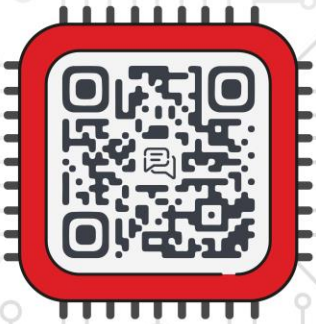

3

16

1920 sp

عملي

RB Informatics; 27/05/2025

التقدير الكمي لحجم البرمجيات

د. ماهر الصارم

هندسة برمجيات ٢

السلام عليكم ورحمة الله وبركاته

مقدمة:

في المحاضرة السابقة، انتقلنا من المفاهيم النظرية العامة لهندسة البرمجيات نحو التطبيقات العملية والأدوات التي تدعم إدارة مشاريع البرمجيات بكفاءة أعلى. استعرضنا أولاً أداة MS Project كأداة متقدمة لتخطيط وجدولة المشاريع، وتعرفنا على مفاهيم المهام(Tasks)، وهيكل تجزئة العمل(WBS)، والموارد(Resources)، والعلاقات بين المهام(Dependencies)، وخط الأساس(Baseline).

ثم ناقشنا أدوات هندسة البرمجيات المدعومة بالحاسوب CASE، والتي تساهم في أتمتة مراحل مختلفة من دورة حياة تطوير البرمجيات، من التحليل والتصميم وحتى الاختبار والتوثيق.

بعدها تطرقنا إلى Capability Maturity Model Integration (CMMI) الذي يوفر إطاراً لتقييم وتحسين نضج العمليات داخل المؤسسات البرمجية عبر خمس مستويات متدرجة، من العمليات غير المنضبطة وحتى التحسين المستمر المبني على القياس.

وأخيراً، ركزنا على أداة Jira كأداة فعالة في إدارة مشاريع البرمجيات وفق منهجيات Agile و Scrum تعلمنا كيفية تنظيم الـ Backlog، وإدارة الـ Sprint، وتقدير قصص المستخدم باستخدام مفهوم Story Points، كما تعرفنا على كيفية كتابة User Stories وخصائصها الأساسية.

وفي نهاية المحاضرة السابقة بدأنا تمهيداً مبدئياً لموضوع **التقدير** Estimation، وتعرفنا على أهدافه الأساسية، وأنواعه المختلفة ما بين التقدير الزمني والتقدير بالنقاط(Story Points) ، كما استعرضنا طرقه العملية مثل Planning Poker وAffinity Estimation والعوامل المؤثرة على دقة التقدير.

بناءً على هذه الأرضية، سنواصل اليوم التعمق بشكل منهجي في موضوع التقدير، وسنتعرف على إحدى الأدوات الهامة فيه وهي Function Point Analysis التي تستخدم لتقدير حجم العمل البرمجي بشكل أكثر دقة وموضوعية

في محاضرة اليوم سوف نتعمق أكثر ونتعلم معاً واحدة من أهم الطرق العلمية الدقيقة لتقدير حجم المشاريع البرمجية، وهي:

تحليل نقاط الوظائف (Function Point Analysis)

هذه الطريقة تستخدم في الشركات الكبيرة والدوائر الحكومية والشركات البرمجية لحساب حجم النظام بشكل عادل ودقيق

لماذا التخمين باستخدام Function Points مهم؟

لأننا كمطوري ومهندسي برمجيات، سنواجه في سوق العمل واقعًا مفاده أن كتابة الكود ليست سوى جزء من المهمة.
فالخطوة الأولى غالبًا ما تكون إدارية بامتياز:
ما حجم هذا المشروع؟ كم يتطلب من وقت وجهد؟ وما هي تكلفته المتوقعة؟
وهنا تبرز أهمية تحليل نقاط الوظائف، كطريقة علمية ومنهجية للإجابة على هذه الأسئلة بدقة وموضوعية.

المحور الأول: التقدير في هندسة البرمجيات (Estimation)

قبل أن نكتب أي سطر برمجي، أول سؤال دائماً نسأله في المشاريع البرمجية هو:
كم سيأخذ هذا المشروع من جهد وزمن وتكلفة؟
هذه العملية اسمها ببساطة: التقدير أو التخمين (Estimation)

لماذا نحتاج التقدير؟

- لأن كل مشروع له موازنة محددة.
- لأن الإدارة تحتاج أن تضع خطة زمنية.
- لأن الفريق يحتاج أن يعرف كم شخص سيعمل، ومتى يبدأ ومتى ينتهي.
- ولأن الزبون يسألنا دائماً: متى ستسلّم النظام؟ وكم ستكلفني؟

التحدي في التقدير البرمجي

تقدير المشاريع البرمجية أصعب من المشاريع العادية للأسباب التالية:

- أحياناً المتطلبات غير واضحة بالكامل منذ البداية.
- المشاكل تظهر أثناء التطوير(bugs)، تغيرات، تحسينات.
- كل مبرمج له سرعة مختلفة.
- لغات البرمجة نفسها تختلف في حجم الكود.

الطرق التقليدية التي كانت تستخدم سابقاً:

1. التقدير حسب الزمن مباشرة:(Time-Based Estimation)

- أقول مثلاً هذه المهمة تحتاج 100 ساعة عمل.
- المشكلة: من يحدد هذا الرقم؟ وعلى أي أساس؟

2. التقدير حسب عدد الأسطر البرمجية:(LOC – Lines of Code)

- نحسب كم سطر كود نحتاج.
- المشكلة: لغة البرمجة تؤثر على عدد الأسطر.
- المبرمج الجيد يكتب أقل كود ليحقق نفس الوظيفة

لذلك ظهر مفهوم التقدير بالحجم الوظيفي

أي أننا لا نقدّر كم ساعة سنعمل أو كم سطر سنكتب، بل نقدّر كم وظيفة سيقدمها النظام للمستخدم؟

وهنا جاءت فكرة الـ Function Point Analysis وهي موضوع المحاضرة اليوم بشكل أساسي

الفكرة الأساسية التي سنبني عليها كل المحاضرة:

كل وظيفة يقدمها النظام، لها وزن معين.

كلما زادت الوظائف أو زادت تعقيداتها، زاد حجم المشروع

المحور الثاني: ما هو Function Point؟

بعد أن عرفنا أن الطرق التقليدية في التقدير تعاني من مشاكل، ظهر لنا أسلوب علمي اسمه:

تحليل نقاط الوظائف (Function Point Analysis)

الفكرة الرئيسية ببساطة:

نحن هنا لا نهتم بعدد ساعات العمل، ولا بعدد الأسطر البرمجية.

بدلاً من ذلك، نقيس:

- كم وظيفة سيقدمها النظام؟
- كم حجم المعالجة التي سينفذها؟
- كم كمية البيانات التي سيتعامل معها؟

لماذا هذه الطريقة أكثر دقة؟

- لأنها تعتمد على ما يراه المستخدم النهائي.
- لأنها لا تتأثر بلغة البرمجة المستخدمة.
- لأن وظيفة "إضافة مستخدم" مثلاً، سواء تمت كتابتها ب java أو python و C# — تبقى نفس الوظيفة.
- لأنها تجعلنا نركز على ما سيقدمه النظام فعلاً.

وظيفة Function Point في المشروع:

تعطيك رقم يمثل حجم المشروع،

ومن هذا الرقم نستطيع أن نستخرج لاحقاً:

- الزمن المتوقع.
- الكلفة التقديرية.
- الموارد المطلوبة.

مكونات Function Point

لنفهم كيف نحسب حجم المشروع بهذه الطريقة، لا بد أن نتعرف على ما يسمى بـ:

"الأنواع الخمسة للوظائف"

وهي الأساس الذي سنبني عليه كامل الحساب.

الرقم	نوع الوظيفة	ماذا تمثل
1	ILF	ملفات داخلية يديرها النظام
2	EIF	ملفات خارجية يقرأ منها النظام
3	EI	مدخلات يدخلها المستخدم للنظام
4	EO	مخرجات وتقارير يصدرها النظام
5	EQ	استعلامات (بحث واسترجاع بيانات فقط)

لماذا حددنا بالضبط هذه الخمس أنواع؟

أي نظام برمجي في العالم، مهما بلغت درجة تعقيده، فإن كل ما يقدّمه للمستخدم يندرج ضمن واحدة من الوظائف الخمس التالية:

- **إدخال بيانات**(External Input - EI)
- **إخراج بيانات**(External Output - EO)
- **استعلام أو بحث عن بيانات**(External Inquiry - EQ)
- **إدارة ملفات داخلية**(Internal Logical File - ILF)
- **التفاعل مع بيانات من أنظمة خارجية**(External Interface File - EIF)

المحور الثالث: الأنواع الخمسة للوظائف في حساب Function Point

كما قلنا سابقًا:

كل وظيفة داخل النظام يجب أن نصنفها في واحد من هذه الأنواع الخمسة.

هذه الخطوة هي أساس عملية التقدير الدقيقة

النوع الأول: Internal Logical File (ILF)

هي قواعد البيانات أو الملفات التي يُديرها النظام بنفسه.

النظام مسؤول عن إدارتها وتعديلها وتحديثها.

أمثلة على ILF:

- **جدول بيانات الطلاب في نظام الجامعة.**
- **جدول بيانات الحسابات البنكية في نظام المحاسبة.**

متى نحسبها؟

عندما يكون النظام هو من يُنشئ ويُعدل ويحذف البيانات داخل هذا الملف.

النوع الثاني: External Interface File (EIF)

هذه ملفات بيانات موجودة خارج النظام.

النظام يقرأ منها أو يستخدم بياناتها لكنه لا يتحكم في تعديلها أو إدارتها.

أمثلة على EIF:

- **قراءة بيانات الموظفين من نظام الموارد البشرية.**
- **قراءة بيانات الموردين من نظام مشتريات خارجي.**

الفرق بين ILF و EIF؟

- **ILF ← الملف بالكامل تحت سيطرة النظام.**

- EIF ← النظام فقط يقرأ أو يستخدم بياناته دون تعديلها.

النوع الثالث: External Input (EI)

كل عملية يقوم فيها المستخدم أو نظام خارجي بإدخال بيانات للنظام.

أمثلة على EI:

- إدخال بيانات طالب جديد.
- تعديل بيانات حساب مستخدم.
- تسجيل فاتورة شراء.

لماذا EI مهم؟

كل إدخال بيانات يحتاج فحص وتحقق وأحياناً معالجة قبل تخزينها.

النوع الرابع: External Output (EO)

هذه هي المخرجات التي يصدرها النظام للمستخدم بعد معالجة البيانات داخليًا.

أمثلة على EO:

- تقرير علامات الطلاب.
- كشف حساب شهري للعميل.
- تقارير إحصائية للمبيعات.

النوع الخامس: External Inquiry (EQ)

عمليات استعلام من النظام لاسترجاع بيانات بدون أي تعديل عليها أو معالجة معقدة.

أمثلة على EQ:

- البحث عن اسم طالب.
- الاستعلام عن فاتورة برقمها.

متى نحسبها كـ EQ؟

عندما تكون العملية استرجاع مباشر للبيانات كما هي بدون عمليات حسابية أو معالجة إضافية.

الفرق بين EO وEQ؟

- EO ينتج بيانات جديدة بعد معالجة.
- EQ فقط يسترجع بيانات كما هي.

ملخص سريع

وظيفة النظام	الوظيفة
يدير البيانات داخليًا	ILF
يقرأ بيانات خارجية	EIF
يستقبل بيانات جديدة	EI
ينتج تقارير أو مخرجات	EO
يسترجع بيانات للاستعلام	EQ

لماذا هذه الأنواع مهمة جدًا؟

لأنها:

- تُعطينا صورة كاملة عن حجم مسؤوليات النظام.
- هي الأساس الذي نبني عليه كل حساب Function Point.
- بدون هذا التصنيف لا يمكننا حساب الحجم بدقة.

ملاحظة هامّة: عملية التقدير الصحيحة تبدأ دائمًا بفهم دقيق لهذه الأنواع الخمسة، لأن الخطأ في التصنيف سيعطيك حجم خاطئ للمشروع بالكامل.

المحور الرابع: تقييم درجة التعقيد لكل نوع من الوظائف

الآن بعد أن صنّفنا وظائف النظام إلى الأنواع الخمسة:

(ILF, EIF, EI, EO, EQ)

يبقى أن نُحدد لكل وظيفة: هل هي بسيطة (Simple)، متوسطة (Average)، أم معقدة (Complex)؟

لماذا هذا مهم؟

لأن كل درجة تعقيد لها وزن معين من النقاط، وهي التي سنجمعها لاحقاً لحساب CFP (Crude Function Points)

كيف نُحدد درجة التعقيد؟

يتم تحديد درجة التعقيد بناءً على:

1. كمية البيانات التي تتعامل معها الوظيفة

عدد الحقول أو البيانات التي تدخل أو تخرج، وتسمى Data Element Type – DET

2. عدد الجداول أو الملفات المرتبطة بالعملية

تسمى File Type Referenced - FTR أو Record Element Type – RET

الجداول الأساسية لتحديد التعقيد

1. تعقيد ملفات ILF وEIF الجداول والملفات

Record Element Type (RET)	Number of Data Element type (DET)		
	1 to 19 DET	20 to 50 DET	51 or more DET
1 RET	Low	Low	Average
2-5 RET	Low	Average	High
6 or more RET	Average	High	High

2. تعقيد المدخلات EI

	1 to 4 DET	5 to 15 DET	16 or more DET
0 to 1 FTR	Low	Low	Average
2 FTR	Low	Average	High
3 or more FTR	Average	High	High

3. تعقيد المخرجات EO

	1 to 5 DET	6 to 19 DET	20 or more DET
0 to 1 FTR	Low	Low	Average
2 or 3 FTR	Low	Average	High
4 or more FTR	Average	High	High

4. تعقيد الاستعلامات EQ

عادة له نفس تصنيف الـ EI من حيث الجداول

بعد تحديد درجة التعقيد لكل وظيفة

نعطي لكل درجة النقاط المخصصة لها حسب الجدول Unadjusted Function Point (UFP):

الوظيفة	Low	Average	High
ILF	×7	×10	×15
EIF	×5	×7	×10
EI	×3	×4	×6
EO	×4	×5	×7
EQ	×3	×4	×6

مثال تطبيقي صغير:

لنفترض أن لدينا:

- 3 **مدخلات بسيطة** (EI Simple) ⟶ 3 × 3 = 9
- 2 **مخرجات معقدة** (EO Complex) ⟶ 2 × 7 = 14
- 1 **استعلام متوسط** (EQ Average) ⟶ 1 × 4 = 4

إجمالي النقاط لهذه الوظائف فقط:

9 + 14 + 4 = 27 **نقطة**

لماذا كل هذه الدقة؟

لأننا نريد أن يكون التقدير علمي قدر الإمكان بحيث:

- **إذا أعاد أي فريق آخر تحليل النظام نفسه، يحصل على نتائج قريبة جداً.**
- **لا يعتمد التقدير على اجتهاد شخص واحد بل على قواعد واضحة**

بهذا نكون أنهينا مرحلة تقييم التعقيد لكل وظيفة

الخطوة القادمة الآن هي:

المحور الخامس: كيف نحسب مجموع النقاط CFP

ثم نحسب النقاط المعدلة FP

بعد أن صنفنا وظائف النظام إلى الأنواع الخمسة، وحددنا درجة التعقيد لكل واحدة منها، أصبحت لدينا الأرقام التي سنبني عليها حساب الحجم الكلي للمشروع.

الخطوة الأولى: حساب مجموع النقاط الخام (CFP - Crude Function Points)

- في هذه المرحلة نقوم بضرب عدد كل نوع في وزنه المحدد حسب درجة التعقيد.
- ثم نجمع جميع النقاط لنحصل على مجموع CFP

مثال توضيحي:

الوظيفة	العدد	درجة التعقيد	الوزن (حسب الجدول)	المجموع
ILF	3	متوسطة	10	30
EIF	2	بسيطة	5	10
EI	5	معقدة	6	30
EO	4	متوسطة	5	20
EQ	2	بسيطة	3	6

CFP = 30 + 10 + 30 + 20 + 6 = 96 نقطة

ملاحظة مهمة:

في هذه المرحلة نحن نقيس فقط الحجم الوظيفي الخام للنظام دون النظر لأي ظروف إضافية معقدة في المشروع

الخطوة الثانية: إدخال عامل تعديل التعقيد (RCAF)

الآن ننتقل إلى مراعاة بعض العوامل الخاصة التي قد تزيد من تعقيد تنفيذ هذا المشروع، حتى لو أن عدد الوظائف لا يعكسها بشكل مباشر.

ما هو RCAF؟

هو اختصار لـ:

Relative Complexity Adjustment Factor

أي: عامل تعديل التعقيد النسبي.

GENERAL SYSTEM CHARACTERISTICS	DEGREE OF INFLUENCE
1. Data Communications	
2. Distributed Processing	
3. Performance	
4. Heavily Used Configuration	
5. Transaction Rates	
6. Online Data Entry	
7. Design for End User Efficiency	
8. Online Update	
9. Complex Processing	
10. Usable in Other Applications	
11. Installation Ease	
12. Operational Ease	
13. Multiple Sites	
14. Facilitate Change	
Total Degree of Influence (TDI)	

لماذا نحتاجه؟

لأنه قد يكون عندك نظام فيه نفس عدد الوظائف مثل مشروع آخر،

لكن:

- الأداء المطلوب أعلى.
- النظام موزع على أكثر من موقع.
- حجم البيانات ضخم.
- هناك شروط أمان عالية.
- أو صعوبات تركيب وصيانة.

كل هذه العوامل نريد أن تدخل معنا في الحساب لتعطي صورة أدق.

كيف نحسب RCAF؟

- هناك 14 عامل (الجدول في الصفحة السابقة)
- كل عامل يأخذ درجة من 0 إلى 5 حسب تأثيره.
- نجمع الدرجات كلها لنحصل على قيمة RCAF.

أمثلة على بعض العوامل:

الشرح البسيط	العامل
هل هناك متطلبات استجابة سريعة ومعالجة ضخمة؟	الأداء(Performance)
هل النظام موزع على أكثر من خادم أو موقع؟	البيانات الموزعة(Distributed Processing)
هل النظام يحتاج سياسات أمان معقدة؟	الأمان(Security)
هل النظام يسمح بالتعديل المباشر للبيانات بشكل دائم؟	التحديثات المباشرة(Online Update)
هل النظام مصمم ليسهل تطويره لاحقًا؟	سهولة الصيانة(Facilitate Change)

مثال عددي:

لو بعد تقييم كل العوامل جمعنا:

RCAF=41

الخطوة 3: حساب Function Point النهائي(FP)

نطبق على مثالنا:

CFP = 96

RCAF = 41

حسب القانون التالي: FP = CFP × (0.65 + 0.01 × RCAF)

FP = 96 × (0.65 + 0.01 × 41)

FP = 96 × (0.65 + 0.41)

FP = 96 × 1.06 = 101.76 **نقطة وظائف**

ماذا تمثل لنا هذه الـ Function Points؟

- هذا الرقم الآن يمثل **الحجم الكامل للنظام** بعد مراعاة الوظائف وتعقيد التنفيذ.
- ومنه نستطيع لاحقاً:
  - تقدير ساعات العمل.
  - تقدير عدد المبرمجين المطلوبين.
  - تقدير الزمن الكلي للمشروع.
  - تقدير الكلفة المادية.

بهذا نكون أنهينا الحساب الأساسي.

الآن ننتقل إلى المرحلة الأخيرة:

المحور السادس: كيف نحول Function Point إلى جهد زمني أو أسطر كود أو تكلفة مالية

بعد أن حسبنا حجم المشروع باستخدام النقاط الوظيفية(FP) ، نحتاج الآن لتحويل هذا الرقم إلى أرقام قابلة للاستخدام في التخطيط:

- كم ساعة عمل نحتاج؟
- كم شخص يجب أن يعمل على المشروع؟
- كم سطر كود نتوقع؟
- كم ستكون تكلفة المشروع؟

1. أولاً: تحويل Function Point إلى ساعات عمل (Person-Hours)

هذا يعتمد على ما يسمى بـ:

Productivity Rate أو معدل إنتاجية الفريق

أي: كم نقطة وظيفة يمكن للفريق (أو للفرد) أن ينجزها في الساعة أو في اليوم

مثال:

لو كانت إنتاجية الفريق FP = 5 في اليوم الواحد (8 ساعات)

إذن:

$$\text{الزمن المطلوب} = \frac{FP}{\text{معدل الانتاج اليومي}} = \frac{100}{5} = 20 \text{ يوم عمل}$$

أو:

$$20 \text{ يوم} \times 8 \text{ساعات} = 160 \text{ساعة عمل}$$

ملاحظة:

معدل الإنتاجية يختلف حسب نوع المشروع، كفاءة الفريق، واستخدام أدوات مساعدة.

في الشركات يتم حساب هذا المعدل بناءً على مشاريع سابقة.

2. ثانيًا: تحويل Function Point إلى LOC أو KLOC (Kilo line of code) أسطر كود

تُستخدم هذه الخطوة عندما نحتاج أن نقدّر:

- حجم الكود المتوقع.
- حجم الملفات.
- أو لمقارنة مع مشاريع قديمة مكتوبة بلغة معينة.

وتُسمى هذه العملية بـ Backfiring

كيف يتم التحويل؟

لكل لغة برمجة، يوجد معدل تقريبي لعدد أسطر الكود التي تُنتجها نقطة وظيفة واحدة

Programming Language	LOC per Function point avg.	median	low	high
Ada	154	-	104	205
Assembler	337	315	91	694
C	162	109	33	704
C++	66	53	29	178
COBOL	77	77	14	400
Java	63	53	77	-
JavaScript	58	63	42	75
Perl	60	-	-	-
PL/1	78	67	22	263
Powerbuilder	32	31	11	105
SAS	40	41	33	49
Smalltalk	26	19	10	55
SQL	40	37	7	110
Visual Basic	47	42	16	158

مثال:

لو أن المشروع يحتوي على:

FP=100

ونريد معرفة كم LOC سنحتاج بـ لغة C++:

LOC=100×64=6400 **سطر برمجي** ← 6.4 KLOC

3. ثالثًا: تحويل Function Point إلى تكلفة مالية

إذا كنا نعرف تكلفة كل ساعة عمل (مثلاً: 15 دولار/ساعة)،

نستخدم الناتج السابق لحساب التكلفة:

التكلفة = عدد ساعات العمل × تكلفة الساعة

مثال:

- **عدد الساعات** = 160
- **تكلفة الساعة** = 15 دولار

التكلفة الكلية = 160 ×15 =2400 دولار

ملاحظات ختامية:

وظيفة Function Point ليست فقط في التقدير الأكاديمي، بل تُستخدم فعلياً في العقود والمناقصات الدولية.

تُعتبر وسيلة عادلة للمقارنة بين عروض الشركات المختلفة.

تُستخدم في مؤسسات كبرى مثل IBM ، HP، والمراكز الحكومية.

نقاط الوظائف (Function Points) **لا تعطينا فقط حجم النظام، بل تعطينا وسيلة علمية لتحويل هذا الحجم إلى جهد حقيقي، تكلفة مالية، وعدد موارد — وهي بذلك تربط بين الجانب التقني والإداري في هندسة البرمجيات**

انتهت المحاضرة