤثر في جودة البرمجيات:**
  - القدرة على إدامة و صيانة البرمجيات.
  - القدرة على توافقها مع أنظمة الحاسوب المختلفة.
  - القدرة على أداء وظائفها بجودة عالية.
  - تفصيل العوامل الشكل١ ص١٢.

النقل : هل تنفذ البرمجيات على جهاز حاسوب من نوع آخر؟

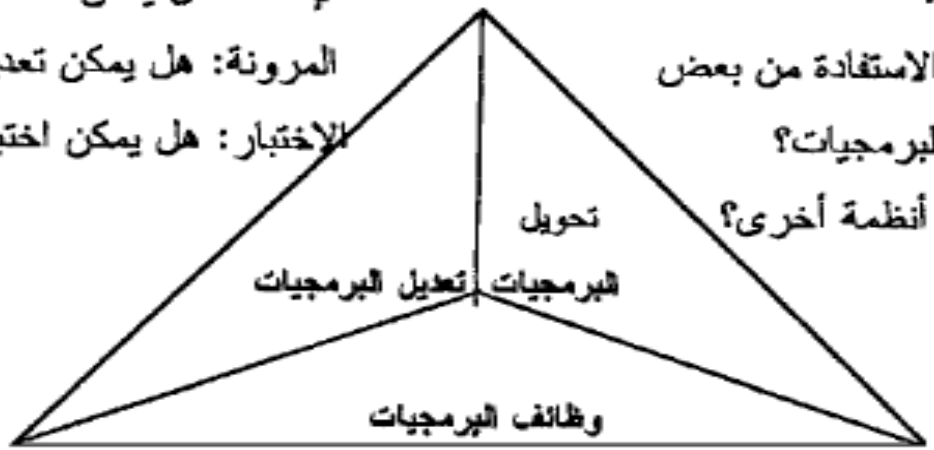
إعادة الاستعمال: هل يمكن الاستفادة من بعض مقاطع البرمجيات؟

الربط: هل يمكن ربطها مع أنظمة أخرى؟

الإدامة: هل يمكن تصحيح الخلل؟

المرونة: هل يمكن تعديله بسهولة؟

الاختبار: هل يمكن اختبار البرمجيات؟



صحة البرمجيات : هل تؤدي البرمجيات الوظيفة المطلوبة بصورة صحيحة؟

درجة الاعتمادية : هل تؤدي البرمجيات وظيفتها بصورة دقيقة في مرات التنفيذ كافة؟

الكفاءة : هل تنفذ البرمجيات على أجهزة الحاسوب بالسرعة المطلوبة؟

التكامل : هل البرمجيات متكاملة بحيث يمكن تمرير البيانات بين اجزائها، مع منع غير المخولين من استغلالها؟

الشكل (1) : العوامل المؤثرة في جودة البرمجيات

٣. تطور هندسة البرمجيات

أسس و مفاهيم تطوير هندسة البرمجيات

ثالثًا: جودة البرمجيات **Software Quality**

- الهدف الرئيسي من هندسة البرمجيات هو إنتاج أنظمة برمجية ذات جودة عالية.
- **مفهوم الجودة من ثلاث جهات:**
  - **الجهة الأولى:** الممولون الذين يتحملون تكاليف تطوير البرمجيات.
    - توفير الأموال.
    - زيادة الإنتاجية.
    - المرونة.
    - الاعتمادية.
    - الكفاءة.

٣. تطور هندسة البرمجيات

أسس و مفاهيم تطوير هندسة البرمجيات

ثالثًا: جودة البرمجيات **Software Quality**

- **مفهوم الجودة من ثلاث جهات:**

- **الجهة الثانية:** المستخدمون الذين يستخدمون البرمجيات و يتعاملون معها.
  - تأدية المهام المطلوبة.
  - سهولة الاستخدام.
  - سهولة التعلم و التذكر.
  - الاعتمادية.
  - الكفاءة.

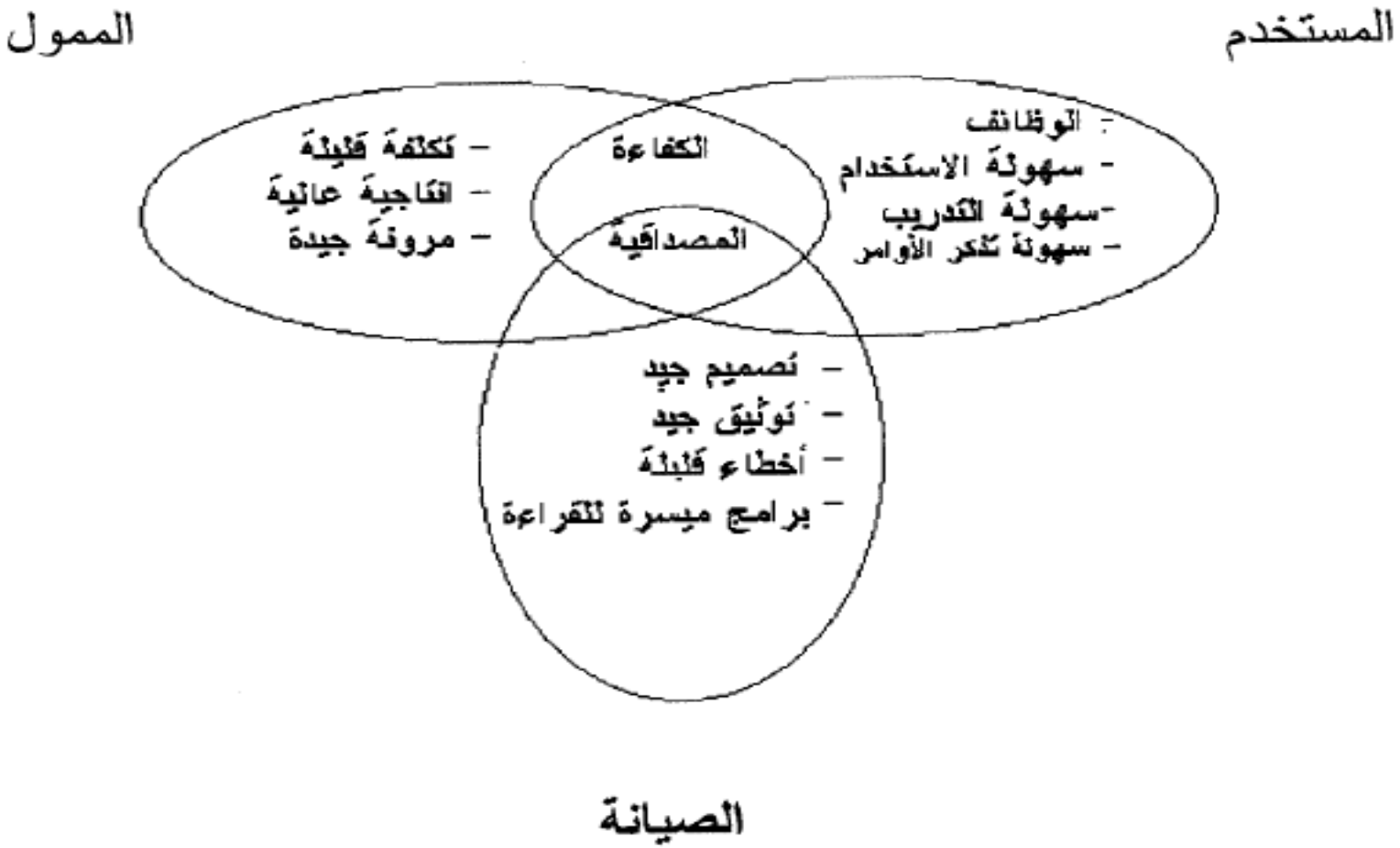
٣. تطور هندسة البرمجيات

أسس و مفاهيم تطوير هندسة البرمجيات

ثالثًا: جودة البرمجيات Software Quality

- **مفهوم الجودة من ثلاث جهات:**

- **الجهة الثالثة:** فريق الصيانة الذي يقوم بتصحيح الأخطاء، و تطوير البرمجيات، و الإضافة عليها.
  - أقل عدد ممكن من الأخطاء.
  - سهولة القراءة، مكتوبة بوضوح.
  - التوثيق الجيد (الملاحظات و التوضيحات المناسبة).
  - التصميم الجيد و البنية الجيدة.
  - الاعتمادية.



الشكل (2) : مفهوم الجودة لدى الجهات ذات العلاقة

٣. تطور هندسة البرمجيات

إن الجودة ضرورية، قد لا يمكن تحقيقها للجهات الثلاث، و هي ميزة لا يمكن إضافتها متى يتطلب الأمر، حيث يجب التخطيط لها منذ بداية تطوير نظام البرمجيات.

٤. صفات البرمجيات الجيدة

١. البرمجيات الأقصر هي الأيسر متابعة.
٢. تفضيل البرمجيات ذات القرارات الأقل.
٣. تجنب تداخل القرارات.
٤. التحديد الجيد لتركيب البيانات.
٥. الإكثار من التوضيح والشرح.
٦. الانسجام: تصميم فقرات البرمجيات بأسوب موحد ونمطي.
٧. التكامل.
٨. الكفاءة: استخدام أقل حجم ممكن من الموارد (حجم التخزين، زمن المعالجة).
٩. الفاعلية: قيام البرمجية بالوظائف المطلوبة وفقا لرغبة المستخدم.

٥. دورة حياة نظام البرمجيات ومراحل تطويره

- **دورة حياة البرمجيات:** هي مجموعة من الأنشطة التي تبدأ منذ بداية التفكير بالبرمجية، مروراً بتحليلها وتصميمها وإنتاجها واستخدامها. وتقسم الدورة إلى عدة مراحل:
  ١. تحديد المتطلبات وتحليل النظام.
  ٢. التصميم.
  ٣. التحويل.
  ٤. الفحص والاختبار.
  ٥. التشغيل والتطبيق.
  ٦. التوثيق.
- **مراحل إضافية:** مرحلة وضع البرمجيات قيد التطبيق، مرحلة الخروج من الخدمة.

٥. دورة حياة نظام البرمجيات ومراحل تطويره

المجموعات المؤثرة في تقدم البرمجيات وتطويرها:

- الإدارة.
- محللو النظام.
- المبرمجون ومبرمجو الأنظمة.
- مستخدمو النظام المقترح.
- قسم الصيانة.

٥. دورة حياة نظام البرمجيات ومراحل تطويره

المرحلة الأولى: تحديد المسألة ومتطلبات المستفيد Requirements Phase

تهدف إلى دراسة احتياجات ومتطلبات المستخدمين وتحديدها، وبالتالي صياغة الأهداف والوظائف التي يجب ان يقوم بها النظام البرمجي المطلوب.

المرحلة الثانية: صياغة المتطلبات Specification Phase

دراسة المتطلبات التي تم جمعها من الجهة المستفيدة بهدف تبويبها وترتيبها لإحداث الربط والتسلسل بين مختلف أجزائها، ثم صياغة هذه المتطلبات إما بأسلوب لغوي، او باستخدام إحدى اللغات المناسبة لهذه المرحلة.

٥. دورة حياة نظام البرمجيات ومراحل تطويره

المرحلة الثالثة: التخطيط Planning Phase

- تتطلب دراسة النظام القائم وتحليله وتحديد امكانية تعديله او بناء نظام جديد وقد يستلزم الامر حساب الجدوى الاقتصادية.

- **يجب أن تبدأ الدراسة بما يلي:**

١ـ تحديد المخرجات والمدخلات للنظام المقترح.

٢ـمعالجة البيانات.

٣ـالمعلومات او النتائج الخارجة من المعالجة.

٥. دورة حياة نظام البرمجيات ومراحل تطويره

المرحلة الرابعة: تحليل النظام Analysis Phase

- تهدف إلى تقرير الحاجة إلى نظام جديد أو تعديل النظام السابق.
- يتضمن التحليل خطوتين هما:

١. **تجميع الحقائق:** ويمكن تجميع الحقائق من مصادر عدة مثل:
- الأشكال المكتوبة.
- الاستبيانات.
- المقابلات.
- المشاهدات.

٢. **تحليل الحقائق وفرزها وتصنيفها:** ومن الوسائل المساعدة في ذلك:
- المخططات الانسيابية System Flow Charts
- جداول القرارات Decision Tables
- انسيابية البيانات Data Flow Diagram

٥. دورة حياة نظام البرمجيات ومراحل تطويره

المرحلة الخامسة: تصميم النظام Design Phase

- **تتلخص خطوات تصميم النظام بما يلي:-**
  - تحديد الأهداف.
  - تطوير تصميم نظام البرمجيات.
  - تحليل التكاليف والمنافع.
  - إعداد تقرير عن التصميم.
- **يجب توضيح الأهداف:** من ناحية الإدارة ، العمل ، التكاليف
- **وخلال مرحلة تصميم النظام ينبغي دراسة:**
  - مستلزمات العمل والمدخلات والمخرجات.
  - الملفات والبيانات.
  - الرقابة والضبط على النظام.
  - الخطوات المتبعة في تنفيذ العمل (خطة العمل).
- لتحليل التكاليف والمنافع تعد قائمة بالتكاليف وقائمة بالمنافع.
- بعد الدراسة والتحليل يجب تلخيص النتائج وكتابة تقرير إلى الإدارة العليا.

٥. دورة حياة نظام البرمجيات ومراحل تطويره

المرحلة السادسة: تنفيذ النظام Implementation Phase

- **يتم خلال هذه المرحلة بناء نظام المعلومات المقترح ويشمل:**
  ١. شراء المعدات أو استئجارها.
  ٢. كتابة البرامج وبناؤها.
  ٣. اختبار النظام.

- **أهم الأنشطة التي تتضمنها:**
  - كتابة البرامج وفحصها.
  - مرحلة التحويل المتوازي.

٥. دورة حياة نظام البرمجيات ومراحل تطويره

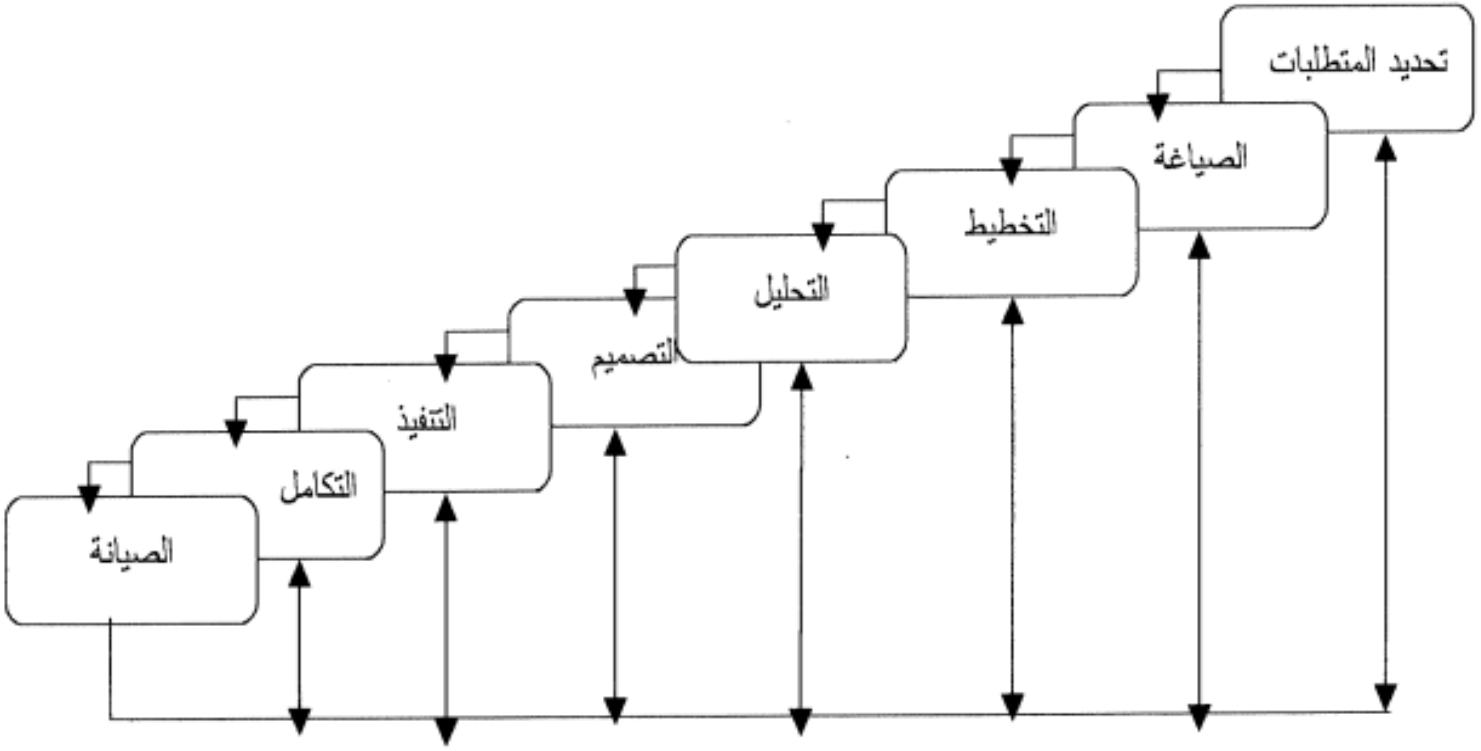
المرحلة السابعة: مرحلة التكامل Integration Phase

تعني جمع أجزاء النظام المختلفة لبناء النظام المتكامل(المنتج المطلوب)

المرحلة الثامنة: مرحلة الصيانة Maintenance Phase

تستمر هذه المرحلة مع النظام طوال دورة حياته، وتعد من أكثر المراحل تكلفة.

٥. دورة حياة نظام البرمجيات ومراحل تطويره



الشكل (3) :الشلال المائي

٦. أساسيات البرمجة الكينونية

البرمجة الكينونية
Object-Oriented Programming

- تتجه في تصميم البرامج نحو تمثيل مسائل العالم الحقيقي باستخدام الكائنات والأصناف.
- توفر إمكانيات متعددة منها: التنظيم الجيد، والمقاطع البرمجية المتكاملة، والوضوح والانسجام، والكفاءة، والاستخدام المتعدد.
- تعد البرمجة الكينونية أسلوبا جديدا في التفكير وبناء البرامج بجمع حقول البيانات مع البرامج الفرعية التي تستند في عملها على تلك الحقول في بنية واحدة تسمى كينون Object.

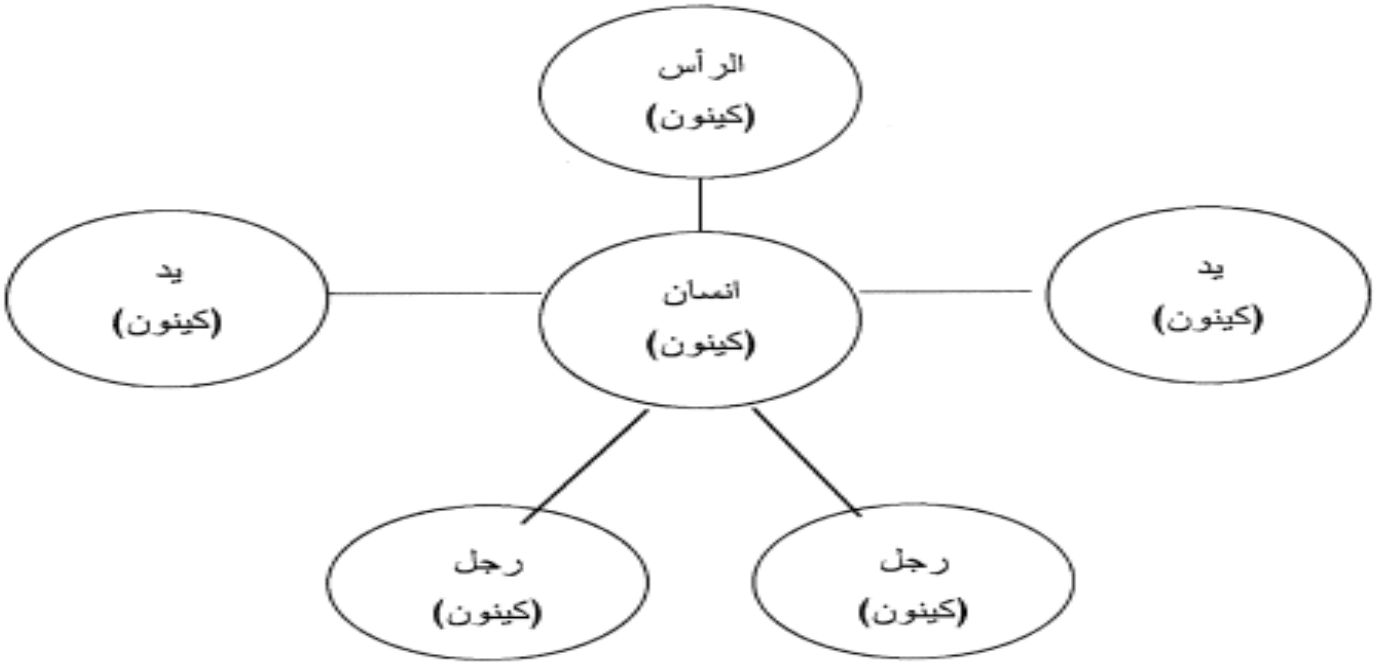
٦. أساسيات البرمجة الكينونية

الكينون Object: هو وحدة برمجية لها استقلاليتها وهو تجسيد لشيء واقعي يؤدي مهمة محددة، يحتوي بياناته الخاصة به كما يحتوي على العمليات التي يسمح بتنفيذها على تلك البيانات.

ومن اجل بناء برامج كينونية يستلزم الأمر اتباع أربع خطوات هي:

- تحديد تركيب الكينون المطلوب
- تحديد المتغيرات التي يتضمنها كل كينون وتعريف كل متغير
- تحديد البرامج الفرعية (الوسائل) المطلوبة لتنفيذ عمليات الكينون.
- تحديد أهداف كل برنامج فرعي وأهمية المتغيرات الأساسية وتحديد الشروط الاولية والنتائج المتوقعة.

٦. أساسيات البرمجة الكينونية



شكل (4) : نموذج وكينونات

٦. أساسيات البرمجة الكينونية

- **الصف Class:** يستخدم لوصف مجموعة كينونات لها نفس السمات او الملامح، وتقوم بعمليات متشابهة. ومن الممكن ان يحتوي النظام المتكامل عدة صفوف وعادة ما يوجد ترابط بين كينونات تنتمي لصفوف مختلفة.
- مهام الكينونة هي بمثابة الخدمات التي تقدمها للكينونات الاخرى.
- يجب تحديد كينونات نظام البرمجيات وخصائصها والعمليات المصاحبة لها.
- ينظر إلى الأسماء الرئيسة باعتبارها كينونات والأفعال باعتبارها عمليات.

٦. أساسيات البرمجة الكينونية

مزايا البرمجة الكينونية:

١. خصوصية المعلومات وإمكانية اخفائها.
٢. **التوارث Inheritance:** استخدام نفس السمات والملامح الموجودة لدى كينون بواسطة كينون اخر. حيث يمكن ان يرث كينون ما بعض سماته وخصائصه من كينون اخر.
٣. **الاستعمال المتعدد Polymorphism:** استدعاء الكينون بمتغيرات مختلفة الانواع والحصول على نتائج مختلفة، حيث يستخدم برنامج فرعي بالاسم نفسه لأكثر من غرض.
٤. **استخدام الكينونات الديناميكية:** حيث يتم تخصيص جزء من الذاكرة للكينونات خلال مرحلة تنفيذ البرامج، ويلغي التخصيص ويعاد حجم الذاكرة المقتطع الى وضعه الاصلي حال انتهاء دور الكينون.

٧. مخططات انسياب التحكم
Control Flow Graphs

- تصف هذه المخططات التركيب المنطقي لمقاطع البرمجيات، حيث المقطع هو عبارة عن وحدة برمجة من برامج (برنامج فرعي أو دالة)
- يتكون المخطط الذي يصف مقطعاً من رؤوس(Nodes) او دوائر صغيرة تمثل الجمل وحواف(Edges) تمثل انسياب التحكم بين الرؤوس.

Questions or Comments?