Mantenimiento del Software

S8

Francisco Ruiz, Macario Polo

Grupo Alarcos

Dep. de Informática

ESCUELA SUPERIOR DE INFORMÁTICA

UNIVERSIDAD DE CASTILLA-LA MANCHA

http://alarcos.inf-cr.uclm.es/doc/mso/

Ciudad Real, 2000/2001

Índice - Sesión 8

- Métricas para el mantenimiento de software
  - De producto
  - De proceso
  - Técnicas de estimación
  - Análisis de la cartera de aplicaciones
  - Ejemplo de aplicación: detección de módulos propensos a fallos
- Herramientas para el mantenimiento de software

Métricas

"Un método y una escala cuantitativos que pueden ser usados para determinar el valor que toma cierta característica en un producto software concreto".

(ISO/IEC, 1991)

"Una función que toma como entrada cierta información del software que se está midiendo, y que devuelve como salida un valor numérico sencillo, el cual es interpretado como el grado en que el producto software posee un atributo dado que afecta a su calidad".

(IEEE, 1992)

Métricas de producto

Aquélla que representa una característica del producto en un instante de tiempo claramente determinado.

(Henderson-Sellers, 1996)

Métricas de producto comunes

- Complejidad ciclomática (McCabe, 1976)
- Puntos de función (Albretch, 1979)
- Densidad de comentarios
- Cohesión
- Acoplamiento
- Tamaño
- Longitud

Ejemplo: número de McCabe (1976)

```
procedimiento A(f : Fichero; var fSalida : Fichero);
variables i : integer;
                          r : registro;
inicio
      i=0
      mientras i<Tamaño(f) hacer
         Leer(r)
         si r.saldo<100000 entonces
             si r.edad>21 entonces
                Escribir(fSalida, r)
             fin si  si no
             si r.saldo<1000000 entonces
                Escribir(fSalida, r)
             fin si
         fin si
         i = i+1
     fin mientras
fin A
```

$Vg(A)=5$

```
procedimiento B(var f : Fichero);
inicio
       OrdenacionPorMezcla(f)
fin B
```

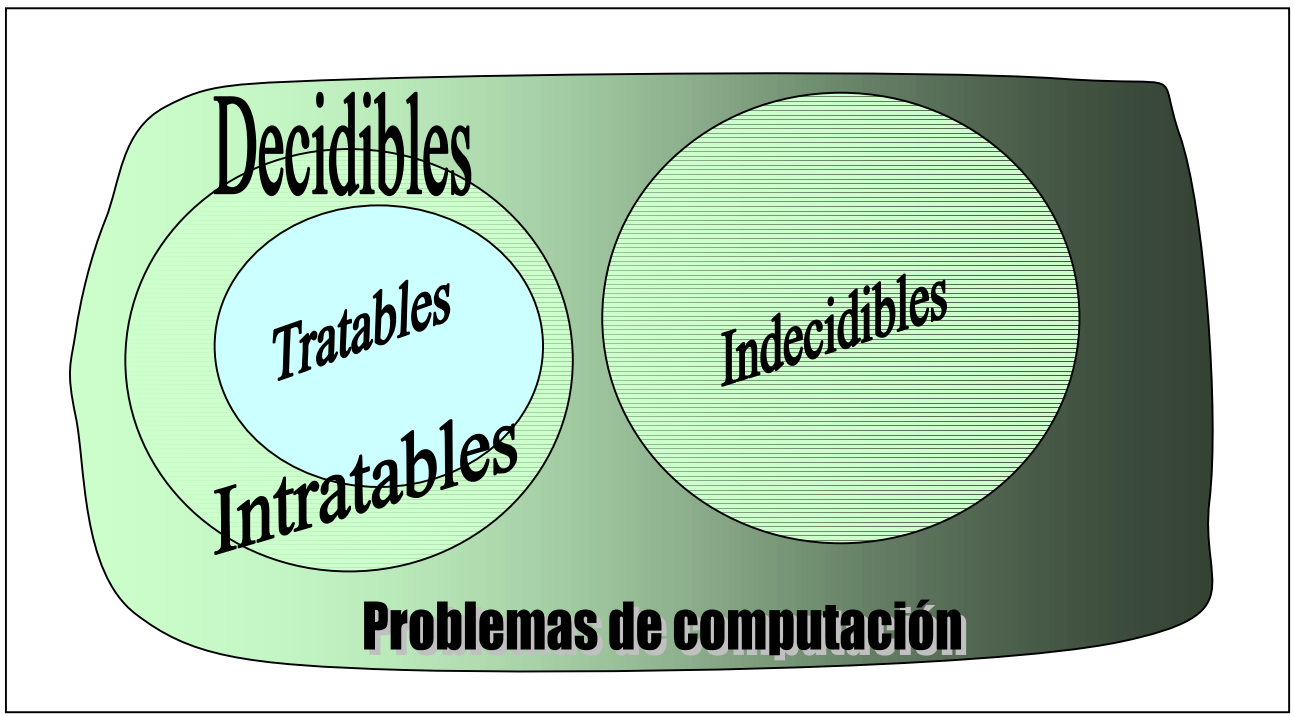
$Vg(B)=1$

```
procedimiento C(f : Fichero)
variables aux : Fichero;
inicio
      A(f, aux);
      B(aux);
fin C
```

$Vg(C)=1$

Otros conceptos de complejidad

- Computacional ≈ problema
- Algorítmica ≈ método de resolución



Métricas de proceso

Evalúan diferentes aspectos del proceso software, permitiendo a sus responsables efectuar previsiones de costes y esfuerzo, detectar anomalías, mejorar actividades, etc.

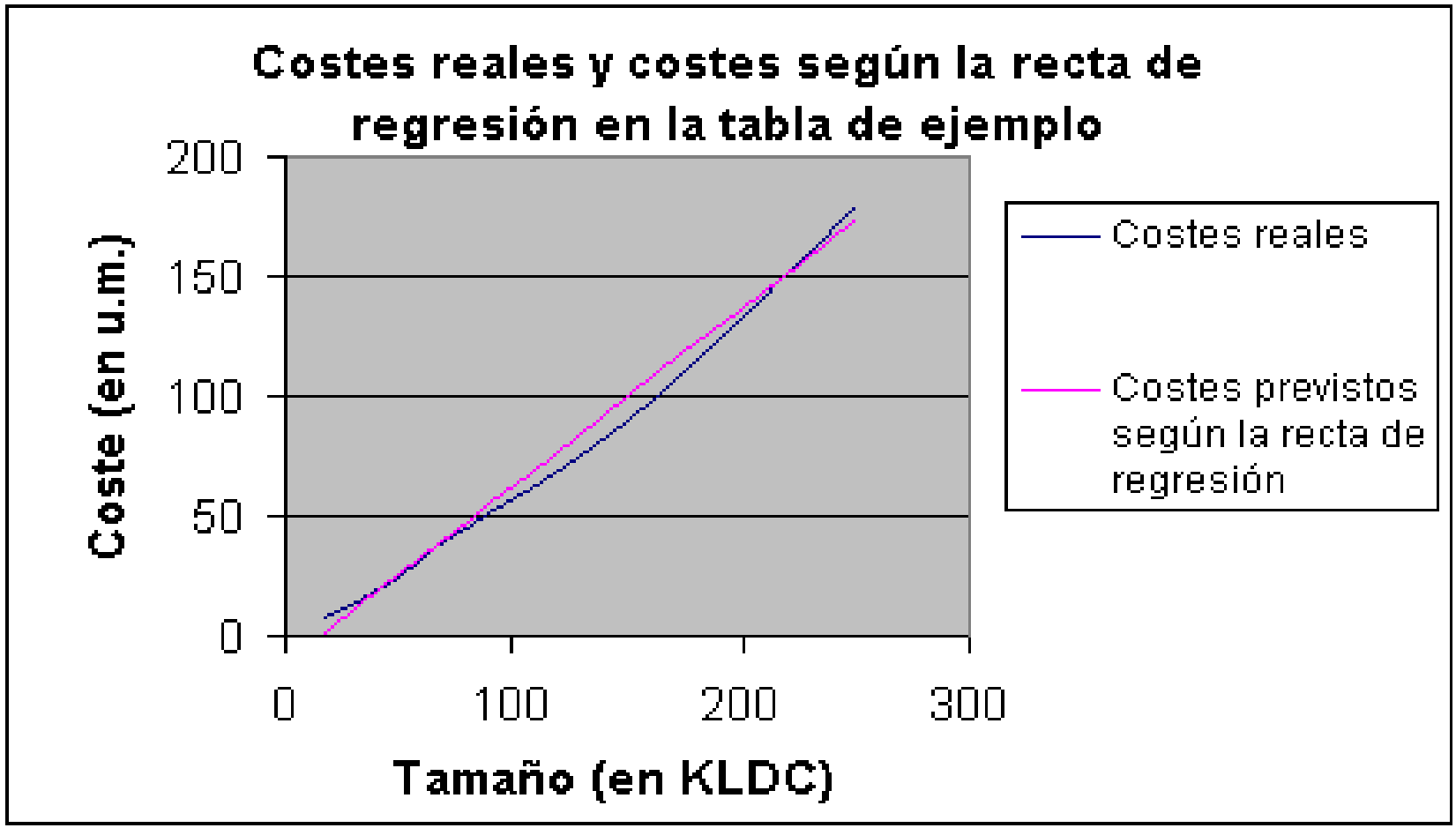
Métricas de producto como métricas de proceso

VERSION	SRC_NAME	No. Of routines	No. Of blanks	Bytes	COMMENTSLOC	COMPDIRSS	DATADECLSS	DECISIONS
v0.0	AGE_SN.C	6	57	6778	28	5	15	29
v1.0	AGE_SN.C	6	57	6778	28	5	15	29
v2.0	AGE_SN.C	6	57	6778	28	5	15	29
v3.0	AGE_SN.C	6	57	6778	28	5	15	29
v0.0	AGE_SN.H	1	29	1712	35	5	0	0
v1.0	AGE_SN.H	1	29	1712	35	5	0	0
v2.0	AGE_SN.H	1	29	1712	35	5	0	0
v3.0	AGE_SN.H	1	29	1712	35	5	0	0
v0.0	AGR2_CNF.H	1	10	363	0	3	0	0
v1.0	AGR2_CNF.H	1	10	363	0	3	0	0
v2.0	AGR2_CNF.H	1	10	363	0	3	0	0
v3.0	AGR2_CNF.H	1	10	363	0	3	0	0
v0.0	AGR2_INI.C	2	30	4929	3	0	11	10
v1.0	AGR2_INI.C	2	30	4929	3	0	11	10
v2.0	AGR2_INI.C	2	30	4929	3	0	11	10
v3.0	AGR2_INI.C	2	30	4929	3	0	11	10
v0.0	AGR2_MSG.C	28	117	20710	43	0	78	95
v1.0	AGR2_MSG.C	28	117	20710	43	0	78	95
v2.0	AGR2_MSG.C	28	117	20710	43	0	78	95
v3.0	AGR2_MSG.C	28	117	20710	43	0	78	95
v0.0	AGR2_MSG.H	1	32	4989	23	18	0	0
v1.0	AGR2_MSG.H	1	32	4989	23	18	0	0
v2.0	AGR2_MSG.H	1	32	4989	23	18	0	0
v3.0	AGR2_MSG.H	1	32	4989	23	18	0	0
v0.0	AGR2_PRI.C	2	17	2243	0	0	3	7
v1.0	AGR2_PRI.C	2	17	2243	0	0	3	7
v2.0	AGR2_PRI.C	2	17	2243	0	0	3	7
v3.0	AGR2_PRI.C	2	17	2243	0	0	3	7

Métricas de producto como métricas de proceso

Proyecto	Tamaño en KLDC	Coste de mantenimiento (en u.m.)
A	50	25
B	250	180
C	75	42
D	18	8
E	150	90

Métricas de producto como métricas de proceso



Técnicas de estimación

- Analogía
- Modelo COCOMO para mantenimiento
- Sistemas dinámicos
- Estimación con puntos-función

Estimación por analogía

Comparar las características del sistema que vamos a modificar con el mismo conjunto de características de otras modificaciones previamente realizadas al mismo o a otros sistemas

ANGEL

- Herramienta para estimar esfuerzo por analogía (Shepperd et al., 1996)
- 20 atributos a comparar de 21 proyectos necesitaron 42 horas de computación
- 12 atributos utilizaron 10 minutos para el mismo número de proyectos

Modelo COCOMO para el mantenimiento (Granja y Barranco, 1997)

$$TCA = \frac{NLN + NLM}{NLT}$$

...siendo:
NLN : nº de líneas nuevas.
NLM : nº de líneas modificadas.
NLT: nº total de líneas.

$$CM = TCA * CD$$

Coste de desarrollo

COCOMO

- El *Índice de Mantenibilidad* (IM) mide la facilidad de mantenimiento del producto considerado
- Toda acción de mantenimiento puede dividirse en tres tareas:
  - Comprensión de los cambios que deben hacerse.
  - Realización de las modificaciones necesarias.
  - Pruebas de los cambios realizados.

$$CM = CM_C + CM_R + CM_P$$

$$CM = TCA * CD * IM$$

COCOMO

- Comprensión de los cambios que deben hacerse:

  I_C : *índice de comprensibilidad*

- Realización de las modificaciones necesarias:

  I_R : *IM de la realización de cambios*

- Pruebas de los cambios realizados:

  I_P : *IM de las pruebas*

COCOMO

Tendremos que:

- $CM_C = TCA * CD * I_C$
- $CM_R = TCA * CD * I_R$
- $CM_P = TCA * CD * I_P$

COCOMO

Con lo que la expresión

$$CM = CM_C + CM_R + CM_P$$

...puede desglosarse en:

$$CM = TCA * CD * (I_C + I_R + I_P)$$

COCOMO

Los diferentes “índices de mantenibilidad” se calculan usando datos históricos y una serie de matrices a partir de:

- X_C : porcentaje de líneas comentadas por cada cien, para calcular I_C.
- X_R : porcentaje de líneas sin datos constantes por cada cien, para calcular I_R.
- X_P : número de errores comprobados por cada cien líneas de código, para I_P.

Sistemas dinámicos (Calzolari et al., 1998)

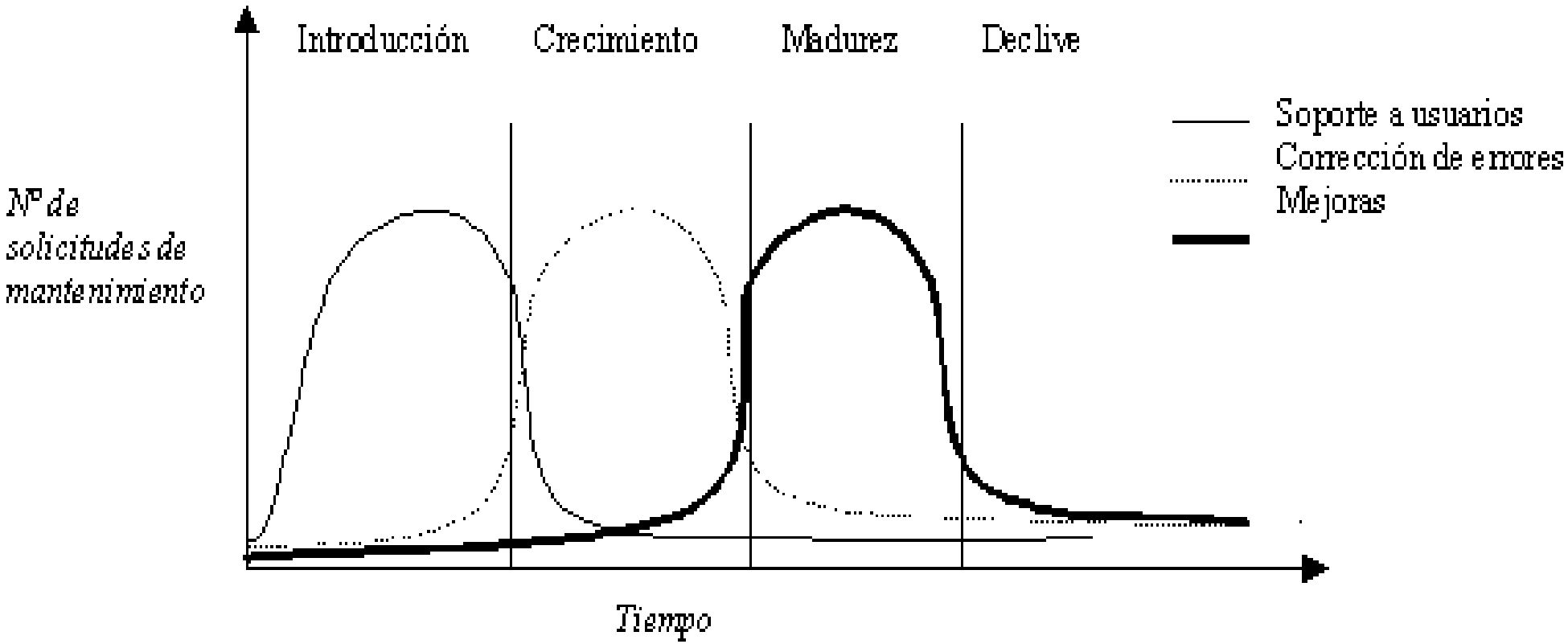
- Basado en el modelo predador/presa introducido en 1972 por May
- En ausencia de presas, los predadores se extinguen
- En ausencia de predadores, la población de presas alcanza y sobrepasa la capacidad del entorno para alimentarlas.
- En los casos intermedios se alcanza un equilibrio estable

Sistemas dinámicos (Calzolari et al., 1998)

- El mantenimiento correctivo es esencialmente predador de defectos software, y el esfuerzo de mantenimiento se alimenta de errores descubiertos por el usuario.

- Los mantenimientos perfectivo y adaptativo se alimentan de necesidades del usuario, y el esfuerzo de estos dos tipos de mantenimiento se adapta a la cantidad de solicitudes de estos dos tipos de mantenimiento.

Distribución de la llegada de solicitudes de modificación (Kung y Hsu, 1998).



Estimación con Puntos-función (Niessink y Van Vliet, 1997)

- Realizan ciertos análisis comparativos entre el esfuerzo de mantenimiento de varias intervenciones y el número de puntos-función modificados, contados según métodos diferentes

- No obtienen los resultados que esperaban

- Sin embargo, concluyen que el esfuerzo de mantenimiento es mucho más dependiente del tamaño del componente que se va a cambiar, que del tamaño del propio cambio (medidos ambos parámetros en puntos-función)

Estimación con Puntos-función (Niessink y Van Vliet, 1997)

- Sin embargo, concluyen que el esfuerzo de mantenimiento es mucho más dependiente del tamaño del componente que se va a cambiar, que del tamaño del propio cambio (medidos ambos parámetros en puntos-función)

Es decir:

$$Esfuerzo \approx K \times Tamaño\ del\ componente \times (1 + \varepsilon \times Tamaño\ del\ cambio)$$

En vez de:

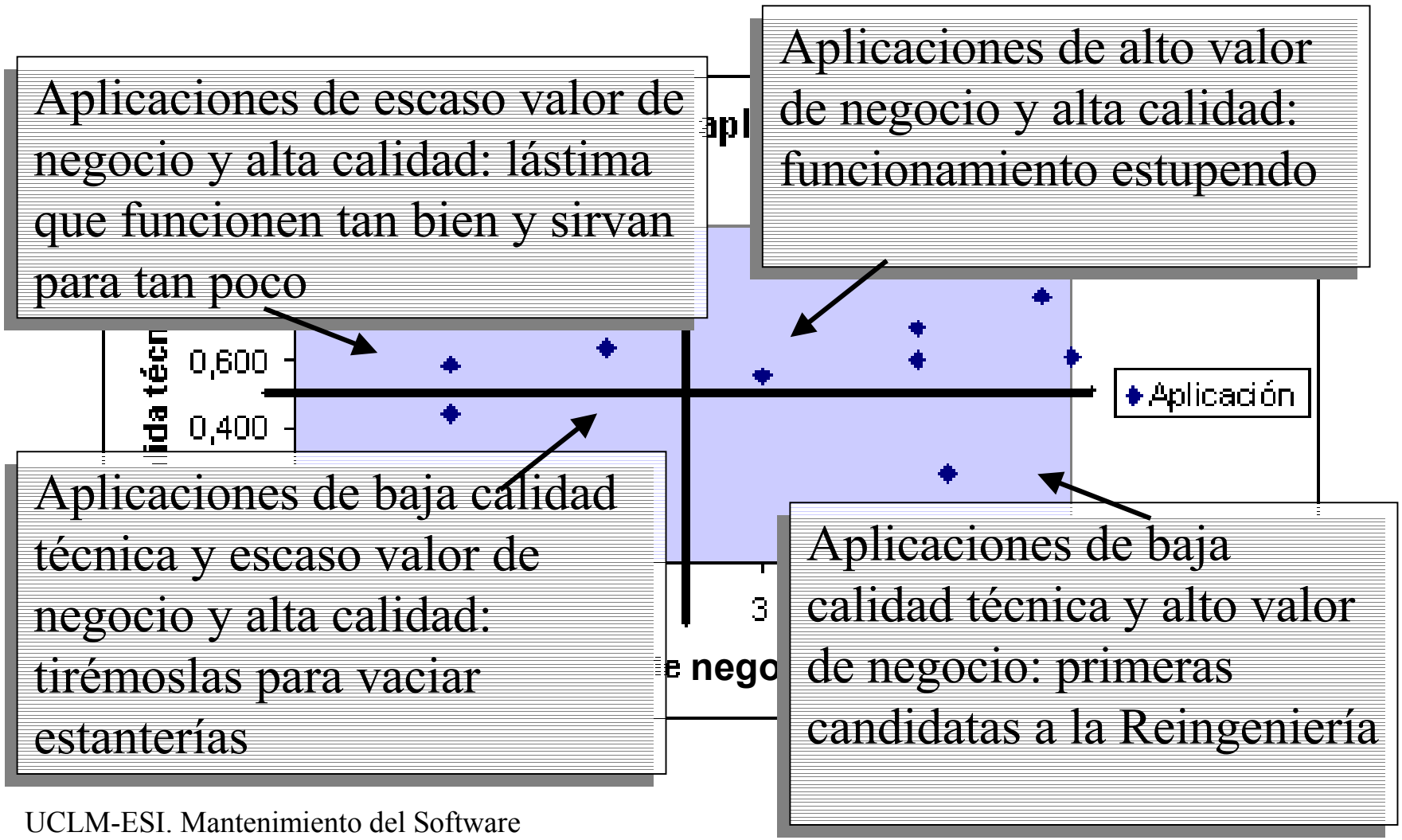
$$Esfuerzo \approx K \times Tamaño\ del\ cambio$$

Reingeniería

Reingeniería: ¿lo hacemos?

- Alternativas:
  - Dejar el software como está
  - Comprar la aplicación
  - Desarrollar de nuevo
  - Aplicar Reingeniería

Análisis de cartera (Sneed, 1995)



¿Es realmente conveniente aplicar Reingeniería?

Partimos de la siguiente “verdad universal”:

Dejar el software como está...

Comprar la aplicación...

Desarrollar de nuevo...

Aplicar Reingeniería...

...cuesta dinero.

Factores que se consideran (I)

- En todos los casos, incurriremos en...
  - Coste de mantenimiento
  - Coste de operaciones

- De todos los casos, obtendremos...
  - Valor de negocio

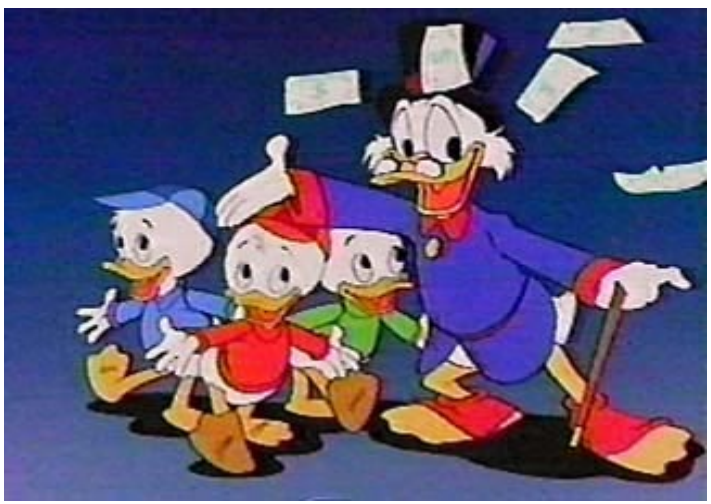
Factores que se consideran (II)

- En la Reingeniería y Nuevo desarrollo...
  - Coste de la Reingeniería o el nuevo desarrollo
  - Coste de las operaciones tras la Reingeniería o el nuevo desarrollo
  - Duración de la Reingeniería o el nuevo desarrollo

- En cualquiera de los casos...
  - La vida del sistema será limitada

Solución:

Realizar un Análisis de costes y beneficios y quedarnos con lo que más provechoso nos resulte



Si lo dejamos como está...

$$B_M=(VN-(CM+COp))\cdot \text{Vida del sistema}$$

VN--> valor de negocio

CM--> coste de mantenimiento

COp--> coste de las operaciones

Si desarrollamos de nuevo...

$$BB_D=((VN'-CM'-COp')\cdot(Vida-DurND)-$$
$$-(CND\cdot FRND))$$

$$BN_D=BB_D-B_M$$

Vida--> vida esperada del sistema

DurND--> duración del nuevo desarrollo

CND--> coste del nuevo desarrollo

FRND--> factor de riesgo del nuevo desarrollo

Si aplicamos Reingeniería...

$$BB_R=((VN''-CM''-Cop'')\cdot(Vida-DurRe)-(CRe\cdot FRRe))$$

$$BN_R=BB_R-B_M$$

Vida--> vida esperada del sistema

DurRe--> duración del proyecto de Reingeniería

CRe--> coste de proyecto de Reingeniería

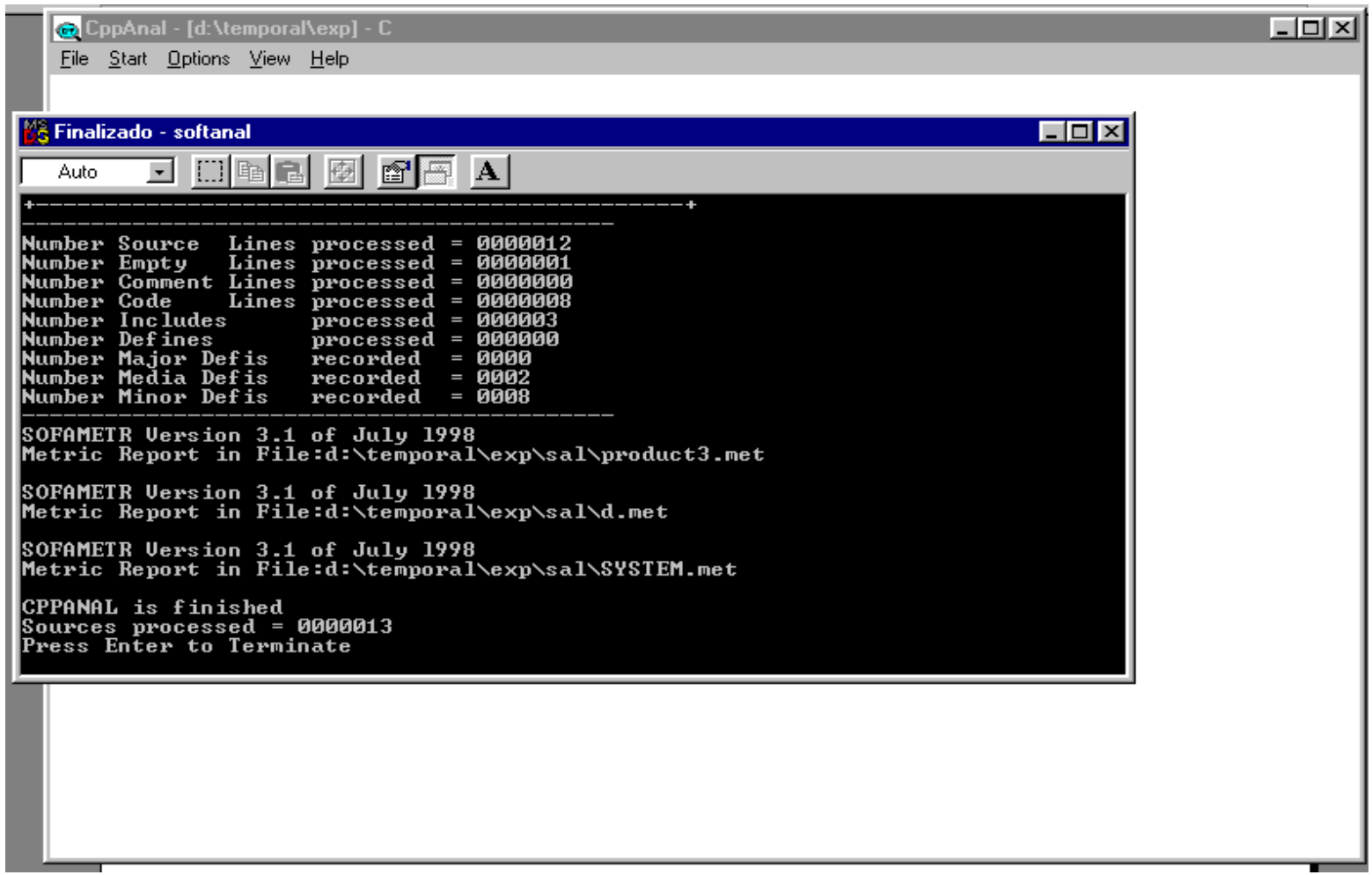
FRRe--> factor de riesgo de la Reingeniería

Herramientas para métricas

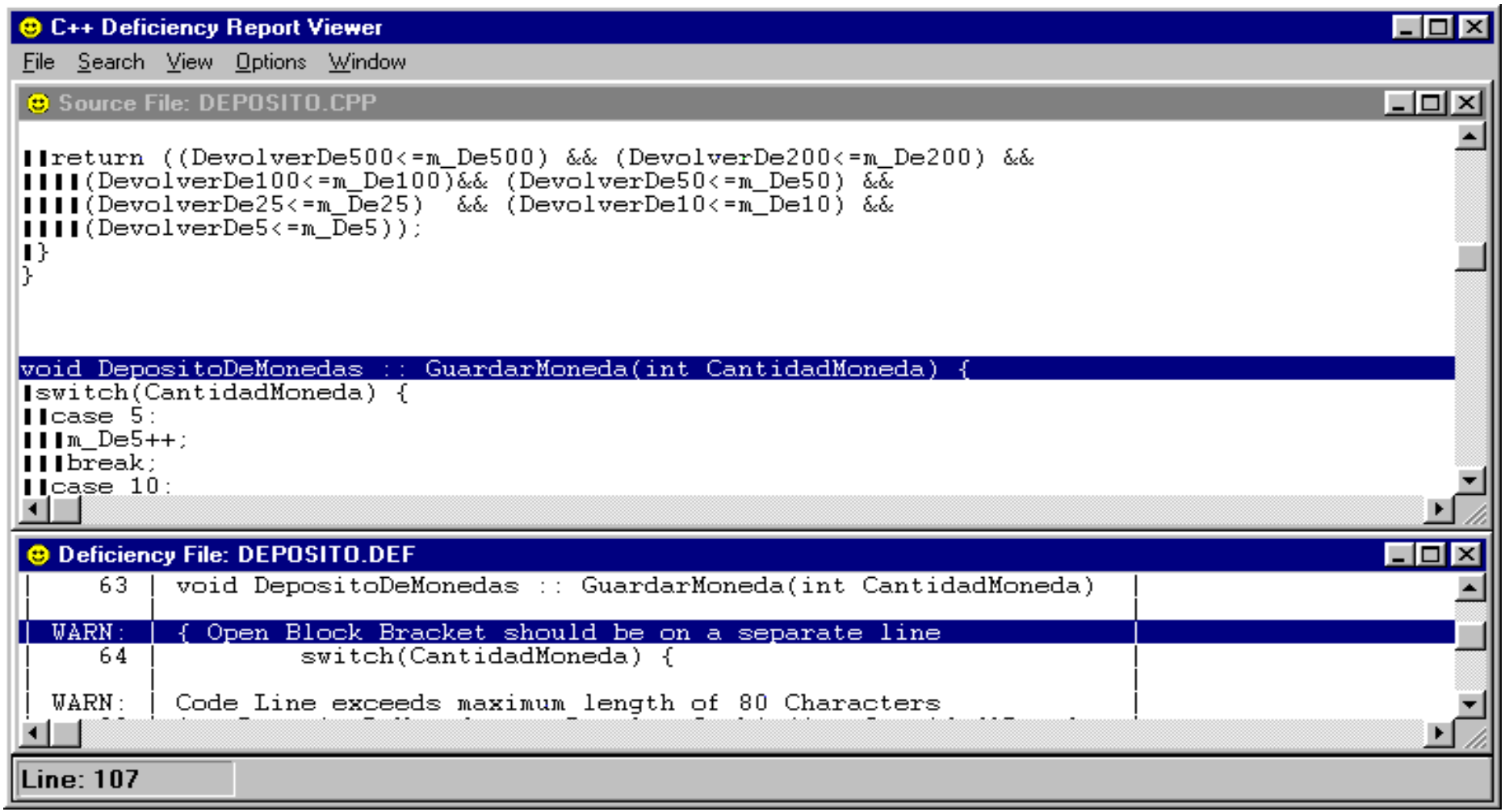
Herramientas de Sneed

- Hacer herramientas es su hobby
- Ejemplos:
  - Analizador de C++ y C
  - Visor de defectos
  - Adquisición de docenas de métricas

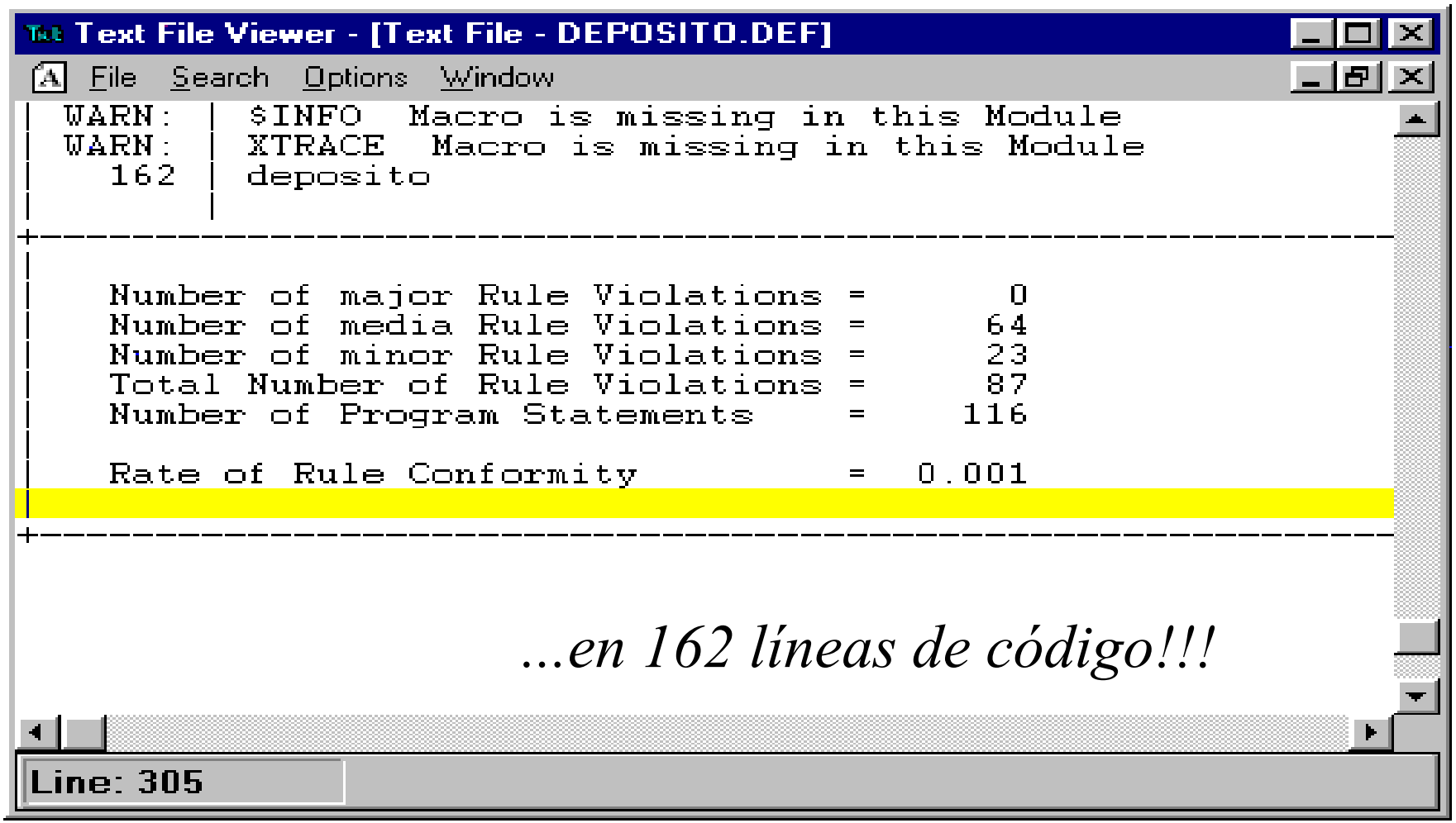
Analizador de C y C++



Visor de defectos



Visor de defectos



Métricas

- Number of Source Members analyzed
- Number of Source Lines in all
- Number of Genuine Code Lines
- Number of Comment Lines
- Number of Major Rule Violations
- Number of Medium Rule Violations
- Number of Minor Rule Violations
- Number of Header Files
- Number of Classes declared
- Number of Classes inherited
- Number of Methods declared
- Number of Methods inherited
- Number of Procedures declared
- Number of Object-Points
- Number of Panels processed
- Number of Reports produced
- Number of Files declared
- Number of Data Bases accessed
- Number of Data Structures
- Number of Defined Definitions

Métricas

- Number of Data Variables declared
- Number of Data Variables inherited
- Number of Data Constants/Enums declared
- Number of Redefinitions (Unions)
- Number of Arrays (Vectors)
- Number of external Data Elements
- Number of different Data Types used

- Number of Data References
- Number of Arguments / Input Variables

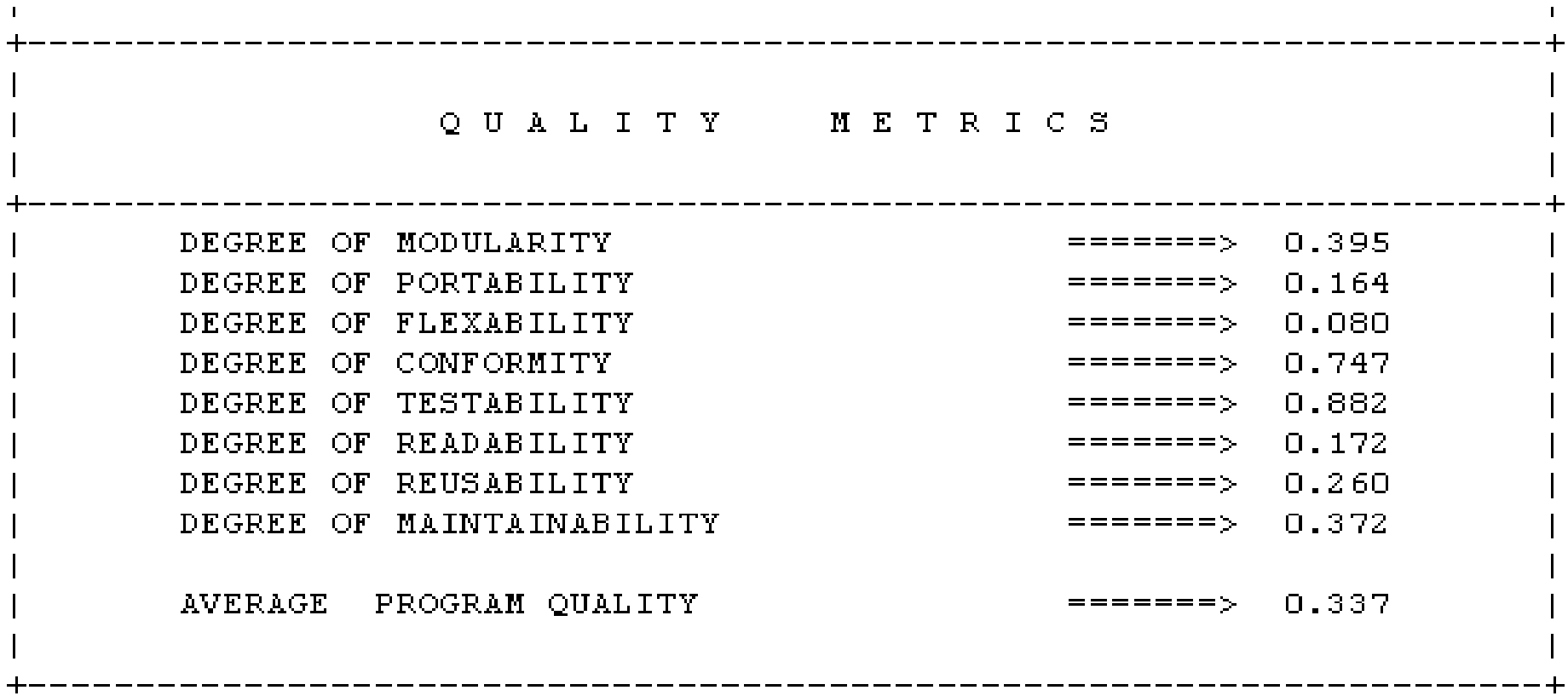
- Number of Results / Output Variables

- Number of Predicates / Conditional Data
- Number of Parameters / Function Arguments
- Number of Data-Points
- Number of Statements
- Number of Input Operations
- Number of Output Operations
- Number of File & Database Accesses
- Number of Function References
- Number of Foreign Functions referenced
- Number of Macro References

Métricas

- Number of Macros referenced
- Number of If Statements
- Number of Switch Statements
- Number of Case Statements
- Number of Loop Statements
- Number of GOTO Branches
- Number of Return statements
- Number of Control Flow Branches
- Number of all Control Statements
- Number of Nesting Levels (Maximum)
- Number of Literals in Statements
- Number of different Statement Types
- Number of Function-Points
- DATA COMPLEXITY
- DATA FLOW COMPLEXITY
- DATA ACCESS COMPLEXITY
- INTERFACE COMPLEXITY
- CONTROL FLOW COMPLEXITY
- DECISIONAL COMPLEXITY
- BRANCHING COMPLEXITY
- LANGUAGE COMPLEXITY

Calidad técnica



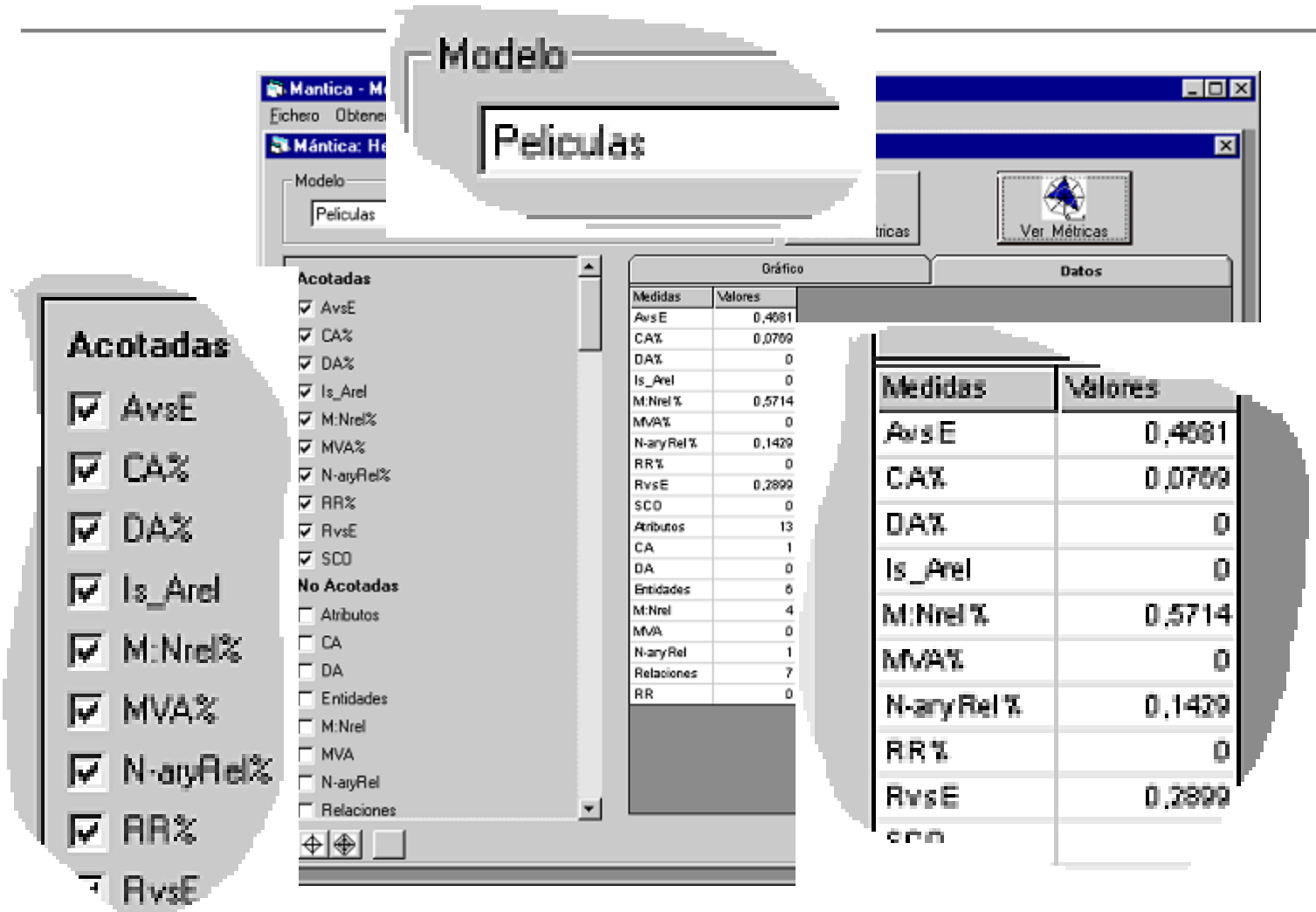
Q U A L I T Y M E T R I C S		
DEGREE OF MODULARITY	=======>	0.395
DEGREE OF PORTABILITY	=======>	0.164
DEGREE OF FLEXABILITY	=======>	0.080
DEGREE OF CONFORMITY	=======>	0.747
DEGREE OF TESTABILITY	=======>	0.882
DEGREE OF READABILITY	=======>	0.172
DEGREE OF REUSABILITY	=======>	0.260
DEGREE OF MAINTAINABILITY	=======>	0.372
AVERAGE PROGRAM QUALITY	=======>	0.337

Herramientas del Grupo Alarcos

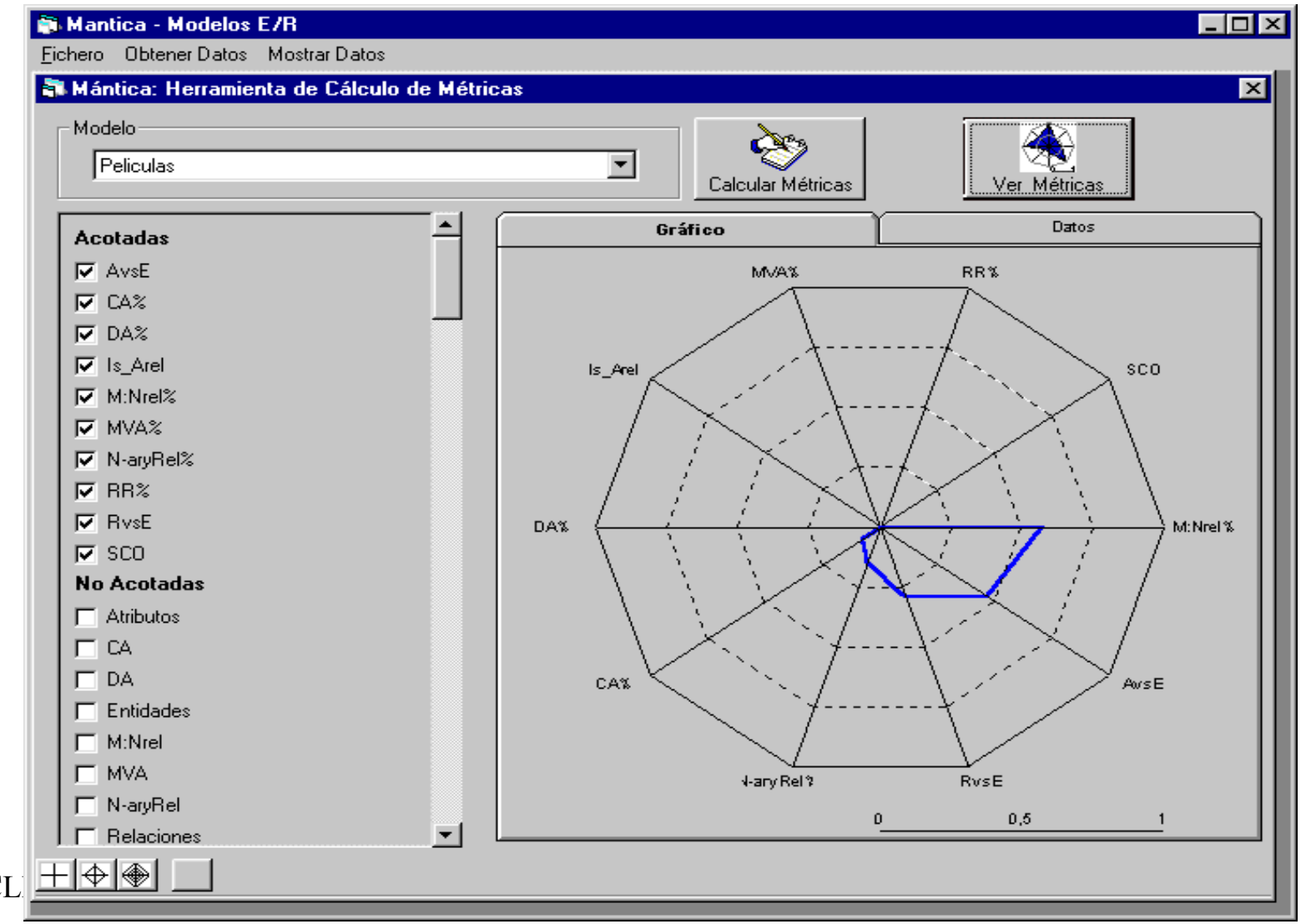
- MÁNTICA: una Herramienta de Métricas para Modelos de Datos

- MANTOOL: gestión del Proceso de Mantenimiento conforme a la Metodología MANTEMA

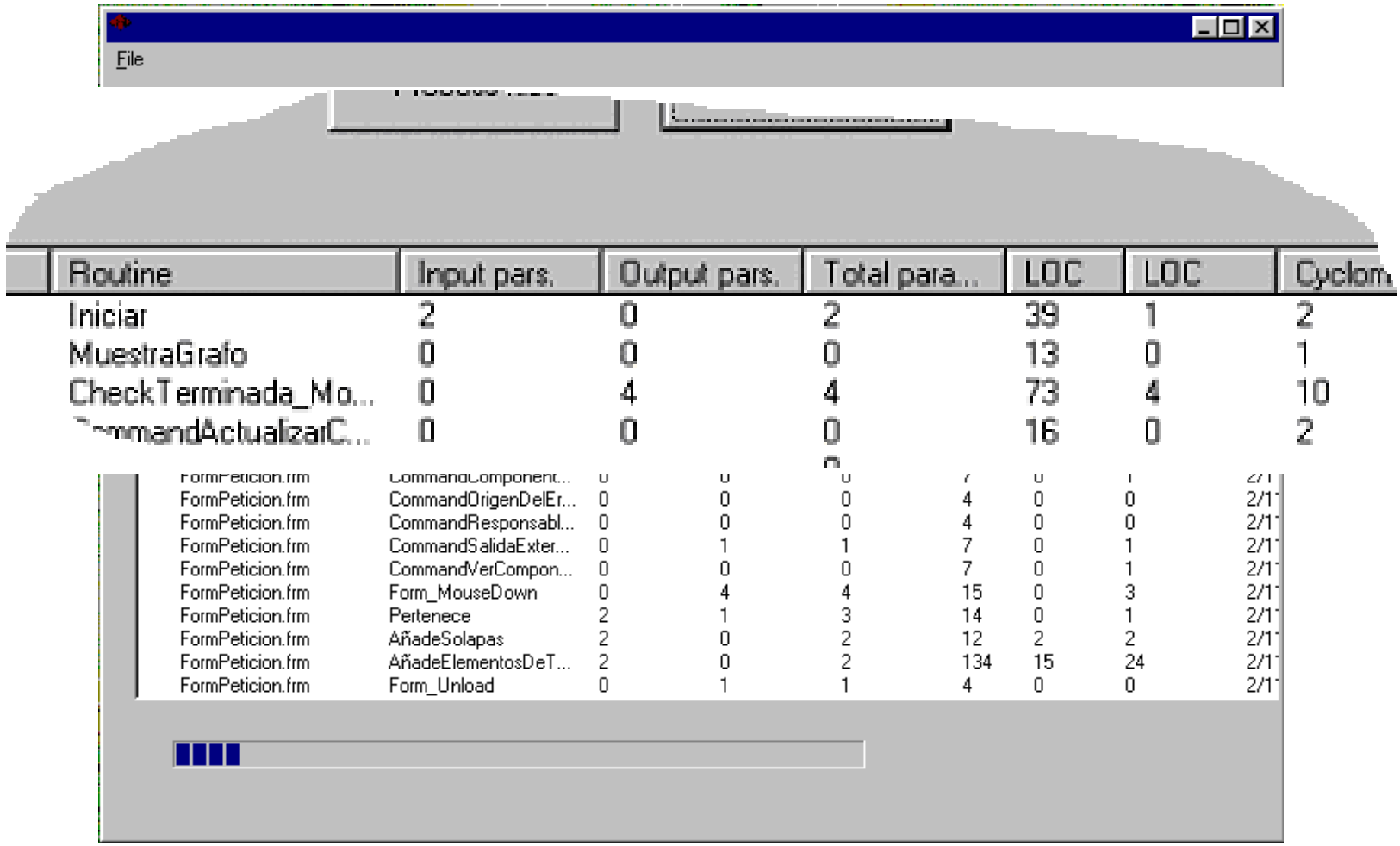
MÁNTICA


Modelo
Peliculas
Ver Métricas
Gráfico
Datos
Acotadas
AvsE
CA%
DA%
Is_Arel
M:Nrel%
MVA%
N-aryRel%
RR%
RvsE
SCO
No Acotadas
Atributos
CA
DA
Entidades
M:Nrel
MVA
N-aryRel
Relaciones
Medidas
Valores
AvsE 0,4081
CA% 0,0769
DA% 0
Is_Arel 0
M:Nrel % 0,5714
MVA% 0
N-aryRel % 0,1429
RR% 0
RvsE 0,2899
SCO 0
Atributos 13
CA 1
DA 0
Entidades 6
M:Nrel 4
MVA 0
N-aryRel 1
Relaciones 7
RR 0

MÁNTICA


Mantica - Modelos E/R
Fichero
Obtener Datos
Mostrar Datos
Mántica: Herramienta de Cálculo de Métricas
Modelo
Peliculas
Calcular Métricas
Ver Métricas
Acotadas
AvsE
CA%
DA%
Is_Arel
M:Nrel%
MVA%
N-aryRel%
RR%
RvsE
SCO
No Acotadas
Atributos
CA
DA
Entidades
M:Nrel
MVA
N-aryRel
Relaciones
Gráfico
Datos
MVA%
RR%
Is_Arel
SCO
DA%
M:Nrel%
CA%
AvsE
N-aryRel%
RvsE
0
0,5
1

MANTOOL

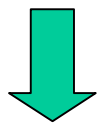

File
Routine
Input pars.
Output pars.
Total para...
LOC
LOC
Cyclom
Iniciar 2 0 2 39 1 2
MuestraGrafo 0 0 0 13 0 1
CheckTerminada_Mo... 0 4 4 73 4 10
CommandActualizarC... 0 0 0 16 0 2
FormPeticion.frm CommandOrigenDelEr... 0 0 0 4 0 0 2/1
FormPeticion.frm CommandResponsabl... 0 0 0 4 0 0 2/1
FormPeticion.frm CommandSalidaExter... 0 1 1 7 0 1 2/1
FormPeticion.frm CommandVerCompon... 0 0 0 7 0 1 2/1
FormPeticion.frm Form_MouseDown 0 4 4 15 0 3 2/1
FormPeticion.frm Pertenece 2 1 3 14 0 1 2/1
FormPeticion.frm AñadeSolapas 2 0 2 12 2 2 2/1
FormPeticion.frm AñadeElementosDeT... 2 0 2 134 15 24 2/1
FormPeticion.frm Form_Unload 0 1 1 4 0 0 2/1

Módulos propensos a fallos

Módulos propensos a fallos

- Un pequeño nº de módulos contiene la mayoría de los errores detectados durante la fase de pruebas

- Si un pequeño nº de módulos contiene la mayoría de los errores detectados durante la fase de pruebas, es porque esos módulos contienen la mayor parte del tamaño del sistema

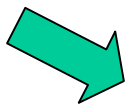


- Un pequeño número de módulos contiene la mayoría de los errores detectados tras la entrega al cliente

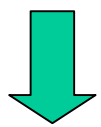
Módulos propensos a fallos

- Si un pequeño número de módulos contiene la mayoría de los errores detectados tras la entrega al cliente, es porque esos módulos ocupan la mayor parte del tamaño del sistema

- Si un pequeño número de módulos contiene la mayoría de los errores detectados tras la entrega al cliente, es porque esos módulos tuvieron la mayor incidencia de errores en la fase de pruebas

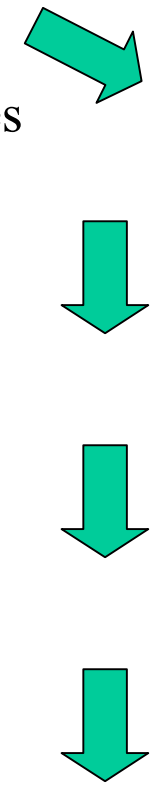


- Los módulos pequeños son menos propensos a fallos que los grandes



Módulos propensos a fallos

- Las métricas de tamaño (LDC) son buenas predictoras del número de errores antes de la entrega
- Las métricas de tamaño (LDC) son buenas predictoras del número de errores después de la entrega
- Las métricas de tamaño (LDC) son buenas predictoras de la densidad de errores antes de la entrega
- Las métricas de tamaño (LDC) son buenas predictoras de la densidad de errores después de la entrega



Módulos propensos a fallos

- Las métricas de complejidad son mejores predictoras que las métricas de tamaño para detectar la propensión a fallos

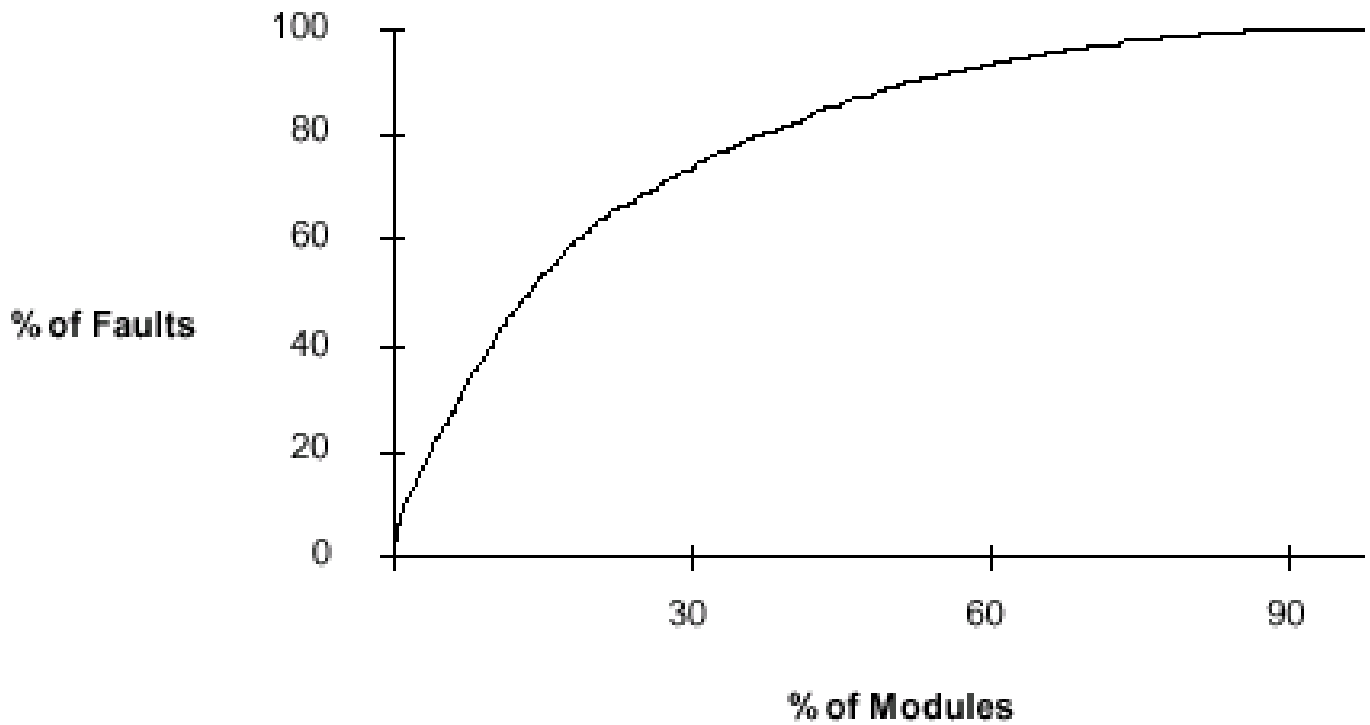
- Los sistemas producidos en entornos similares tienen densidades de fallos más o menos similares, tanto antes como después de la entrega

http://www.agena.co.uk/new_direction_metrics/HelpFileWhat_you_can_and_cannot_do_with_the_basic_metrics.html

Publicado en IEEE TSE (mayo 2000, Fenton y Ohlsson)

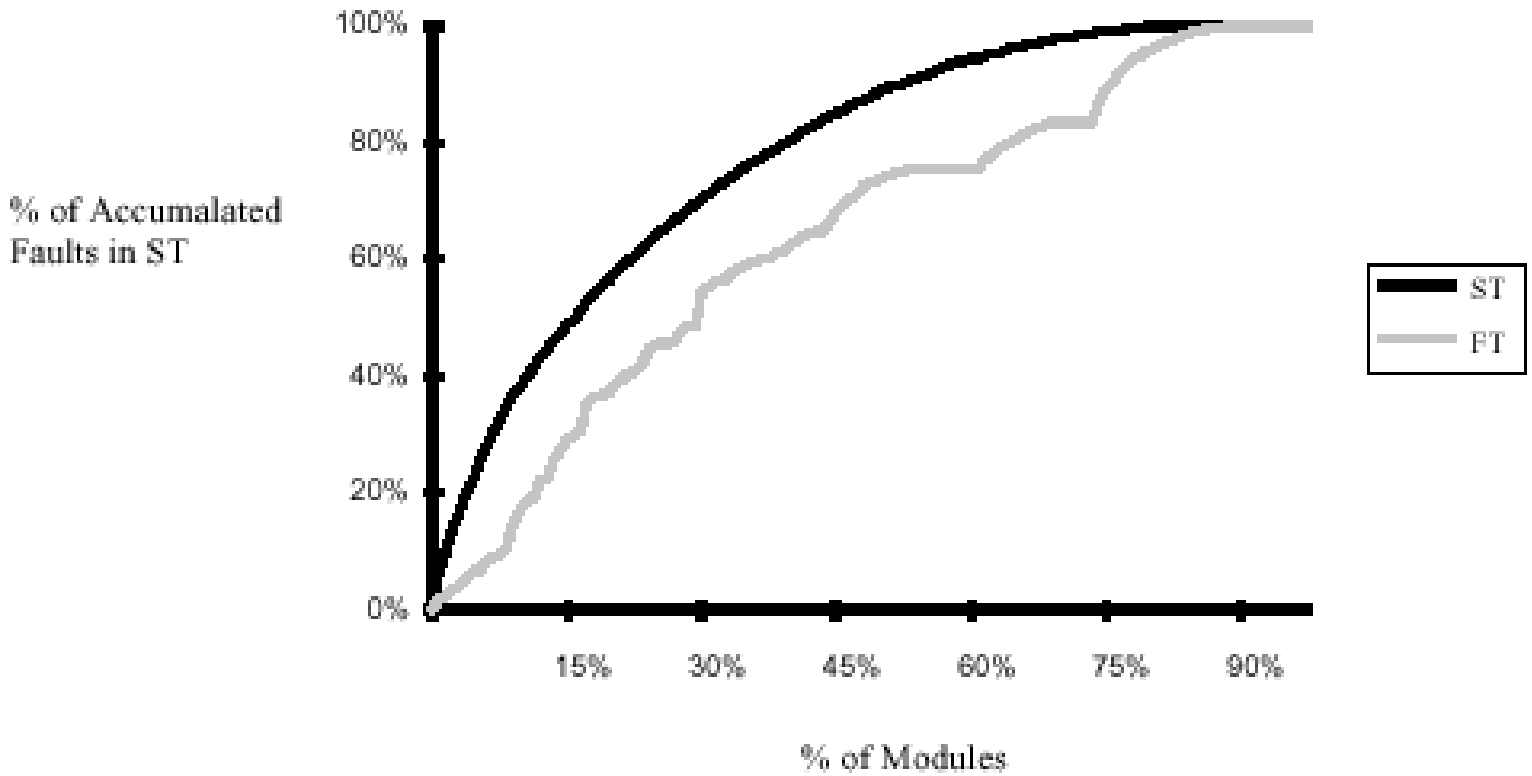
Estudio empírico

- Hipótesis 1: Un pequeño nº de módulos contiene la mayoría de los errores detectados durante la fase de pruebas



Estudio empírico

- H3: La alta incidencia de fallos en las pruebas de funciones (FT) implica alta incidencia de fallos en las pruebas del sistema (FT).



Estudio empírico

- H6: Las métricas de complejidad son mejores predictoras que las métricas de tamaño para detectar la propensión a fallos

Cierta durante las pruebas, y falsa tras ellas. Parece debido a que son los módulos complejos probados con más cuidado que los simples.

Detección de módulos propensos a fallos

Detección de módulos propensos a fallos (Zage, Zage y Wilburn, 1994)

- Proponen dos métricas de medida de la calidad del diseño:
  - *De*, para medir la calidad del diseño de un módulo, desde el punto de vista de sus relaciones con el exterior

Detección de módulos propensos a fallos (Zage, Zage y Wilburn, 1994)

- Con el exterior:

$$D_e = e_1 * (\text{inflows} * \text{outflows}) + e_2 * (\text{fan\_in} * \text{fan\_out})$$

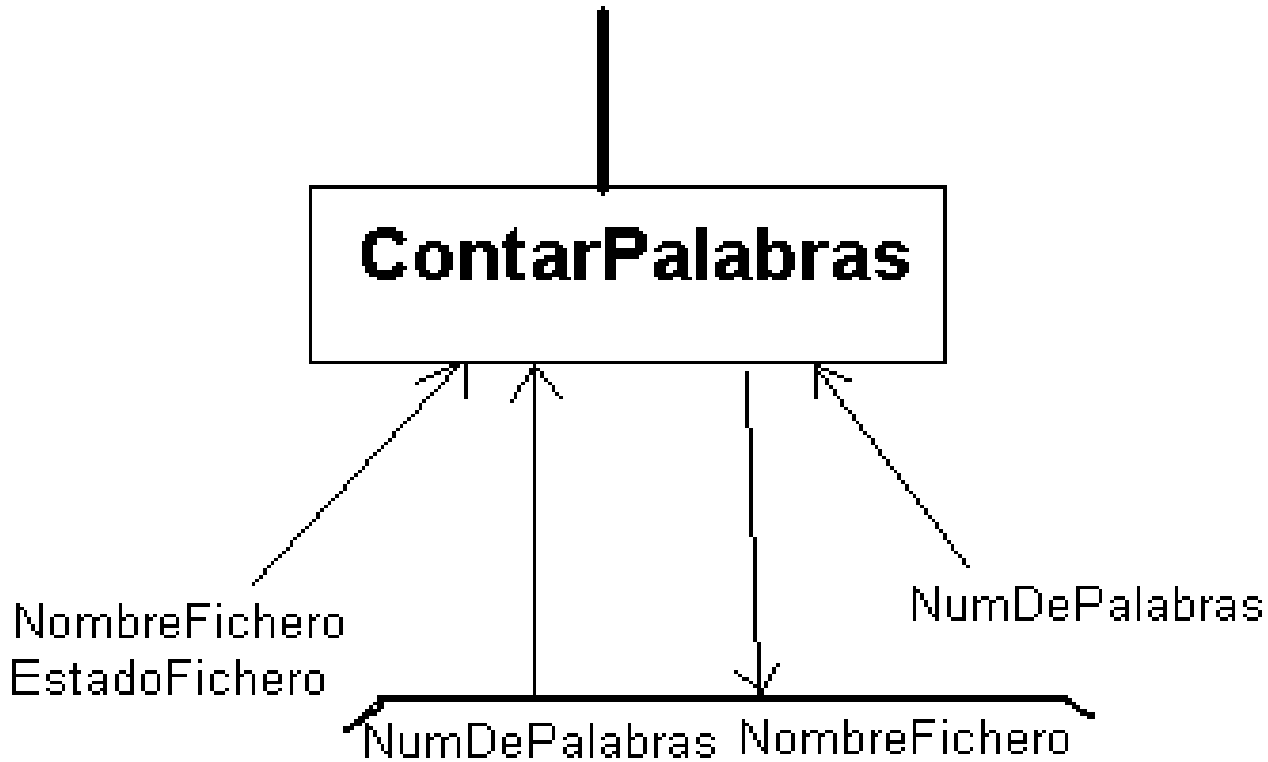
inflows: nº de entidades de datos pasadas al módulo desde otros módulos

outflows: nº de entidades de datos pasadas desde el módulo a otros módulos

fan_in: nº de módulos de nivel superior a los que está conectado

fan_out: nº de módulos de nivel inferior a los que está conectado

Detección de módulos propensos a fallos (D_e)



$D_e=(4*1)+(1*3)=7$ *(pesos unitarios)*

Resultados experimentales con D_e

- Proyectos de la universidad de entre 2.000 y 30.000 LDC
  - 12% tenían valores altos de D_e , un 50% con errores
  - 53% de los errores detectados estaban en esos módulos
- Proyecto industrial (532 programas, 2,5 M LDC)
  - 12% valores altos de D_e , un 50% con errores
  - 67% de los errores contenidos en ellos

Detección de módulos propensos a fallos (Zage, Zage y Wilburn, 1994)

- Con el interior:

$$D_i = i_1 * CC + i_2 * DSM + i_3 * I\_O$$

- CC (Central Calls): nº de llamadas a funciones o procedimientos
  - DSM (Data Structure Manipulations): nº de referencias a tipos de datos complejos
  - I_O (Input/Output): nº de acceso a dispositivos externos

Detección de módulos propensos a fallos (D_i)

```
procedure ContarPalabras
   Obtener(NombreFichero, EstadoFichero)
   if EstadoFichero=Error1 then
        Error_Table(1)=NombreFichero
        Escribir("Error: fichero inválido")
   else
        ContarNumPalabras(NombreFichero, NumPalabras)
        if NumPalabras=-1 then
               Error_Table(2)="Fichero no existe"
               Escribir("Error: no existe")
        else
               Procesar(NumPalabras)
        endif
   endif
end
```

$D_i = CC + \mathbf{DSM} + I\_O = 3+2+2 = 7$ *(pesos unitarios)*

Resultados experimentales con D_i

- Aplicación “Advanced Field Artillery Tactical Data System”
- Obtuvieron los mejores resultados cuando:
  - $i_1=i_3=1$
  - $i_2=2.5$
- 94% de los errores en el 89% de los módulos detectados

Comparativa

	Vg	LDC	D_i
Módulos clasificados	11%	11%	11%
Módulos clasificados con errores	44%	56%	89%
Errores encontrados	37%	51%	94%
Falsos positivos	56%	44%	11%

Métrica de diseño

$$D(G)=D_e+D_i$$

	D_e	D_i	D(G)
Módulos clasificados	12%	11%	12%
Módulos clasificados con errores	50%	89%	100%
Errores encontrados	53%	94%	97%
Falsos positivos	50%	11%	0%

Conclusión: factores influyentes

- El acoplamiento en general
  - Nº de módulos utilizados
  - Nº de llamadas a funciones y procedimientos
  - Parámetros
  - Nº de estructuras de datos manipuladas
  - Nº de accesos a dispositivos externos

Conclusión: factores influyentes