CS xxxx: ວິສະວະກຳ ຊອບແວຣ໌
2011-2012

ການບໍລິຫານການຜະລິດຊອບແວຣ໌

ບົດທີ 5

ການປະເມີນຕົ້ນທຶນຂອງຊອບແວຣ໌

Software Cost Estimation

ເນື້ອໃນຫຍໍ້

- ການປະເມີນຕົ້ນທຶນຂອງຊອບແວຣ໌
- ການປະເມີນຂະໜາດຂອງຊອບແວຣ໌
  - ການນັບຈຳນວນແຖວຂອງໂຄດ
  - ການນັບຈຳນວນ Function
- ເທັກນິກການປະເມີນຕົ້ນທຶນ ແລະ ຄວາມພະຍາຍາມ
- ເທັກນິກການປະເມີນແບບ COCOMO

ການປະເມີນຕົ້ນທຶນຂອງຊອບແວຣ໌

- ການປະເມີນຕົ້ນທຶນຂອງຊອບແວຣ໌ເປັນກິດຈະກຳທີ່ສຳຄັນທີ່ສຸດໃນການວາງແຜນໂຄງການ
- ເປັນການປະມານຄ່າໃຊ້ຈ່າຍທີ່ເກີດຂຶ້ນທັງໝົດໃນການຜະລິດຊອບແວຣ໌ເພື່ອເອົາມາເປັນຕົ້ນທຶນຂອງຊອບແວຣ໌ ແລ້ວນຳໄປປະເມີນລາຄາຂອງຊອບແວຣ໌
- ຄ່າໃຊ້ຈ່າຍທີ່ສຳຄັນທີ່ສຸດແມ່ນຄ່າແຮງງານ (Effort), ຄ່າໃຊ້ຈ່າຍໃນການຊື້ອັດຖຸດິບຕ່າງໆ
- ຕົ້ນທຶນຂອງໂຄງການແມ່ນຕົ້ນທຶນຂອງການຜະລິດຊອບແວຣ໌ລວມກັບຕົ້ນທຶນອື່ນໆນຳ

ການປະເມີນຕົ້ນທຶນຂອງຊອບແວຣ໌

- ຕົ້ນທຶນຂອງໂຄງການປະກອບດ້ວຍ:
  - ຄ່າໃຊ້ຈ່າຍດ້ານ Hardware, Software ແລະ ການບຳລຸງຮັກສາ
  - ຄ່າໃຊ້ຈ່າຍໃນການເດີນທາງ ແລະ ການຝຶກອົບຮົມ
  - ຄ່າໃຊ້ຈ່າຍເປັນຄ່າແຮງງານ
    - ເງິນເດືອນບຸກຄະລາກອນ
    - ຄ່າໃຊ້ຈ່າຍໃນການກະກຽມງານ
    - ຄ່າໃຊ້ຈ່າຍໃນການບໍລິຫານ
    - ຄ່າຕິດຕໍ່ສື່ສານ
    - ຄ່າສະຫວັດດີການສັງຄົມ

ການປະມານຂະໜາດຂອງຊອບແວຣ໌

- ການຄຳນວນຫາຕົ້ນທຶນແມ່ນໃຊ້ຄ່າປະສິດທິຜົນໃນການເຮັດວຽກ (Productivity)
- ປະສິດທິຜົນຂອງການເຮັດວຽກສາມາຄຳນວນໄດ້ຈາກຈຳນວນຂອງວຽກທີ່ເຮັດ (Size) ຫານດ້ວຍຈຳນວນເວລາທີ່ຕ້ອງການໃນການຜະລິດ (Effort) ຊຶ່ງອາດມີຫົວໜ່ວຍເປັນ Person-Hours, Man-Day, Man-Month

$$\text{Productivity} = \text{Size}/\text{Effort}$$

- ການອັດແທກຂະໜາດຂອງຊອບແວຣ໌ນັ້ນມີ 2 ປະເພດຄື:
  - ນັບຈຳນວນແຖວຂອງໂຄດ ແລະ ນັບຈຳນວນ Function (Line of Code: LoC and Function Point:FP)

ການປະມານຂະໜາດຂອງຊອບແວຣ໌

- ການນັບຈຳນວນແຖວຂອງໂຄດ
  - Simple Line Count ເປັນວິທີນັບໂຄດທຸກແຖວທີ່ມີຢູ່ໃນ Source File
  - Physical Line (LINES) ບໍ່ນັບແຖວທີ່ເປັນນິຍາມຂອງຕົວປ່ຽນ
  - Physical Line of Code ບໍ່ນັບຈຳນວນແຖວຫວ່າງ ແລະ comment
  - Logical Lines of Code (LLOC) ຄ້າຍຄືກັບວິທີແບບ physical ແຕ່ແຕກຕ່າງຢູ່ບ່ອນວ່າ Logical ນັ້ນຈະນັບແຖວທີ່ມີການເຊື່ອມຕໍ່ກັນດ້ວຍເຄື່ອໝາຍ “_” ເປັນແຖວດຽວກັນ
  - Statements (STMT) ເປັນການນັບຈຳນວນປະໂຫຍກຄຳສັ່ງ

ການປະມານຂະໜາດຂອງຊອບແວຣ໌

- ການນັບຈຳນວນແຖວຂອງໂຄດ
  - ການວັດແທກຂະໜາດຂອງຊອບແວຣ໌ດ້ວຍການນັບຈຳນວນແຖວເຫັນວ່າຍັງບໍ່ທັນຖືກຕ້ອງເທົ່າທີ່ຄວນ ຫາກຕ້ອງການປຽບທຽບໂຄດທີ່ຂຽນຈາກພາສາໂປຣແກຣມທີ່ແຕກຕ່າງກັນ
  - ຈຳນວນແຖວທີ່ນັບໄດ້ຂຶ້ນຢູ່ກັບພາສາຂຽນໂປຣແກຣມທີ່ເລືອກໃຊ້ ແລະ ຄຸນນະພາບໃນການອອກແບບໂປຣແກຣມ

ການປະມານຂະໜາດຂອງຊອບແວຣ໌

- ການນັບຈຳນວນແຖວຂອງໂຄດ
  - ຕົວຢ່າງການຄຳນວນຫາ Productivity ຂອງຜູ້ຂຽນໂປຣແກຣມ

	ວິເຄາະ	**ອອກແບບ**	**ຂຽນໂປຣແກຣມ**	**ທົດສອບ**	**ສ້າງເອກະສານ**
ໂຄດ Assembly	4 ອາທິດ	6 ອາທິດ	10 ອາທິດ	12 ອາທິດ	2 ອາທິດ
ໂຄດພາລະດັບສຸງ	4 ອາທິດ	6 ອາທິດ	5 ອາທິດ	7 ອາທິດ	2 ອາທິດ

	Size	**Effort**	**Productivity**
ໂຄດ Assembly	6000 ແຖວ	34 ອາທິດ	705 ແຖວ/ເດືອນ
ໂຄດພາລະດັບສຸງ	2500 ແຖວ	24 ອາທິດ	416 ແຖວ/ເດືອນ

ການປະມານຂະໜາດຂອງຊອບແວຣ໌

- ການນັບຈຳນວນ Function (Function Point: FP)
  - ເປັນການວັດແທກຂະໜາດຂອງຊອບແວຣ໌ຕາມຈຳນວນ function ຂອງໂປຣແກຣມຈາກຂໍ້ກຳໜົດຄວາມຕ້ອງການ
  - ບໍ່ຂຶ້ນກັບພາສາຂຽນໂປຣແກຣມທີ່ເລືອກໃຊ້ ແລະ ການອອກແບບ
  - ມີສູດດັ່ງນີ້:

$$FP = UFP \times VAF$$

  - ຈາກສູດ, ຈຳນວນ Function ຄຳນວນໄດ້ຈາກຄ່າ FP ທີ່ບໍ່ທັນໄດ້ຖືກປັບແຕ່ງ (Unadjusted Function Point : UFP) ຄູນກັບຄ່າປັດໃຈຄຸນລັກສະຂອງລະບົບ (Value Adjustment Factor: VAF)

ການປະມານຂະໜາດຂອງຊອບແວຣ໌

- ການນັບຈຳນວນ Function (Function Point: FP)
  - ການຄຳນວນຫາ FP ທີ່ຍັງບໍ່ທັນໄດ້ປັບແຕ່ງ (UFP)
    - ຈຳແນກປະເພດຂອງ Function ໂດຍແບ່ງເປັນ 5 ປະເພດຄື: Internal Logical Files (ILF), External Interface Files (EIF), External Inputs (EI), External Outputs (EO) ແລະExternal Queries (EQ)
    - Function ແຕ່ລະປະເພດເກີດຈາກການປະມວນຜົນລາຍການຂໍ້ມູນ (Transaction) ຂອງຜູ້ໃຊ້ ຈຶ່ງມີຄວາມຊັບຊ້ອນແຕກຕ່າງກັນຕາມ ຈຳນວນຂອງຂໍ້ມູນ (Data Element Type: DET), Record (Record Element Type: RET) ແລະ File ທີ່ກ່ຽວຂອງ (File Type Reference: FTR) ທີ່ປະກອບເປັນ Transaction ນັ້ນ

ການປະມານຂະໜາດຂອງຊອບແວຣ໌

- ການນັບຈຳນວນ Function
  - ການຄຳນວນຫາ FP ທີ່ຍັງບໍ່ທັນໄດ້ປັບແຕ່ງ (UFP)
    - ຈຳແນກປະເພດຂອງ Function

Functions	ລາຍລະອຽດ
External Inputs (EI)	ຂໍ້ມູນທີ່ຮັບເຂົ້າມາໃນລະບົບເພື່ອເອົາໄປ update ໃນ ILF
External Output (EO)	ຂໍ້ມູນທີ່ເປັນຜົນໄດ້ຮັບຈາກການປະມວນຜົນອອກໄປສະແດງ
External Queries(EQ)	ຂະບວນການດຶງຂໍ້ມູນແລະການປະມວນຜົນເພື່ອສະແດງຜົນຕໍ່ຜູ້ໃຊ້
Internal Logical Files (ILF)	ເປັນ File ທີ່ກ່ຽວຂ້ອງກັບຂໍ້ມູນທີ່ຢູ່ໃນລະບົບຕະລອດອາຍຸການໃຊ້ງານລະບົບທີ່ຖືກບຳລຸງຮັກສາ ແລະ update ຈາກ EI
External Interface Files (EIF)	ເປັນ File ທີ່ກ່ຽວຂ້ອງກັບຂໍ້ມູນທີ່ໃຊ້ເພື່ອການອ້າງອີງ ແລະ ໃຊ້ຮ່ວມກັບລະບົບອື່ນ

ການປະມານຂະໜາດຂອງຊອບແວຣ์

- ການນับจำนอม Function (Function Point: FP)
  - ການคำนอมหา FP ที่ยังบ่ทันได้ปับแต่ง (UFP)
    - สะนั้น ການนับ Function แต่ละปะเพด จิ่งต้องนับจำนอม DET, RET และ FTR ที่ก่ยงอ้อง แล้วเอ๊ามาหยุบกับละดับถอามซับซ้อมของ Function ที่แบ่งเป้น 3 ปะเพดถื๊: Low, Average, High
    - ให้นับอ่าแต่ละปะเพด Function มีละดับ Low, Average และ High จำนอมเท่าใด แล้วเอ๊ามาถุมกับ ต๊อถ่องหมากของแต่ละละดับใม Function แต่ละปะเพด
    - จากมั้นให้คำนอมหาจำนอมลอมทั้งหมดที่นับได้ก่จะได้ค่า UFP

ການປະມານຂະໜາດຂອງຊອບແວຣ์

- ການນับจำนอม Function (Function Point: FP)
  - ສะแดงละดับถอามซับซ้อมของ Function EI, EO, EQ

EI

FTR	DET		
	1-4	5-15	>15
< 2	Low	Low	Average
2	Low	Average	High
> 2	Average	High	High

(a)

EO, EQ

FTR	DET		
	1-5	6-19	>19
< 2	Low	Low	Average
2/3	Low	Average	High
> 3	Average	High	High

(b)

ການປະມານຂະໜາດຂອງຊອບແວຣ໌

- ການນັບຈຳນວນ Function (Function Point: FP)
  - ສະແດງລະດັບຄວາມຊັບຊ້ອນຂອງ Function ILF ແລະ EIF

ILF, EIF

RET	DET		
	1-19	20-50	>50
1	Low	Low	Average
2-5	Low	Average	High
> 5	Average	High	High

(c)

ສົມມຸດວ່າ ຂໍ້ມູນສິນຄ້າທີ່ຈະເອົາເຂົ້າໄປໃນລະບົບ (EI) ກ່ຽວຂ້ອງກັບ file 2 ຊະນິດ (FTR) ແລະ ຂໍ້ມູນສິນຄ້ານີ້ປະກອບດ້ວຍ field ຂໍ້ມູນບໍ່ເກີນ 15 field(DET) ເມື່ອທຽບກັບຕາຕະລາງ EI ແລ້ວ ເຫັນວ່າ function EI ມີລະດັບຄວາມຊັບຊ້ອນເທົ່າກັບ Average, ແຕ່ຂໍ້ມູນທີ່ຈະເອົາເຂົ້າໄປໃນລະບົບທັງໝົດມີ 10 ຊະນິດ ເມື່ອປະເມີນແລ້ວເຫັນວ່າ ຢູ່ໃນລະດັບ Low 2 ຊະນິດ, ລະດັບ Average 5 ຊະນິດ ແລະ ລະດັບ High 3 ຊະນິດ

ການປະມານຂະໜາດຂອງຊອບແວຣ໌

- ການນັບຈຳນວນ Function (Function Point: FP)
  - ສະແດງຕາຕະລາງຕົວຖ່ວງໜັກ ແລະ ການຄຳນວນ UFP

ປະເພດຂອງ Function	ບັນທັດຖານຄວາມຊັບຊ້ອນ			ລວມ
	Low	Average	High	
EI	2 x 3 = 6	5 x 4 = 20	3 x 6 = 18	44
EO	__ x 4 = __	__ x 5 = __	__ x 7 = __	
EQ	__ x 3 = __	__ x 4 = __	__ x 6 = __	
ILF	__ x 7 = __	__ x 7 = __	__ x 15 = __	
EIF	__ x 5 = __	__ x 10 = __	__ x 10 = __	
	ລວມ UFP			

(d) = (a) + (b) + (c)

ການປະມານຂະໜາດຂອງຊອບແວຣ໌

- ການນັບຈຳນວນ Function (Function Point: FP)
  - ຄຳນວນຄ່າປັດໃຈຄຸນລັກສະນະຂອງລະບົບ (VAF)
    - ປັດໃຈທີ່ສົ່ງຜົນຕໍ່ຄວາມແຕກຕ່າງລະຫວ່າງກັນຂອງແຕ່ລະລະບົບປະກອບດ້ວຍຄຸນລັກສະນະເດັ່ນທັງໝົດ 14 ດ້ານ
    - ໃຫ້ກຳໜົດລະດັບອິດທິພົນຂອງຄຸນລັກສະນະໃນແຕ່ລະດ້ານວ່າມີຄວາມກ່ຽວຂ້ອງກັບລະບົບຫລາຍປານໃດ ໂດຍມີຄ່າຕັ້ງແຕ່ 0 ເຖິງ 5 (0 ບໍ່ກ່ຽວຂ້ອງ, 5 ກ່ຽວຂ້ອງຫຼາຍທີ່ສຸດ)
    - ໃຫ້ລວມລະດັບອິດທິພົນທັງ 14 ດ້ານເຂົ້າກັນ ແລ້ວເອົາມາຄຳນວນ VAF ຕາມສູດດັ່ງນີ້

$$VAF = 0.65 + (0.01 \times \text{ຜົນບວກຂອງຄຸນລັກສະນະ 14 ດ້ານ})$$

ການປະມານຂະໜາດຂອງຊອບແວຣ໌

↳ ຄຳນວນຄ່າປັດໃຈຄຸນລັກສະນະຂອງລະບົບ (VAF)

ລ/ດ	ຄຸນລັກສະນະ	ຄ່າ	ລ/ດ	ຄຸນລັກສະນະ	ຄ່າ
1	ການຕິດຕໍ່ສື່ສານຂໍ້ມູນ (Data Communication)		8	ການປັບປຸງຂໍ້ມູນແບບ Online (Online Update)	
2	ການປະມວນຜົນຂໍ້ມູນແບບກະຈາຍ (Distributed Data Processing)		9	ຄວາມຊັບຊ້ອນຂອງການປະມວນຜົນ (Complex Processing)	
3	ປະສິດທິພາບຂອງລະບົບ (Performance)		10	ການນຳໄປໃຊ້ຄືນໄດ້ (Reusability)	
4	ການປ່ຽນແປງແກ້ໄຂຄ່າຂອງລະບົບ (Configuration)		11	ຄວາມງ່າຍໃນການຕິດຕັ້ງ (Installation Ease)	
5	ປະລິມານລາຍການຂໍ້ມູນ (Transaction)		12	ຄວາມງ່າຍໃນການດຳເນີນງານ (Operational Ease)	
6	ການປ້ອນຂໍ້ມູນເຂົ້າສູ່ລະບົບແບບ Online (Online Data Entry)		13	ການໃຊ້ງານໄດ້ຫລາຍ site (Multiple Site)	
7	ປະສິດທິພາບການໃຊ້ງານຂອງຜູ້ໃຊ້ (End-user Efficiency)		14	ຮອງຮັບການປ່ຽນແປງຄວາມຕ້ອງການຂອງຜູ້ໃຊ້ (Change Requirement)	

ການປະມານຂະໜາດຂອງຊອບແວຣ໌

- ການນັບຈຳນວນ Function (Function Point: FP)
  - ຄຳນວນຄ່າ FP ທີ່ປັບແຕ່ງແລ້ວ
    - ເມື່ອຄຳນວນ UFP ແລະ VAF ໄດ້ແລ້ວ ເອົາມາຄູນກັນຈະໄດ້ຄ່າ ຂອງ FP ທີ່ປັບແຕ່ງແລ້ວຕາມຄຸນລັກສະນະຂອງລະບົບ ຕາມສູດ ດັ່ງນີ້

$$FP = UFP \times VAF$$

ເທັກນິກການປະເມີນຕົ້ນທຶນ ແລະ ຄວາມພະຍາຍາມ

- ການປະເມີນຕົ້ນທຶນ ແລະ Effort ທີ່ດີຈະຕ້ອງໃຫ້ໄກ້ຄຽງກັບຄວາມເປັນຈິງ ຊຶ່ງເປັນການເຮັດໄດ້ຍາກ ສະນັ້ນ ຈຶ່ງໄດ້ມີການຄິດຄົ້ນເທັກນິກການປະເມີນຂຶ້ນມາຫຼາຍແບບດັ່ງນີ້

ເທັກນິກ	ລາຍລະອຽດ
Algorithmic Cost Modeling	ການໃຊ້ແບບຈຳລອງທາງຄະນິດສາດເພື່ອປະເມີນໂດຍແບບຈຳລອງນັ້ນຖືກພັດທະນາມາຈາກການລວບລວມຂໍ້ມູນຕົ້ນທຶນຈິງໃນອະດີດທີ່ມີຄວາມສຳພັນກັບການວັດແທກບາງຢ່າງຂອງຊອບແວຣ໌ ເຊັ່ນ: ຂະໜາດຂອງມັນ
Expert Judgement	ການໃຊ້ຄວາມເຫັນຜູ້ຊ່ຽວຊານໃນການປະເມີນ ປຽບທຽບກັບຂໍ້ມູນໃນອາດີດ ເພື່ອປຶກສາ ແລະ ຕົກລົງກຳໜົດຕົ້ນທຶນຮ່ວມກັນ
Estimation by Analogy	ການປະເມີນດ້ວຍການວິເຄາະ ໂດຍອາໃສຂໍ້ມູນຈາກໂຄງການໃນທຸລະກິດດຽວກັນທີ່ເຮັດປະສົບຜົນສຳເລັດມາແລ້ວເປັນຂໍ້ມູນຫລັກໃນການວິເຄາະ
Parkinson's Law	ເປັນການແຈກຢາຍວຽກໃຫ້ກັບບຸກຄະລາກອນຕາມໄລຍະເວລາທີ່ມີຢູ່
Pricing to Win	ການປະເມີນເພື່ອໃຫ້ຊະນະການປະມູນ

ເທັກນິກການປະເມີນແບບ COCOMO

- COCOMO (Constructive Cost Model) ເປັນແບບຈຳລອງການປະເມີນຕົ້ນທຶນ, Effort ແລະ ການຈັດຕາຕະລາງເຮັດວຽກ
- ແບບຈຳລອງດັ່ງກ່າວແມ່ນພິຈາລະນາຈາກຂະໜາດຂອງຊອບແວຣ໌, ຄຸນລັກສະນະຂອງຊອບແວຣ໌ທີ່ຜູ້ໃຊ້ຕ້ອງການ ແລະ ປັດໃຈແວດລ້ອມອື່ນໆ ເຊັ່ນ: ຄວາມແນ່ນອນຂອງຂະບວນການ ແລະ ຄວາມສາມາດໃນການຜະລິດຊອບແວຣ໌ຂອງທີມງານ, ຄວາມຢືດຢຸ່ນ, ຄວາມສ່ຽງ ແລະ ວິທີການຈັດການຄວາມສ່ຽງ ເປັນຕົ້ນ
- ສາມາດຄຳນວນແບບ Exponential

ເທັກນິກການປະເມີນແບບ COCOMO

- COCOMO ໄດ້ຖືກພັດທະນາເປັນລຸ້ນທີ່ 2 ໃນປີ 1997 ໂດຍ ລວບລວມຂໍ້ມູນຈາກໂຄງການທັງໝົດ 161ໂຄງການ
- COCOMO II ໄດ້ແບ່ງແບບຈຳລອງອອກເປັນ 3 ຊະນິດເພື່ອ ປະເມີນຕາມໄລຍະຕ່າງໆຂອງພັດທະນາຊອບແວຣ໌
  1. Application-composition Model ເໝາະສົມກັບການຜະລິດ ຊອບແວຣ໌ແບບ Component ແລະໃຊ້ຢູ່ໃນໄລຍະສະຫລຸບ concept ໃນການດຳເນີນງານ, ໃຊ້ Object Point ແທນຂະໜາດ ຂອງຊອບແວຣ໌
  2. Early Design Model ໃຊ້ປະເມີນຢູ່ໃນໄລຍະກ່ອນອອກແບບ ຊອບແວຣ໌ ຫລັງຈາກການກຳໜົດຄວາມຕ້ອງການ, ໃຊ້ FP
  3. Post-architecture Model ໃຊ້ຫລັງການອອກແບບ

ເທັກນິກການປະເມີນແບບ COCOMO

- Application-composition Model
  - ເປັນແບບຈຳລອງຂອງ COCOMO II
  - ເໝາະສົມກັບການຜະລິດຊອບແວຣ໌ດ້ວຍວິທີທາງແບບ component ຊຶ່ງ ແຕ່ລະ component ສາມາດນັບເປັນ Object point ໄດ້ (ຂະໜາດ)
  - ບັນດາ Object component ຈະມີຈຳນວນ Object point ຕ່າງກັນ ຂຶ້ນຢູ່ກັບລະດັບຄວາມຊັບຊ້ອນ ທີ່ແບ່ງອອກເປັນ 3 ລະດັບດັ່ງນີ້

	ງ່າຍ	ຊັບຊ້ອນທຳມະດາ	ຊັບຊ້ອນຫຼາຍ
Screen	1	2	3
Report	2	5	8
3GL Modules	4	10	-

ເທັກນິກການປະເມີນແບບ COCOMO

- Application-composition Model
  - ກໍລະນີ component ຂອງຊອບແວຣ์ຖືກອອກແບບໃຫ้สามาดเอ้ากับมาใช้ใหม่ได้ และ มีการใช้ Library นำ จะต้องเอ้าอัดตาการเอ้ากับมาใช้ใหม่มาลบออกจำนอม Object point ที่นับได้ทั้งหมด ซึ่งเอิ้มว่า Revised Object Point (ROP)

$$ROP = ObjectPoint \times (100-\%reuse)/100$$

  - เอ้า ROP ที่ได้ไปคำนอมหา Effort ดั่งนี้

$$MME = ROP/(Productivity\ constant)$$

MME = ManMonthEffort

ເທັກນິກການປະເມີນແບບ COCOMO

Application-composition Model

- ຕາຕະລາງສະແດງຄ່າຄົງທີ່ປະສິດທິຜົນໃນການຜະລິດຊອບແວຣ໌

ລະດັບປະສົບການ ແລະ ຄວາມສາມາດ	ຕ່ຳຫຼາຍ	ຕ່ຳ	ປານກາງ	ສູງ	ສູງຫຼາຍ
Productivity Constant (NOP per Month)	4	7	13	25	50

- ຕົວຢ່າງ: ໂຄງການ ກ ມີ Object point = 40, ອັດຕາການການເອົາ code ໄປໃຊ້ໃໝ່ແມ່ນ 10% ແລະ ຄວາມສາມາດຂອງທີມງານແມ່ນປານກາງ ຊອກຫາ Effort ທີ່ຕ້ອງການໃຊ້ໃນໂຄງການດັ່ງກ່າວ

ROP = 40 x (100-10)/100
= 40X0.9 = 36

⇨

MME = 36/13
≈ 3 Man-Months

ເທັກນິກການປະເມີນແບບ COCOMO

- Early Design Model
  - ເປັນແບບຈຳລອງຂອງ COCOMO II ທີ່ໃຊ້ໃນໄລຍະກ່ອນການອອກແບບຊອບແວຣ ຊຶ່ງມີສູດດັ່ງນີ້:

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

MME ແມ່ນ Effort ທີ່ມີຫົວໜ່ວຍເປັນ Man Month Effort

A ແມ່ນຄ່າຄົງທີ່ຂອງປະສິດທິຜົນໃນການຜະລິດຊອບແວຣ ໃນລະດັບປານກາງ

B ແມ່ນຄ່າຂອງປັດໃຈທີ່ສົ່ງຜົນກະທົບຕໍ່ Effort ທີ່ເອີ້ນວ່າ Cost Driver

Size ແມ່ນຂະໜາດຂອງຊອບແວຣ์ມີຫົວໜ່ວຍເປັນ KLoC (Kilo of Line of Code = 1000 LoC)

$$B = 0.91 + 0.01 \left(\sum_{1}^{5} Ratings\right)$$

ເທັກນິກການປະເມີນແບບ COCOMO

- Early Design Model
  - Cost Driver ທີ່ເອົາມາໃຊ້ມີທັງໝົດ 5 ປັດໃຈ ຊຶ່ງແບ່ງອອກເປັນ 4 ລະດັບ ດັ່ງຕາຕະລາງ:

Factor Code	ຕ່ຳຫຼາຍ	ຕ່ຳ	ປານກາງ	ສູງ	Factor Name
PREC	6.20	4.96	3.72	2.48	Precedentness
FLEX	5.07	4.05	3.04	2.03	Flexibility
RESL	7.07	5.65	4.24	2.83	Risk Resolution
TEAM	5.48	4.38	3.29	2.19	Team Cohesion
PMAT	7.80	6.24	4.68	3.12	Process Maturity

  - ໂດຍເບື້ອງຕົ້ນຈະຕ້ອງປະເມີນລະດັບແລະໃຫ້ຄະແນນແຕ່ລະປັດໃຈ ແລ້ວຈຶ່ງຄຳນວນຫາຄ່າຂອງ Cost Driver ຈາກສູດຂ້າງເທິງ

ເທັກນິກການປະເມີນແບບ COCOMO

Early Design Model

- ຄວາມໝາຍຂອງແຕ່ລະປັດໃຈ

Factor	ຄວາມໝາຍ
PREC	ຄວາມຄ້າຍຄືກັນຂອງຊອບແວຣ໌ໃໝ່ກັບຊອບແວຣ໌ທີ່ເຄີຍພັດທະນາມາແລ້ວ ຖ້າຄ້າຍຄືກັນຫຼາຍຈະມີຄ່າໜ້ອຍ ໝາຍຄວາມວ່າມັນສົ່ງຜົນກະທົບໜ້ອຍ ແລະ ກົງກັນຂ້າມ
FLEX	ການອັດແທກລະດັບຄວາມຢືດຢຸ່ນໃນການບໍລິຫານຈັດການ ແລະ ດຳເນີນໂຄງການ
RESL	ການອັດແທກລະດັບຄວາມສາມາດໃນການຈັດການຫຼືຄວບຄຸມຂອງອົງກອນຫຼືທີມງານໂຄງການ
TEAM	ການອັດແທກລະດັບການເຮັດວຽກເປັນທີມຂອງອົງກອນຫຼືທີມງານໂຄງການ
PMAT	ການອັດແທກລະດັບວຸດທິຄວາມສາມາດຂອງອົງກອນຫຼືທີມງານໂຄງການ ແຕ່ລະດັບຕ່ຳສຸດແມ່ນ 1 ຈົນເຖິງລະດັບສູງສຸດແມ່ນ 5

ເທັກນິກການປະເມີນແບບ COCOMO

Early Design Model

- ຕົວຢ່າງ: ສົມມຸດວ່າ ປັດໃຈທັງຫ້າຖືກຈັດໃຫ້ຢູ່ໃນລະດັບຕ່ຳຫຼາຍ ແລະ ກຳໜົດໃຫ້ຂະໜາດຂອງຊອບແວຣ໌ທີ່ປ່ຽນມາເປັນ LoC ແລ້ວເທົ່າກັບ 10 KLoC. ໃຫ້ຄຳນວນຫາແຮງງານໂດຍປະມານບົນພື້ນຖານຄ່າຄົງທີ່ຂອງປະສິດທິຜົນໃນການຜະລິດໃນລະດັບປານກາງ
- ຄຳນວນຫາ Cost Driver B ດັ່ງນີ້:

$$B = 0.91 + 0.01\,(6.20+5.07+7.07+5.48+7.80)$$
$$= 0.91 + 0.01 \times 31.62$$
$$= 1.2262$$

- $\Rightarrow B > 1 \Rightarrow$ Cost Driver ມີຜົນກະທົບຕໍ່ຂະໜາດຂອງຊອບແວຣ໌ ແລະ Effort
- ຄຳນວນຫາ Effort ດັ່ງນີ້:

$$MME = A \times (Size)^{B} = 13 \times (10)^{1.2262} = 13 \times 16.8344$$
$$= 218.84$$ ຫຼື ປະມານ 219 Man-Month

ເທັກນິກການປະເມີນແບບ COCOMO

- Post-architecture Model
  - ນອກຈາກ Cost Driver ທັງ 5 ຍັງມີປັດໃຈອື່ນທີ່ມີຜົນກະທົບຕໍ່ Effort ທັງໃນດ້ານຄຸນລັກສະນະຂອງຜະລິດຕະພັນ (Product Factor), ດ້ານ Platform, ດ້ານບຸກຄະລາກອນ (Personnel Factor)
  - ປັດໃນທີ່ມີຜົນກະທົບລວມທັງໝົດ 16 ປັດໃຈ ທີ່ໃຫ້ຄ່າຄະແນນເປັນລະດັບຕ່ຳຫຼາຍ, ຕ່ຳ, ປານກາງ, ແລະ ສູງ
  - ເອົາຄະແນນລະດັບທີ່ປະເມີນທັງ 16 ປັດໃຈມາຄູນກັນຈະໄດ້ Effort Multiplier ດັ່ງນີ້:

$$MME\ (Modified) = MME \times EM$$

EM ແມ່ນ Effort Multiplier ທີ່ເປັນຜົນຄູນຂອງບັນດາປັດໃຈທີ່ສົ່ງຜົນກະທົບ

EI

FTR	DET		
	1-4	5-15	>15
< 2	Low	Low	Average
2	Low	Average	High
> 2	Average	High	High

EO, EQ

FTR	DET		
	1-5	6-19	>19
< 2	Low	Low	Average
2/3	Low	Average	High
> 3	Average	High	High

RET	DET		
	1-19	20-50	>50
1	Low	Low	Average
2-5	Low	Average	High
> 5	Average	High	High

ປະເພດຂອງ Function	ບັນທັດຖານຄວາມຊັບຊ້ອນ			ລວມ
	Low	Average	High	
EI	2 x 3 = 6	5 x 4 = 20	3 x 6 = 18	44
EO	__ x 4 = __	__ x 5 = __	__ x 7 = __	
EQ	__ x 3 = __	__ x 4 = __	__ x 6 = __	
ILF	__ x 7 = __	__ x 7 = __	__ x 15 = __	
EIF		__ x 10 = __	__ x 10 = __	
	ລວມ UFP			