Stima dei costi del Software

Riferimenti

- Sommerville, Capitolo 26 (stima dei costi)

I problemi fondamentali della stima

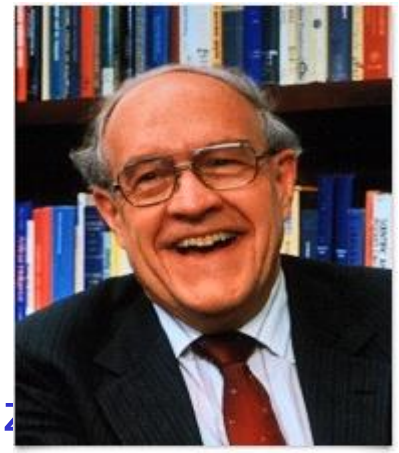
- Quanto vale lo **sforzo** richiesto per completare una attività?
- Quanto **tempo** (di calendario) è necessario per completare una attività?
- Qual è il **costo** totale di una attività?

- La stima dei costi e la tempistica di un progetto sono in genere eseguite insieme.

Metriche di sforzo

- Per sforzo (effort) si intende la misura del quantitativo di lavoro umano necessario per svolgere un lavoro relativo ad un processo di sviluppo software
- L'unità di misura più comunemente usata è il mese/uomo (man-month) e tutti i suoi multipli e sottomultipli
  - Un mese/uomo si definisce, teoricamente, come il quantitativo di lavoro che un singolo dipendente può svolgere in un mese lavorativo

The Mythical Man-Month



Frederick Brooks
*1931

- Problemi della definizione del mese-uomo
  - Ogni persona ha una sua diversa produttività, che può variare nel tempo (con l'età e la conoscenz ad esempio)
  - La produttività di una persona dipende dalla complessità del problema da risolvere
  - Aumentando la dimensione del gruppo di lavoro aumenta proporzionalmente la produttività?
    - Se una persona può svolgere 1 man-month di lavoro in un mese, quanto lavoro potranno svolgere 10 persone nello stesso mese?
- Tutti questi problemi furono espressi compiutamente per la prima volta da Brooks nel famoso libro The Mythical Man-Month: Essays on Software Engineering, nel 1975
- In conclusione: si tratta di una metrica di facile misura (ma solo a posteriori) ma di difficilissima stima

Brooks (1975). *The Mythical Man-Month*. Addison-Wesley.

I fattori fondamentali dei costi

- Costi dell'Hardware e del software.
- Costi per viaggi e formazione.
- Costi dello sforzo (la voce più rilevante in molti progetti), pari agli stipendi degli ingegneri/programmatori coinvolti.
- Altri costi da considerare…
  - Costi di rifornimento, riscaldamento, illuminazione
  - Costi di tutto il personale a supporto: amministrazione, tecnici, personale delle pulizie, etc.
  - Costi di rete e telecomunicazioni
  - Costi di biblioteche e altri servizi
  - Costi di previdenza sociale
- I costi delle risorse "a contorno" di solito valgono almeno quanto i costi legati agli stipendi delle risorse addette direttamente alla produzione

Costi e Prezzi

- Le stime servono a prevedere i costi di sviluppo di un sistema software.
- Il **prezzo** da pagare dovrebbe essere la somma del **costo** più il **profitto**
- Ma la relazione fra costi di sviluppo e prezzo richiesto al committente non è, in genere, semplice.

- Ci possono essere svariate considerazioni di tipo organizzativo, economico, politico ed aziendali che possono influire sul prezzo che verrà richiesto.

Fattori per la determinazione dei prezzi del Software

Opportunità di Mercato	Una organizzazione può fissare un prezzo di sviluppo più basso per entrare in un nuovo settore di mercato
Incertezza della Stima del Costo	Se si è incerti sui presunti costi di sviluppo, si può maggiorare il prezzo
Termini di contratto	Il prezzo può variare a seconda se il contratto richiede la consegna del codice sorgente o no
Volatilità dei requisiti	Se si prevede che i requisiti cambieranno, si abbassa il prezzo per lo sviluppo, prevedendo di guadagnare anche per la manutenzione.
Salute Finanziaria	Se lo sviluppatore è in crisi finanziaria, può anche abbassare il prezzo, per sopravvivere

Produttività di sviluppo Software

- Per stimare i costi di sviluppo, può essere necessario conoscere la produttività degli ingegneri.
- La produttività è una misura della velocità con cui gli ingegneri producono software e la relativa documentazione.

- Un problema: in tali valutazioni della produttività non si tiene conto della qualità di quanto prodotto!
  - Confrontare la produttività di due team che sviluppano software con diversi requisiti di qualità non è realmente significativo.

Problemi delle misure di produttività

- Tutte le misure di produttività basate sul rapporto volume/unità di tempo sono difettose perchè considerano solo la quantità e non la qualità di ciò che si produce.
- La produttività potrebbe essere ‘aumentata’ a discapito della qualità.

- La metrica di produttività più semplice (LOC/pm), usando le LOC, non è un indicatore valido in assoluto ma dipende dai vari fattori indicati precedentemente.
- Se si usano I FP, il conteggio di FP dipende da valutazioni soggettive sulla complessità del software.

Tecniche per la stima dello sforzo

- Non c'è un modo semplice ed efficace per fare stime accurate nei primi stadi del progetto.
  - Le stime iniziali sono formulate in genere su requisiti definiti solo ad alto livello;
  - Il software può dover funzionare su computer 'sconosciuti' o usando nuove tecnologie;
  - Il personale assegnato al progetto può essere sconosciuto.
- Le tecnologie che cambiano continuamente comportano che le tradizionali tecniche di stima possono non funzionare più correttamente!

Tecniche di Stima tradizionali

- Pricing to win.
- Expert judgement.
- Algorithmic cost modelling.
- Estimation by analogy.
- Parkinson's Law.

Tecniche per la stima dei costi

- Price-to-win estimating
  - Il costo viene stimato in base a quanto il cliente è disposto a spendere. La stima dipende dal budget del cliente più che dalle funzionalità del software
- Giudizio di esperti
  - Esperti sulle tecniche di sviluppo e sul dominio vengono consultati. Il costo del progetto viene stimato da ognuno, vengono quindi comparate e discusse le stime finché non viene trovato un accordo
- Legge di Parkinson
  - “il lavoro si espande per occupare il tempo disponibile” … il costo viene determinato unicamente in base ai vincoli di tempo imposti e alle risorse disponibili
- Stima tramite analogia
  - Si cerca di stimare il costo trovando delle analogie tra il progetto da realizzare e progetti precedentemente realizzati
- Modellazione algoritmica
  - Viene sviluppato un modello sulla base di informazioni storiche e del legame empiricamente trovato tra alcune metriche di prodotto e di processo e il costo del software

Pricing to win

- Il progetto costerà quanto il committente ha intenzione di pagare.
- Vantaggi:
  - Si ottiene il contratto.
- Svantaggi:
  - La probabilità che il cliente ottenga ciò che richiede è piccola…I costi pagati non garantiscono tutto ciò che sarebbe richiesto.
- É usato soprattutto quando non c'è un documento dei requisiti del sistema, ma il committente si fida dell'azienda che fa lo sviluppo, e sa che la cifra offerta rientra nel suo budget.

Approcci di stima Top-down e bottom-up

- Le tecniche precedenti possono essere usate sia in modo top-down che bottom-up.
- Top-down
  - Si parte dall'intero sistema e si analizza l'intera funzionalità e come questa sia via via ottenuta attraverso I vari sottosistemi.
- Bottom-up
  - Si parte dal livello dei singoli componenti e si stima lo sforzo di ciascuno di essi. Gli sforzi dei componenti si sommano per ottenere quello globale.

Stima Top-down

- Richiede solo la conoscenza delle funzionalità del sistema.
- É usabile anche senza alcuna conoscenza dell'architettura del sistema e dei suoi componenti.
- Considera anche I costi di integrazione, configuration management e di documentazione.
- Può sottostimare I costi legati a problemi tecnici di basso livello di specifici componenti.

Stima Bottom-up

- Usabile solo se l'architettura è nota ed I componenti ben identificati.
- Può essere un metodo di stima accurato se il sistema è stato progettato in dettaglio.
- Può sotto-stimare il costo di attività a livello sistema quali l'integrazione e la documentazione.

Uso dei metodi di stima

- Ogni metodo ha I suoi vantaggi e svantaggi, per cui la stima dovrebbe basarsi su differenti metodi.
- Se I vari metodi non forniscono risultati comparabili, si deve concludere che non ci sono dati sufficienti per fare una stima valida.
- Occorrerà raccogliere ulteriori dati sul prodotto, sul processo o il team.
- La tecnica del *Pricing to win* è talora la sola tecnica applicabile.
  - Ci si accorda su un costo di progetto in base ad una bozza di proposta e lo sviluppo sarà vincolato da tale costo.
  - La specifica di dettaglio sarà negoziata successivamente o si userà un approccio di sviluppo evolutivo.

Stima per analogia

- Si basa sul riuso delle esperienze accumulate in precedenti progetti
  - valuta le analogie tra il nuovo progetto e progetti simili svolti in passato
- Si applica quando:
  - Non esiste, in azienda, una metodologia adottata per la stima dei costi
  - E' richiesta una rapida, semplice e ragionevolmente accurata macro stima
  - Esistono informazioni di precedenti progetti ma non un data base storico dettagliato
- Requisiti per l'uso:
  - Dimensione e scopi del nuovo prodotto simile a quelli del prodotto di riferimento
  - Metodo di lavoro del nuovo progetto simile a quello del progetto di riferimento
  - Disponibilità di informazioni molto dettagliate o di persone che ricordano il lavoro in maniera accurata

Stima per analogia

- La stima viene eseguita valutando la quantità di lavoro (in mesi/uomo M) e il tempo di sviluppo (in mesi) previsti
  - applicazione di coefficienti moltiplicativi al costo del progetto di riferimento
    - Modifiers (m) che tengano conto degli scostamenti di determinate caratteristiche (C) da quelle del caso in esame

- Requisiti essenziali nella scelta del progetto di riferimento:
  - Conoscenza di quali caratteristiche sono simili e quali no
  - Conoscenza della qualità dei dati sullo sforzo e i costi di manodopera disponibili per il progetto di riferimento

$$M = M_{rif} \bullet \Pi_i m_i$$

- Le caratteristiche da considerare sono quelle che fanno riferimento direttamente al software

Esempio

- Caratteristiche del sistema più complesse — circa il 15%
- Più menù e schermate — circa il 10%
- Struttura dei file nel data base meno complessa — circa il 5%
- Più interfacce di sistema — circa il 20%
- Uso di DBMS meno familiari — circa il 20%
- Più potenti tool di sviluppo — circa il 10%
- Team di sviluppo con maggiore esperienza — circa il 20%
- Testing di accettazione più rigoroso — circa il 5%

$M = M_{rif} \bullet 1.15 \bullet 1.10 \bullet 0.95 \bullet 1.20 \bullet 1.20 \bullet 0.90 \bullet 0.80 \bullet 1.05 = 1.31 \bullet M_{rif}$

- In teoria nessun limite al numero di modifiers che possono essere usati.
- In pratica conviene non usarne più di 15 in generale e più di 10 se ne esiste più di uno con una variazione percentale maggiore del 30%, altrimenti si perde il concetto stesso di analogia tra sistemi

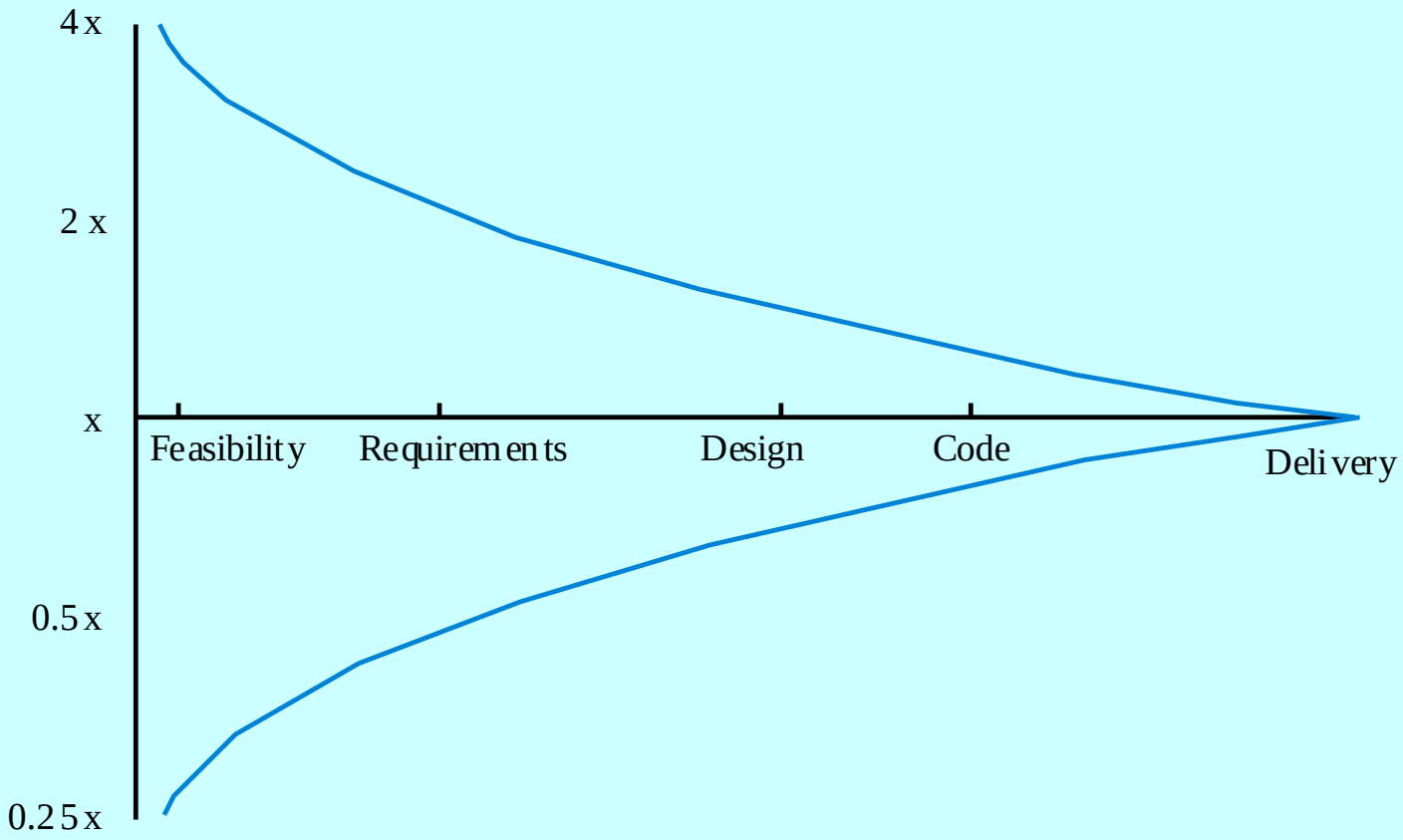
Stima media

- Per tener conto dell'incertezza nella stima dei modificatori, spesso viene utilizzata una metodologia nella quale vengono considerate tre possibili combinazioni di valori dei parametri
  - Corrispondenti a stima ottimistica, pessimistica e media
  - Il costo (medio) viene valutato come stima ottimistica+4*stima attesa+stima pessimistica

Stima per analogia durante lo sviluppo del progetto

- Metodo delle percentuali
  - Da dati storici sono note le percentuali relative ai costi nelle varie fasi del ciclo di vita
  - Noto il costo effettivo di una fase è possibile ottenere una stima via via più precisa del costo finale assumendo che le proporzioni tra i costi siano costanti

Incertezza della stima



Modellazione algoritmica dei costi

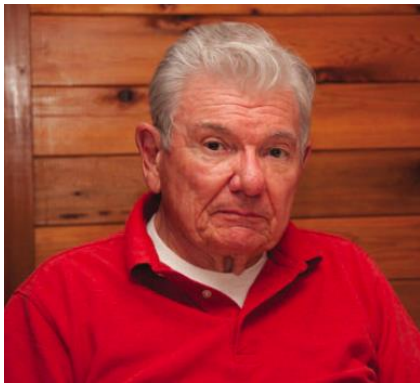
- Il Costo è stimato come una funzione matematica che tiene conto di attributi del prodotto, del processo e del progetto, stimati da project managers:

$$Effort = A \times Size^{B} \times M$$

  - A è una costante dipendente dall'organizzazione
  - B tiene conto della non linearità tra lo sforzo e la dimensione (B>1) indica che lo sforzo aumenta più che linearmente con le dimensioni
  - M è un moltiplicatore dipendente dal prodotto da realizzare, dal processo scelto e dalla tipologia delle persone coinvolte

- L'attributo di prodotto più usato per la stima è la dimensione del codice (Size- espressa in LOC o FP)
- I diversi modelli di stima algoritmica usano valori diversi per A, B ed M. La stima dei fattori che contribuiscono a B e M è soggettiva.

Function Point Analysis



- Tecnica proposta da **Albrecht** [*Measuring Application Development Productivity, 1979*]
- Approccio indipendente dal linguaggio di programmazione per misurare le funzionalità del sistema.
- Può essere usata per:
  - Stimare il costo richiesto per programmare, testare il software
  - Prevedere il numero di errori che si rileveranno durante il testing
  - Prevedere il numero di componenti o di LOC del sistema

Cosa sono i Function Point (FP)

- I FP sono una misura di funzionalità basata su entità logico-funzionali che l'utente facilmente comprende (es. Input, Output,etc.)
- I FP sono pertanto indipendenti dal linguaggio di programmazione, quindi la produttività può essere confrontata tra diversi linguaggi
- Un FP non è una singola caratteristica ma una combinazione di caratteristiche del sistema
  - Destinati a supportare stime, pianificazioni relative a sistemi Sw
  - Nati per essere applicati a Sistemi Sw di tipo Business
  - Misura che può essere effettuata ‘presto’ nel CVS
  - Misura solo i requisiti funzionali

Formulazione originaria del metodo dei Function Point

- I FP vengono derivati usando una relazione empirica che si basa su:
  - Misure dirette (espresse in valori interi) del Dominio delle informazioni del software
  - Valutazione della complessità del software

- Le Misure del Dominio delle Informazioni sono catturate con riferimento al Confine (Boundary) del sistema software.

Boundary

- Secondo il concetto di Boundary, esiste una linea chiusa che definisce il confine del sistema software cui è rivolta la misura.
- Con riferimento al confine, sono misurate o valutate le seguenti caratteristiche del programma:
  - External input (EI)
  - External output (EO)
  - External inquiry (EQ)
  - Internal Logical File (ILF)
  - External Interface File (EIF)

External Input (EI)

- Un EI rappresenta un tipo di input unico (proveniente dall'esterno dell'applicazione) fornito o da un utente o da un altro sistema, che verrà elaborato dall'applicazione.
  - Un input è unico se il formato è unico, o la logica con cui viene elaborato è unica.
- Questi input tipicamente devono aggiungere, cambiare o cancellare dati in un File Logico Interno (ILF).
- I dati possono essere informazioni di controllo o informazioni del dominio applicativo (business information). Se i dati sono informazioni di controllo non devono aggiornare nessun ILF.

External Output (EO)

- Un EO è un tipo di dati unico che viene creato nell'applicazione ed è destinato all'utente, o ad altre applicazioni, attraversando il confine dell'applicazione.
  - Un output è unico se il formato o la logica elaborativa è unica.
- Un EO consiste in report, schermate, file, messaggi d'errore…
- Un EO può addizionalmente aggiornare un ILF.
- Questi reports o files sono creati a partire da uno o più ILF e EIF.

External Inquiry (EQ)

- Una interrogazione esterna EQ è definita come una coppia input/output unica, dove un input online produce la generazione di una risposta immediata del software, sotto forma di output online.
  - Un'interazione è unica se il formato o la logica di elaborazione è unica.
- Una EQ ricerca dati o informazioni di controllo (da uno o più ILF e/o EIF) per una risposta immediata.
- Il processo di input non deve aggiornare nessun ILF, mentre il processo di output non contiene dati “derivati”
- Se c'è un coinvolgimento di updating del file logico interno o una produzione di dati derivati o calcolati, deve essere considerato un input seguito da un output.

Internal Logical File

- Un ILF è un raggruppamento unico di dati logicamente correlati, o di informazioni di controllo, identificabile dall'utente, usato ed aggiornato dall'applicazione.

- I dati relativi non attraversano il confine, ma risiedono internamente al confine dell’applicazione e sono gestiti attraverso gli external input (EI).

External Interface File

- Una EIF è un unico gruppo di dati logicamente correlati, o di informazioni di controllo, identificabile dall'utente che viene usato dall'applicazione.

- I dati relativi attraversano il confine e risiedono interamente all'esterno del confine dell'applicazione e sono mantenuti da un'altra applicazione.

- Un EIF è un ILF di un'altra applicazione.

Calcolo dei Function Point

- Esistono diverse teorie, più o meno complesse per calcolare i FP in funzione del conteggio degli elementi prima descritti
  - Il livello di complessità di ciascun elemento individuato per ognuna delle 5 precedenti categorie viene classificato in:
    - Basso (semplice)
    - Medio
    - Alto (complesso)
  - Ad ognuno dei livelli è associato un peso che varia da 3 (più semplice) a 15 (più complesso).

Matrice dei pesi: un esempio

	Componenti	Livello di complessità		
		Basso	Medio	Alto
Transactional function types	Ext. Input (EI)	3	4	6
	Ext.Output (EO)	4	5	7
	Ext. Inquiry (EQ)	3	4	6
Data function types	Internal Logical files (ILF)	7	10	15
	External Interface files (EIF)	5	7	10

UFC e FP

Una prima stima dei FP è detta **UFC** (Unadjusted Function Point Count):

$$UFC = \sum(\text{number of elements of given type}) \times (\text{weight})$$

Lo UFC è poi corretto da un fattore moltiplicativo TFC compreso tra 0.65 e 1.35

DFP = UFC * TCF

(Delivered Function Point)

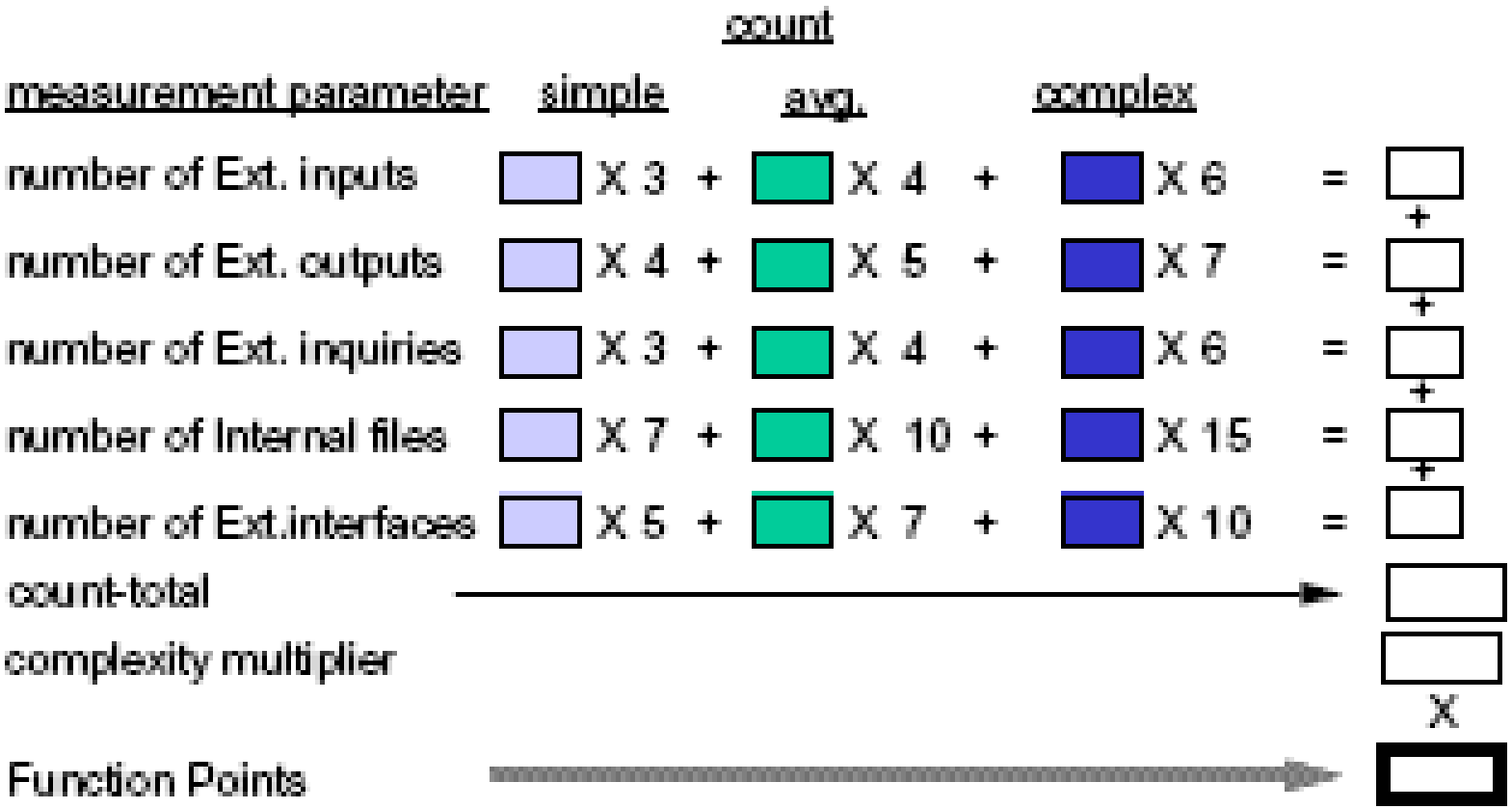
TCF- Fattore di Complessità Totale

- Il TCF è calcolato sulla base di 14 Fattori di complessità (CF), valutati su una scala da 0 (ininfluente) a 5 (essenziale):

1. Il sistema richiede backup e recovery affidabili?
2. Sono richieste comunicazioni specializzate per trasferire dati da/verso l'applicazione?
3. Vi sono funzionalità richiedenti elaborazioni distribuite?
4. Le prestazioni sono un elemento critico?
5. Il software funzionerà in un ambiente operativo esistente già pesantemente utilizzato?
6. Il sistema richiede inserimenti online di dati?
7. L'input online di dati richiede che la transazione relativa sia progettata su più schermate o più operazioni?
8. Il file principali IFL sono aggiornati online?
9. I dati di I/O, i file, le interrogazioni sono complesse?
10. L'elaborazione interna è complessa?
11. Il codice è progettato/scritto per essere riusabile?
12. Il progetto comprende anche l'installazione e la conversione?
13. Il software è stato progettato per essere installato presso diversi utenti?
14. Il software è stato progettato per facilitare le modifiche e per un semplice uso da parte dell'utente finale?

$$TCF = 0.65 + 0.01 * \Sigma_{i=1..14} CF_i$$

Schema di calcolo FP



Applicazione dei FP

- Dalla loro definizione, molti gruppi si sono internazionalmente occupati di poter fornire definizioni e metodologie di stima dei valori dei FP che fossero il più soddisfacenti possibili. Ad esempio:
  - International Function Point User Group, http://www.ifpug.org/
  - Gruppo Utenti Function Point Italia, http://www.gufpi-isma.org/
- Ciò nonostante, le metodologie che portano alla migliore stima sono quelle che sono state pensate, provate e migliorate con l'utilizzo nell'ambito di un gruppo di lavoro stabile e nell'utilizzo di cicli di vita e tecnologie ben stabilizzati
- Una presentazione dettagliata della metodologia di calcolo dei FP secondo la versione 4.1.1 è riportata nel materiale didattico
  - Function_Point_411.pdf

Utilizzi dei FP

- Per stimare la dimensione finale del codice:
  - Studi hanno cercato di rilevare una correlazione tra FP e numero di LOC, variabile a seconda del linguaggio di programmazione.
  - Es. COBOL: 1 FP -> 100 LOC
  - PL1 : 1FP -> 80 LOC
  - OO lang : 1 FP ->60 LOC
- Per stimare lo sforzo (in ore/uomo) di sviluppo:
  - Es. se 1 mese/uomo ->12 FP , allora …
- Per valutare la completezza del testing
  - studi hanno misurato la correlazione fra FP e numero di difetti scoperti durante il testing

Use Case Point

- Una proposta più recente sposta l'attenzione verso la misura degli use case (Karner, 1993)
- Nella tecnica degli use case point vengono tenuti in conto nella misura:
  - Casi d'uso
  - Attori
  - Scenari
- La tecnica di stima è simile a quella dei FP

Gautam Banerjee,Use case points, http://www.bfpug.com.br/Artigos/UCP/Banerjee-UCP_An_Estimation_Approach.pdf

Use Case Point

- Gli attori sono pesati in base alla loro complessità (altri programmi, utente esperto, utente inesperto)
- I casi d'uso sono pesati in base alla complessità delle interazioni (in funzione del numero di interazioni dei suoi scenari, ad esempio)
  - Nella complessità del caso d'uso si tiene conto anche della possibilità di riutilizzare software esistente
- Si aggiusta il conteggio con un fattore tecnico dipendente dai requisiti non funzionali richiesti e da condizioni ambientali legate al processo e agli strumenti di sviluppo
- Si calcola una stima generale dello sforzo in giorni/uomo

Vantaggi e svantaggi dei FP

Vantaggi:

- Indipendenti dal linguaggio
- Ottime indicazioni per applicazioni di elaborazione dati, che usano linguaggi convenzionali o non procedurali
- Basati su quei dati che hanno la maggior probabilità di essere noti all'inizio di un progetto

Svantaggi

- Soggettività nell'assegnazione dei pesi
- Dati sul dominio delle informazioni possono essere difficili da reperire
- Nessuna valutazione della complessità dell'algoritmo (a parità di pochi input e output potrebbe essere banale ed estremamente complicato)
- I FP non hanno un diretto significato fisico

Feedback e Tuning

- La stima dello sforzo con FP e UCP può diventare più affidabile con l'esperienza che si acquisisce nel loro utilizzo, specie in un ambiente di sviluppo invariato

- Al termine di ogni progetto è possibile cercare di ricavare dei feedback per il tuning dei parametri del metodo di calcolo dei FP

Appendice:
COCOMO

Il modello Constructive Cost Model (COCOMO)

- Un modello empirico basato su dati relativi ad esperienze di progetto proposto da Barry Boehm.

- É un modello ben documentato ed indipendente dal venditore del software.
- Ha avuto una lunga storia di modifiche dalla versione iniziale pubblicata 1981 (COCOMO-81) fino alla più recente, COCOMO 2.
- COCOMO 2 (diversamente da COCOMO 81 basato sul pocesso a cascata) tiene conto di diversi approcci allo sviluppo software (quali prototipazione, component-based, riuso…)

COCOMO 81

- Modello a tre livelli (base, intermedio e dettagliato) in base al livello di accuratezza della stima proposta (da rozza a più fine)
- Ogni livello propone, inoltre, stime diverse corrispondenti a tre diverse tipologie di progetti:
  - **Semplice**
    - Applicazioni piccole, per le quali il team di sviluppo è pienamente consapevole delle problematiche da affrontare
  - **Moderato**
    - Applicazioni più complesse, nelle quali si possono incontrare problematiche rispetto alle quali l'esperienza del team è limitata
  - **Integrato**
    - Applicazioni complesse, anche con problematiche hardware e vincoli legati a norme

COCOMO 81

- Per ognuno dei tre livelli sono definite le formule per la stima di sforzo e tempo di sviluppo secondo un dettaglio via via maggiore

$PM = a \bullet S^b$ (sforzo in person-month)

$T = c \bullet PM^d$ (tempo di sviluppo)

  - S misurato in migliaia di DSI (Delivered Source Instructions)
  - a,b,c,d sono parametri dipendenti dal tipo di applicazione
    - Non vengono stimati dall'esperto ma estratti da una tabella di valori che sono considerati plausibili

Tipo di applicazione	**a**	**b**	**c**	**d**
Semplice	2.4	1.05	2.5	0.38
Moderata	3.0	1.12	2.5	0.35
Integrata	3.6	1.20	2.5	0.32

COCOMO 81-livello base

Project complexity	Formula	Description
Simple	$PM = 2.4\ (KDSI)^{1.05} \times M$	Well-understood applications developed by small teams.
Moderate	$PM = 3.0\ (KDSI)^{1.12} \times M$	More complex projects where team members may have limited experience of related systems.
Embedded	$PM = 3.6\ (KDSI)^{1.20} \times M$	Complex projects where the software is part of a strongly coupled complex of hardware, software, regulations and operational procedures.

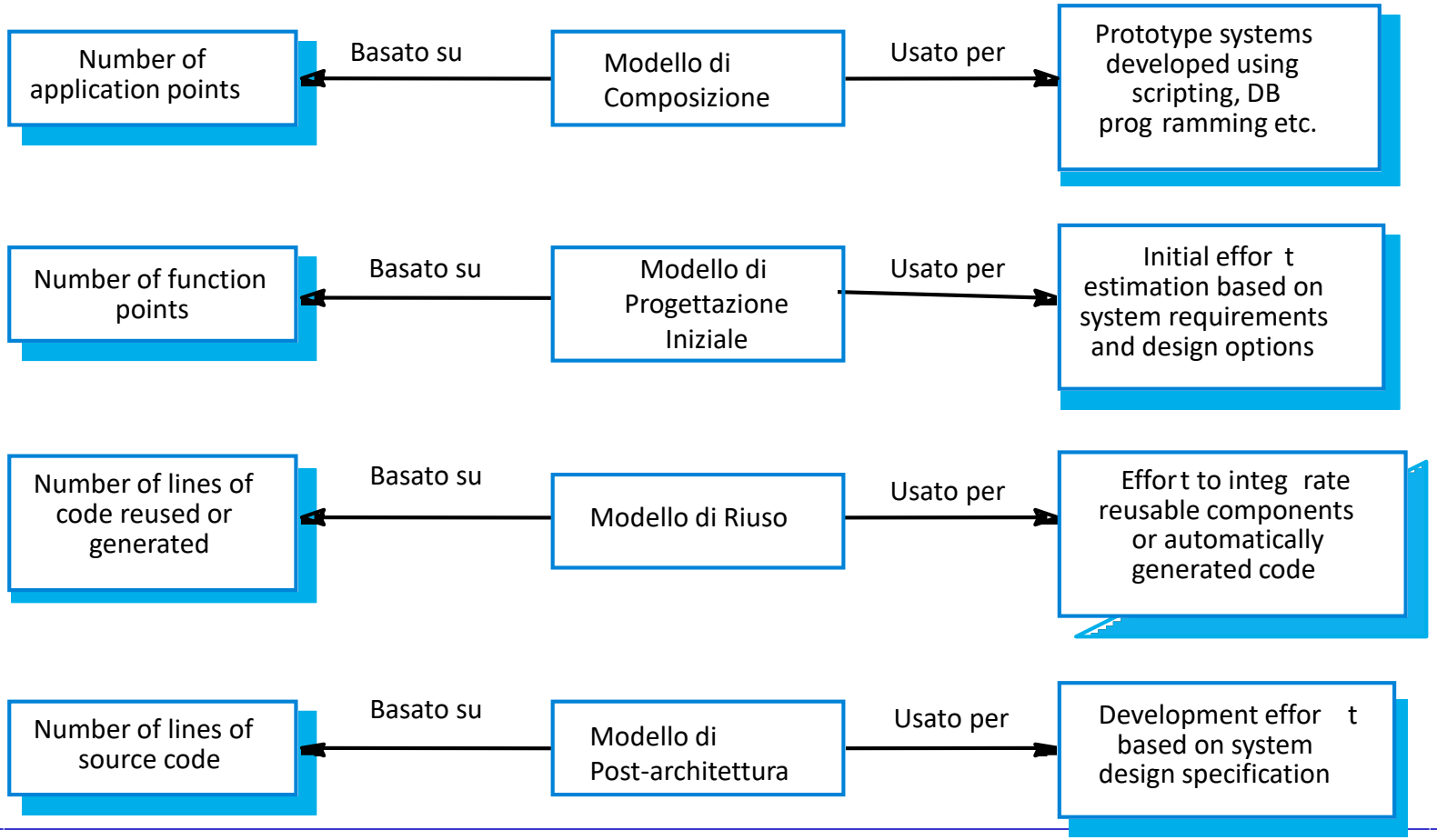
Limiti di COCOMO 81

- COCOMO 81 (Basic COCOMO) proponeva un modello estremamente semplificato sia delle applicazioni che del ciclo di vita adottato
  - In pratica venivano utilizzati solo due parametrI.
    - DSI, come metrica di prodotto
    - Tipologia, come ulteriore metrica che tenesse in conto di tutti i fattori di processo, ma che può assumere solo tre valori possibili
  - A causa di queste limitazioni si sono introdotti modelli COCOMO più precisi → COCOMO II

COCOMO 2

- COCOMO 2 comprende un insieme di modelli applicabili al variare delle modalità di realizzazione dei software, che permettono stime più dettagliate e affidabili.
  - Modello di composizione delle applicazioni. Utile per la stima di sistemi ottenuti a partire dal riuso di componenti
  - Modello di progettazione iniziale. Basata fondamentalmente sui Function Point; utile per ottenere stime in fase di analisi
  - Modello di riutilizzo. Utile per stimare il costo relativo all'integrazione di diversi componenti riusabili.
  - Modello di post-architettura. Modelli utilizzabili a valle della definizione dell'architettura per ottenere stime più affidabili

Uso dei modelli COCOMO 2



1. Modello di composizione delle applicazioni

- E' un modello utile per stimare i costi di realizzazione di un prototipo o di realizzazione di un software a partire da componenti esistenti.

$$PM = (NAP \times (1 - \%reuse/100)) / PROD$$

  - PM è lo sforzo in mesi-uomo, NAP il numero di Application Point, PROD la produttività
  - In pratica, tale modello esclude dal calcolo dello sforzo la parte del sistema riusata.

Tabella per la valutazione della produttività

Developer s experience and capability	Very low	Low	Nominal	High	Very high
ICASE maturity and capability	Very low	Low	Nominal	High	Very high
PROD (NOP/month)	4	7	13	25	50

2. Modello di progettazione iniziale

- Viene usato una volta che I requisiti utente sono stati definiti e la progettazione iniziata.
- E' spesso utilizzato per confrontare diverse soluzioni progettuali possibili.

$$Effort = A \times Size^{B} \times M$$

  - A = 2.94 (almeno in prima istanza)
  - Size è misurata in KLOC (calcolata a partire dai FP)
  - B varia tra 1.1 e 1.24 in base ad una serie di fattori quali l'innovatività del progetto, la flessibilità, le tecniche di gestione del rischio, la maturità del processo …
  - M = PERS × RCPX × RUSE × PDIF × PREX × FCIL × SCED;

I Moltiplicatori

- I moltiplicatori riflettono vari aspetti:
  - PERS: capacità del personale
  - RCPX: complessità del prodotto
  - RUSE: livello di riuso
  - PDIF: difficoltà legate alla piattaforma utilizzata
  - PREX: esperienza del personale
  - FCIL: funzionalità di supporto
  - SCED: tempistica
  - Ogni moltiplicatore è valutato su una scala da 1(molto basso) a 6 (molto alto)

3. Modello di riuso

- Considera sia il riuso di codice in modalità black-box, senza cambiamenti, sia il riuso di codice che deve essere adattato per integrarlo con nuovo codice.

- Ci sono due versioni del modello di stima:
  - **Black-box reuse** (senza modifiche del codice riusato). Si calcola una stima dello sforzo (PM).
  - **White-box reuse** (con modifiche del codice). Si calcola una stima del numero equivalente di linee di nuovo codice (a partire dal numero di linee riusate). Tale valore viene poi integrato con quello delle linee generate ex novo.

Modello di riuso

- Per codice generato automaticamente (PM è lo sforzo di integrazione):

$$PM_{Auto} = (ASLOC * AT/100)/ ATPROD$$

  - **ASLOC**: numero di linee di codice complessivo
  - **AT**: percentuale di codice automaticamente generato
  - **ATPROD**: produttività degli ingegneri che si occupano dell'integrazione del codice (tipico: 2400 istruzioni al mese)

Modello di riuso

- Per codice riutilizzato e modificato:

  ESLOC = ASLOC * (1-AT/100) * AAM.

  - **ESLOC**: numero equivalente di righe di nuovo codice (da utilizzare nelle formule che richiedono il numero di linee di codice generato ex novo)
  - **ASLOC**: numero di linee di codice riutilizzato (ma da adattare)
  - **AT**: percentuale di codice automaticamente generato
  - **AAM** coefficiente che indica la difficoltà di adattamento del codice. É' la somma di **AAF** (costo per effettuare le modifiche), **SU** (costo per comprendere il codice esistente), **AA** (costo legato alle decisioni sul riutilizzo eventuale)

- In pratica si paragona lo sforzo di adattamento del codice riusato ad uno sforzo di scrittura di nuovo codice.

4. Modello di Post-Architettura (P.A.)

- Viene utilizzato quando è disponibile un progetto architetturale iniziale ed è conosciuta la struttura dei sottosistemi.
- Viene riutilizzata la stessa formula del modello di composizione:

$$\text{Effort} = A \times \text{Size}^{B} \times M$$

  - Stavolta però è possibile una migliore stima della Size

Modello di Post-Architettura (P.A.)

- La dimensione del codice (**Size**) è stimata sulla base della somma di 3 componenti:
  - Numero di linee di codice da sviluppare da zero;
  - Stima di numero di linee di codice equivalente calcolate tramite il modello di riuso;
  - Stima del numero di linee di codice che devono essere adattate ai requisiti specifici dell'applicazione (stima non semplice!!!)

- Il termine esponenziale **B** invece varia in modo continuo (a partire da 1.01) e dipende da:
  - 5 fattori di scala (valutati in una scala tra 0 e 5- da extra alto a molto basso)
  - In COCOMO 81 B assumeva solo 3 possibili valori per modello.

- Il coefficiente **M** dipende da 17 attributi

Fattori di scala in COCOMO 2-modello P.A.

Precedenti	Indica l'esperienza precedente dell'organizzazione con questo tipo di progetto (da bassa a alta)
Flessibilità di sviluppo	Indica se si usa un processo prefissato col cliente, oppure c'è libertà di scelta.
Architettura/ Risoluzione dei rischi	Indica se viene eseguita o meno una accurata analisi dei rischi
Coesione del Team	Indica il livello di conoscenza e familiarità del team di sviluppo
Maturità del Processo	Indica il grado di maturità del processo adottato (secondo CMM)

Moltiplicatori (costi guida)

- **Attributi di prodotto**
  - RELY, Affidabilità richiesta
  - CPLX, Complessità dei moduli
  - DOCU, documentazione
  - DATA, dimensione del database
  - RUSE, percentuale di componenti riusabili
- **Attributi informatici**
  - TIME, tempo
  - PVOL, precarietà della piattaforma di sviluppo
  - STOR, vincoli di memoria
- **Attributi di personale**
  - ACAP, capacità degli analisti
  - PCON, continuità del personale
  - PCAP, capacità dei programmatori
  - PEXP, esperienza dei programmatori
  - AEXP, esperienza degli analisti
  - LTEX, esperienza del linguaggio e degli strumenti
- **Attributi di progetto**
  - TOOL, utilizzo di strumenti
  - SCED, concentrazione della pianificazione di sviluppo
  - SITE, qualità delle comunicazioni tra i siti di sviluppo

Effetti dei cost drivers sulla stima

Exponent value	1.17
System size (including factors for reuse and requirements volatility)	128, 000 DSI
Initial COCOMO estimate without cost drivers	**730 person-months**
Reliability	Very high, multiplier = 1.39
Complexity	Very high, multiplier = 1.3
Memory constraint	High, multiplier = 1.21
Tool use	Low, multiplier = 1.12
Schedule	Accelerated, multiplier = 1.29
Adjusted COCOMO estimate	**2306 person-months**
Reliability	Very low, multiplier = 0.75
Complexity	Very low, multiplier = 0.75
Memory constraint	None, multiplier = 1
Tool use	Very high, multiplier = 0.72
Schedule	Normal, multiplier = 1
Adjusted COCOMO estimate	**295 person-months**

Qualche osservazione su COCOMO

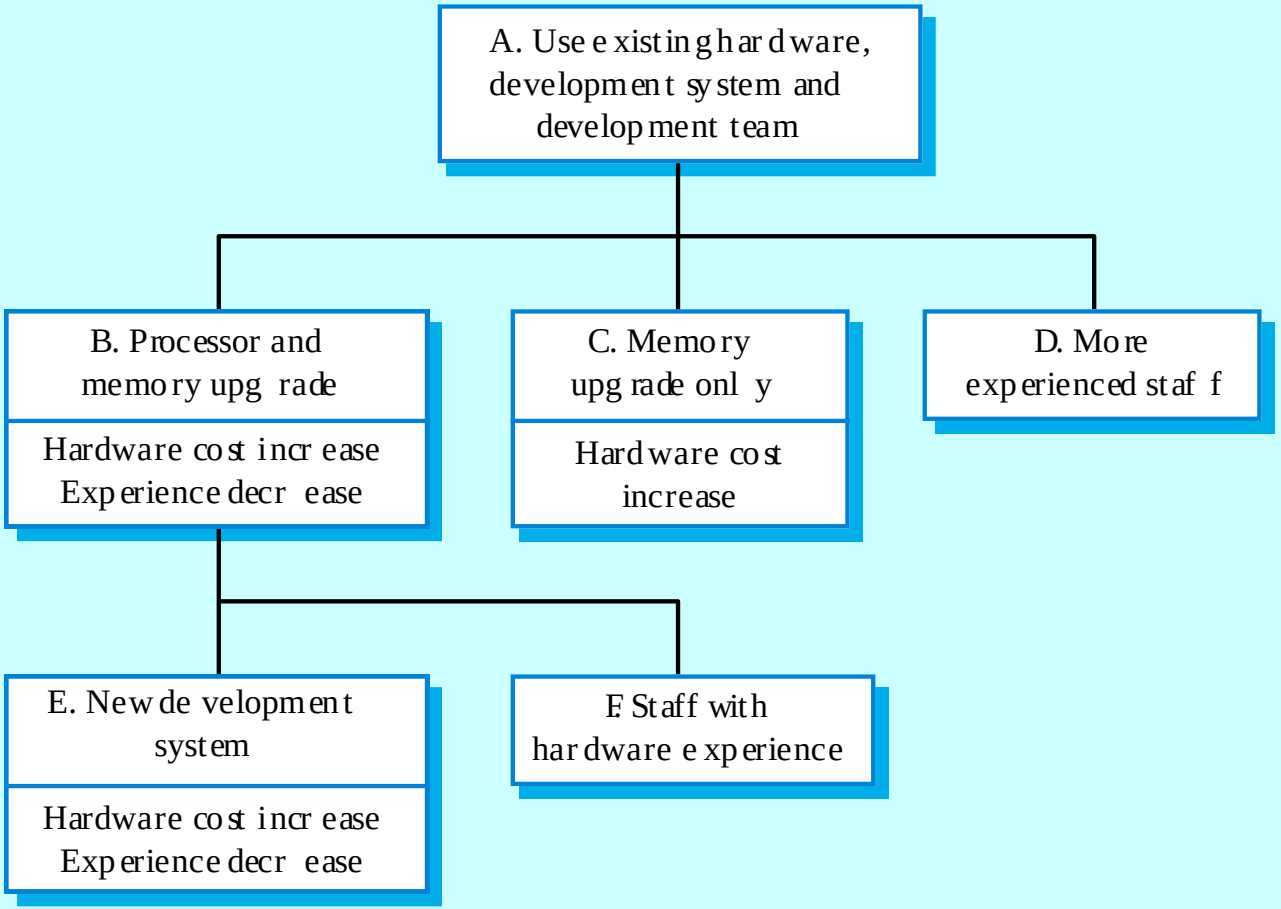
- Il modello è stato proposto sulla base di esperienza e dati storici raccolti dagli autori del modello
  - Dovrebbe dunque essere tarato prima dell'uso in altre organizzazioni
- Prevede molti attributi la cui valutazione è spesso difficile e imprecisa
  - Anche per gli attributi sarebbe necessaria una taratura in base a dati storici di progetto
- Nella pratica è difficile che le aziende posseggano tali dati!
- Chi può permetterselo, paga un esperto di COCOMO per adattarlo ed usarlo nella propria realtà.

Modelli di costo nel Project planning

- I modelli di costo algoritmici consentono di eseguire la pianificazione dei progetti, in quanto permettono la valutazione di strategie alternative.

- Un esempio: Sistema per controllo di esperimenti spaziali
  - Deve essere molto affidabile;
  - Deve minimizzare il peso hardware (number of chips);
  - Pertanto, I moltiplicatori su affidabilità e vincoli informatici saranno > 1.
- I componenti di costo del progetto:
  - Costi dell'hardware;
  - Costi della piattaforma di sviluppo;
  - Costo dello Sforzo di sviluppo.

Opzioni di gestione



I costi delle Opzioni di gestione

Option	RELY	STOR	TIME	TOOLS	LTEX	Total effort	Software cost	Hardware cost	Total cost
A	1.39	1.06	1.11	0.86	1	63	949393	100000	1049393
B	1.39	1	1	1.12	1.22	88	1313550	120000	1402025
C	1.39	1	1.11	0.86	1	60	895653	105000	1000653
D	*1.39*	*1.06*	*1.11*	*0.86*	*0.84*	*51*	*769008*	*100000*	*897490*
E	1.39	1	1	0.72	1.22	56	844425	220000	1044159
F	1.39	1	1	1.12	0.84	57	851180	120000	1002706

Costo software= stima di sforzo*RELY*STOR*TIME*TOOLS*LTEX*15.000

Costo medio di 1mese/persona

La scelta dell'opzione

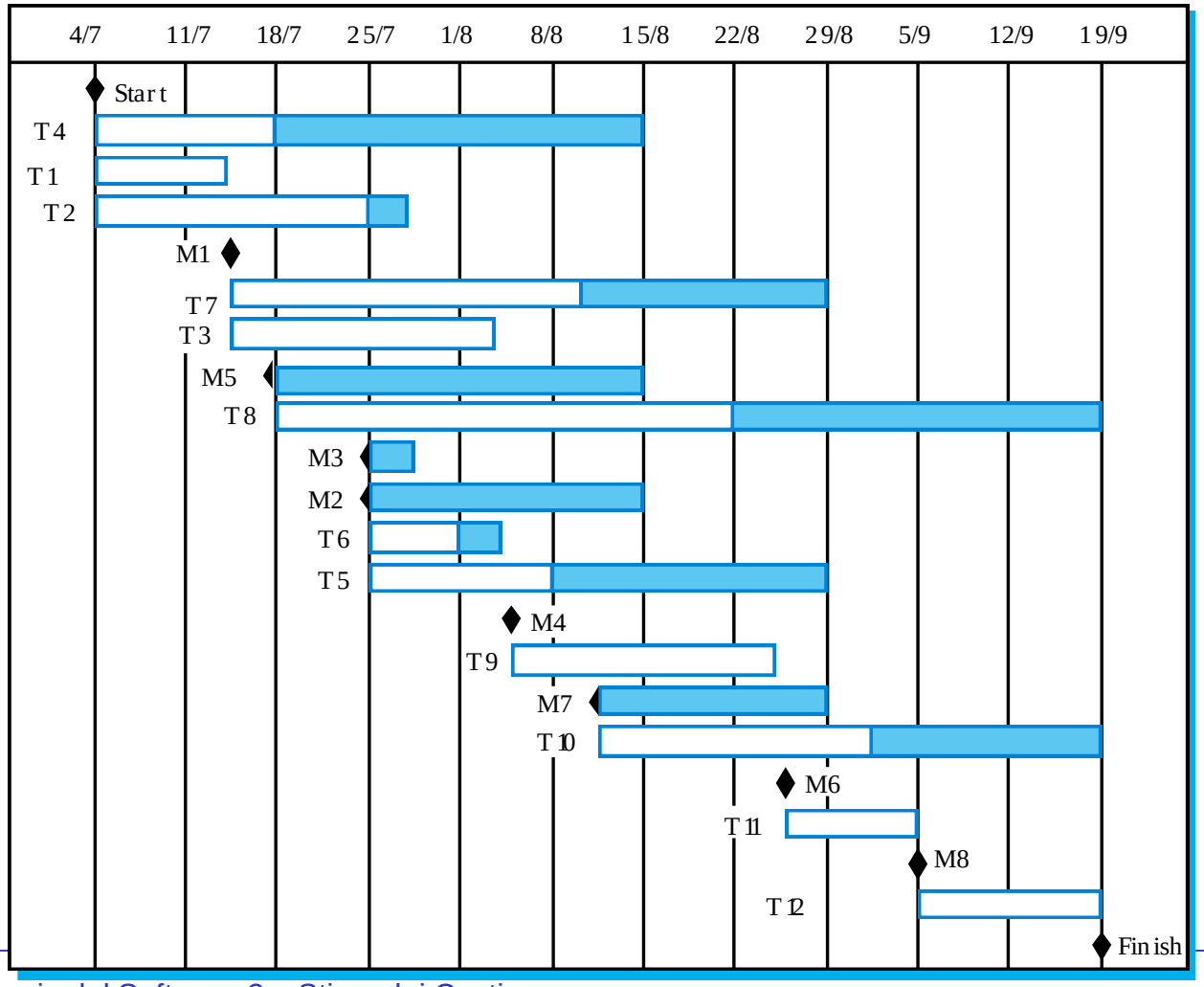
- L'opzione D (uso di uno staff più esperto) sembra la migliore alternativa rispetto ai costi
  - Ma ha un alto rischio associato, per la difficoltà di trovare uno staff molto esperto.
- L'opzione C (upgrade memory) consente di risparmiare meno ma presenta bassi rischi.

- Il modello rivela l'importanza dell'esperienza dello staff nello sviluppo software.

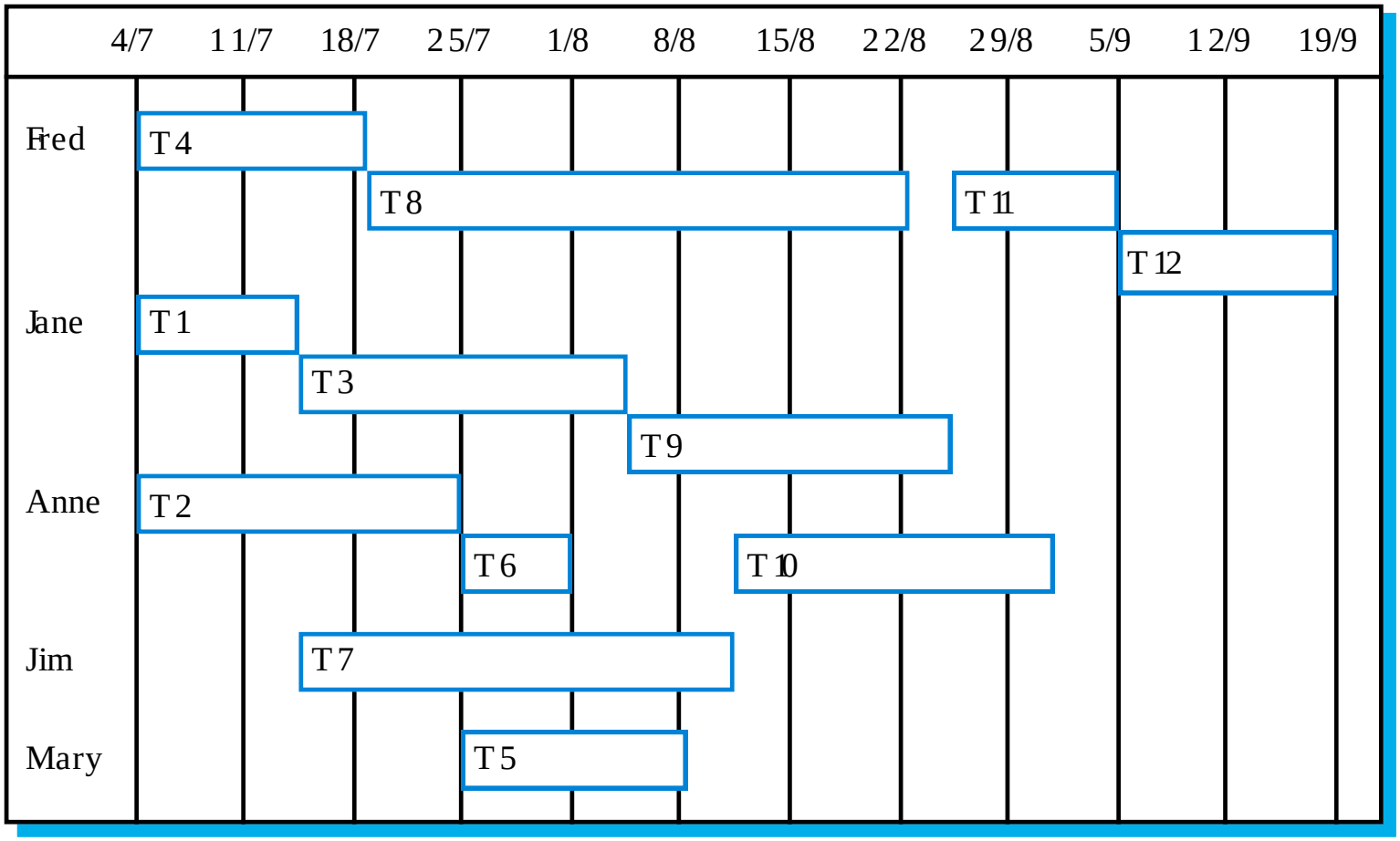
Durata del progetto e gestione dello staff

- Analogamente alla stima dell'effort, I managers devono stimare anche:
  - il tempo di calendario necessario a completare il progetto
  - Quando lo staff dovrà lavorare sul progetto.

Tempistica di progetto- Diagramma di Gant



Allocazione del personale nel tempo



Durata del progetto

- COCOMO II ha anche una formula per la stima del tempo di sviluppo:

$$TDEV = 3 \bullet PM^{[0.33+0.2\bullet(B-1.01)]} \bullet SCED$$

- Dove:
  - PM è lo sforzo calcolato precedentemente
  - B è il fattore esponenziale calcolato precedentemente
  - SCED è il moltiplicatore che indica la presumibile dilatazione nei tempi di sviluppo.

- Si può notare che il tempo di sviluppo dipende direttamente dallo sforzo ma non direttamente dal numero di sviluppatori

Requisiti sullo staff

- Lo staff richiesto non si può determinare dividendo lo sforzo di sviluppo richiesto per la tempistica richiesta:
  - Es. 50 pm/ 10 mesi = 5 persone

- Il numero di persone rischiesto per lo sviluppo varia in base alla fase del progetto
- Inoltre più persone lavorano, e maggiore è lo sforzo totale richiesto.
- Un rapido incremento dello staff in un progetto spesso comporta ritardi nello schedule!

Alcune considerazioni conclusive

- Non c'è una relazione semplice tra costi di sviluppo e prezzi pagati.
- Ci sono molti fattori che influiscono sulla produttività (es. Attitudine personale, esperienza nel dominio, il progetto di sviluppo, la dimensione del progetto, I tool e l'ambiente di lavoro).
- In genere il prezzo del software viene stabilito per ottenere il contratto, e la funzionalità viene rapportata al prezzo pagato.

Alcune considerazioni conclusive

- Si dovrebbero usare più tecniche di stima dei costi contemporaneamente.
- Il modello COCOMO predice lo sforzo di sviluppo considerando vari fattori relativi al progetto, al prodotto, al personale e all'hardware.
- I modelli di costo algoritmici supportano una analisi quantitativa di diverse opzioni consentendo il confronto di opzioni diverse.
- Il tempo di un progetto non è proporzionale al numero di persone impiegate.

Appendice:
Gestione dei Rischi

Riferimenti

- Sommerville, Capitolo 5 (gestione dei rischi)

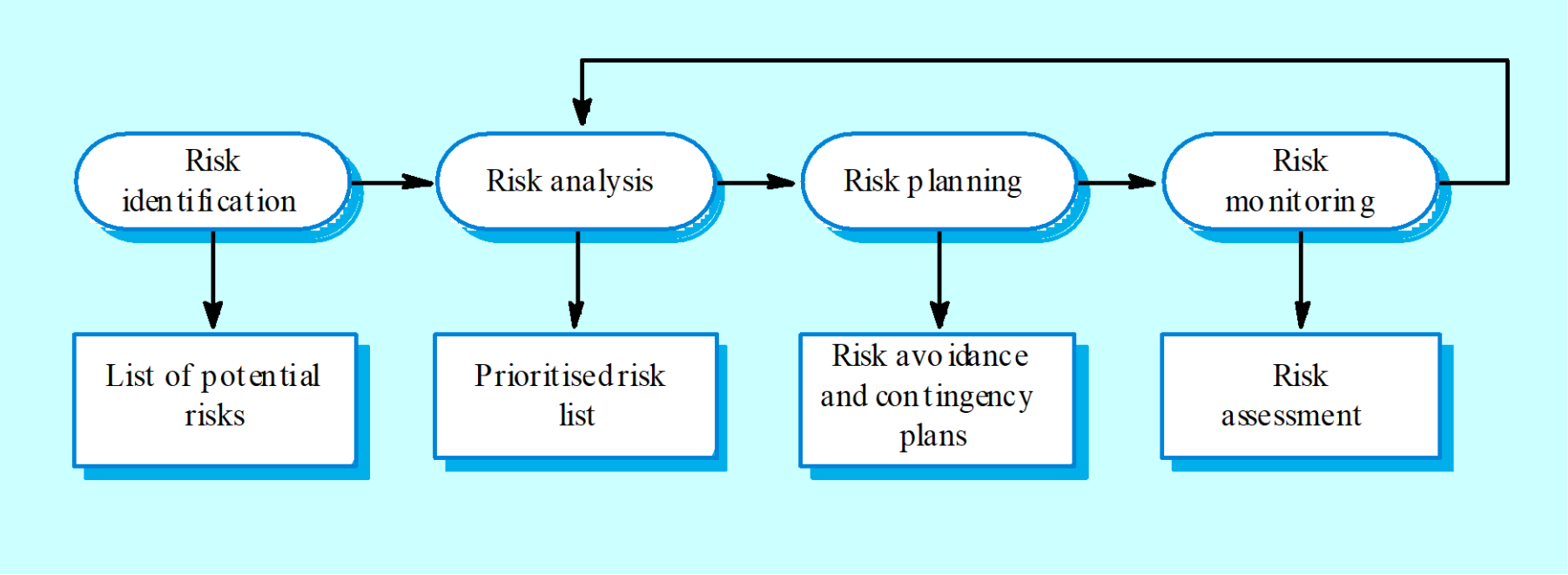
Gestione dei rischi

- La gestione dei rischi consiste nella loro identificazione e nella proposta di soluzioni alternative che possano minimizzarne gli effetti
- Un rischio è definito come la probabilità che alcuni eventi avversi si verifichino in un progetto software
  - Rischi per il progetto
    - Allungamento dei tempi
  - Rischi per il prodotto
    - Perdita di qualità
    - Riduzione di funzionalità
  - Rischi per l'organizzazione
    - Incremento dei costi

Rischi possibili

Risk	Affects	Description
Staff turnover	Project	Experienced staff will leave the project before it is finished.
Management change	Project	There will be a change of organisational management with different priorities.
Hardware unavailability	Project	Hardware that is essential for the project will not be delivered on schedule.
Requirements change	Project and product	There will be a larger number of changes to the requirements than anticipated.
Specification delays	Project and product	Specifications of essential interfaces are not available on schedule
Size underestimate	Project and product	The size of the system has been underestimated.
CASE tool under-performance	Product	CASE tools which support the project do not perform as anticipated
Technology change	Business	The underlying technology on which the system is built is superseded by new technology.
Product competition	Business	A competitive product is marketed before the system is completed.

Processo di gestione del rischio



Processo di gestione del rischio

- Identificazione del rischio
  - Sono individuati i possibili rischi
- Analisi del rischio:
  - sono valutate probabilità e conseguenze dei possibili rischi
- Pianificazione del rischio
  - Si disegnano piani per evitare rischi o minimizzarne gli effetti
- Monitoraggio del rischio
  - Il rischio è costantemente controllato

1. Identificazione dei rischi- Condotta mediante checklist

- Rischi tecnologici
  - Dipendenti dalle tecnologie utilizzate e dalla loro possibile evoluzione/dismissione
- Rischi relativi alle persone
  - Abbandono di persone del team di progetto
  - Difficoltosa integrazione di nuovo personale
- Rischi organizzativi
  - Relativi all'ambiente di lavoro nel quale si sta realizzando il progetto
- Rischi strumentali
  - Dipendenti dagli strumenti di sviluppo e di supporto utilizzati
- Rischi relativi ai requisiti
  - Variazione dei requisiti del cliente
- Rischi di stima
  - Imprecisione nelle stime sulle quali viene dimensionato il processo di sviluppo

Esempi di possibili rischi

Risk type	Possible risks
Technology	The database used in the system cannot process as many transactions per second as expected. Software components that should be reused contain defects that limit their functionality.
People	It is impossible to recruit staff with the skills required. Key staff are ill and unavailable at critical times. Required training for staff is not available.
Organisational	The organisation is restructured so that different management are responsible for the project. Organisational financial problems force reductions in the project budget.
Tools	The code generated by CASE tools is inefficient. CASE tools cannot be integrated.
Requirements	Changes to requirements that require major design rework are proposed. Customers fail to understand the impact of requirements changes.
Estimation	The time required to develop the software is underestimated. The rate of defect repair is underestimated. The size of the software is underestimated.

2. Analisi del rischio

- Valutazione della probabilità con la quale l'evento avverso si verifica e della serietà delle conseguenze connesse
  - La *probabilità* viene di solito stimata con una scala discreta che assuma valori come (molto bassa, bassa, moderata, alta, molto alta)
  - Gli *effetti* degli eventi avversi possono essere stimati anch'essi su scala discreta che assuma valori come (insignificante, serio, tollerabile, catastrofico)

Esempi di analisi di rischio

Risk	Probability	Effects
Organisational financial problems force reductions in the project budget.	Low	Catastrophic
It is impossible to recruit staff with the skills required for the project.	High	Catastrophic
Key staff are ill at critical times in the project.	Moderate	Serious
Software components that should be reused contain defects which limit their functionality.	Moderate	Serious
Changes to requirements that require major design rework are proposed.	Moderate	Serious
The organisation is restructured so that different management are responsible for the project.	High	Serious

Esempi di analisi di rischio

Risk	Probability	Effects
The database used in the system cannot process as many transactions per second as expected.	Moderate	Serious
The time required to develop the software is underestimated.	High	Serious
CASE tools cannot be integrated.	High	Tolerable
Customers fail to understand the impact of requirements changes.	Moderate	Tolerable
Required training for staff is not available.	Moderate	Tolerable
The rate of defect repair is underestimated.	Moderate	Tolerable
The size of the software is underestimated.	High	Tolerable
The code generated by CASE tools is inefficient.	Moderate	Insignificant

3. Pianificazione dei rischi

- Strategie per evitare i rischi
  - Cercano di ridurre la probabilità connessa al rischio
- Strategie per minimizzare i rischi
  - Cercano di ridurre l'impatto che eventualmente avrebbe l'evento avverso sul progetto
- Piani precauzionali
  - Cercano di prevedere una soluzione alternativa di compromesso in anticipo rispetto all'effettivo verificarsi dell'evento avverso

Esempi di strategie

Risk	Strategy
Organisational financial problems	Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business.
Recruitment problems	Alert customer of potential difficulties and the possibility of delays, investigate buying-in components.
Staff illness	Reorganise team so that there is more overlap of work and people therefore understand each other's jobs.
Defective components	Replace potentially defective components with bought-in components of known reliability.

Esempi di strategie

Risk	Strategy
Requirements changes	Derive traceability information to assess requirements change impact, maximise information hiding in the design.
Organisational restructuring	Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business.
Database performance	Investigate the possibility of buying a higher-performance database.
Underestimated development time	Investigate buying in components, investigate use of a program generator

4. Monitoraggio del rischio

- Serve a valutare, dinamicamente
  - se i rischi diventano più o meno probabili
  - Se I loro effetti cambiano di intensità (ad esempio se aumenta la serietà del rischio)
- Vengono monitorati un insieme di indicatori di rischio, in maniera automatica o soggettiva
- Ad ogni riunione del comitato che si occupa della gestione del progetto si dovrebbero riconsiderare I rischi effettivi, sulla base dei dati aggiornati provenienti dal monitoraggio

Possibili indicatori di rischio

Risk type	Potential indicators
Techno logy	Late delivery of hardware or support software, many reported technology problems
Peop le	Poor staff morale, poor relationships amongst team member, job availability
Organisational	Organisational gossip, lack of action by senior management
Tools	Reluctance by team members to use tools, complaints about CASE tools, demands for higher-powered workstations
Requirements	Many requirements change requests, customer complaints
Estimation	Failure to meet agreed schedule, failure to clear reported defects