دورة PDCA في إدارة كلفة مشروع

دورة PDCA (خطط، اعمل، تحقَّق، تصرَّف) في إدارة كلفة المشروع:

- **خطّط (Plan)**
  - يجري تقدير كلفة المشروع البرمجي ككل
  - اعتماداً على هذا التقدير، يجري تقسيم الكلفة بين أطوار المشروع (التحليل، التصميم، البرمجة، والاختبار مثلاً) ومن ثم وضع ميزانية من اجل كل طور
  - يمكن حساب التقدير الكلي من خلال تقدير الأطوار
- **اعمل (Do)**
  - تجميع نتيجة كلفة المشروع قيد العمل (Accumulate the cost result of the running project)
- **تحقَّق (Check)**
  - المقارنة بين قيمة الميزانية والنتيجة في ذلك الوقت
- **تصرَّف (Action)**
  - إذا كان هناك فرقاً بين الميزانية والنتيجة، سيجري تحليل الأمر واتخاذ الإجراءات المناسبة
  - إذا تجاوزت النتيجة الميزانية المقدَّرة، يجري اتخاذ الاحتياطات اللازمة للبقاء ضمن حدود الميزانية
  - قد نضطر إلى تغيير جزء الميزانية المقدَّر لكل طور، وهنا يجب على مدير المشروع التعامل مع المشكلة بدقة
  - إذا كان هناك فرق كبير بين التقدير والنتيجة، يجب التحقق من الطريقة المستخدمة لوضع التقديرات
  - بعد تحليل سبب خطأ التقدير، يجب تحسين طريقة التقدير، وذلك لاستخدامها في المشاريع المستقبلية وتحسين دقة التقدير(Estimation Accuracy)

تقسيم الكلفة الكلية للمشروع البرمجي

يمكن تقسيم الكلفة الكلية للمشروع البرمجي إلى كلفة أولية (Initial Cost) وكلفة جارية (Running Cost):

- **كلفة أولية**

هي النفقات اللازمة لتطوير النظام البرمجي المطلوب، وتتضمن كلفة البرمجيات والأجهزة وغيرها. كلفة الأجهزة تتمثَّل بنفقات شراء الأجهزة مثل المخدّمات (Servers) والحواسيب الشخصية (PC) وغيرها. أما كلفة البرمجيات فتتمثَّل بشراء منتجات معينة مثل أنظمة إدارة قواعد المعطيات (Database Management System) وغيرها من الأدوات والبرمجيات اللازمة للعمل، بالإضافة إلى نفقات تطوير البرمجيات.

- **كلفة جارية**

هي النفقات اللازمة لصيانة وإدارة النظام الذي يجري تطويره، وتتضمن كلفة صيانة الأجهزة (Hardware Maintenance) وكلفة إدارة البرمجيات (Software Administration).

يبيّن الجدول التالي كيفية تقسيم كلفة المشروع البرمجي:

كلفة أولية	**أجهزة**		مخدمّات، حواسيب شخصية، أجهزة شبكات
	برمجيات	منتجات	أنظمة إدارة قواعد المعطيات، أنظمة تشغيل، أدوات،...
		برنامج التطبيق	برنامج التطبيق
	غير ذلك		نفقات متنوعة
كلفة جارية	أجهزة		كلفة صيانة الأجهزة
	برمجيات		كلفة إدارة البرمجيات

قياسات البرمجيات

يمكن تصنيف أخذ القياسات إلى قياسات مباشرة (Direct Measurement)، مثل طول المسمار، وقياسات غير مباشرة (Indirect Measurement) مثل جودة المسمار، مقاسةً بعدد المسامير المرفوضة مثلاً. ويمكن تصنيف قياسات البرمجيات بأسلوب مشابه.

- **القياسات المباشرة للبرمجيات (Software Direct Measurement)**

تتضمن القياسات المباشرة لإجرائية هندسة البرمجيات المستخدمة في تطوير البرمجية الكلفة والجهد المطبَّقين، وتتضمن القياسات المباشرة للمنتج البرمجي الناتج أسطر الترميز (Lines Of Code LOC) المنتَجة، سرعة التنفيذ، حجم الذاكرة اللازمة للتنفيذ، والأعطال المعلن عنها خلال فترة معينة.

- **القياسات غير المباشرة للبرمجيات (Software Indirect Measurement)**

تتضمن قياسات المنتج البرمجي غير المباشرة الوظيفية (Functionality)، الجودة (Quality)، درجة التعقيد (Complexity Degree)، الفعالية (Efficiency)، الموثوقية (Reliability)، قابلية الصيانة (Maintainability) ، وغير ذلك.

- **ملاحظة**

تستخدم أحياناً كلمة مقياس (Measure) للتعبير عن نفس المفهوم، وهذا قد يختلف من مرجع إلى آخر.

من الضروري وضع أسس معينة تمكن من أخذ قياسات البرمجيات، بحيث يمكن إجراء مقارنات واستنتاجات بناءً على هذه القياسات. ولذلك تصنّف القياسات إلى قياسات تعتمد على حجم البرمجية وأخرى تعتمد على الوظيفة التي تقوم بها هذه البرمجية.

القياسات الحجميّة التوجه للبرمجيات

تُستنتج المقاييس الحجميّة التوجّه (Size-Oriented Metrics) اعتماداً على حجم البرمجية الناتجة. يبيّن الجدول التالي المقاييس الحجميّة التوجّه في مشاريع تطوير البرمجيات في منظمة ما:

المشروع	LOC	الجهد	$(000)	الصفحات/ الوثيقة	الأخطاء	العيوب	العنصر البشري
ألفا	12100	24	168	365	134	29	3
بيتا	27200	62	440	1224	321	86	5
غاما	20200	34	314	1050	256	64	6
...	...	...	...	...	...	...	...

على سبيل المثال، في المشروع ألفا، جرى تطوير 12100 سطراً من الترميز بجهد 24 شخص/شهر، وبكلفة 168000$. يجب الانتباه إلى أن الجهد والكلفة السجلين في الجدول السابق يمثّلان جميع مراحل تطوير البرمجية (التحليل والتصميم والترميز والاختبار) وليس فقط الترميز. تُشير المعلومات الأخرى عن المشروع ألفا إلى أنَّه جرى تطوير 365 صفحة من الوثائق، وتسجيل 134 خطأً قبل إصدار البرمجية، ومصادفة 29 عيباً بعد تسليم البرمجية للزبون خلال العام الأول من التشغيل. وقد عمل ثلاثة أشخاص في تطوير برمجيات المشروع ألفا.

يمكن أن نستنتج، اعتماداً على الجدول السابق، مجموعة من المقاييس الحجميّة التوجه للمشروع:

- عدد الأخطاء في كل ألف سطر من الترميز
- عدد العيوب في كل ألف سطر من الترميز
- كلفة كل سطر من الترميز
- عدد صفحات الوثائق لكل ألف سطر من الترميز

يمكن إضافةً إلى ذلك حساب مقاييس أخرى مثيرة للاهتمام:

- عدد الأخطاء لكل شخص/شهر
- عدد سطور الترميز لكل شخص/شهر
- كلفة كل صفحة وثائق

لا تعدّ المقاييس الحجميّة التوجّه عموماً أفضل طريقة لقياس البرمجيات. إذ يدور معظم الجدل حول استخدام عدد أسطر الترميز (LOC) قياساً أساسياً. يدّعي مناصرو هذا القياس أن عدد أسطر الترميز هو نتاج إنساني في جميع مشاريع تطوير البرمجيات ويمكن إحصاؤه بسهولة، وأن العديد من نماذج التقدير البرمجية الحالية تستخدم (LOC) أو (KLOC) دخلاً رئيسياً. من جهة أخرى، يدّعي المعارضون أن قياسات (LOC) تابعة للغة البرمجة المستخدمة في تطوير البرمجية، وأنها تعاقب البرامج المصممة تصميماً جيداً لكنها أقصر.

المقاييس الوظيفية التوجُّه للبرمجيات

تعتمد المقاييس الوظيفية التوجه (Function-Oriented Metrics)على قياس الوظيفة التي تقوم بها البرمجية الناتجة عن المشروع. ولما كان من غير الممكن قياس "الوظيفية" قياساً مباشراً، فلا بدّ من استنتاجها بأسلوب غير مباشر باستخدام قياسات مباشرة أخرى.

- **نقطة الوظيفة (Function Point)**

جرى اقتراح أحد المقاييس الوظيفية التوجُّه، والذي يُسمى نقطة الوظيفة (Function Point). يجري استنتاج نقاط الوظيفة باستخدام علاقة تجريبية تستند إلى قياسات (مباشرة) قابلة للعد لنطاق المعلومات البرمجية، وإلى تقييم تعقيد البرمجيات.

تُحسب نقاط الوظيفة بإكمال الجدول المبيَّن:

			عامل التثقيل Weighting Factor				
موسطات أخذ القياسات	**العدد count**		**معقّد Complex**	**متوسط Average**	**بسيط Simple**		
عدد مدخلات المستخدمين		×	6	4	3	=	
عدد مخرجات المستخدمين		×	7	5	4	=	
عدد استفسارات المستخدمين		×	6	4	3	=	
عدد الملفات		×	15	10	7	=	
عدد الواجهات الخارجية		×	10	7	5	=	
المجموع							

لنشرح مداخل الجدول السابق:

- عدد مدخلات المستخدمين

يحصى كل دخل (Input) للمستخدم يوفِّر معطيات تطبيقية التوجُّه مميزة للبرمجية، يجب التمييز بين المدخلات والاستفسارات (Inquiries) التي تحصى على حدة.

- عدد مخرجات المستخدمين

يحصى كل خرج (Output) للمستخدم يوفِّر معطيات تطبيقية التوجُّه مميزة للمستخدم. ضمن هذه السياق، يشير الخرج إلى التقارير، الشاشات، رسائل الأخطاء وغير ذلك.

- عدد استفسارات المستخدمين

يحصى كل استفسار مميَّز، ويعرَّف الاستفسار بأنه دخل آني يؤدي إلى توليد استجابة فورية من البرمجية على شكل خرج آني.

- عدد الملفات

يُحصى كل كلف رئيسي منطقي (أي تجميع منطقي لمجموعات من المعطيات التي قد تكون جزءاً واحداً من قاعدة معطيات واسعة أو ملف مستقل).

- عدد الواجهات الخارجية

تُحصى جميع الواجهات التي يمكن للآلة قراءتها (كملفات المعطيات على الأقراص) والتي تستخدم لنقل المعلومات من نظام لآخر.

تطور المنظمات التي تستخدم نقاط الوظيفة، معايير لكي تحدّد فيما إذا كان مدخل ما من مداخل الجدول السابق معقداً أو بسيطاً أو متوسطاً. ومع ذلك، فإن تحديد درجة التعقيد هو شيء شخصي إلى حدٍّ ما.

المقاييس الوظيفة التوجُّه للبرمجيات (متابعة)

- **حساب نقاط الوظيفة**

لحساب نقاط الوظيفة نستخدم العلاقة التالية:

$$FP = \text{count-total} \times [0.65 + 0.01 \times \text{SUM}(Fi)]$$

حيث التعداد الكلي هو مجموع جميع المداخل التي حصلنا عليها من الجدول السابق.

إن قيم Fi هي "قيم تعديل درجة التعقيد" وتستند إلى الإجابات على الأسئلة التالية:

1. هل يتطلب النظام إجراء نسخ احتياطي دوري واستعادة موثوقين؟
2. هل هناك حاجة إلى اتصالات المعطيات؟
3. هل تتطلب الوظائف معالجة موزعة؟
4. هل تشكّل فعالية الأداء (Efficiency of Performance) أمراً أساسياً؟
5. هل ستعمل البرمجية ضمن بيئة تشغيل موجودة وتستخدم بكثافة؟
6. هل تتطلب البرمجية إدخالا آنياً للمعطيات؟
7. هل يتطلب إدخال المعطيات المباشر أن يجري بناء مناقلات الدخل (Input Transaction) عبر شاشات أو عمليات متعددة؟
8. هل يُحدَّث الملف الرئيسي تحديثاً مباشراً (On-Line)؟
9. هل المدخلات أو المخرجات أو الملفات أو الاستفسارات معقَّدة؟
10. هل المعالجة الداخلية معقَّدة؟
11. هل تم تصميم الترميز بحيث يمكن إعادة استخدامه؟
12. هل يتضمن التصميم عمليتي التحويل (Conversion) والتجهيز (Installation)؟
13. هل تم تصميم البرمجية من أجل تجهيز متعدد (Multiple Installation) في مؤسسات مختلفة؟

14. هل تم تصميم البرمجية بحيث تُيَسِّر التغيير وتوفِّر سهولة الاستخدام من قبل المستخدم؟

تُستخدم نقاط الوظيفة فور حسابها على وجهٍ مشابه لعدد أسطر الترميز (LOC)، لتنظيم قياسات أخرى للبرمجية، مثل:

- عدد الأخطاء في كل نقطة وظيفة
- عدد العيوب في كل نقطة وظيفة
- كلفة كل نقطة وظيفة
- عدد صفحات الوثائق لكل نقطة وظيفة
- عدد نقاط الوظيفة لكل شخص/شهر

الطرق الأساسية لتقدير الكلفة في مشاريع تطوير البرمجيات

تشكّل نفقات الأشخاص المشاركين في مشاريع تطوير البرمجيات النسبة الأكبر من نفقات هذه المشاريع. لذلك يتطلب تقدير الكلفة أن يجري تقدير الجهد الذي يقوم به الشخص بشكلٍ أساسي. وهناك عدة طرق تستخدم لهذه الغاية:

- **تقدير حجم البرنامج (Program Size Estimation)**

وهي طريقة لتقدير الجهد اعتماداً على التميز المصدري للبرنامج، تعرف باسم نموذج CoCoMo (COnstructive COst MOdel). يجري تقدير الجهد اعتماداً على المواصفات الداخلية (Internal Program Specification) فقط، وبالتالي تكون دقة التقدير (Accuracy of Estimation) منخفضة في المراحل الأولى من تطوير البرمجية، وهي غير مناسبة في هذه المراحل.

- **تقدير نقطة الوظيفة (Function Point Estimation)**

يجري في هذه الطريقة تقدير الجهد اعتماداً على تعقيد البرمجية (Software Complexity)، باستخدام الوحدة "نقطة الوظيفة" (Function Point)، بالإضافة إلى استخدام مواصفات خارجية للبرنامج (External Program Specification).

- **طريقة التشابه (Similarity Method)**

يجري في هذه الطريقة تقدير الجهد والكلفة اعتماداً على النتائج السابقة لمشاريع مشابهة للمشروع الحالي. قد يكون من الصعب – أحياناً– تحديد فيما إذا كانت مشاريع قديمة مشابهة للمشروع الحالي وكذلك تقدير الفرق بينها.

- **طريقة التجميع أو المراكمة (Accumulation Method)**

يجري التقدير في هذه الحالة من خلال التحقّق من العمليات وتجميع الجهد المطلوب لها. من الضروري التحقق من جميع العمليات. تعتمد دقة التقدير على دقة العمليات التي يجري التحقق منها، لذلك من الضروري أن يجري تدقيق هذه العمليات في مرحلة مبكرة من إجرائية التطوير.

مقارنة بين طرق تقدير الكلفة

يبيِّن الجدول التالي مقارنة بين الطرق الرئيسية لتقدير الكلفة في مشاريع تطوير البرمجيات:

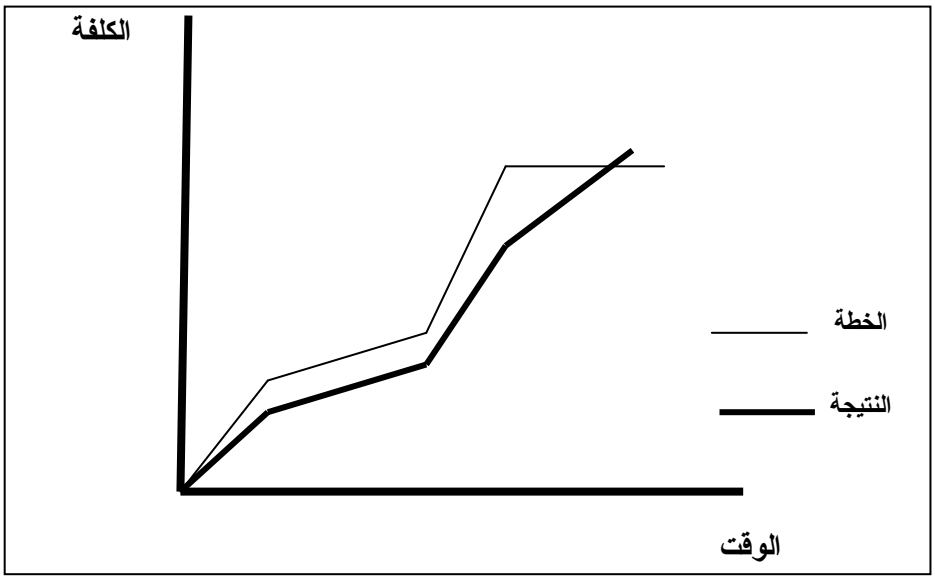
الطريقة	ملخّص	المزايا	تعليمات الاستخدام
تقدير حجم البرنامج (نموذج CoCoMo)	تقدير الجهد اعتماداً على الترميز المصدري للبرنامج.	من الممكن تقدير الجهد والمدة من التحليل/التصميم إلى الاختبار اعتماداً على امتداد البرنامج (Program Scale).	من الضروري تقدير الترميز المصدري بدقة, يكون الفرق بين التقدير والنتيجة كبيراً في المرحلة الأولى من إجرائية التطوير.
تقدير نقطة الوظيفة أو تحليل نقطة الوظيفة (FPA)	تقدير الجهد اعتماداً على الدخل والخرج أشياء أخرى.	من الممكن إجراء تقدير دقيق نسبياً، حتى في المرحلة المبكرة حيث يكون توصيف البرنامج غير مقرَّر بعد.	غير مناسبة للبرمجيات التي تعتمد كثيراً على المواصفات الداخلية مثل المنطق المعقد.
طريقة التشابه	تقدير الجهد اعتماداً على نتائج مشاريع سابقة مشابهة.	يستخدم لإجراء تقدير أولي.	ضرورية لتجميع معلومات.
طريقة التكديس (من الأسفل إلى الأعلى Bottom-Up)	تقدير الجهد من خلال التحقق من عناصر العمل وتكديس الجهد.	تعتمد دقة التقدير على دقة عملية التحقق من عناصر العمل.	ضرورية للتحقق من المهام التفصيلية في المرحلة المبكرة من إجرائية التطوير.

دقة تقدير كلفة البرمجيات

تكون المعلومات المطلوبة لإجراء تقدير الكلفة في المراحل المبكرة من المشروع غامضة عموماً أكثر من المراحل اللاحقة، ويكون الفرق كبيراً بين قيمة التقدير والقيمة الفعلية. على سبيل المثال، في التقدير المعتمد على حجم البرنامج مثل نموذج CoCoMo، يمكن حساب التقدير الدقيق في مرحلة تصميم البرنامج (Program Design) حيث يكون منطق البرنامج قد أصبح واضحاً. بينما في تحليل نقطة الوظيفة الذي يعتمد على مواصفات خارجية مثل معلومات دخل وخرج وظيفة معينة، يكون التقدير أكثر دقة في المراحل المبكرة مثل مرحلة التحليل.

يجب أن يجري، وعلى نحوٍ منتظم، التحقّق من القيم الفعلية للكلفة مقارنةً مع القيم المخطط لها، بحيث يجري تحديد فيما إذا كانت الكلفة تتغير حسب الخطة. كذلك يجب التحقّق من القيمة الفعلية لكلفة كل عنصر مقارنةً مع الكلفة المخطط لها لهذا العنصر، واتخاذ الإجراءات المناسبة عندما تتجاوز القيمة الفعلية القيمة المخطط لها.

في مشاريع تطوير البرمجيات، ستزداد الكلفة نتيجة لتوسّع نطاق البرمجية الناتج عن تغيرات المواصفات، أو نتيجة لكلف إضافية ناتجة عن تأخر في المهام. لاحظ الشكل التالي:



التحقق من الكلفة الفعلية لكل عنصر كلفة

لتحديد الكلفة الكلية، يجب التحقق من الكلفة الفعلية لكل عنصر كلفة. يمكن استخدام جدول شبيه بالجدول التالي لإدارة التفاصيل المتعلقة بكل عنصر:

			شهر نيسان		شهر أيار	
			الخطة	النتيجة	الخطة	النتيجة
كلفة أولية	أجهزة	مخدّمات، حواسيب شخصية، أجهزة شبكات	0	964	3,002	1,988
		Installation		98	200	196

	برمجيات	شراء برمجيات	1,505	1,500	0	0
		نفقات شخصية (بالساعة)	1,000	1,006	1,000	1,002
		Outsourcing	0	0	0	0
كلفة جارية	أجهزة	كلفة صيانة الأجهزة				
	برمجيات	كلفة إدارة البرمجيات				

عندما تتجاوز القيم الفعلية القيم المقدَّرة، وهذا ما هو متوقع، لن يجري إتمام المشروع ضمن حدود الميزانية إلا إذا جرى اتخاذ الإجراءات المناسبة. لذلك، من الضروري التحقق من العناصر المتعلقة بالكلفة وتحديد العوامل التي تسبب زيادة الكلفة ومن ثم اتخاذ الإجراءات المناسبة.

أساسيات نموذج CoCoMo

- **نموذج (CoCoMo 2.0)**

نموذج تقدير برمجي جديد للكلفة والجدولة الزمنية، وهو مناسب للنماذج الجديدة في تطوير البرمجيات، مثل برمجيات الأعمال (Business Software)، البرمجيات الغرضيّة التوجه (Object-Oriented Software)، نماذج التطوير التطوّرية أو الحلزونية (Spiral or Evolutionary Development Models)، وغير ذلك.

- **غايات نموذج (CoCoMo 2.0)**
  - تطوير نموذج لتقدير الكلفة والجدولة الزمنية للبرمجيات
  - تطوير قاعدة معطيات لكلفة البرمجيات وأدوات تدعم المقدرات اللازمة للتحسين المستمر للنموذج
  - لتوفير إطار عمل تحليلي كمّي (Quantitative Analytic Framework)، ووضع مجموعة من الأدوات والتقنيات لتقويم جهود تحسين تكنولوجيا البرمجيات، وذلك من خلال دورة حياة كلفة ووقت هذه البرمجيات.

- **القطاعات المستقبلية لسوق البرمجيات**

يبيِّن المخطط التالي قطاعات سوق البرمجيات المستقبلية:

برمجيات المستخدم (End-User Programming)		
تكامل الأنظمة (Integration System)	**تركيب التطبيقات (Application Composition)**	**مولّدات التطبيقات وأدوات التركيب (Application Generators and Composition Aids)**
البنية التحتية (Infrastructure)		

- **نماذج CoCoMo 2.0 الخاصة بقطاعات سوق البرمجيات**
  - لا يحتاج قطاع برمجيات المستخدم إلى نموذج CoCoMo 2.0
  - نموذج تركيب التطبيقات (Application Composition Model)
  - مولّدات التطبيقات، تكامل الأنظمة، والبنية التحتية
    - نموذج التصميم المبكر (Early Design Model)
    - نموذج البناء البعدي (Post-Architecture Model)

- **قياس حجم البرمجيات**

يستخدم نموذج CoCoMo 2.0 ثلاثة مقاييس مختلفة لقياس حجم المشروع البرمجي:

  - نقاط الغرض (Object Points)
  - نقاط الوظيفة غير المضبوطة (Unadjusted Function Points)
  - أسطر الترميز المصدري (Source Line Of Code SLOC)

نمذجة الجهد

- **الجهد المفتَرض أو المتصوَّر (Nominal Effort)**

يُعطى الجهد المفتَرض من أجل حجم مشروع ما بالعلاقة التالية:

$$PM_{nominal} = A \times (Size)^{B}$$

ويعبَّر عن هذا الجهد باستخدام الوحدة شخص/شهر (Person/Month ***PM***). سنوضح بقية الرموز المستخدمة في العلاقة السابقة:

  - يعرِّف نموذج CoCoMo عاملاً أسيّاً من أجل التوفير النسبي (Relative Economies) أو الخسارة النسبية (Relative Diseconomies) للامتداد (Scale) الذي نصادفه عندما يزداد حجم المشروع البرمجي. يجري تمثيل هذا العامل من خلال الأس (B).
  - يُستخدَم الثابت (A) لتمثيل التأثيرات الخطية (Linear Effects) على الجهد في المشاريع ذات الحجم المتزايد، ويكون (A=2.94).

- **عوامل الامتداد الأسية (Exponent Scale Factors)**

يجري حساب العامل (B) باستخدام المعادلة التالية:

$$B = 0.91 + 0.01 * SUM_{i=1 \text{ to } 5}(W_i)$$

يبيِّن الجدول التالي مستويات التقدير (Rating Levels) الخاصة بعوامل الامتداد الأسية في نموذج CoCoMo 2.0:

يجري جمع التقديرات الرقمية للمشروع (W_i) من أجل كل العوامل، وتستخدم لتحديد عامل الامتداد (B).

عوامل الامتداد (W_i)	منخفض جداً (5)	منخفض (4)	مفترض (3)	مرتفع (2)	مرتفع جداً (1)	مرتفع جداً جداً (0)

- **مثال 1**

إذا كان لدينا مشروع برمجي ذو حجم (100 KLOC)، وتقدير مرتفع جداً جداً (0) من أجل جميع العوامل. سيكون لدينا:

- $W_i = 0$
- $B = 0.91$
- $E = PM = 2.94*100^{0.91} = 2.94*66 = 194$ PM (الجهد النسبي)

- **مثال 2**

إذا كان لدينا مشروع برمجي ذو تقدير منخفض جداً (5) من أجل جميع العوامل. سيكون لدينا:

- $W_i = 25$
- $B = 1.16$
- $E = PM = 2.94*100^{1.16} = 2.94*209 = 614$ PM (الجهد النسبي)

- **التوفير والخسارة الناتجة عن الامتداد**
  - إذا كان ($B < 1.0$)، سيبدي المشروع توفيراً نتيجة الامتداد. إذا تضاعف حجم المشروع، فإن الجهد اللازم سيكون أقل من ضعف الجهد السابق. تزداد إنتاجية المشروع (Project Productivity) بازدياد حجم المشروع
  - إذا كان ($B=1.0$)، سيكون هناك توازناً بين التوفير والخسارة الناتجة عن الامتداد. يستخدم هذا النموذج الخطي (Linear Model) من أجل تقدير كلفة المشاريع الصغيرة. يستخدم من أجل نموذج تركيب التطبيقات (Applications Composition Model)
  - إذا كان ($B > 1.0$)، سيبدي المشروع خسارةً نتيجة الامتداد. هذا يعود بشكل رئيسي إلى سببين أساسيين، نمو عبء التواصل بين الأشخاص (Interpersonal Communications Overhead)، ونمو عبء تكامل الأنظمة الضخمة (Large-System Integration Overhead). سيكون للمشاريع الأضخم عدد اكبر من الأشخاص، وبالتالي سيكون هناك مسارات تواصل أكثر بين الأشخاص. إن دمج منتج صغير كجزء من منتج أضخم، يتطلب ليس فقط جهد تطوير المنتج الصغير، وإنما يتطلب جهد إضافي لتصميم ودمج واختبار واجهات هذا المنتج الصغير مع باقي المنتج.

القسم الرابع عشر والخامس عشر

إدارة المخاطر

الكلمات المفتاحية:

إدارة المخاطر، الاستراتيجيات المنفعلة للمخاطرة، الاستراتيجيات الفاعلة للمخاطرة، الخسارة، مخاطر المشروع، المخاطر التقنية، مخاطر الأعمال، المخاطر المعروفة، المخاطر القابلة للتنبؤ، المخاطر غير القابلة للتنبؤ، تحديد المخاطرة، المخاطرة العمومية، المخاطرة الخاصة بالمنتج، قائمة تفقُّد عناصر المخاطرة، مكونات المخاطرة البرمجية، توقُّع المخاطرة، أثر المخاطرة، احتمال المخاطرة، جدول المخاطرة، مراقبة المخاطرة، تخفيف المخاطرة، تجنب المخاطرة.

ملخص:

يناقش هذا الفصل كيفية إدارة المخاطر المتعلقة بالمشروع البرمجي

أهداف تعليمية:

- شرح الفرق بين الاستراتيجيات المنفعلة والاستراتيجيات الفاعلة لإدارة المخاطرة
- توضيح صفات المخاطر البرمجية
- توضيح أصناف المخاطر البرمجية
- توضيح أهمية تحديد المخاطر وتعيين هوية المخاطرة
- شرح كيفية إنشاء قائمة تفقُّد عناصر المخاطر
- شرح مكونات المخاطرة البرمجية
- شرح كيفية إجراء توقُّع للمخاطرة
- شرح كيفية إنشاء جدول للمخاطر
- شرح كيفية تقييم أثر المخاطرة
- شرح كيفية تقييم المخاطرة
- شرح كيفية تخفيف ومراقبة وإدارة المخاطرة

خطوات أساسية في إدارة المخاطر

- **تحديد المخاطر**

تحديد المخاطر المتعلقة بكلفة المشروع، الجدول الزمني للمشروع، مستوى الجودة المرغوب، التقنيات المستخدمة، وغير ذلك.

- **تقييم المخاطر**

تقييم احتمال حدوث كل مخاطرة وتأثيرها على المشروع.

- **وضع الإجراءات المضادة**

تحديد المخاطر ذات التأثير الكبير على المشروع، ومحاولة التخفيف منها ووضع الإجراءات المناسبة لذلك.

- **مراقبة وإدارة المخاطر**

مراقبة آلية التعامل مع المخاطر، وتوثيق جميع المعلومات المتعلقة بذلك وبالمشاكل التي قد تنتج عن مخاطرة معينة، واعتماد نظام سليم للتواصل بين أعضاء فريق المشروع.

الاستراتيجيات المنفعلة والاستراتيجيات الفاعلة للمخاطرة

- **استراتيجيات المخاطرة المنفعلة (Reactive Risk Strategies)**

تعتمد معظم الفرق البرمجية على الاستراتيجيات المنفعلة لإدارة المخاطرة. حيث تراقب الإستراتيجية المنفعلة وقوع المخاطر المحتملة، وتوضع الموارد اللازمة لمعالجتها عندما تصبح مشاكل حقيقية. والأكثر شيوعاً هو أن الفريق البرمجي لا يفعل شيئاً تجاه المخاطر إلى أن يحدث شيء خاطئ. عندها، "يطير" الفريق إلى العمل في محاولة لتصحيح المشكلة بسرعة. يسمى هذا الأسلوب غالباً "نمط إطفاء الحرق" (Fire Fighting Mode). وعندما يخفق ذلك يجري اعتماد أسلوب "إدارة الأزمات" (Crisis Management)، ويصبح المشروع في خطر حقيقي.

- **استراتيجيات المخاطرة الفاعلة (Proactive Risk Strategies)**

الإستراتيجية الأكثر ذكاءً لإدارة المخاطرة هي الإستراتيجية الفاعلة، والتي تبدأ قبل بدء العمل التقني بكثير. حيث يجري تحديد المخاطر (Risks Identification) المحتملة وتقدير احتمال وقوعها (Occurrence Probability Estimation) وأثرها (Impact)، وتصنيفها أيضاً في جدول أولوية حسب أهميتها. بعد ذلك يقوم الفريق البرمجي بوضع خطة لإدارة المخاطرة (Risk Management Plan). الهدف الأول هو تجنُّب المخاطرة، ولكن لأنه لا يمكن تجنُّب جميع المخاطر، يعمل الفريق على تطوير خطة طوارئ (Contingency Plan) تمكِّنه من الاستجابة بطريقة مضبوطة وفعالة.

صفات المخاطر البرمجية

بالرغم من أن هناك العديد من الاقتراحات لتعريف المخاطرة البرمجية (Software Risk)، إلا أن هناك اتفاق عام على أن المخاطرة تتصف دوماً بصفتين:

- **عدم اليقين (Uncertainty)**

قد يحصل الحدث المميِّز للمخاطرة وقد لا يحصل، أي لا يوجد مخاطر احتمالها 100%.

- **الخسارة (Loss)**

إذا أصبحت المخاطرة حقيقية فإن التبعات أو الخسائر الناتجة عن ذلك ستحصل.

أصناف المخاطر البرمجية

من الضروري عند تحليل المخاطر إعطاء قيمة كمية لمستوى الشك ودرجة الخسارة المتعلقة بكل منها. ولتحقيق ذلك تُصنَّف المخاطرة إلى الأصناف التالية:

- **مخاطر المشروع (Project Risks)**

تهدّد مخاطر المشروع خطة المشروع، أي إذا أصبحت هذه المخاطر حقيقية، فمن المرجَّح أن يحصل انزياح في الجدول الزمني للمشروع، ومن ثم تزداد الكلفة. تحدِّد مخاطر المشروع ماهية المشاكل المحتملة في الميزانية، الجدول الزمني، والكادر، الموارد، الزبون، ومشاكل المتطلبات وأثر كل ذلك على المشروع.

- **المخاطر التقنية (Technical Risks)**

تهدّد المخاطر التقنية جودة ودقة وتوقيت البرمجية المطلوب إنتاجها. وعند حصول مخاطرة تقنية، يصبح تحقيق البرمجية صعباً أو مستحيلاً. تحدِّد المخاطر التقنية المشاكل المحتملة في التصميم، التحقيق، الواجهات، الاختبار، والصيانة. إضافةً إلى ذلك هناك عوامل أخرى للمخاطرة في هذا الإطار، كغموض المواصفات (Specification Ambiguity)، الشك التقني (Technical Uncertainty)، التقادم التقني، والتقنيات الجديدة. تحصل المخاطرة التقنية في أغلب الأحيان لأن المسألة أصعب حلاً مما تصورنا.

- **مخاطر الأعمال (Business Risks)**

تحدِّد مخاطر الأعمال قابلية حياة البرمجيات التي ستبنى، حيث غالباً ما تعرِّض مخاطر الأعمال المشروع أو المنتج للخطر. سنورد أهم مخاطر الأعمال:

  - بناء منتج أو نظام ممتاز لا يرغب فيه أحد (مخاطرة السوق)
  - بناء منتج لم يعد متناسباً مع الإستراتيجية العامة لأعمال الشركة (مخاطرة إستراتيجية)
  - بناء منتج لا يعرف فريق المبيعات كيفية بيعه أو تسويقه
  - فقدان دعم الإدارة العليا بسبب تغيير اهتمامها أو إجراء تغير في الأشخاص (مخاطرة إدارية)
  - فقدان توفر الميزانية أو الموظفين (مخاطرة الميزانية)

من المهم ملاحظة أن هذا التصنيف البسيط لن يعمل دوماً، فبعض المخاطر لا يمكن التنبؤ بها سلفاً.

أصناف عامّة المخاطر البرمجية

جرى اقتراح تصنيف عام للمخاطر يتضمن:

- **المخاطر المعروفة (Known Risks)**

هي المخاطر التي يمكن اكتشافها بعد تقييم متأنٍ لخطة المشروع، ولبيئة الأعمال والبيئة التقنية التي يجري فيها تطوير المشروع، والمصادر الموثوقة الأخرى للمعلومات (مثل: تاريخ توريد غير معقول، افتقار للمتطلبات الموثَّقة أو لنطاق البرمجية، أو بيئة تطوير فقيرة).

- **المخاطر القابلة للتنبؤ (Predictable Risks)**

تستخلص من الخبرة السابقة، مثل تغيير العاملين، اتصالات ضعيفة مع الزبون، تخفيف جهود الموظفين مع تخديم طلبات الصيانة المستمرة.

- **المخاطر غير القابلة للتنبؤ (Unpredictable Risks)**

وهذه تشبه "جوكر" ورق اللعب، يمكن أن تحدث، وقد تحدث فعلياً ولكن يصعب فعلياً تعيين هويتها مقدَّماً.

تحديد المخاطر

تحديد المخاطرة أو تعيين هوية المخاطرة (Risk Identification) هو إجرائية لتوصيف التهديدات التي تعترض خطة المشروع (التقديرات، الجدول تلزمني، تحميل الموارد، ...). عندما يقوم مدير المشروع بتعيين هوية المخاطر، فإنه يكون قد خطا الخطوة الأولى باتجاه تجنبها عند الإمكان والتحكم فيها أيضاً عند الضرورة.

هناك نوعان متميزان من المخاطر من اجل جميع أصناف أو فئات المخاطر. المخاطر العامة (Generic Risks) والمخاطر الخاصة بالمنتج (Product-Specific Risks). المخاطر العامة هي تهديد محتمل لكل مشروع برمجي، أما المخاطر الخاصة بالمنتج فلا يستطيع تعيينها إلا أولئك الذين لديهم فهم واضح للقضايا التقنية وللناس وللبيئة الخاصة بالمشروع الجاري تنفيذه. يجري فحص خطة المشروع وبيان نطاق البرمجيات بهدف تحديد المخاطر، ويوضع جواب عن السؤال التالي: "ما هي الصفات الخاصة لهذا المنتج التي قد تهدد خطتنا للمشروع؟". يجب تعيين كل من المخاطر العامة والخاصة بالمنتج بشكل نظامي، وكما يقال "إذا لم تهاجِم المخاطر بفعالية، ستهاجمك المخاطر بفعالية".إحدى طرائق تحديد المخاطر هي أن نقوم بإنشاء قائمة حصر لعناصر المخاطرة (Risk Item Checklist).

قائمة حصر عناصر المخاطرة

إحدى طرائق تحديد المخاطر هي أن نقوم بإنشاء قائمة حصر عناصر المخاطرة (Risk Item Checklist). يمكن استخدام هذه القائمة لتحديد المخاطر وتركيز الاهتمام في مجموعة جزئية من المخاطر المعروفة والقابلة للتنبؤ بها في الفئات الجزئية العمومية (والتي قد تختلف من مشروع إلى آخر، وقد تختلف كذلك قائمة تفقُّد العناصر المتعلقة بكل منها) التالية:

- **حجم المنتج (Product Size)**

المخاطر المتعلقة بالحجم الكلي للبرمجية المراد بناؤها أو تعديلها.

- **أثر الأعمال (Business Impact)**

المخاطر المتعلقة بالقيود التي تفرضها الإدارة أو يفرضها مكان التسويق.

- **مزايا الزبون (Client Characteristics)**

المخاطر المتعلقة بمستوى تمرُّس الزبون وقابلية المطوّر للاتصال بالزبون بطريقة مخططة زمنياً.

- **تعريف الإجرائية (Process Definition)**

المخاطر المتعلقة بالدرجة التي وصل إليها تعريف الإجرائية البرمجية والتي تتَّبعها مؤسسة التطوير.

- **بيئة التطوير (Development Environment)**

المخاطر المتعلقة بتوفُّر وجودة الأدوات المستخدمة لبناء المنتج.

- **التقنية المطلوب بناؤها (Required Technology)**

المخاطر المتعلقة بتعقيد النظام المطلوب بناؤه ومستوى التقنيات المطلوب توفيها في النظام.

- **حجم الكادر وخبرته (Staff Size and Experience)**

المخاطر المتعلقة بخبرة مهندسي البرمجيات الذين سيقومون بالعمل فيما يخصّ خبرتهم التقنية الكلية وخبرتهم في المشروع.

مخاطر حجم المشروع

- **قائمة حصر عناصر المخاطرة**

تُعيِّن قائمة حصر عناصر المخاطرة التالية المخاطرَ المتعلقة بحجم المنتج البرمجي:

- الحجم التقديري للمنتج مقاساً باستخدام عدد أسطر الترميز (LOC) أو نقاط الوظيفة (FP)
- درجة الثقة في تقدير الحجم التقديري للبرمجية
- الحجم التقديري للمنتج مقاساً بعدد البرامج والملفات والمناقلات
- الانزياح النسبي (Relative Shift) في حجم المنتج عن وسطي المنتجات السابقة
- حجم قاعدة المعطيات (Database) التي يُنشئها أو يستخدمها المنتج
- عدد مستخدمي المنتج
- عدد التغيرات المتوقعة في متطلبات المنتج، قبل التسليم، وبعد التسليم
- مقدار البرمجيات التي أُعيد استخدامها

يجب في كل حالة من هذه الحالات إجراء مقارنة معلومات المنتج المطلوب تطويره بالخبرة السابقة. فإذا حصلت نسبة انزياح كبيرة، أو كانت الأرقام متماثلة ولكن النتائج السابقة أقل بكثير من المقبول، فإن احتمال المخاطرة عالٍ جداً.

مخاطر أثر الأعمال

يخضع قسم التسويق لاعتبارات الأعمال التي تتعارض مباشرةً أحياناً مع المخاطر التقنية.

- **قائمة حصر عناصر المخاطرة**

تحدد قائمة حصر عناصر المخاطرة التالية المخاطرَ العمومية المقابلة لأثر الأعمال:

- أثر هذا المنتج في عائدات الشركة
- وضوح هذا المنتج في نظر الإدارة العليا
- إمكانية الالتزام بالمواعيد النهائية للتسليم (هل هذه مواعيد معقولة ومنطقية؟)
- عدد الزبائن الذين سيستخدمون هذا المنتج وانسجام احتياجاتهم معه
- درجة تمرُّس المستخدم النهائي
- مقدار وجودة توثيق المنتج اللازم توفيرها وتسليمها للزبون
- القيود الحكومية على بناء المنتج (إن وجدت)
- الكلفة المترتبة على التسليم المتأخر
- الكلفة المترتبة على منتج فيه عيوب

يجب في كل حالة من هذه الحالات إجراء مقارنة معلومات المنتج المطلوب تطويره بالخبرة السابقة. فإذا حصلت نسبة انزياح كبيرة، أو كانت الأرقام متماثلة ولكن النتائج السابقة أقل بكثير من المقبول، فإن احتمال المخاطرة عالٍ جداً.

أنماط مختلفة من الزبائن

- **اختلاف الزبائن**

لا يوجد زبون مماثل لآخر. سنوضح ذلك من خلال النقاط التالية:

- تختلف الاحتياجات من زبون لآخر:

فزبون يعرف ما يريد، وآخر يعرف ما لا يريد. وزبون يبذل قصارى جهده لمعرفة التفاصيل، على حين يكتفي زبون آخر بالوعود حتى لو كانت غير دقيقة.

- تختلف الشخصية من زبون لآخر:

فزبون يستمتع بكونه زبوناً، ويستمتع بالمفاوضات وبالنتائج السيكولوجية للمنتج الجديد، بينما يفضّل آخر عدم كونه زبوناً. وزبون يقبل سعيداً أي شيء يُسلَّم إليه، ويجعل أسوأ منتج هو الأفضل، في حين يشتكي آخر بأسىً عندما يفتقر المنتج إلى الجودة. ويبدي البعض تقديره عندما تكون الجودة عالية، وقلة منهم يشتكون لمجرّد الشكوى بقطع النظر عن الأسباب.

- تختلف الصلة بالموردين من زبون لآخر:
- فزبون يعرف جيداً المنتَج والمنتِج، وآخر لا يتواصل مع المنتِج إلا بمراسلات مكتوبة، وببعض المكالمات الهاتفية المختصرة
- غالباً ما يناقض الزبون نفسه:

فهو يريد الحصول على كل شيء "البارحة" ومجاناً. غالباً ما يعاني المنتِج من تناقض الزبون مع نفسه.

المخاطر المتعلقة بالزبون

يمكن أن يكون للزبون السيئ أثر واضح في مقدرة الفريق البرمجي على إنهاء المشروع في وقته المحدد وبموازنته المحددة. يمثِّل الزبون السيئ تهديداً كبيراً لخطة المشروع ومخاطرة كبيرة أيضاً لمدير المشروع.

- **قائمة حصر عناصر المخاطر المتعلقة بالزبون**

تعيِّن قائمة حصر عناصر المخاطر التالية المخاطرَ العمومية المتعلقة بالزبون:

- هل عملت مع الزبون في الماضي؟
- هل يمتلك الزبون فكرة واضحة عن المطلوب؟ هل قضى الزبون وقتاً كافياً لكتابتها؟ أو هل كتبها بالأصل؟
- هل سيوافق الزبون على قضاء وقتٍ في اجتماعات رسمية لجمع المتطلبات بهدف تحيد نطاق المشروع؟
- هل الزبون مستعد لتأسيس قنوات اتصال سريعة مع المطوِّر؟
- هل الزبون متمرِّس تقنياً في مجال المنتج؟
- هل الزبون مستعد ليدع موظفيك يقومون بعملهم، أي هل سيقاوم الزبون نفسه بألا يقف فوق رأسك خلال الأعمال التقنية التفصيلية؟
- هل يفهم الزبون الإجرائية البرمجية (Software Process) التي يتَّبعها الفريق لتطوير المنتَج البرمجي؟

إذا كان الجواب عن أيٍّ من الأسئلة السابقة نفياً، فيجب إجراء تقصٍّ آخر لتقييم احتمال المخاطرة.

المخاطر المتعلقة بالإجرائية

إذا كانت الإجرائية البرمجية "سيئة التعريف"، وإذا أجري التحليل والتصميم والاختبار بأسلوب مناسب، وإذا وافق الجميع على أن الجودة مفهوم هام، ولكن لا أحد يتصرّف لتحقيق ذلك بأي طريقة ملموسة فعلياً، فسيكون في المشروع عندئذٍ مخاطرة. استخلصت الأسئلة التالية من ورشة عمل لتقييم ممارسات هندسة البرمجيات التي طورتها شركة Pressman، واقتُبِست الأسئلة نفسها من استبيان معهد هندسة البرمجيات (Software Engineering Institute) لتقييم الإجرائية البرمجية.

- **قضايا الإجرائية (Process Issues)**

- هل تدعم إدارتك العليا سياسة مكتوبة تؤكّد أهمية الإجرائية القياسية لهندسة البرمجيات؟
- هل طوّرت مؤسستك وصفاً مكتوباً للإجرائية البرمجية التي ستطبَّق في هذا المشروع؟
- هل يحبِّذ أعضاء الكادر المكلَّف بالعمل الإجرائيةَ البرمجية كما هي موثقة ومستعدّون لاستخدامها؟
- هل تُستخدم الإجرائية البرمجية لمشاريع أخرى؟
- هل طورت مؤسستك أو طلبت سلسلة من الدورات التدريبية في هندسة البرمجيات للمديرين والكادر التقني؟
- هل تُقدَّم معايير منشورة لهندسة البرمجيات لكل مطوِّر برمجيات أو مدير برمجيات؟
- هل تجرى المراجعات التقنية الرسمية (Formal Technical Reviews) لتوصيف المتطلبات والتصميم والترميز بشكل منتظم؟
- هل تُجرى المراجعات التقنية الرسمية للإجراءات الاختبار وحالات الاختبار بشكل منتظم؟
- هل توثَّق نتائج كل مراجعة تقنية رسمية ، بما في ذلك الأخطاء الموجودة والموارد المستخدمة؟

- هل توجد آلية ما لضمان توافق العمل المنفَّذ ضمن المشروع مع قياسات هندسة البرمجيات؟
- هل استُخدمت آلية للتحكم في تغيير متطلبات الزبون التي تؤثِّر في البرمجية؟
- هل يوجد توثيق لبيان العمل، توصيف متطلبات البرمجية، وخطة تطوير البرمجية، وغيرها، وذلك لكل عقد فرعي؟
- هل يُتَّبع إجراء ما لمتابعة ومراجعة أداء المتعاقدين الفرعيين؟

المخاطر المتعلقة بالإجرائية (متابعة)

- **قضايا تقنية (Technical Issues)**
  - هل تُستَخدم وسائل مناسبة للمساعدة على التواصل بين الزبون والمطور، مثل تقنيات توصيف التطبيق الميسَّرة (Fast Application Specification Techniques FAST)؟
  - هل تُستَخدم طرائق محدّدة لتحليل البرمجيات؟
  - هل تُستَخدم طريقة محدّدة لتصميم المعطيات وتصميم بنية البرمجية؟
  - هل كُتِبَ أكثر من 90% من ترميز برنامجك بلغة عالية المستوى؟
  - هل عُرِّفَت واستخدمت مصطلحات محدّدة لتوثيق الترميز؟
  - هل تُستخدم طرائق محدّدة لتصميم حالات اختبار البرمجية؟
  - هل تُستخدم أدوات برمجية لدعم فعاليات التخطيط والمتابعة؟
  - هل تُستخدم أدوات برمجية لدعم إجرائية تحليل البرمجيات وتصميمها؟
  - هل تُستخدم أدوات لإنشاء نماذج أولية (Prototype) برمجية
  - هل تُستخدم أدوات برمجية لدعم إنتاج الوثائق وإدارتها؟
  - هل تُجمع مقاييس الجودة لجميع المشاريع البرمجية؟
  - هل تُجمع مقاييس الإنتاجية لجميع المشاريع البرمجية؟

إذا أجيب بالنفي عن معظم هذه الأسئلة، فإن الإجرائية البرمجية ضعيفة ويكون احتمال المخاطرة عالياً. يجب الانتباه إلى أن قائمة تفقُّد عناصر المخاطرة المتعلقة بالإجرائية قد تختلف من مشروع إلى آخر ولكن النقاط المحددة سابقاً تعتبر أكثر شيوعاً وأهمية بين المشاريع البرمجية.

المخاطر المتعلقة بالتكنولوجيا

يُعتبر دفع التكنولوجيا إلى حدودها أمراً مثيراً ومصدراً للتحدي كذلك، وهو حلم كل إنسان تقني، وذلك لأن التكنولوجيا تُجبر الممارس على استخدام مهارته لأقصى درجة، ولكنها أيضاً محفوفة بالمخاطر.

- **قائمة حصر بنود المخاطر المتعلقة بالتكنولوجيا**

تميِّز قائمة حصر المخاطر التالية المخاطرَ العمومية المتعلقة بالتكنولوجيا أو التقنية التي ستُبنى:

- هل التكنولوجيا التي ستبنى جديدة على المؤسسة؟
- هل تحتاج متطلبات الزبون إلى بناء خوارزميات جديدة أو تقنية دخل أو خرج جديدة؟
- هل للبرمجيات واجهة مع أجهزة جديدة أو غير مجرَّبة؟
- هل للبرمجيات التي ستُبنى واجهة مع نظام قواعد معطيات لم تثبت بعد وظيفته وأداؤه في مجال التطبيق المراد بناؤه؟
- هل تحتاج متطلبات المنتَج إلى واجهة مستخدم خاصة؟
- هل تحتاج متطلبات المنتَج إلى استخدام طرائق جديدة للتحليل والتصميم والاختبار؟
- هل تحتاج المتطلبات إلى استخدام طرائق غير تقليدية لتطوير البرمجيات، مثل الطرائق الصورية (Formal Methods)؟ والطرائق المعتمدة على الذكاء الاصطناعي (Artificial Intelligence)، والشبكات العصبونية (Neural Networks)؟
- هل تضع المتطلبات قيود أداء فائقة على المنتج؟
- هل الزبون غير متأكد من إمكانية تحقيق الوظيفة المطلوبة؟

إذا كان الجواب على أي من هذه الأسئلة بالإيجاب (نعم)، فيجب إجراء تحليل إضافي لتقييم احتمال المخاطرة.

المخاطر المتعلقة ببيئة التطوير

تدعم بيئة هندسة البرمجيات فريق العمل والإجرائية والمنتَج، ولكن إذا كان هناك عيوباً في هذه البيئة، فقد تكون مصدر مخاطرة كبيرة.

- **قائمة حصر عناصر المخاطر المتعلقة ببيئة التطوير**
  - هل تتوفر الأدوات المناسبة لإدارة المشروع البرمجي؟
  - هل تتوفر الأدوات المناسبة لإدارة الإجرائية البرمجية؟
  - هل تتوفر الأدوات المناسبة للتحليل والتصميم؟
  - هل تتوفر مترجمات (Compilers) أو مولِّدات ترميز (Code Generator)، مناسبة للمنتَج المراد بناؤه؟
  - هل تستخدم البيئة قاعدة معطيات أو مخزن (Repository)؟
  - هل جميع الأدوات البرمجية المستخدمة متكاملة مع بعضها؟
  - هل جرى تدريب أعضاء فريق المشروع على الأدوات التي تلاءم مهام كل عضو؟
  - هل هناك خبراء محليين للإجابة عن الأسئلة المتعلقة بالأدوات؟
  - هل هناك توثيق كافٍ لهذه الأدوات؟
  - هل المساعدة المتاحة (بطريقةٍ ما، عبر الهاتف، الانترنت،...) كافية؟

إذا كانت الإجابة عن معظم هذه الأسئلة بالنفي، فإن بيئة تطوير البرمجيات ضعيفة واحتمال المخاطرة عالٍ جداً.