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

محتويات الوحدة

المقدمة

تمهيد

عزيزي الدارس ،،

سنتعرف من خلال هذه الوحدة "مقاييس البرمجيات" وجودتها إلى وأهم السمات التي يجب أخذها في الاعتبار لإنتاج برمجيات ذات جودة عالية.

- تبدأ الوحدة بتعريف الجودة وفقاً لمقياس ايزو المعياري "B204" مع الأخذ في الاعتبار مجموعة من السمات الأساسية لإنتاج برمجيات ذات جودة عالية ، وعند الانتهاء من إنتاج البرمجية يتم التأكد من جودتها أو عدمه من خلال قياس خواص البرمجية ومراقبة وتنظيم عملية تطوير البرمجية.
- إدارة جودة البرمجيات: تشمل الأنشطة أو العمليات التي تضمن أن المشروع البرمجي يحقق أهدافه بمعنى أخر يحقق توقعات العملاء ، ويجب أن تنفصل إدارة الجودة عن إدارة المشروع لضمان الاستقلالية.
- مقاييس البرمجيات وفي هذا القسم من الوحدة نتعرف على مقاييس البرمجيات ، ونتناول خصائص مقاييس البرمجيات الجيدة ، وتصنيف مقاييس البرمجيات ، مقاييس تحديد مستوى مؤسسات تطوير البرمجيات.

عزيزي الدارس أرجو أن لا يغيب عن ذهنك عند قراءة مادة الوحدة محاولة الإجابة عن الأسئلة التقويمية ، والعودة إلى النص للتأكد من صحة الإجابة ، كما أرجو عدم الرجوع إلى إجابات التدريبات إلا بعد محاولة حلها أولاً.

نأمل أن تؤدي دراستك إلى هذه الوحدة إلى إثراء فهمك وتعرفك إلى خطوات تقدير تكلفة البرمجيات.

أهداف الوحدة

عزيزي الدارس، بنهاية دراسة هذه الوحدة ينبغي أن تكون قادراً على أن :

- تعرِّف جودة البرمجيات.
- تعدد أهم السمات الأساسية التي يجب أخذها في الاعتبار لإنتاج برمجيات ذات جودة عالية.
- تصف بإيجاز أهم السمات التي تحدد مدى جودة البرمجيات.
- تعرِّف إدارة جودة البرمجيات.
- تلخص أهم أنشطة إدارة جودة البرمجيات.
- تعدد الأسئلة الواجب الإجابة عليها أثناء التخطيط لإدارة جودة البرمجيات.
- تسرد المحتويات الواجب تضمينها في وثيقة خطط إدارة الجودة.
- تعرِّف عملية تأمين جودة البرمجيات.
- تعرِّف المقاييس المعيارية للبرمجيات مع ذكر أمثلة عليها.
- تصف الفرق بين المقاييس المعيارية والإجراءات.
- تعدد أهم أنشطة تأمين جودة البرمجيات.
- تذكر أهمية إدارة جودة البرمجيات.
- تعرِّف مقاييس جودة البرمجيات.
- تلخص في نقاط أهمية مقاييس البرمجيات.
- تذكر خصائص مقاييس البرمجيات الجيدة.
- تصنف مقاييس البرمجيات إلى أنواعها المختلفة.
- تقيس حجم البرمجية ومدى درجة تعقيدها باستخدام مقاييس ماكابي وهالستيد.
- تعدد مقاييس الجودة.
- تصنف مقاييس تحديد مستوى مؤسسات تطوير البرمجيات.

1.جودة البرمجيات (Software Quality)

أصبحت مصطلحات الجودة وتأمين الجودة للمنتجات في العالم الصناعي اليوم مهمة وشائعة الاستخدام وخاصة في صناعة البرمجيات وذلك للاعتماد المطلق على هذه الصناعة في جميع مناحي الحياة. وكحد أدنى يمكننا القول بأن برمجية ما عالية الجودة إذا ما اتسمت بالسمات العامة التالية :

- تلبي احتياجات المستخدم.
- خالية من العيوب.
- يمكن الاعتماد عليها في العمل.
- يمكن إجراء صيانة لها (تصحيح ، تغيير،.....).

وتُعرف **الجودة** (Quality) وفقاً لمقياس أيزو المعياري **"ISO Standard 8204"** بأنها : المجموع الكلي لميزات وخصائص منتج أو خدمة ما والتي تؤثر على قدرته / قدرتها على تلبية حاجات المستخدم المحددة أو الضمنية.

"The totality of features and characteristics of a product or service that bear on its ability to satisfy specified or implied needs"

فمثلاً نعني بجودة التصميم **(Quality of Design)** مجموعة الميزات التي يحددها المصممون لمنتج ما ، فعندما يُطلب إنتاج منتج ذي مواصفات أداء عالية ، لا مناص من زيادة جودة تصميم المنتج إذا جرى تصنيعه وفقاً للمواصفات المطلوبة. أما جودة التوافق **(Quality of Conformance)** فهي درجة اتباع مواصفات التصميم خلال مرحلة التصنيع ، وكلما ازدادت درجة التوافق ارتفع مستوى جودة التوافق ، وبالتالي جودة المنتج.

ويعرف المنتج ذو الجودة . أيضاً . بأنه المنتج الذي يلبي ويستمر في تلبية الاحتياجات التي من أجله تم إنتاجه.

والجودة لها سمة أساسية وهي أنها يمكن قياسها ، وفي هندسة البرمجيات هذه القياسات يشار لها بالمقاييس **(Metrics)** ، فعلى سبيل المثال يمكن قياس التكاليف والحجم الذي تشغله البرمجية ، وتعقيد البرمجية الخ.

ولإنتاج برمجيات ذات جودة عالية يجب الأخذ في الاعتبار مجموعة من السمات الأساسية من أهمها :

- مدى كبر أو حجم البرمجية.
- مدي تعقيد عناصر مكونات البرمجية.
- مدى الاعتماد على البرمجية.
- تكاليف إنتاج البرمجية.
- الجهد اللازم لتغيير البرمجية لتلبى احتياجات المستخدم.

حيث إن ذلك يسهم إلى حد كبير في :

- التنبؤ بجودة البرمجية خلال مرحلة تطويرها أو إنتاجها.
- قياس جودة جزء من برمجية تم إنتاجها.
- مراقبة وتحسين عملية إنتاج البرمجية.
- المقارنة بين الطرق المختلفة لإنتاج البرمجية واختيار الأفضل منها.

ويمكن الأخذ في الاعتبار سمات الجودة سابقة الذكر في واحده أو أكثر من الأوضاع التالية :

1. خارج عملية التطوير لتوضيح الأهداف.
2. خلال عملية التطوير لتوجيه عملية التطوير لتحقيق الأهداف.
3. في النهاية لتقويم إنتاج البرمجية.

ويمكن إيجاز أهم السمات التي تحدد مدى جودة البرمجيات فيما يلي:

1. الصحة (Correctness) : وهي تمثل المدى التي تفي فيه البرمجية بمتطلبات المستخدم.
2. الاعتمادية (Reliability) : وهي تمثل الحد الذي يمكن أن تعمل فيه البرمجية باستمرار دون حدوث توقف أو فشل.
3. الكفاءة (Efficiency) : وهي كمية RAM ووقت المعالج الذي تستخدمه البرمجية.
4. التكاملية (Integrity) : وهي الدرجة التي تمكن فيها البرمجية المستخدم من الدخول وتنظيم إدخال المعلومات.
5. الاستخدام (Usability) : سهولة استخدام البرمجية.
6. الصيانة (Maintainability) : وهي تعبر عن الجهد المطلوب لاكتشاف خطأ وإصلاحه.
7. المرونة (Flexibility) : وهي تعبر عن الجهد اللازم لتغير البرمجية لمقابلة التغيرات المطلوبة.
8. الاختبارية (Testability) : وهي تعبر عن الجهد اللازم لاختبار البرمجية بفعالية.
9. الانتقالية (Portability) : تعبر عن الجهد اللازم لنقل البرمجية إلى حاسبات ذات مكونات وأنظمة تشغيل مختلفة.
10. إعادة الاستخدام (Re-usability) : تعبر عن الحد الذي يمكن به إعادة استخدام البرمجية أو جزء منها من خلال برمجية أخرى.
11. تضافرية العمل (Interoperability) : تعبر عن الجهد اللازم لجعل البرمجية تعمل بالتعاون مع برمجية أخرى.

وكما هو ملاحظ فإن هناك العديد من هذه السمات متناقض مع بعضه فعلى

سبيل المثال لإنتاج برمجية ذات كفاءة عالية فإن إمكانية النقل أو الحمل ربما تتناقض معه ، ولهذا فإن لكل مشروع تحقيق مجموعة من الأهداف المحددة توضع في الاعتبار قبل عملية التطوير والإعداد.

ولكن عندما تنتهي من إنتاج برمجية كيف تتأكد من جودتها من عدمه؟ ، هناك طريقتان للإجابة عن هذا التساؤل :

1) قياس خواص البرمجية التي تم إنتاجها.
2) مراقبة وتنظيم عملية تطوير البرمجية.

ولتوضيح الرؤيا دعنا نلاحظ أوجه الشبه بين إنتاج برمجية وإعداد وجبة غذائية من الطعام، فإذا أردت إعداد وجبة معينة فإنك تقوم بالخدمة عليها وبناءً عليه تحصل على وجبة ذات جودة يمكن معرفتها بقياس الطعم ، ورؤية اللون ، وساخنة أم لا ، وإذا كان فيها شيء غير مضبوط فإنه ليس هناك شيء يمكن عمله لضبط جودتها ، وعلى هذا فإنه يعاد التحكم في الكميات للمكونات والطريقة لضبط جودة الوجبة، ولإعداد وجبة ذات جودة يجب أن تمر عملية الإعداد بعدة مراحل بداية من ناحية إعداد المكونات وحتى الطهي بمراحله المختلفة ، وفي كل خطوة يمكن تصحيح الخطأ إذا ما تم شيء غير صحيح على سبيل المثال يمكن شراء مكونات جديدة إذا ما اتضح أن أحدها فاسد أو غير طازج أو أننا نعيد غسيل أحد المكونات إذا ما أتضح أن نظافته غير كافية ، وهكذا فإن جودة الوجبة يمكن تأكيدها بعمل اختبارات خلال عملية الإعداد، ويمكن تطبيق نفس المبدأ في عملية تطوير البرمجيات حيث يمكن إجراء قياسات ومراجعات مماثلة للتأكد من جودة البرمجية المنتجة. وكذلك فإنه لإعداد وجبه يلزم (وصفة للإعداد) والتي يمكن اتباعها وهذا مماثل تماما لاستخدام وسائل وطرق جيده خلال عملية تطوير البرمجية.

تدريب (1)

إذا ما طلب منك تطوير برمجية تقوم بعملية تحكم آلي لطائرة ما هي العوامل الهامة التي يجب أن تؤخذ في الاعتبار؟

تدريب (2)

ما هي معاملات الجودة التي تساعد قياس كثافة الخطأ (Fault Density) في تحقيقها؟

أسئلة تقويم ذاتي

عرف مصطلح الجودة وفقا لمقياس آيزو المعياري، وضح ذلك بأمثلة على:

- جودة التصميم.
- جودة التوافق.

حدد بإيجاز أهم السمات التي تحدد مدى جودة البرمجيات.

ما هي أهم أنشطة إدارة جودة البرمجيات.

2. إدارة جودة البرمجيات

(Software Quality Management)

إدارة جودة البرمجيات تشمل الأنشطة (Activities) أو العمليات (Processes) التي تضمن أن المشروع البرمجي يحقق أهدافه ، وبمعنى أخر فإن المشروع البرمجي يحقق توقعات العملاء، ويجب أن تنفصل إدارة الجودة عن إدارة المشروع لضمان الاستقلالية.

1.2 أنشطة إدارة جودة البرمجيات

:(Software Quality Management Activities)

يمكن تلخيص أنشطة إدارة البرمجيات في التالي :

1.1.2 نشاط تخطيط الجودة

(Quality Planning Activity)

هي عملية وضع خطة تضمن جودة البرمجية المطلوب تطويرها وذلك من خلال الإجابة على الأسئلة التالية :

- كيف يتم تقدير الجودة المطلوبة للبرمجية تحت التطوير؟.
- ما هي السمات المؤثرة في جودة البرمجية؟.
- كيف يتم تقويم الجودة؟.
- ما هي المعايير القياسية التنظيمية التي يجب تطبيقها عند اختبار جودة البرمجية ؟ ، وما هي المعايير الجديدة التي يجب أخذها في الاعتبار عند الضرورة؟.

وتُعد مرحلة تخطيط الجودة أهم مراحل إدارة جودة البرمجيات حيث إن

التخطيط الجيد يضمن تحقيق المراحل التالية وتحقيق أهدافها ، ويجب أن تكون خطط الجودة قصيرة ومدونة في وثائق محددة موجزة حتى يمكن قراءتها جيداً. ويجب أن تشمل :

- غاية الوثيقة وأفقها ، وأهداف الجودة **(Quality Objectives)** .
- قوائم بجميع المعايير القياسية **(Standards)** الخاصة بالمشروع ، وبعملية البرمجة ، والمنتج نفسه ، وكيفية قياسها والتحقق من إنجازها.
- المسئوليات المحددة لأنشطة الجودة مثل : الفحوصات **(Inspections)** ، والمراجعات والتدقيقات **(Reviews and Audits)** ، والاختبارات **(Tests)** بما في ذلك اختبار التحقق والمصادقة ، والتوثيق **(Documentation)** ، والتحكم في العيوب وإجراء التصحيح في حالة حدوثها.
- الخطط المفصلة للأنشطة السابقة ، والتي يجب أن تنفذ متضمنة : الجداول الزمنية ، والموارد ، وتصديق المسئولين عليها.
- إدارة المخاطر **(Risks Management)**.
- الطرق والأدوات التي يجب أن يتم توظيفها لإتمام النشاطات والمهام المتعلقة بخطة تأمين جودة البرمجيات.

ويتمثل دخل مرحلة التخطيط في :

- سياسية جودة المنظمة **(Organization 's Quality Policy)**.
- المعايير القياسية للمنظمة **(Organization Standards)**.
- المعايير القياسية الصناعية وثيقة الصلة بالمشروع **(Relevant Industry Standards)**.
- التنظيمات **(Regulations)**.
- أفق عمل المشروع **(Scope of Project Work)**.
- متطلبات المشروع **(Project Requirements)**.

ويتمثل خرج مرحلة التخطيط في :

- المعايير القياسية المُعرفة للمشروع (Standards Defined for the Project).
- خطة الجودة (Quality Plan).

2.1.2 نشاط تأمين جودة البرمجيات

(Software Quality Assurance Activity) :

- **مبادئ وتعريفات :**

تُعرف عملية تأمين جودة البرمجيات بأنها : منهجية مخططة ونظامية (Planned and Systematic Approach) لتقويم الجودة (Evaluation of the Quality) ، وتحديد مدى الالتزام بالمقاييس المعيارية (Standards) في إنتاج المنتج البرمجي ، وكذلك العمليات (Processes) ، والإجراءات (Procedures) الخاصة بذلك ، وتشمل ، أيضا ، عملية التأكيد من أن المعايير والإجراءات قد تم تأسيسها والالتزام باتباعها خلال دورة حياة تطوير المنتج، وتتم عملية التأكد هذه من خلال أنشطة مراقبة العمليات (Processes Monitoring) ، وتقويم المنتج (Product Evaluation) ، ومراجعة وتتبع (تدقيق) (Auditing) جميع النشطات والإجراءات التي تم اتباعها في تطوير المنتج البرمجي لتحديد مدى كفاءته لتحقيق كامل أهدافه.

ويمكن القول أيضاً إن تأمين جودة البرمجيات تعني التأكد من أن البرمجية تم إنتاجها لتلبي أهداف الجودة المحددة في خطة الجودة ، حيث تختلف هذه الأهداف من مشروع لآخر ، حيث إنه لا بد أن تكون هذه الأهداف قد تم انتقاؤها من قائمة معاملات الجودة المتعارف عليها في مجال البرمجيات، والتي تم ذكرها سابقاً، ولتحقيق هذه الأهداف فإنه لا بد أن تتم عملية الاختبارات خلا ل تطوير البرمجية للتأكد من صحة تنفيذ العملية.

- **المقاييس المعيارية وإجراءات تطوير البرمجيات :**

إن وضع المقاييس المعيارية والإجراءات لتطوير البرمجيات عملية حرجة جداً ، حيث إنها تمثل إطار العمل **(Framework)** الذي يتم من خلاله عملية تطوير المنتج البرمجي ، أي أنها تؤسس طرق وصفية **(Prescribed Methods)** لتطوير البرمجيات، وتُعد عملية توثيق المقاييس المعيارية والإجراءات والأنشطة الخاصة بها من خلال تعريفات صريحة يمكن قياس مدى الالتزام عملية مهمة جداً في عملية تأمين جودة البرمجيات

- **المقاييس المعيارية (Standards) :**

هي معايير مُؤسسة **(Established Criteria)** يتم من خلالها مقارنة المنتجات البرمجية ، وتعد أساس إدارة الجودة الفعالة ، وقد تكون هذه المعايير دولية أو قومية أو مؤسسية أو معايير مشروع، وتعرّف معايير المنتج القياسية الخصائص التي يجب أن تكون عليها مكونات البرمجية مثل أسلوب البرمجة الشائع ، وتعرف الحد أو المدى أو التفاوت لبعض الكميات المقيسة التي يمكن الرضوخ أو التسامح فيها. ومن المعايير القياسية للمنتج البرمجي :

1. **المعايير القياسية للتوثيق (Documentation Standards)** : وهي تشمل معايير بنية هيكل المستند ، ومعايير تحديثه ، ومعايير تنسيق المحتويات ، الخ
2. **المعايير القياسية للتصميم (Design Standards)** : وتشمل الشكل ومحتويات تصميم المنتج ، وكذلك الطرق التي تتم عن طريقها ترجمة متطلبات البرمجيات إلى التصميم المناسب لها.
3. **المعايير القياسية للشيفرة (Code Standards)** : وتشمل اللغة التي سيتم بها تشفير البرنامج ، والقيود المفروضة عليها ، حيث يتم فيها بيان توضيحي للهياكل القانونية الصحيحة **(Legal Language Structures)** واصطلاحات الأسلوب **(Style Conventions)** المستخدم في عملية البرمجة ، وهياكل البيانات

(Data Structures) ، وواجهات الاستخدام (Interfaces) ، وكذلك تشمل التوثيق الداخلي للشيفرة (Internal Code Documentation).

وتُعد **المعايير القياسية آيزو (ISO)** من أهم المعايير القياسية الدولية لإدارة الجودة القابلة للتطبيق على نطاق المنظمات من التصنيع إلى الخدمات الصناعية. ومن خلال المعايير القياسية أيزو يمكن تعريف نظام تأمين الجودة (Quality Assurance System) بأنه : البنية التنظيمية والمسؤوليات والإجراءات وعمليات البرمجة والموارد اللازمة لإنجاز إدارة الجودة.

ويُعد مقياس **ISO 9001** هو مقياس تأمين الجودة المطبق في هندسة البرمجيات ، حيث يتضمن هذا المقياس عشرين متطلباً يجب توفرها لكي يتحقق تأمين الجودة فعلياً ، حيث تتضمن هذه المتطلبات البنية التنظيمية ، ونظام الجودة ، وعمليات البرمجة ، وعمليات التفتيش والاختبارات ، وأعمال مراقبة المنتج ، وأعمال التصحيح والوقاية ، وعمليات التوثيق ، وعمليات التخزين والتغليف والحفظ والتسليم ، والتدريب ، وخلافه من العمليات.

ولكي تحصل شركة برمجيات ما على شهادة **ISO 9001** ، عليها أن تعتمد مجموعة من السياسات والإجراءات التي تشمل المتطلبات المطلوب تحقيقها في مقياس **ISO 9001** ، وعليها أن تثبت أنها تطبق هذه السياسات والإجراءات فعلياً. ولمزيد من المعلومات حول مقياس **ISO 9001** يمكن الرجوع لصفحات الويب المختلفة ذات الصلة بالموضوع والمنتشرة عبر الإنترنت.

- **الإجراءات (Procedures) :**

هي معايير مُؤسسة يتم من خلالها مقارنة عمليات التطوير (Development) والتحكم بالعمليات (Control Processes) ، ويتم صياغتها في خطوات واضحة يمكن أن تتبع لتنفيذ عملية ، وكل العمليات لابد أن يكون لها إجراءات موثقة، ومن الأمثلة على العمليات التي تحتاج إلى إجراءات الاختبارات والفحوصات الرسمية (Formal

.Inspections)

ولكي تكون عملية تأمين جودة البرمجية مؤكدة فإنه لابد من التخطيط لها مسبقا من خلال التخطيط الكلي للمشروع حيث يتبع مدير التخطيط للمشروع الخطوات التالية :

1. يقرر **(Decide)** : ما هي معاملات الجودة الأكثر أهمية لهذا المشروع على سبيل المثال (الاعتمادية ، قابلية الصيانة) وهي مشابهة تماما في إعداد وجبة فإن الاهتمام ينصب على النكهة والقيمة الغذائية كمتغيرات هامة.
2. **الانتقاء (Selection)** : اختيار المقاييس والطرق الموافقة للأهداف المنتقاة ، على سبيل المثال استخدام مقياس التعقيد لقياس الصيانة.
3. **التجميع (Assembles)** : تجميع ما سبق في البندين السابقين ليمثل خطة تأمين جودة للمشروع.

وعلى المؤسسات منتجة البرمجيات أن تقنع المستهلك باستخدام طرق مؤثرة في عملية الإنتاج وذكر ماهية هذه الطرق المؤثرة ، إضافة إلى عمل بيان عملي **(Demonstration)** لتوضيح هذه الطرق وتأثيرها ، وفي هذه الحالة يمكن القول إن تأمين جودة البرمجية لا تشمل فقط تنظيم عملية إنتاجية البرمجية وفقا لعوامل جودة المشروع ، ولكن مع بيان كامل موثق لها.

- **أنشطة تأمين جودة البرمجيات (Software Quality Assurance Activities) :**

يوجد العديد من الأنشطة التي يجب أن يقوم بها فريق تأمين جودة البرمجيات لتطوير منتج عالي الجودة ، حيث تهتم هذه النشاطات بالتخطيط لضمان الجودة ، والتقويم ، والمراقبة ، ومتابعة السجلات ، والتحليل وإصدار التقارير ، وإجمالاً يمكن ذكر أهم هذه النشاطات فيما يلي:

1. تقويم المنتج **(Product Evaluation)** : وهو عبارة عن أحد أنشطة تأمين جودة

البرمجيات حيث يتم فيه التأكد من أن المعايير القياسية والإجراءات قد تم اتباعها بصورة صحيحة وتتوافق مع المعايير القياسية التي تم تحديدها في وثيقة خطة الجودة ، وأن المنتج يعكس طبيعة المتطلبات التي قد تم وصفها في خطة المشروع.

2. مراقبة العمليات **(Processes Monitoring)** : عبارة عن أحد أنشطة تأمين جودة البرمجيات حيث يتم فيه التأكد من أن الخطوات المناسبة لتنفيذ العمليات (الإجراءات) قد تم إتباعها بصورة صحيحة وتتوافق مع الإجراءات التي تم تحديدها في وثيقة خطة الجودة.

3. التدقيق والمراجعات **(Auditing)** : وهي تُعد من الأنشطة المهمة لضمان جودة البرمجيات حيث تنظر بعمق داخل العمليات والمنتج واستخلاص المعايير القياسية والإجراءات التي تم اتباعها ومقارنتها بما هو مدون بخطة تأمين جودة البرمجيات. أي أن الغرض الأساسي لعملية التدقيق والمراجعات هي مراجعة المنتج البرمجي ، والتأكد من أن المقاييس المعيارية والإجراءات المعتمدة ونقاط تأمين الجودة قد تم اتباعها وأخذها في الاعتبار ، وإرسال تقرير دوري بالنتائج لمدير المشروع.

3.1.2 نشاط مراجعات الجودة

(Quality Reviews Activity)

هي آلية لتفحص كامل وبدقة بالغة كل نظام البرمجيات ومستندات التوثيق المرتبطة به ، حيث يقوم مجموعة من الأشخاص بمراجعة الشيفرة والتصميم والمواصفات وخطط الاختبار والمعايير القياسية ، بهدف اكتشاف خلل أو عدم تماسك ومتانة النظام ، ويتم تودين استنتاجات هذه المراجعات وإرسالها لمؤلفها أو للشخص المسئول عن تصحيح هذه الأخطاء (إن وجدت). ويجب أن يكون فريق المراجعة صغيراً ، ويتضمن أعضاء من فريق المشروع قد تكون لهم مساهمة فعالة ، فمثلاً في حالة مراجعة تصميم لفرعية معينة يجب أن يكون المهندسون الذين صمموا هذه الفرعية من البرنامج من ضمن أعضاء الفريق ، فقد يساهمون

برؤى مهمة للواجهات البينية لهذا الجزء وقد لا يتم اكتشافها إذا اعتبر الجزء كنظام قائم بذاته، ويجب أن تكون المراجعات مبنية على مراجعة مستنديه وليست مقتصرة على المواصفات أو التصاميم أو الشيفرة فقط ، حيث إنه يجب أن تتم مراجعة وثائق المشروع مثل نماذج العمليات ، وخطط الاختبار، وإجراءات إدارة التصميم ، ومعايير العمليات وكتيبات التشغيل.

وهناك العديد من أنماط المراجعة التي يمكن القيام بها في إطار هندسة البرمجيات، ومن أهم هذه الأنماط المراجعات التقنية ذات الصفة الرسمية والتي تتم بناءً على طلب رسمي يتناول تصميم البرمجية بحضور العملاء وكوادر تقنية وإدارية.

وتُعد المراجعات التقنية الرسمية **(Formal Technical Review "FTR")** أحد الأنشطة المتعلقة بتأمين جودة البرمجيات ، كما تعتبر آلية فعالة لتحسين جودة المنتج البرمجي ، وذلك من خلال تحقيق الأهداف التالية :

- كشف الأخطاء الوظيفية والمنطقية بالبرمجية.
- حصر التحسينات التي يجب إدخالها على المنتج لزيادة جودته.
- تثبيت أجزاء المنتج التي لا تحتاج أية تحسينات.
- التحقق من تلبية البرمجية للاحتياجات المطلوبة منها.
- التأكد من توافق البرمجية مع المعايير القياسية المُعرَّفة سلفاً.
- توفير التناسق والانسجام في عملية تطوير البرمجية.
- جعل المشاريع أكثر قابلية للإدارة.

وبصفة عامة يوجد عدة أنواع من المراجعات **(Types of Reviews)** منها :

- فحوصات التصميم أو البرنامج **(Design or Program Inspection)** لكشف الأخطاء التفصيلية في التصميم أو في الشيفرة ، وفحص ما إذا كان قد تم اتباع المعايير القياسية ، وتتم المراجعات بدلالة قائمة فحص تتضمن الأخطاء الممكنة.

- مراجعات تقدم سير المشروع **(Project Progress Reviews)** لتوفير معلومات عن مدى تقدم سير المشروع للإدارة المعنية ، حيث يتم مراجعة العمليات والمنتج في نفس الوقت ، وتهتم بالتكلفة والخطط والجداول الزمنية.
- مراجعات الجودة **(Quality Reviews)** للقيام بتحاليل فنية لمكونات المنتج أو الوثائق لاكتشاف الأخطاء (الاختلافات) بين المواصفات وبين تصميم المكونات ، وقد تهتم بموضوعات الجودة مثل البعد عن المعايير القياسية أو سمات الجودة الأخرى.

ولكن ما هي نتائج المراجعات التقنية الرسمية؟ .

في نهاية المراجعات يجب على أعضاء فريق المراجعات أن يقروا إحدى القرارات التالية :

- قبول المنتج دون تعديلات إضافية.
- رفضه بسبب أخطاء مؤثرة (وهذا يستلزم مراجعة جديدة بعد إصلاح هذه الأخطاء).
- قبول المنتج مؤقتاً بشرط تصحيح الأخطاء القليلة التي اكتُشفت ، ولكن بدون أن يستلزم ذلك إعادة المراجعة.

2.2 أهمية إدارة جودة البرمجيات

:(Software Quality Management Importance)

تتركز أهمية إدارة جودة البرمجيات في تزويد الإدارة بالبيانات الكافية لتكون على دراية بجودة منتجها ، وبهذه الطريقة يتسنى للإدارة الرؤية العميقة والثقة بأن منتجها عن مستوى طموحها ، وفي حالة وجود أي مشاكل في عملية تأمين جودة المنتج سيكون دور الإدارة مواجهة تلك المشاكل وتخصيص الموارد اللازمة لحلها وصولاً إلى الجودة المرجوة ، هذا بالإضافة إلى أن إدارة جودة البرمجيات تقلل من تكلفة فشل النظام **(System Failure Costs)**.

ويمكن تقسيم تكلفة فشل النظام إلى تكلفة إخفاق داخلية وأخرى خارجية. فالتكلفة الداخلية هي التي تتكبدها الشركة أو المؤسسة المنتجة عند اكتشاف العيوب في المنتج قبل شحنه للعميل ، وتشمل تكلفة :

- إعادة التشغيل.
- أصلاح العيوب.
- تحليل أنماط الفشل.

أما تكلفة الفشل الخارجي فهي التكلفة المتعلقة باكتشاف العيوب بعدما أصبح المنتج بين يدي العميل ، ويمكن أن تتضمن تكلفة :

- معالجة الشكاوى.
- إصلاح العيوب.
- إعادة المنتج أو استبداله.
- الدعم التقني المستمر.
- أعمال الكفالة.

وبصفة عامة تتضمن تكلفة الجودة **(Cost of Quality)** جميع تكلفة النشاطات المتعلقة بالجودة ومتابعتها، ويمكن توزيع تكلفة الجودة على ثلاث نواحي هي:

1. **تكلفة الفشل** : تم مناقشتها أعلاه ويمكن حذفها في حالة عدم وجود عيوب في المنتج قبل شحنه للعميل وبعده.
2. **تكلفة منع الأعطال** : وتتضمن تكلفة مايلي :
   - تخطيط الجودة.
   - المراجعات التقنية الرسمية.
   - أدوات الاختبار.
   - التدريب.
3. **تكلفة التقويم** : وتتضمن تكلفة النشاطات اللازمة للحصول على رؤية عميقة

لحالة المنتج ، ومن أمثلة تكلفة التقويم مايلي:

- الفحوصات المختلفة.
- معايرة التجهيزات وصيانتها.
- الاختبارات.

3. مقاييس البرمجيات

(Software Metrics)

إن الطريقة المنطقية الوحيدة لتحسين أي عملية برمجة هي قياس السمات المميزة لها ، وتطوير مجموعة من المقاييس **"Metrics"** ذات مغزى تستند إلى هذه السمات ، ومن ثم استخدام هذه المقاييس في توفير مؤشرات تقود إلى وضع استراتيجية للتحسين.

وبديهياً ، يمكن تخمين أن مقاييس البرمجيات **"software metrics"** متعلقة بالأعداد وقياس السمات المختلفة لعملية تطوير البرمجيات ، ولكن لإعطاء فكرة دقيقة عما تعنيه مقاييس البرمجيات فإننا نحتاج إلى تعريف واضح ، وإذا رجعنا إلى ما تم نشره بهذا الخصوص فسنجد فيه العديد من التعريفات والتي تحمل في معظم الأحيان نفس المضمون.

1.3 تعريف مقاييس البرمجيات

(Definition of Software Metrics)

لقد أخذ تعريف مقاييس البرمجيات أشكالاً متعددة منذ أن استهل ، نذكر منها :

- التعريف الذي اقترحه نورمان فونتن [6,7] حيث عرف مقاييس البرمجيات بأنها تعبير متراكم **"Collective Term"** يستخدم لوصف المدى الواسع للأنشطة المتعلقة بالقياسات **"Measurements"** في هندسة البرمجيات. وهذه الأنشطة تتدرج من إنتاج

الأعداد التي تصف سمات مصدر شيفرة البرمجية **"Software Source Code"** (وهي مقاييس البرمجة الكلاسيكية) إلى النماذج التي تساعد في التنبؤ بمتطلبات موارد البرمجية **"Software Resource Requirement"** وجودتها ، وكذلك السمات الكمية **"Quantitative Aspects"** الخاصة بتأمين الجودة ، وهذا يغطي أنشطة مثل تسجيل ومراقبة العيوب أثناء التطوير والاختبار.

- التعريف الذي اقترحه بول غودمان [6,8] حيث عرف مقاييس البرمجيات بأنها : تطبيق مستمر لتقنيات أساسها القياس **"Measurement-Based Techniques"** على عملية التطوير وما تنتجه من منتجات للحصول على معلومات إدارية ذات فائدة ومغزي بصفة دورية ، وذلك لتحسين العمليات ومنتجاتها.
- ومن منطلق أن الغرض الأساسي من استخدام مقاييس البرمجيات هو اتخاذ القرارات **"Decision Making"** والذي تم التأكيد عليه من قبل جرادي [9,10] حيث أشار إلى أن مقاييس البرمجيات تستخدم لقياس صفات معينة **"Specific Attributes"** لمنتج البرمجية أو عملية تطويرها للمساعدة في اتخاذ قرارات أفضل.
- أيضاً يمكن تعريف مقاييس البرمجيات بأنها [9]: الممارسة الفعلية لقياس الخصائص المختلفة لعملية تطوير البرمجية والمنتجات البرمجية للحصول على معلومات مفيدة وذات علاقة، لجعل العملية الإدارية تتم بكفاءة خلال كامل عملية التطوير.
- ووفقاً لمسرد تعريفات معهد المهندسين الكهربائيين والإلكترونيين"IEEE" الخاص بهندسة البرمجيات ، فإن مقاييس البرمجيات هي قياس كمي **"Quantitative Measure"** للدرجة التي يمتلك فيها نظام، أو مكون، أو عملية برمجة ما سمة معينة.

من خلال ما سبق يمكن تعريف مقياس سمة معينة **"Specific Attribute Metric"** لنظام ، أو مكون ، أو عملية برمجة بأنه : مؤشر كمي **"Quantitative Indicator"** أو مجموعة من المؤشرات الكمية الناتجة عن تحليل مجموعة من

البيانات المجمعة عن طريق الممارسة الفعلية لقياس هذه السمة على امتداد فترات طويلة من الزمن والربط بينها ، بحيث توفر هذه المؤشرات نظرة ثاقبة لمدير المشروع أو الممارسين ، أو مهندسي البرمجيات لاتخاذ قرار سليم لتحسين هذه السمة ، وإجراء المقارنات الموضوعية بين التقنيات والعمليات المستخدمة في عملية البرمجة، وبالتالي تحسين جودة المنتج أو العملية البرمجية على المدى القصير والبعيد.

فمثلاً يمكن أن نشتق مقياس لسمة مدى فعالية عملية البرمجية **"Programming Process Efficacy Attribute "** ، وذلك من خلال قياسات الأخطاء المكتشفة قبل إصدار البرمجية ، والعيوب المسلّمة للمستخدمين والتي يبلّغون عنها بعد إصدارها ، ومدى مطابقة البرنامج التنفيذي لتطلعات العميل ، والجهد الإنساني المستهلك ، والوقت المستهلك ، .... وغيرها من القياسات ، ومن ثم تحليل هذه القياسات والربط بينها للحصول على مقياس يتضمن مؤشرات كمية تُمكن المديرين والممارسين ومهندسي البرمجيات من اتخاذ القرار الصائب.

أسئلة تقويم ذاتي

؟	لقد أخذ تعريف مقاييس البرمجيات أشكالا متعددة منذ ان استهل ما هو التعريف الذي يمثل خلاصة هذه التعريفات إجمالا من وجهة نظرك؟

2.3 لماذا مقاييس البرمجيات؟ Why Software (Metrics?)

تسمح لنا مقاييس البرمجيات بتفهم بيئة تطوير البرامج بشكل أفضل ، وتعطي معلومات مفصلة حول المشروع، وتستخدم المقاييس للتوقع ، وللإدارة ، ولتنظيم المنتج. ومع المقاييس الجيدة يمكن التخطيط للمشروع بطريق صحيحة، هذا وتبين

مقاييس البرمجيات الخطوط العريضة المساعدة التي توضح أين نحن الآن وإلى أين نتجه خلال عملية التطوير، ويمكن تلخيص بعض النقاط التي توضح لماذا نحتاج مقاييس البرمجيات كما يلي [13] :

- بدون قياس عملية البرمجية أو جودة المنتج النهائي فإن التقدير الشخصي فقط محتمل ، وهو ليس مرغوباً فيه.
- بالمقاييس المتينة فإن المتطلبات يمكن أن يتم تعضيدها ، وبالتالي فإن أخطاء المكونات يمكن تشخيصها في المراحل المبكرة ، وبذلك يمكن تحسين أو ضمان تأمين الجودة.
- التنبؤ بمصدر المتطلبات فائدة أخرى مهمة لمقاييس البرمجيات.

وطريقة أخرى للإجابة عن السؤال هو تحديد المتاعب والمشاكل التي يمكن أن تنشأ في حالة عدم استخدام المقاييس في المشاريع البرمجية، وهنا نعرض ثلاث صعوبات تواجه المطورين والمدراء الذين لا يستخدمون مقاييس البرمجيات :

1. لا يمكنهم وضع أهداف يمكن قياسها للمنتج ، حيث إنهم لا يعرفون هل تم الوصول إليها أم لا ، وعلى سبيل المثال يمكن أن يَعدوا بأن المنتج سيكون سهل الاستخدام ، ويمكن الاعتماد عليه ، وسهل الصيانة، وحيث إنهم ليست لديهم تعريفات لهذه المقاييس فإن وصولهم لهذه الأهداف لا يمكن معرفته.
2. لا يستطيع المديرون قياس مكونات التكلفة ، وبالتالي فإنه من شبه المستحيل تقدير التكلفة الكلية ، وبذلك لا يمكن إعطاء تقدير دقيق للعميل.
3. المطورون والمديرون يخفقون في تحديد أو توقع جودة المنتج الذي يتم إنتاجه ، وبالتالي إذا احتاج العميل إلى معرفة مدى موثوقية المنتج أو كم العمل اللازم لتغيير المنتج ، فإنهم لا يستطيعون تزويده بإجابة.

ويمكن تلخيص أهمية مقاييس البرمجيات في أنها :

1. تساعد فيما يجب عمله خلال عملية التصميم ، وترشدنا إلى اختيار برمجية واضحة ، وبسيطة ، وسهلة التعامل ، وموثوق بها.
2. تقودنا إلى طريقة جيدة لقياس جودة البرمجة.
3. تساعدنا في التنبؤ بالجهد اللازم للتطوير.
4. تساعدنا في المفاضلة بين الطرق المختلفة لتطوير البرمجية.
5. تساعدنا في تحسين الممارسات العملية.

أسئلة تقويم ذاتي

؟ وضح اهم النقاط التي توضح لماذا نحتاج لمقاييس البرمجيات.

3.3 خصائص مقاييس البرمجيات الجيدة

(Characteristics of Good Software Metrics) :

المقاييس الجيدة لا بد أن تسهل عملية تطوير النموذج القادر على توقع العملية أو متغيرات المنتج ، ليس فقط بالوصف ولكن يجب أن تشمل النقاط التالية :

- أن تكون سهلة ، مُعرفة بالضبط ، وبالتالي يكون من الواضح كيفية تقويم المقاييس.
- تكون موضوعية **"Objective"** إلى أقصى حد ممكن.
- يمكن الحصول عليها بسهولة وبتكلفة معقولة.
- قابلية للتطبيق بفاعلية ، يجب أن تقيس المقاييس الهدف المطلوب قياسه.
- متينة **"Robust"** ، لا تكون حساسة نسبياً أمام التغييرات البسيطة للعملية أو المنتج.
- يجب أن تشمل على قيم للبيانات الملائمة للكميات المقيسة.

أسئلة تقويم ذاتي

؟ ماهي أهم خصائص مقاييس البرمجيات الجيدة؟.

4.3 تصنيف مقاييس البرمجيات [9 , 14]

(Classification of Software Metrics)

ووفقاً لما أورده إيفرالد [14] فإن مقاييس البرمجيات يمكن تقسيمها إلى : مقاييس للمنتج **"Product Metrics"** ومقاييس للعملية **"Process Metrics"**.

- مقاييس المنتج "Product Metrics" :

مقاييس المنتج تقيس البرمجية كمنتج ابتداءً من مراحله الأولية وحتى تسليمه للعميل ، وهي تتعلق كذلك بسمات : شيفرة المصدر **"Source Code"** ، وقياس المتطلبات **"Measure the Requirements"** ، وحجم البرنامج **"Size of Program"**، والتصميم **"Design"** وخلافه، وإذا ما تم تعريف مقاييس المنتج في المراحل الأولى من عملية التطوير فإن هذا سيكون مساعداً إلى حد كبير في التحكم في عملية تطوير المنتج. وسوف نتناول بعض من مقاييس المنتج ، مثل : مقاييس الحجم**"Size Metrics"** ، ومقاييس التعقيد **"Complexity Metrics"** ، ومقاييس هالستيد للمنتج **"Halstead's Product Metrics"** ، ومقياس الجودة **"Quality Metrics"** ، وذلك على سبيل المثال.

- مقاييس الحجم **"Size Metrics"** :

تُعد مقياس حجم البرمجية من أبسط المقاييس حيث يمكن حسابها كميا بطريقتين :

الأولى : (الحجم بعدد البايت) : ويرتبط هذا بذاكرة المعالج وحجم القرص الصلب مما يؤثر تلقائيا على أداء الحاسب.

الثانية : الحجم بعدد الجمل ، وذلك عن طريق قياس حجم البرمجية عن طريق قياس عدد سطور الشيفرة **"LOC"** ، أو حساب عدد نقاط الوظيفة **"FP"** " ، أو باستخدام طرق أخرى كما أوضحنا ذلك في الوحدة الخامسة. ويتم تحديد حجم البرمجية في المرحلة المبكرة للمشروع مما يكون له فائدة كبيرة ، حيث يرتبط هذا بجهد التطوير وتكاليف الصيانة.

◈ مقاييس التعقيد **"Complexity Metrics "** :

في الأيام الأولى للبرمجة كانت الذاكرة الرئيسية للحاسبات صغيرة وكانت المعالجات بطيئة للغاية وكان الهدف هو إنتاج برمجيات تعمل بكفاءة مما جعل فئة المبرمجين تلجأ للحيل **"Tricks"** للتغلب على هذه الصعاب مما زاد تعقيد إنتاج هذه البرامج ، أما في هذه الأيام فقد تغير الوضع حيث أصبح التركيز على خفض زمن تطوير البرمجية وسهولة صيانتها وأصبح التركيز كذلك على كتابة برامج سهلة ، ومفهومة ، وواضحة ، وبسيطة ، مع إمكانية تعديلها. وبالتالي يصبح من السهل اختبارها ، وفهمها وكذلك تعديلها.

ولكن ما هي أهم السمات الأساسية للبرامج البسيطة؟ :

- سهولة اكتشاف الأخطاء فيها.
- أسرع في الاختبار.
- يمكن الاعتماد عليها.
- أسرع في إجراء التعديلات عليها.

وبرغم المميزات سابقة الذكر للبرمجيات البسيطة إلا أن كثيرا من مصممي البرمجيات غالبا ما يسلكون سلوكاً معاكساً تماما حيث يبتعدون تماما عن الحلول

البسيطة بغرض الكسب من تعقيدات برامجهم، إلا أن هذا لا يعني أن هناك كثيرا من محللي ومصممي البرمجيات في هذه الأيام يبذلون قصارى جهدهم لتطوير برمجيات تتميز بكل من الوضوح والبساطة. وقد ينهي المبرمج تطوير البرمجية معتقدا أنها تعمل بصحة ومكتوبة بوضوح ولكن كيف نعرف أنها مكتوبة بوضوح؟ هل البرنامج القصير يلزم بأن يكون أبسط من البرنامج الطويل لنفس الهدف؟ هل البرمجية المتداخلة في كتابتها أبسط من البرمجية المعدة بدون تداخل؟ هذه أسئلة يهتم بها الناس وحتى الآن ليس لها إجابات محددة واضحة.

وللوصول إلى تعريف مقياس مفيد لقياس سمة التعقيد دعنا نضع مجموعة من الفروض التي تؤثر في تعقيد برنامج معين فمنها على سبيل المثال طول البرنامج ، وعدد المسارات خلال البرنامج ، وأماكن استقبال البيانات ، حيث يمكن استخدام هذه العوامل في الحصول على معادلة لتعريف سمة التعقيد بحيث يمكن اختبار هذه المعادلة ودراسة مدى خرج هذه المعادلة والوقت المأخوذ لكتابة أو فهم برنامج ، حيث إنه باستخدام مقاييس التعقيد يمكن إدارة وتنظيم عملية تطوير البرمجية وقياس مدى التعقيد فيها ، ومن أشهر هذه المقاييس :

- مقياس تعقيد الانحناءات **"Cyclomatic Complexity Metric"** : لأي برمجية يمكن أن ترسم له خريطة تدفق التحكم البيانية **"Control Flow Graph"** (G) ، حيث تمثل كل عقدة **(Node)** فيه كتلة من الشيفرة المتسلسل **"Block of Sequential Code"** ، وكل قوس يمثل تفرع أو نقطة قرار **"Decision Point"** في البرنامج ، وباستخدام نظرية الرسم يمكن اكتشاف مدى تعقيد البرنامج ، والذي يقيس مباشرة عدد المسارات الخطية غير المعتمدة **"Linearly Independent Paths"** بشيفرة البرنامج **"Program's Source Code"** ، وذلك وفقاً للعلاقة الرياضية التالية :

$$V(G) = E - N + 2$$

حيث **V(G)** تمثل مدى تعقيد خريطة تدفق تحكم البرنامج **(G)** ، **"E"** تمثل عدد

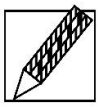
المسارات القوسية "Arcs" (أسهم التوصيل) بخريطة التدفق ، و "N" عدد العقد بالرسم البياني.

وللبرامج الهيكلية "Structured Program" يمكن حساب مدى تعقيد خريطة تدفق تحكم البرنامج V(G) مباشرة بحساب عدد نقاط القرارات في جسم البرنامج ، وذلك طبقاً لاقتراح ماكابي لسمة التعقيد "McCabes Cyclomatic Complexity" [15] ، حيث اقترح بأن سمة التعقيد لا تعتمد على عدد الجمل ولكن تعتمد اعتماد مطلقاً على هيكلية جمل القرارات (Decision Statements Structure) في بنية البرمجية (مثل عدد جمل : If و While ، والجمل المشابهة). ولحساب التعقيد وفقا لهذه الفرضية يكون رقم التعقيد = عدد الحلقات المغلقة + 1

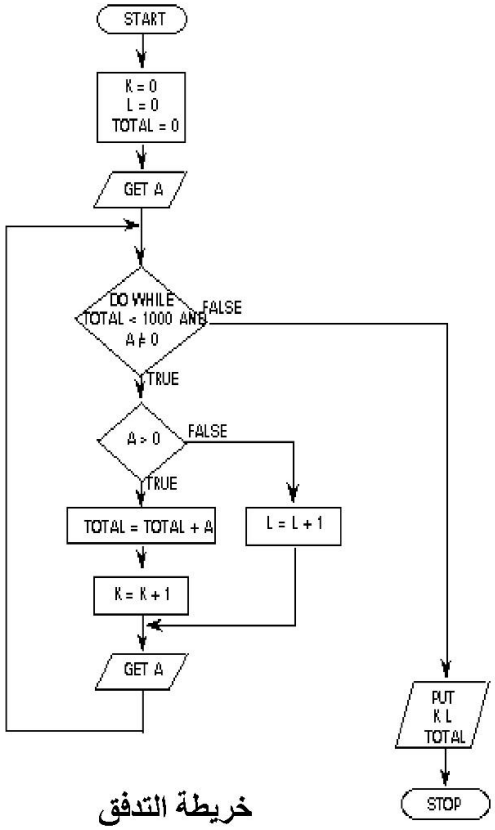
(Cyclomatic Number) = "No. of Closed Loops" + 1
Or = " No. of Decision Points " + 1

ومقياس ماكابي لتعقيد الانحناءات "McCabe's Cyclomatic Complexity Metric" يُعد مؤشراً للجهد المبذول في عملية البرمجة "Programming Effort" ، **وللجهد المطلوب لعملية الصيانة "Maintenance Effort"** ، ولأداء عملية فحص وتحديد موضع الأخطاء وتصحيحها في شيفرة البرنامج (تنقيح البرنامج) " Debugging Performance ".

تدريب (3)



موضح أدناه خريطة التدفق لبرنامج ما ارسم خريطة تدفق التحكم الخاصة به ، ومن ثم أوجد درجة تعقيد البرنامج طبقاً لمقياس ماكابي (McCabe).



خريطة التدفق

Flowchart

وهناك طريقتان لاستخدام مقياس ما كابي لدعم تطوير البرمجيات :

الأول : إذا كان لدينا خوارزميتان **"Two Algorithm"** لحل نفس المشكلة فإنه يمكن استخدام هذا المقياس لاختيار الخوارزمية الأبسط والأبسط.

الثاني : على حسب فرضية ماكابي إذا كان مقياس التعقيد لجزئية يزيد عن 10 فإنه

يعتبر كبير جدا ، وفي هذه الحالة يقترح ما كابي إعادة كتابة البرمجية مرة أخرى أو يمكن تقسيمها إلى عدة أجزاء صغيرة.

إلا انه يعاب على المقياس السابق النقاط التالية :

1. لماذا تم اختيار الرقم 10 كحد أقصى أو كحد نهائي لقياس التعقيد؟ ما هي الأسس التي تم عليها اختيار هذا الرقم.
2. لم يضع المعيار حد محدد لطول البرمجية ، حيث إنه وفقاً لهذا المقياس فإن تعقيد برمجية طولها صفحة واحدة بدون قرارات يساوي تعقيد برمجية طولها العديد من الصفحات المماثلة.
3. يعتمد المقياس كلية على تنظيم السريان للجمل (جمل التوجيه) مهملا البيانات وإدخالها ، فهناك برنامج يعتمد على مداخل قليلة للبيانات مباشرة وآخر عكسه تماما يحوي مداخل كثيرة للبيانات من مداخل غير مباشرة.

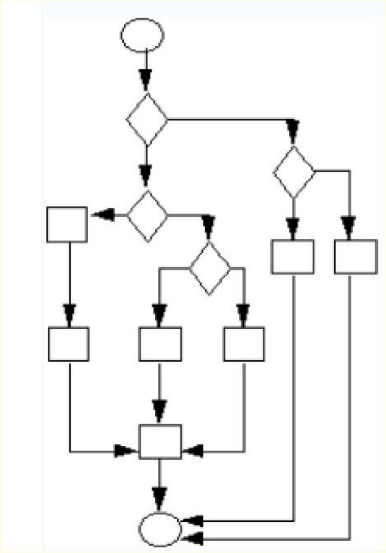
ومن هنا يتضح إهمال قياس ماكابي لعوامل أخرى يمكن أن تؤخذ في الاعتبار بغرض تحسينه ، إلا أنه وحتى الآن يُعد مقياساً مهماً جدا وله تأثير فعال ويمكن الانطلاق منه لتطوير معادلات أكثر دقة أخذه في الاعتبار العوامل الأخرى التي تم الإشارة لها.

ويمكننا القول إن وجود مقياس لسمة التعقيد يساعد مطوري البرمجيات في الحيثيات التالية :

- تقدير أو تخمين الجهد اللازم لعمل جزئية.
- اختيار أبسط التصميمات من التصميمات المتاحة والمرشحة.
- إرسال إشارة حينما تكون هناك جزئية مركبة والتفكير في إمكانية تجزئتها أو تقسيمها.

تدريب (4)

الشكل التالي يوضح خريطة تدفق التحكم لبرنامج "P" ، أحسب درجة تعقيد هذا البرنامج.



خريطة تدفق التحكم للبرنامج "P"

تدريب (5)

للإجرائية التالية ، ارسم خريطة تدفق التحكم ، ومن ثم احسب درجة تعقيدها.

```
        Procedure sort (var x: array of integer; n: integer);
        var i, j, save : integer;
        begin
(1.)        for i:=2 to n do                    (node 1)
(2.)        for j:=1 to i do                    (node 2)
(3.)            If x[i]<x[j] then               (node 3)
(4.)            Begin                           (node 4, including rows 4-7)
(5.)                save:=x[i];
(6.)                x[i]:=x[j];
(7.)                x[j]:=save;
(8.)            end                             (node 5)
(9.)    end;                                    (node 6)
```

◈ مقاييس هالستيد للمنتج **"Halstead's Product Metrics" [14, 15] :**

اقترح هالستيد مجموعة من المقاييس والتي يمكن تطبيقها على العديد من المنتجات البرمجية حيث ناقش مفردات البرنامج (n) **"Program Vocabulary"** ، وطول البرنامج (N) **"Program Length"**، وكذلك حجم البرنامج (V) **"Program Volume"** ، وذلك طبقاً للعلاقات الرياضية التالية :

- مفردات البرنامج = $n1 + n2 = n$
- طول البرنامج = $N1 + N2 = N$
- حجم البرنامج = $N \text{ Log}_2 n = V$ ، وهو يمثل قياس حجم الذاكرة المطلوب (مقيساً بالبايت) لتخزين البرنامج .

حيث إن :

n1 : عدد العوامل المفردة **(Unique Operators)** في البرنامج ، حيث يمكن أن تكون هذه العوامل حسابية أو منطقية ، تلك التي تستخدم في عملية البرمجة ، مثل الموضحة بالجدول رقم 8.1.

جدول رقم 8.1 : عوامل هالستيد (Halstead' Operators)

مثال	وصف العوامل
* , + , - , / , ( ) , { } , = , > , < , ...	العوامل الحسابية والمنطقية Mathematical and Logical Operators
If , goto , case, while , ...	الكلمات المحجوزة Reserved Words
Const , friend , volatile , Double , ...	محددات النوع Type Qualifiers
Static , Typedef, ...	مخصصات طبقة الذاكرة Storage Class Specifiers

n2 : عدد المعاملات المفردة (Unique Operands) في البرنامج ، حيث يُعرف المعامل (Operand) بأنها هدف العملية الحسابية أو المنطقية أو التعليمية البرمجية ، فعلى سبيل المثل : في الجملة X = 2 + 3 : X ، 2 و 3 تُعد معاملات ، = و + هما عاملان.

N1 : عدد العوامل الكلية (Total Number of Operators) في البرنامج .

N2 : عدد المعاملات الكلية (Total Number of Operands) في البرنامج .

تدريب (6)

أوجد حجم الذاكرة المطلوب لتخزينه لتخزين المقطع البرمجي التالي

```
if (k < 2)
  {
      if (k > 3)
      x = x*k;
  }
```

◈ مقاييس الجودة **"Quality Metrics"** :

تعتمد جودة البرمجيات على عدة سمات منها : الصحة ، والكفاءة ، وإمكانية الحمل ، وإمكانية الصيانة ، والاعتمادية ، والمتانة ، وغالباً ما يتم عمل مقاييس الجودة خلال كل مرحلة من دورة التطوير ، ومن هذه المقاييس :

1. مقاييس الكشف **"Detect Metric"** : عدد الأخطاء في المنتج البرمجي يجب أن يكون قابلاً للاشتقاق بسهولة ، ويمكن اعتباره مقياساً للمنتج ، وقد اقترح إيفرلاد [14] بعض الإجراءات التي تتبع لحساب العيب في المنتج منها :

- عدد تغييرات التصميم.
- عدد الأخطاء التي تم كشفها أثناء فحص الكود.
- عدد الأخطاء التي تم كشفها أثناء اختبارات البرنامج.
- عدد تغييرات الكود المطلوب.

وباستخدام قيم هذه المقاييس يمكن التنبؤ بجودة المنتج.

2. مقاييس الموثوقية (الاعتمادية) **"Reliability Metrics"** : تمثل موثوقية البرمجيات أحد العناصر الأساسية في جودته الإجمالية ، إذ لا يهمنا كثيراً أن تستوفى البرمجية بعض شروط الجودة المختلفة إن كانت تُخفق في أداء المهمة المطلوبة منها إخفاقاً متكرراً. وموثوقية البرمجية قابلة للقياس والضبط ، وتقديرها ممكن باستخدام البيانات التاريخية ، وتلك التي تتراكم أثناء عملية التطوير. ويمكن تعريف موثوقية برمجية ما بأنها [1] : "احتمال أن يعمل البرنامج بدون عطل في بيئة محددة وخلال زمن معين". لتوضيح ذلك لنفرض أن موثوقية البرنامج **"P"** تُقدر بـ 96% لعمل مدة 8 ساعات ، فهذا يعني أنه إذا قمنا بتشغيل البرنامج **"P"** مئة مرة ، وكانت مدة المعالجة المطلوبة تمتد إلى ثماني ساعات ، فإن من المتوقع أن يجري التنفيذ بدون أعطال في 96 مرة من المرات المئة.

ويُعد معامل قياس كثافة الأخطاء (Fault Density "FD" Parameter) من مقاييس الموثوقية التي تعتمد على احتساب نسبة العيوب في ألف سطر شيفرة (Defects/KLOC) ، حيث تكتشف الأخطاء خلال التحقق والاستخدام العادي للبرمجية، وهناك بعض الأخطاء يمكن تصحيحها ولهذا لا تدخل في حساب المعامل "FD". وهناك أخطاء أخرى في البرمجيات لا تكتشف ويُعد الحد (2 – 50) مقياساً مناسباً لكل KLOC والحد 2 بالضبط الحد الأمثل. وهذا المقياس مفيد جدا إلى حد كبير في زيادة تأثير طرق التحقيق وقياس الصحة (Correctness) في تأمين جودة البرمجيات، ولقد أثبتت معظم الدراسات العملية أن معظم الأخطاء نادرا ما تسبب فشل للنظام في حين أن عدداً صغيراً من الأخطاء قد يسبب فشل النظام ، ولهذا يكون من الأجدى بذل جهد لاكتشاف هذا العدد الصغير من الأخطاء.

إحدى التحديات الرئيسية في استخدام مقياس كثافة الخطأ هو أن بعض الأخطاء تكون واضحة خلاف الأخطاء الأخرى ، ولهذا فإن المقياس الأكثر فائدة للمستخدمين هو **مقياس المتوسط الزمني بين الأعطال** (Mean Time Between Failures (MTBF)) ، **حيث :**

$$MTBF = MTTF + MTTR \ldots\ldots\ldots\ldots\ldots (1)$$

"MTTF" : مقياس المتوسط الزمني للعطل (Mean Time to Failure (MTTF)) ، وهو يعبر عن المتوسط الزمني للنظام للأداء بدون فشل، وهذا يمكن قياسه بتسجيل الأوقات التي تحدث فيها الأعطال وحساب متوسط الوقت بين الأخطاء المتتابعة. وهذا المقياس يمكن أن يعطي تنبؤاً لمعدل الخطأ المستقبلي للنظام.

"MTTR" : مقياس المتوسط الزمني للتصليح (Mean Time to Repair (MTTF)) ، وهو يعبر عن المتوسط الزمني لإصلاح الأعطال.

إضافة إلى وجود مقياس للموثوقية ، لا بد من تطوير مقياس للتوفر (Availability) ، والذي يقيس مدى احتمالية أن تعمل البرمجية في وقت معين بشكل

مطابق للمتطلبات الموضوعة وبدون أعطال. ويتم احتساب قيمة مقياس التوفر "AV" طبقاً للمعادلة التالية :

$$AV = [MTTF / (MTTF + MTTR)] * 100\% \ldots\ldots\ldots (2)$$

نلاحظ من المعادلة (1) أن المقياس **MTBF** ذو حساسية متساوية لكل من **MTTF** ، و **MTTR** ، بينما نلاحظ من المعادلة رقم (2) أن المقياس **AV** ذو حساسية أكبر تجاه المقياس **MTTR** والذي يمكن اعتباره قياساً غير مباشر لقابلية صيانة البرمجية.

◈**مقاييس العملية (Process Metrics) :**

تقيس هذه المقاييس خواص إجراء عملية تطوير البرامج : مثل الوقت ، وعدد الأشخاص المشتركين في التطوير ، وأنواع المنهجيات. وهنا ينصب التركيز على السمات المناسبة لبناء النموذج الذي منه توقع وتنظيم خطة التطوير، وتستخدم هذه المقاييس لتحسين العملية وإدارة المتطلبات والتي يتم تنفيذها في عملية التطوير لإنتاج منتج ذي جودة عالية ويحقق رغبات العميل.

وقد قسم جرادي [11] مقاييس البرمجيات من حيث طريقة القياس إلى نوعين

◈**المقاييس البدائية "Primitive Metrics" :**

وهي مقاييس يمكن قياسها مباشرة **"Directly Measure"** ، مثل عدد سطور البرمجية **"LOC"** ، أو عدد الأخطاء في البرنامج.

◈**المقاييس المحسوبة "Computed Metrics" :**

وهي مقاييس يمكن قياسها بطريقة غير مباشرة مثل جودة المنتج **"Quality of Product"** عن طريق قياس مقاييس أخرى .

وطبقاً لما جاء في [3] يمكن أن تقسم مقاييس البرمجيات إلى مقاييس ديناميكية ، ومقاييس استاتيكية.

◈المقاييس الديناميكية "Dynamic Metrics" :

وهي مقاييس يتم تجميعها بواسطة قياسات يتم إجراؤها على البرمجية أثناء التشغيل (Execution) ، لذا فهي تتعلق كثيراً بسمات جودة البرمجيات (Software Quality Attributes) ، حيث يتم قياس زمن استجابة النظام (Response Time of a System) ، وسمة الأداء (Performance Attribute) ، وعدد الإخفاقات (Number of Failures) ، وسمة الاعتمادية (Reliability Attribute) .

◈المقاييس الاستاتيكية "Static Metrics" :

وهي مقاييس يتم تجميعها بواسطة قياسات يتم إجراؤها على البرمجية أثناء وضع عدم التشغيل ، ولذا فهي تتعلق بصورة مباشرة بتقويم سمات قابلية التعقيد وقابلية الفهم والصيانة.

هذا الجدول التالي يلخص أشهر المقاييس المستخدمة في عملية اختبار البرمجيات:

المقياس	الوصف
دليل رضاء العميل Customer Satisfaction Index	هذا الدليل يتم طرحه قبل وأثناءه وبعده تسليم المنتج وهو عبارة عن استمارة استبيان تشمل العديد من الأسئلة بغرض تحليل : • عدد طلبات تحسين النظام في السنة. • عدد طلبات الصيانة للنظام في السنة. • عدد طلبات المساعدة عن طريق الخدمة الساخنة. • وقت التدريب لكل مستخدم جديد. • عدد مرات الطرح للنظام من قبل المنتج.
كمية الأعطال المستلمة Delivered Defect Quantities	وهي تحسب لكل نقطة دالة أو لكل عدد سطور من الشيفرة في وقت التسليم (الأشهر الثلاثة الأولى أو السنة الأولى من التشغيل مع تحديد مستويات الخطورة حسب التصنيف أو السبب(عطل متطلبات– عطل تصميم – عطل مساعدة أو توثيق).

المقياس	الوصف
مدى الاستجابة Responsiveness	يعبر عنه بالوقت المأخوذ لتصحيح العطل حسب أولويته ومستوى خطورته.
عدم ثبات المنتج. Product Volatility	ويعبر عنه بالنسبة بين عدد طلبات الصيانة لجعل النظام يلبي المواصفات إلى طلبات تحسين أداء النظام
نسب الأعطال. Defect Ratios	العيوب التي توجد بالمنتج بعد تسليمه لكل نقطة داله. العيوب التي توجد بالمنتج بعد تسليمه لكل عدد سطور من الشيفرة. عيوب قبل التسليم: عيوب ما بعد التسليم السنوية.
كفاءة إزالة العيوب. Defect Removal Efficiency	• عدد العيوب ما بعد التسليم والتي تكتشف من قبل الزبائن العاملين في الحقل. • عدد العيوب التي تكتشف داخليا بالتفتيش قبل التسليم بالنسبة للعدد الكلي للأخطاء • العدد الكلي للعيوب وهو عبارة عن مجموع العيوب السابقة (الداخلية والخارجية) في السنة الأولى بعد تسليم المنتج.
تعقيد المنتج المسلم Complexity of Delivered Product)	• يستخدم مقياس ماكابي. • توقع العيوب وتكاليف الصيانة استنادا على إجراءات التعقيد.
تغطية الاختبار. Test Coverage	• النسبة المئوية للمسارات ، التفريعات ، الحالات التي تم اختبارها. • النسبة بين عدد العيوب المكتشفة إلى عدد العيوب المتوقعة.
تكلفة العيوب. Cost of Defects	• فاقد العمل لكل عطل يحدث أثناء التشغيل. • تكلفة توقف العمل • فاقد المبيعات والنية الحسنة. • تكاليف المقاضاة الناتجة من العيوب • تكلفة الصيانة السنوية لكل نقطة داله. • تكلفة التشغيل السنوية لكل نقطة داله.

المقياس	الوصف
تكلفة جودة النشاطات. Costs of Quality Activities	• تكاليف المراجعات والفحوصات والإجراءات الوقائية. • تكاليف التخطيط للاختبار والإعداد له. • تكاليف إجراء الاختبار وتتبع العطل. • تكاليف تشخيص العيوب وإصلاحها. • تكاليف الوسائل المساعدة. • تكاليف الاختبار والأسئلة المساعدة المصاحبة للمنتج.
إعادة العمل. Re-work	• جهد إعادة العمل بالساعات كنسبة من ساعات التشفير الأصلية. • عدد السطور من الكود المعادة بالنسبة لعدد السطور الكلية المسلمة. • عدد المكونات المعادة بالنسبة لعدد المكونات المسلمة.
الإعتمادية ومقاييس تقويم اختبارات تطبيق النظام. Reliability	تعرف على أنها النسبة بين الوقت الذي فيه النظام متاح إلى الوقت الذي يحتاجه النظام لكي يكون متاحاً. وتقاس بالمقاييس التالية : • متوسط الوقت بين الفشل • متوسط الوقت للتصليح • عدد مرات إعادة طلب النظام أو الإصدارات المعدلة منه.

أسئلة تقويم ذاتي

؟ إرسم جدولاً لخص فيه أشهر المقاييس المستخدمة في عملية اختبار البرمجيات.

5.3 مقياس تحديد مستوى مؤسسات تطوير البرمجيات

يتم تحديد مستوى مؤسسات تطوير البرمجيات عن طريق مقياس يحتوي على خمسة مستويات ، ويعتمد على ما يُسمى بنموذج نضج القدرة **"Capability Maturity Model (CMM)"** ، وذلك بطرح استمارة استقصاء تم تطويرها خصيصا لهذا الغرض من معهد هندسة البرمجيات **(Software Engineering Institute "SEI")** من جامعة ميلون بالولايات المتحدة **(Carnegie Mellon University "CMU")** ، ويمكن تعريف هذه المستويات كالتالي :

- **المستوى الأول (بدائي) (Initial)** : وفيه يتم إنتاج البرمجية بطريقة عشوائية بعمليات بسيطة غير منظمة حيث يعتمد نجاحها على جهد أشخاص محددين وذوي مهارات داخل المؤسسة وليس على طريقة إدارة المؤسسة للعمل.

- **المستوى الثاني (مثبت الجدارة) (Repeatable)** : وفيه تتم إدارة المؤسسة لإنتاج برمجية تراعي فيها كل من التكاليف ، الجدول الزمني والأداء وفيه يمكن أن تثبت المؤسسة نجاحها لنفس نوعية المشاريع التي تقوم بتنفيذها.

- **المستوى الثالث (معرف) (Defined)** : وفيه تتم عملية التطوير لكل من الإدارة وأنشطة هندسة البرمجيات معرفة وموثقة ويراعى فيها القياسية في عملية الإنتاج ، إضافة إلى ما تم توضيحه في المستوى الثاني.

- **المستوى الرابع (مهيأ إداريا) (Managed)** : وفيه يراعى تفاصيل القياسات لعملية التطوير للمنتج مجمعة ومحددة كميا بطريقة منظمة إداريا ، ويشمل كذلك ما تم توضيحه في المستوى الثالث.

- المستوى الخامس (الأمثل) **(Optimizing)** : وفيه تتم عمليات القياس باستمرار لتحسين أداء المنتج حيث يتم اقتراح طرق أخرى ووسائل للاختبار بغرض التحسين وكذلك ما تم طرحه في المستوى الرابع.

ويحبذ في كل الأحوال شراء البرمجيات من المؤسسات ذات المستوى الخامس والتي غالبا ما تصنع برمجيات ذات جودة عالية بأقل تكاليف.

أسئلة تقويم ذاتي

؟

يشمل مقياس تحديد مستوى مؤسسات تطوير البرمجيات خمسة مستويات، ماهي هذه المستويات، ولماذا يحبذ في كل الأحوال شراء البرمجيات من المؤسسات ذات المستوى الخامس.