

# Il programmatore

Lavorare con efficacia e efficienza





È necessario trovare un giusto equilibrio tra efficacia ed efficienza.

## Consigli pratici...

- ...per un metodo di lavoro
- ...per noi stessi e per gli altri
- ...per risultati migliori
- ...



3

- Un buon metodo di lavoro non significa solo un buon modo di lavorare. Significa anche trovare, codificare, far propri percorsi di sviluppo che collaudati dal team diventino percorsi "affidabili".
- Il metodo di lavoro, se è un percorso del team, va condiviso in maniera bidirezionale ("do e ricevo").
- Se il percorso è "affidabile" non può che portare ogni volta che viene seguito a risultati migliori.

## Quali consigli

- Per programmare meglio
- Per programmare sicuro
- Per ri-programmare di meno



Consigli per programmatori ed anche qualcosa per gli analisti.

Programmatore ed analista non sono due figure.

In ognuno di noi convivono le due personalità in misura diversa per attitudini professionali e personali, per ruolo, per competenze.

Un programma che funziona è un programma:

-che "gira" senza errori

-che è funzionale allo scopo prefissato

# Organizzarsi per l'organizzazione



- ORGANIZZAZIONE: **insieme** di risorse orientate al perseguimento di una **finalità comune**, in un rapporto continuo di **coevoluzione** con l'ambiente



5

## L'animale a due teste

C'era una volta un uccello con due teste e un corpo: la testa di destra era vorace e abilissima nella ricerca del cibo, mentre quella di sinistra, altrettanto ghiotta, era maldestra. La testa di destra riusciva sempre a nutrirsi a sazietà, mentre quella di sinistra era incessantemente tormentata dalla fame.

E così un giorno, la testa di sinistra disse alla destra: "Conosco, qui vicino, un'erba squisita di cui ti delizieresti: vieni, ti conduco dove cresce".

In realtà sapeva che quell'erba era velenosa, ma voleva con questo stratagemma uccidere l'altra testa, per poter mangiare a piacimento.

E la testa di destra mangiò l'erba, e il veleno uccise l'uccello dalle due teste.

*Taisen Deshimaru, La tazza e il bastone*

# Organizzarsi per l'organizzazione



- Le organizzazioni sono la **naturale** conseguenza della divisione del lavoro
- Laddove gli sforzi di molti individui sono propriamente organizzati e combinati per il raggiungimento di un obiettivo comune, si **ottengono vantaggi** in termini di efficienza ed efficacia
- Le organizzazioni rappresentano quindi un limite all'iniziativa individuale in termini di regole, strutture, ruoli, ma sono al contempo la **condizione necessaria** per il raggiungimento dei nostri obiettivi



# Senso di orientamento

- Non navigare a caso
- Cercare la strada giusta prima di mettersi in cammino
- Usare la mappa
- Leggere i “cartelli”
- Chiedere informazioni



7

I manuali standard

I manuali di programmazione tradizionale e webgate

I “miei” manuali

Leggere e interpretare i messaggi del sistema

Formulare la domanda giusta alla persona giusta

Formulare la domanda, ascoltare la risposta e capire la risposta

# Cassetta degli attrezzi



- Controllare di avere gli attrezzi prima di cominciare il lavoro
- Conoscere come si usano gli attrezzi
- Usare l'attrezzo giusto per ogni lavoro



8

Non posso stringere i bulloni con un cacciavite o tagliare il fil di ferro con le forbici delle unghie  
Avere sempre con sé i tools

Preparare prima i tools sul sistema; se lo faccio solo quando mi servono diventa più che altro una perdita di tempo

Sperimentare sempre i tools nei casi pratici

Avere una panoramica completa per scegliere il tool migliore. Per poter scegliere devo conoscere.  
Tenere a portata di mano il tool secondario se si "rompe" il principale

ALCUNI TOOLS ESSENZIALI PER PC:

-FileZilla (<http://filezilla-project.org/index.php>): client FTP

-Notepad++ (<https://notepad-plus-plus.org/>): editor file txt avanzato

-KeePass (<http://keepass.info/download.html>): gestore sicuro per password (anche per Android)

-PDF Creator (<https://www.pdfforge.org/pdfcreator/download>): creatore di file pdf

-WinRAR (<http://www.winrar.it/>): compressione

# Analisi: produrla e capirla

1. scrivere per farsi capire
2. leggere per capire
3. capire per riprodurre

• dalla richiesta “disordinata”  
bisogna raggiungere la  
soluzione “ordinata”



1. l'analisi deve essere essenziale, chiara, documentata
2. l'analisi deve essere compresa e contestualizzata
3. se l'analisi è stata capita, assimilata e ragionata, allora diventa un patrimonio prezioso anche per lavori futuri

**Cosa** deve fare e  
**chi** deve essere  
il programmatore?



10

# Programmare in chiaro

- Non criptare il linguaggio di programmazione
- Titolo del programma
- Cosa fa in **2** righe
- L'interfaccia esterna (parametri)
- Ordinare le porzioni di codice



11

Il linguaggio di programmazione HLL (alto livello) istruisce la macchina con parole umanamente intellegibili

Non scriviamo per la macchina ma per l'uomo

In un sorgente non deve mai mancare:

- Titolo
- Scopo (in 2 righe)
- Autore
- Descrizione parametri input/output
- Documentazione modifiche

Le porzioni di codice devono essere raggruppate e ordinate logicamente anche se il linguaggio consente di porre disordine:

- Definizioni file, schiere, DS, campi, costanti
- Parametri
- Chiavi di accesso ai file



## Dare il nome giusto

- Ogni cosa va chiamata con il nome giusto
  - File
  - Campi e descrizioni (TEXT, COLHDG)
  - DS
  - Programmi
  - Menu
  - ...



13

Il nome giusto:

- Identifica
- Describe
- Evita ambiguità

# Dare il nome giusto

Riga	Colonna 1	Sostituisce	Function
001400	A		TEXT('FLAG ANNULLAMENTO')
001400	A		COLMDC('Flag annullamento')
001700	A		REFFLD(CCLI)
001800	A		TEXT('NUMERO RIGHE')
001900	A		COLMDC('Numero righe')
002000	A		TEXT('NUMERO RIGA')
002100	A		COLMDC('Numero riga')
002200	A		REFFLD(CART)
002300	A		REFFLD(CABE)
002400	A	TUSPEFA	
002500	A		REFFLD(DESC)
002600	A		TEXT('DESCRIZIONE ARTICOLO')
002700	A		COLMDC('Descrizione articolo')
002800	A		REFFLD(QCAN)
002900	A		TEXT('QUANTITA')
003000	A		COLMDC('Quantita')
003100	A		REFFLD(LVEN)
003200	A		TEXT('NUMERO DI ORDINE')
003300	A		COLMDC('Num. di ordine')
003400	A		REFFLD(CUSR)
003500	A		TEXT('UTENTE INS')
003600	A		COLMDC('Utente Ins')
003700	A		REFFLD(UTAG)
003800	A		TEXT('DATA INS')
003900	A		REFFLD(HORA)
004000	A		COLMDC('Data Ins')
004100	A		REFFLD(HORA)
004200	A		TEXT('ORA INS')
004300	A		COLMDC('Ora Ins')
004400	A		REFFLD(CUSR)
004500	A		TEXT('UTENTE AGG')
004600	A		COLMDC('Utente Agg')
004700	A		REFFLD(UTAG)
004800	A		TEXT('DATA AGG')

cosa conterrà questo campo anonimo?



- Il nome giusto:
- Identifica
  - Describe
  - Evita ambiguità

## Programmare con la lingua

- L'utente non è una macchina
- Ortografia
- Proprietà di termini
- Chiarezza e semplicità
- Il video e la stampa devono parlare da soli



15

### STAMPE

- Devono essere ancor più chiare del video perché sono avulse dal contesto in cui sono state prodotte

# Programmare con la lingua

The image shows a screenshot of a software application window titled "SoftwareSiro - [WEBGATE400 Session]". The window contains a menu bar with "File", "Comandi", "Modifica", "Visualizza", "Strumenti", and "Ajuda". Below the menu is a toolbar with various icons. The main area of the window displays a dialog box titled "Parametri nuova visualizzazione statistica". Inside this dialog, there are fields for "Data inizio periodo di calcolo" (set to 01/01/2010) and "Data fine periodo di calcolo" (set to 31/03/2010). Annotations in red text with arrows point to specific elements: "Il titolo del video?" points to the window title bar; "cosa significa?" points to the dialog title; "F6=Inizio elaborazione?" points to the "F6" key icon in the toolbar. To the right of the window is a large red question mark and a blue puzzle piece icon. The MKI logo is visible in the bottom left corner, and the number 16 is in the bottom right corner.

STAMPE

- Devono essere ancor più chiare del video perché sono avulse dal contesto in cui sono state prodotte

## Pensare al prossimo...

- **UTENTE:** pensare a chi userà il mio programma
- **PROGRAMMATORE:** pensare a chi modificherà, correggerà il mio programma



17

Seguire tutti i consigli precedenti per essere chiari e limpidi

### UTENTE

Deve essere guidato verso la strada giusta

Deve essere informato di cosa sta facendo per lui la macchina

Deve essere allarmato se sta eseguendo operazioni "pericolose"

### PROGRAMMATORE

Usare (leggere e scrivere) il database delle segnalazioni.

Scrivere commenti utili nel sorgente.

# Pensare al prossimo...

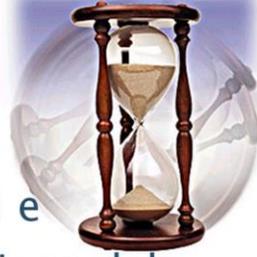
```
Riga 36      Colonna 36      Sostituisci
-----1-----2-----3-----4-----5-----6-----
002200 /* Creazione ..... */
002300      CALL      PGM($CRTF) PARM(&LIBFUSER '.....')
002400
002500 /* Controllo esistenza file ..... */
002600      RTVMBRD   FILE(&LIBFUSER/.....) NBRCURRCD(&NREC)
002700      MONMSG    MSGID(CPF9812) EXEC(DO)
002800          CHGVAR  VAR(&ERR) VALUE('1')
002900      CALL      PGM(.....) PARM(&ERR)
003000      GOTO      CMDLBL(FINE)
003100      ENDDO
003200
003300 /* Controllo esistenza record su ..... */
003400      IF        COND(&NREC *EQ 0) THEN(DO)
003500          CHGVAR  VAR(&ERR) VALUE('2')
003600      CALL      PGM(.....) PARM(&ERR)
003700      GOTO      CMDLBL(FINE)
003800      ENDDO
003900
004000 /* Inserimento ..... */
004100      CALL      PGM(T.....) PARM(&ERR)
004200
004300 FINE:      ENDPGM
```

**attenzione alla  
ridondanza di  
codice**



## Curare il tempo

- il tempo del programmatore e dell'utente è molto più prezioso del tempo della macchina
- **INTERATTIVO** = solo per "interagire" con la macchina
- il resto è solo ed unicamente **BATCH**



19

## La strada non insegna niente?



- pensate che il traffico di rete sia una cosa diversa dal traffico in tangenziale?
  - scegliere la strada più veloce
  - comprimere più possibile
  - evitare trasferimenti remoti non necessari



# Coltivare le cellule neuronali

- usare la formula CA.ME.RA.
  - CAtalogare
  - MEmorizzare
  - RAgionare



21

## CATALOGARE:

-sapere dove trovare le informazioni

## MEMORIZZARE

-tenere a mente solo le cose fondamentali, al limite solo la catalogazione

## RAGIONARE

-comprendere i processi per rielaborare le informazioni catalogate e memorizzate

-chiedersi il perché... e cercare di risponderci

# Costruire la propria KB

- indice dei manuali
- bigini
- appunti
- sorgenti esempio
- siti internet
- avete un disco sul PC? usatelo



22

L'ufficialissimo IBM Document Center 7.4: <https://www.ibm.com/docs/en/i/7.4>

Power 9: <https://www.ibm.com/support/knowledgecenter/POWER9/p9hdx/POWER9welcome.htm>

IBM RedBook: <http://www.redbooks.ibm.com/>

IBM i technical resource roadmap: <https://developer.ibm.com/components/ibm-i/articles/ibm-i-resources-roadmap/>

IBM Rational developer for i: <https://www.ibm.com/support/knowledgecenter/en/SSAE4W>

IBM i Access: <https://www.ibm.com/support/pages/node/633795>

RPG Café: <https://www.ibm.com/support/pages/node/1106229>

IBM News room: <http://www-03.ibm.com/press/photos.wss>

FAQ400: <http://www.faq400.com/>

Il guru Scott Klement: <http://www.scottklement.com/>

IT Jungle: <https://www.itjungle.com/>

MC Press online: <https://www.mcpressonline.com/>

TechChannel (old IBM Systems Magazine): <https://techchannel.com/ibm-i/>

IBM i updates: <https://www.ibmupdates.com/>

Simon Hutchinson: <https://www.rpgpgm.com/>

## Non essere possessivi

- allocare i record senza scopo non è sconsigliato...

**...è ORRENDO!!!**

- se il file è di I/O usare estensione N su tutte le letture fini a se stesse
- controllare record allocati con API



## Essere ordinati



- dove metto i pezzi del programma?
- dove metto i file vuoti?
- dove metto i prototipi?
- dove metto i frammenti di codice?
- ambiente sviluppo e produzione seguono stesse regole?



# Essere risparmiatori



- eseguire UPDATE solo quando serve
- ottimizzare le istruzioni dentro i cicli  
(0,001 sec x 400.000 = 7 minuti) → è un tempo macchina enorme
- non occupare spazio inutile con file work
- usare le CTE di SQL
- il disco costa! tempo e soldi



# Essere ecologici



- riciclare il **codice “sano”** e smaltire il **“nocivo”**
- riciclare le **buone esperienze** proprie e di altri
- non inventare cose che già esistono ma migliorarle
- solo una cosa non si può riciclare: gli **INDICATORI!!!**



26

- Conoscere, studiare ed usare le routine standard
- Creare le proprie routine quando necessario e pensarle perché possano essere usate sempre, ovunque e da chiunque
- Blocchi di codice brevi sono più facili da gestire e producono meno errori
- Fare tesoro dell'esperienza propria e "sfruttare" quella altrui per non cadere nei medesimi errori e per risparmiare tempo

OGNUNO E' PADRONE DEL SUO TEMPO, MA A FINE MESE DOBBIAMO RENDERE CONTO...

# Essere affidabili



- testare prima il programma...
- ... per evitare di “dare testate” dopo
- provare a settori e durante lo sviluppo e non solo alla fine
- individuare carenze di:
  - correttezza
  - completezza
  - affidabilità



## Usare le vecchie novità

- un poco alla volta
- sbirciare, leggere e imparare
- RPG ILE vs. RPG/400
- RPG SQL
- IFS vs. QDLS
- SQL vs. Query/400
- RDi vs. SEU/PDM



Usare la tecnologia  
**F.B.L.**

*Fa' Ballaa L'öcc*



Di fronte a un  
problema?



# VAS (Valutazione Ambientale Strategica)



# VAS (Valutazione Ambientale Strategica)

Non  
fermarsi  
sui  
particolari



Ampliare  
lo sguardo  
e valutare  
tutto il  
contesto



## Usare il panicometro

- conoscere la propria soglia di panico
- c'è una ragione valida per oltrepassarla?
- lavorare in stato di panico:
  - crea confusione
  - irrita
  - non risolve



## Usare le celle di isolamento

- isolare i problemi
- isolare i componenti del programma
- isolare se stessi mentalmente se necessario



# Guardarsi allo specchio



- l'errore che trovi potrebbe essere **come il tuo** oppure essere **proprio il tuo**
- 1° sbloccare il cliente
- 2° segnalare/correggere errore
- 3° trovare e ricostruire i dati errati
- **ultimo (facoltativo) epistemologia dell'errore**



35

L'**epistemologo** indaga come funziona la scienza, si chiede perché funzioni e in che modo possa funzionare al meglio, s'interroga sullo stato delle discipline, le loro forme, la loro organizzazione, le loro possibilità, i loro errori e i loro successi.

I luoghi migliori per praticare questa disciplina filosofica sono le tangenziali intorno a Milano, i balconi delle scale, nel raggio di 1 metro dalle macchinette del caffè e per taluni luogo di vera ispirazione è il bagno.

Recenti studi dimostrano che filosofeggiare davanti al video del PC può recare seri danni alla salute... ☺

☺ ☺

In breve quindi?



# Il programmatore deve avere **T.I.C.**

• Tenacia



• Intuito



• Curiosità



37

TENACIA:

- perseverare senza demordere
- non sempre le soddisfazioni sono proporzionali allo sforzo sostenuto

INTUITO:

- è il presentimento che una certa idea mai provata in precedenza possa funzionare...

CURIOSITA:

- sbirciare
- voglia di sapere sempre cose nuove