Software Quality Assurance (SQA)

Índex

- Introducció
- *Standards*
- *Quality Metrics*

Índex

- **Introducció**
- *Standards*
- *Quality Metrics*

Introducció

- **Objectiu** Enginyeria del Software:
  - SW de **qualitat**!
    - ➔ La qualitat **es construeix**, **no** es pot afegir **al final**

- Qualitat:
  - **Externa**: la percebuda per **l'usuari**
  - **Interna**: la percebuda pels **desenvolupadors**
  - Com s'assoleix?:
    - Complint especificacions: funcionals i no funcionals.
    - Desenvolupant seguint procediments standard
    - Característiques implicites: mantenible, segur, documentat

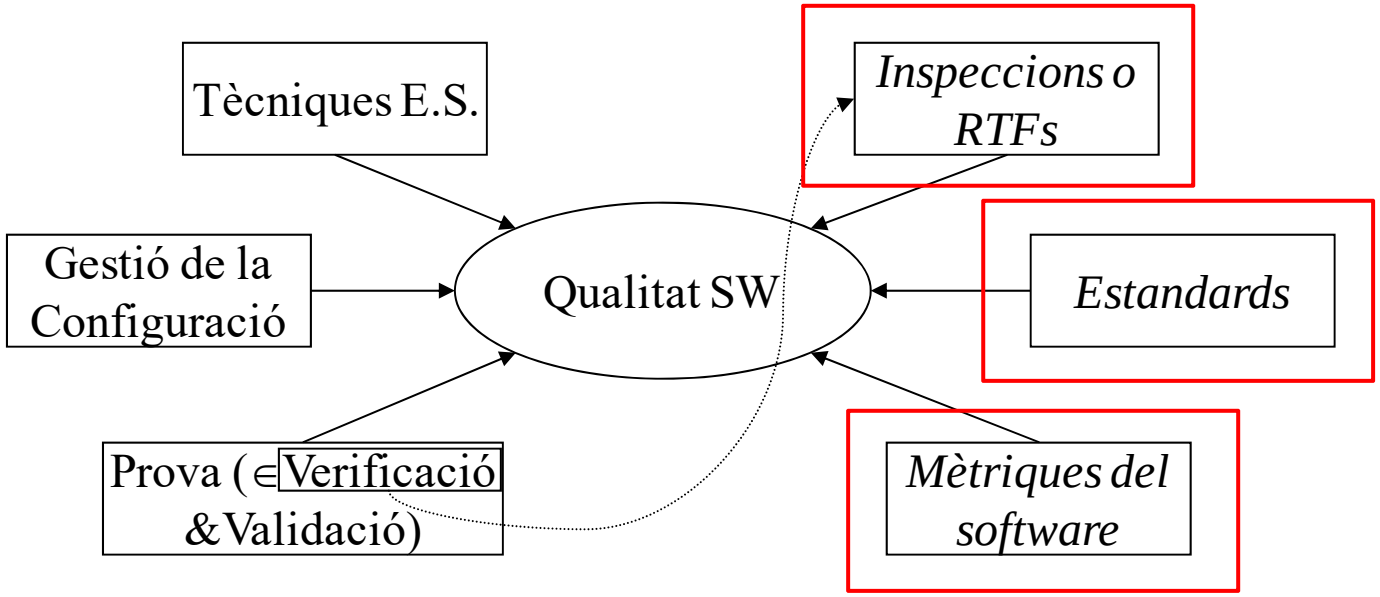
Introducció

- **Quins factors** defineixen la Qualitat? (i què mesuren):
  - Característiques operatives (funcionament del SW)
    - *Correcció*: Grau de compliment dels requeriments
    - *Fiabilitat*: Exactitud del funcionament
    - *Eficiència*: Quantitat de recursos necessaris
    - *Facilitat d'us i aprenentatge*
    - *Seguretat*
  - Revisió del producte (facilitat dels canvis)
    - *Facilitat de manteniment*: Esforç per localitzar i arreglar un defecte
    - *Flexibilitat*: Esforç per fer les millores
    - *Facilitat de prova*
  - Transició del producte (adaptabilitat a nous entorns)
    - *Portabilitat*: Esforç per migrar
    - *Reusabilitat*: Quantitat de SW que es pot reaprofitar per a altres apliacions
    - *Interoperativitat*: Esforç per acoplar el SW a un altre sistema

Com es mesuren? (veure capítol *Mètriques*)

Introducció

- Què contribueix a la Qualitat del SW?



Índex

- Introducció
- ***Standards***
- *Quality Metrics*

Standards

- Defineixen característiques de:
  - Tipus de **productes de SW** (programes, manuals, documents d'especificació, disseny, proves, …)
  - **Processos d'enginyeria** (com fer revisions tècniques formals, com aplicar mètriques, com fer gestió de la configuració, models de desenvolupament, …)
- Avantatges:
  - Descriuen les **millors** (o més apropiades) **pràctiques** de treball: (dictades per l'experiència)
  - Evita repetir errors habituals
  - Assegura que tothom (desenvolupadors, testers, etc)
    - Segueix les mateixes pràctiques.
      - ➔ Afavoreix la continuitat de la feina quan algú abandona un projecte
      - ➔ Redueix esforç d'aprenentatge
    - Es comunica, coordina i enten millor

Standards

- Desavantatges:
  - **Costosos** de **desenvolupar** ➔ Cal recorrer a standards d'**organitzacions** mundials (IEEE, ISO, ANSI …) i **adaptar-los**.
  - La seva **aplicació** (i verificació) **porta** més **feina** (però també més qualitat)
  - **Poden estar desfasats** respecte tècniques més modernes (SW orientat a objecte, nous llenguatges, sistemes operatius…)
  - Han d'estar al servei del projecte, no al revés!
    - S'ha de seguir un standard, però s'ha de modificar o canviar si no és útil.
    - S'ha d'adaptar a les nostres necessitats.

Standards

Fonts dels estandards

- organitzacions nacionals i internacionals d'estandardització
  - **ISO** International Standards Organization `http://www.iso.org`
  - **BSI** British Standards Institute `http://www.bsi.org.uk`
  - **ASTM** American Society for Testing and Materials `http://www.asme.org`
- grups professionals o industrials
  - **IEEE** Institute of Electrical and Electronic Engineers `http://www.ieee.org`
  - **ANSI** American National Standards Institute `http://www.ansi.org`
  - **EIA** Electronic Industries Association `http://www.eia.org`
- agències governamentals
  - **DoD** departament de defensa dels Estats Units
  - **DEF** Britsh Defence Standards
  - **NIST** National Institute of Standards and Technology `http://www.nist.gov`
  - **ESA** European Space Agency `http://www.estec.esa.nl/ecss`
- empreses
  - Bellcore Communications Research `http://www.bellcore.com`
- centres de recerca
  - SEI Software Engineering Institute `http://www.sei.cmu.edu`

Guide to software engineering standards. S.Magee, L.Tripp, Artech House, 1997

Standards

- Estandards d'enginyeria del software de la **IEEE**

  Std. 829-1983 Standard for software test **documentation**
  Std. 1219-1992 Standard for software **maintenance**
  Std. 610.12-1990 Standard for software **engineering terminology**
  Std. 830-1993 Recommended practice for software **requirements specifications**
  Std. 1016-1987 Recommended practice for software **design descriptions**
  Std. 1008-1987 Standard for software **unit testing**
  Std. 1012-1987 Standard for software **verification and validation**
  Std. 1028-1988 Standard for software **reviews and audits**
  Std. 1042-1987 Guide to software **configuration management**
  Std. 982.1-1988 Standard dictionary of measures to produce reliable software
  Std. 1063-1987 Standard for software **user documentation**
  Std. 1062-1993 Recommended practice for software acquisition

....

Recopilats a *Software Engineering : IEEE Standards Collection, IEEE Press, 1994*

Índex

- Introducció
- *Standards*
- ***Quality Metrics***

Mètriques

- Definició (segons la IEEE):
  - Una **mesura quantitativa** del grau en què un element poseeix un cert **grau de Qualitat** respecte algun **aspecte** (p.ex. fiabilitat, reusabilitat, portabilitat …).
  - Una **funció**:
    - Input: Dades sobre el SW
    - Output: Valor numèric (que és la **mesura quantitativa** anterior)

Mètriques

- Objectius:
  - **Facilitar** el control, planificació i **actuació** del procés de gestió de desenvolupament del SW. Basat en:
    - **Desviacions** respecte al grau de ***performance*** previst.
    - **Desviacions** respecte el **timetable** i el **pressupost** previst.
  - **Identificar** moments per **millorar** el procés de desenvolupament (*preventive corrections*). Basat en:
    - **Acumulació de mètriques** sobre el grau de *performance* dels **equips** de desenvolupament, **unitats**, etc.

Mètriques

- Requeriments (una mètrica ha de ser)
  - **Generals**:
    - Rellevant
    - Vàlid
    - Fiable
    - Entenible
    - Mutualment exclusiva amb altres mètriques
  - **Operatius**:
    - Fàcil i simple
    - Inmune a intervencions esbiaixades per usuaris interessats

Mètriques

- Desavantatges
  - **No** existeix una mètrica global **per a tot** ➔ Es limiten a certs aspectes
  - **No** hi ha un **acord** sobre quines són les **millors**
  - **Moltes** prenen **significat** només quan es **comparen** amb els valors per a **altres projectes**
  - Es poden veure **afectades** per **diferents factors** (estil de codi, complexitat del codi, percentatge de codi reutilitzat, volum de documentatió, …)

- Com mesurem la Qualitat?
  - És difícil obtenir una única mesura de Qualitat del SW:

$$F_q = \sum c_i m_i$$ on m_i és una mètrica, i c_i un pes subjectiu

Mètriques

- Tipus de mètriques:
  - Fases del sistema SW
    - **Mètriques de procés**: Relacionades amb el procés de desenvolupament del SW
    - **Mètriques de manteniment**: Relacionades amb la facilitat de manteniment
    - **Mètriques del producte**: Relacionades amb les funcionalitats del SW
  - Mesures de Conceptes concrets
    - Qualitat del codi
    - *Timetable*
    - Efectivitat (d'eliminació d'errors)
    - Productivitat

Mètriques - Exemples

- Mesura de mida/volum del SW
  - **KLOC**: (mètrica clàssica) Nombre de **milers** de **linies** de **codi**.
  - **NFP** (Number of Function Points): Nombre de **recursos humans** necessaris per desenvolupar un SW o funcionalitat

Mètriques - Exemples

- Mesura comptatge d'errors

	Calculation of NCE	Calculation of WCE	
Error severity class	**Number of Errors**	**Relative Weight**	**Weighted Errors**
a	b	c	D = b x c
low severity	42	1	42
medium severity	17	3	51
high severity	11	9	99
Total	**70**	**---**	**192**
NCE	**70**	**---**	**---**
WCE		**---**	**192**

Number of code errors (**NCE**) vs. weighted number of code errors (**WCE**)

Mètriques

- Tipus de mètriques:
  - Fases del sistema SW
    - **Mètriques de procés**: Relacionades amb el procés de desenvolupament del SW
    - **Mètriques de manteniment**: Relacionades amb la facilitat de manteniment
    - **Mètriques del producte**: Relacionades amb les funcionalitats del SW
  - Mesures de Conceptes concrets
    - Qualitat del codi
    - *Timetable*
    - Efectivitat (d'eliminació d'errors)
    - Productivitat

Mètriques - Exemples

- Mètriques de procés
  - Error density metrics

Code	Name	Calculation formula
CED	**Code Error Density**	$CED = \frac{NCE}{KLOC}$
DED	**Development Error Density**	$DED = \frac{NDE}{KLOC}$
WCED	**Weighted Code Error Density**	$WCDE = \frac{WCE}{KLOC}$
WDED	**Weighted Development Error Density**	$WDED = \frac{WDE}{KLOC}$
WCEF	**Weighted Code Errors per Function Point**	$WCEF = \frac{WCE}{NFP}$
WDEF	**Weighted Development Errors per Function Point**	$WDEF = \frac{WDE}{NFP}$

NCE = The number of code errors detected by code inspections and testing.
NDE = total number of development (design and code) errors) detected in the development process.
WCE = weighted total code errors detected by code inspections and testing.
WDE = total weighted development (design and code) errors detected in development process.

Mètriques - Exemples

– Error severity metrics

Code	Name	Calculation formula
ASCE	**Average Severity of Code Errors**	$ASCE = \frac{WCE}{NCE}$
ASDE	**Average Severity of Development Errors**	$ASDE = \frac{WDE}{NDE}$

NCE = The number of code errors detected by code inspections and testing.
NDE = total number of development (design and code) errors) detected in the development process.
WCE = weighted total code errors detected by code inspections and testing.
WDE = total weighted development (design and code) errors detected in development process.

Mètriques - Exemples

– SW process timetable metrics

Code	Name	Calculation formula
TTO	Time Table Observance	$TTO = \frac{MSOT}{MS}$
ADMC	Average Delay of Milestone Completion	$ADMC = \frac{TCDAM}{MS}$

MSOT = Milestones completed on time.
MS = Total number of milestones.
TCDAM = Total Completion Delays (days, weeks, etc.) for all milestones.

Mètriques - Exemples

– Error removal effectivenes metrics

Code	Name	Calculation formula
DERE	Development Errors Removal Effectiveness	$DERE = \frac{NDE}{NDE + NYF}$
DWERE	Development Weighted Errors Removal Effectiveness	$DWERE = \frac{WDE}{WDE+WYF}$

NDE = total number of development (design and code) errors) detected in the development process.
WCE = weighted total code errors detected by code inspections and testing.
WDE = total weighted development (design and code) errors detected in development process.
NYF = number software failures detected during a year of maintenance service.
WYF = weighted number of software failures detected during a year of maintenance service.

Mètriques - Exemples

– Process productivity metrics

Code	Name	Calculation formula
DevP	Development Productivity	$DevP = \frac{DevH}{KLOC}$
FDevP	Function point Development Productivity	$FDevP = \frac{DevH}{NFP}$
CRe	Code Reuse	$Cre = \frac{ReKLOC}{KLOC}$
DocRe	Documentation Reuse	$DocRe = \frac{ReDoc}{NDoc}$

DevH = Total working hours invested in the development of the software system.
ReKLOC = Number of thousands of reused lines of code.
ReDoc = Number of reused pages of documentation.
NDoc = Number of pages of documentation.

[Galin2004]

Mètriques

- Tipus de mètriques:
  - Fases del sistema SW
    - **Mètriques de procés**: Relacionades amb el procés de desenvolupament del SW
    - **Mètriques de manteniment**: Relacionades amb la facilitat de manteniment
    - **Mètriques del producte**: Relacionades amb les funcionalitats del SW
  - Mesures de Conceptes concrets
    - Qualitat del codi
    - *Timetable*
    - Efectivitat (d'eliminació d'errors)
    - Productivitat

Mètriques - Exemples

- Mètriques de manteniment:
  - HD (Help Desk Service) calls density metrics

Code	Name	Calculation formula
HDD	HD calls density	$HDD = \frac{NHYC}{KLMC}$
WHDD	Weighted HD calls density	$WHDD = \frac{WHYC}{KLMC}$
WHDF	Weighted HD calls per function point	$WHDF = \frac{WHYC}{NMFP}$

NHYC = the number of HD calls during a year of service.
KLMC = Thousands of lines of maintained software code.
WHYC = weighted HD calls received during one year of service.
NMFP = number of function points to be maintained.

Mètriques - Exemples

– Severity of HD calls metrics and HD success metrics

Code	Name	Calculation formula
ASHC	Average severity of HD calls	$ASHC = \frac{WHYC}{NHYC}$

NHYC = the number of HD calls during a year of service.

WHYC = weighted HD calls received during one year of service.

Code	Name	Calculation formula
HDS	HD service success	$HDS = \frac{NHYOT}{NHYC}$

NHYNOT = Number of yearly HD calls completed on time during one year of service.
NHYC = the number of HD calls during a year of service.

Mètriques - Exemples

– HD productivity and effectiveness metrics

Code	Name	Calculation formula
HDP	HD Productivity	$HDP = \frac{HDYH}{KLNC}$
FHDP	Function Point HD Productivity	$FHDP = \frac{HDYH}{NMFP}$
HDE	HD effectiveness	$HDE = \frac{HDYH}{NHYC}$

HDYH = Total yearly working hours invested in HD servicing of the software system.
KLMC = Thousands of lines of maintained software code.
NMFP = number of function points to be maintained.
NHYC = the number of HD calls during a year of service.

Mètriques - Exemples

– Failures density metrics

Code	Name	Calculation formula
SSFD	Software System Failure Density	$SSFD = \frac{NYF}{KLMC}$
WSSFD	Weighted Software System Failure Density	$WFFFD = \frac{WYF}{KLMC}$
WSSFF	Weighted Software System Failures per Function point	$WSSFF = \frac{WYF}{NMFP}$

NYF = number of software failures detected during a year of maintenance service.
WYF = weighted number of yearly software failures detected during one year of maintenance service.
NMFP = number of function points designated for the maintained software.
KLMC = Thousands of lines of maintained software code.

Mètriques

- Tipus de mètriques:
  - Fases del sistema SW
    - **Mètriques de procés**: Relacionades amb el procés de desenvolupament del SW
    - **Mètriques de manteniment**: Relacionades amb la facilitat de manteniment
    - **Mètriques del producte**: Relacionades amb les funcionalitats del SW
  - Mesures de Conceptes concrets
    - Qualitat del codi
    - *Timetable*
    - Efectivitat (d'eliminació d'errors)
    - Productivitat

Mètriques - Exemples

- Mètriques de producte

Software metric	Description
Fan in/Fan-out	Fan-in is a measure of the number of functions or methods that call some other function or method (say X). Fan-out is the number of functions that are called by function X. A high value for fan-in means that X is tightly coupled to the rest of the design and changes to X will have extensive knock-on effects. A high value for fan-out suggests that the overall complexity of X may be high because of the complexity of the control logic needed to coordinate the called components.
Length of code	This is a measure of the size of a program. Generally, the larger the size of the code of a component, the more complex and error-prone that component is likely to be. Length of code has been shown to be one of the most reliable metrics for predicting error-proneness in components.
Cyclomatic complexity	This is a measure of the control complexity of a program. This control complexity may be related to program understandability. I discuss how to compute cyclomatic complexity in Chapter 22.
Length of identifiers	This is a measure of the average length of distinct identifiers in a program. The longer the identifiers, the more likely they are to be meaningful and hence the more understandable the program.
Depth of conditional nesting	This is a measure of the depth of nesting of if-statements in a program. Deeply nested if statements are hard to understand and are potentially error-prone.
Fog index	This is a measure of the average length of words and sentences in documents. The higher the value for the Fog index, the more difficult the document is to understand.

Mètriques - Exemples

- Mètriques Orientades a Objecte

Object-oriented metric	Description
Depth of inheritance tree	This represents the number of discrete levels in the inheritance tree where sub-classes inherit attributes and operations (methods) from super-classes. The deeper the inheritance tree, the more complex the design. Many different object classes may have to be understood to understand the object classes at the leaves of the tree.
Method fan-in/fan-out	This is directly related to fan-in and fan-out as described above and means essentially the same thing. However, it may be appropriate to make a distinction between calls from other methods within the object and calls from external methods.
Weighted methods per class	This is the number of methods that are included in a class weighted by the complexity of each method. Therefore, a simple method may have a complexity of 1 and a large and complex method a much higher value. The larger the value for this metric, the more complex the object class. Complex objects are more likely to be more difficult to understand. They may not be logically cohesive so cannot be reused effectively as super-classes in an inheritance tree.
Number of overriding operations	This is the number of operations in a super-class that are over-ridden in a sub-class. A high value for this metric indicates that the super-class used may not be an appropriate parent for the sub-class.

Mètriques - Exemples

- Mètriques d'anàlisi

especifitat o manca d'ambigüetat

$$Q_1 = n_{ui} / n_r$$

n_{ui} nombre de requeriments interpretats de la mateixa manera per tots els revisors
n_r nombre total de requeriments

- Mètriques d'anàlisi i disseny

complexitat del disseny de l'arquitectura de mòduls

complexitat	d'estructura	$S(i) = f_{out}^2(i)$
	de dades	$D(i) = v(i) / (f_{out}(i) +1)$
	del disseny	$C = \Sigma_i S(i) + D(i)$

f_{out} : expansió, nombre de mòduls invocats directament pel modul *i*
$v(i)$: nombre de variables d'entrada i sortida del modul *i*

complexitat MHK

$$MHK = l(i) [f_{in}(i) + f_{out}(i)]$$

f_{in} : concentració, nombre de mòduls que invoquen directament al modul *i*
$l(i)$: nombre de sentències amb que s'ha programat el mòdul *i*

Mètriques - Exemples

mètriques CK (Chidamber i Kemener '94)

WMC *weighted methods per class*

- suma de les complexitats de tots els mètodes d'una classe, calculada amb V(G) per exemple.
- indicador de la quantitat d'esforç d'implementació i prova

DIT *depth of inheritance tree*

- alçada d'un arbre d'herència (d'una jerarquia de classes)
- DIT alt ⇒ probablement les classes de nivells inferiors heredin molts mètodes
  ⇒ disseny complexe (erroni?), però *potser* es reutilitzen molts mètodes

NOC *number of children*

- nombre de classes directament descendents d'una altra
- NOC alt ⇒ s'incrementa la reutilització de mètodes però es redueix l'abstracció de la classe genèrica → error de disseny ?

CBO *coupling between object classes*

- Nombre de classes depenents entre si degut a compartició de variables, atributs.
- CBO "excessiu" va en contra de la modularitat, reutilització (les classes també haurien de ser el més independents possible) i dificulta la prova.

RFC *response for a class*

- nombre de mètodes que poden ser executats potencialment en resposta a un missatge rebut per un objecte de la classe
- RPC alt ⇒ l'esforç de prova (p.e. d'integració) i complexitat del disseny creixen

LCOM *lack of cohesion between methods*

- nombre de mètodes que accedeixen com a mínim a un atribut també accedit per una altra classe
- LCOM alt ⇒ mètodes acoplats mitjançant els atributs ⇒ potser la classe es podria dissenyar millor dividint-la en classes diferents

Mètriques - Exemples

mètriques de Lorenz i Kidd, '94

CS *class size*

- nombre total d'atributs i mètodes d'una classe (incloent els heredats)
- CS gran pot indicar que la classe té massa responsabilitats i s'ha de dividir

NIO *number of invalidated operations for a subclass*

- nombre de mètodes redefinits per una classe derivada (versió especialitzada)
- NIO alt va contra l'abstracció de la classe genèrica

NAO *number of added operations for a subclass*

- NAO hauria de reduir-se a mesura que agumenta la profunditat de la subclasse

SI *specialization index*

- grau d'especialització d'una classe especialitzada (subclasse)

$$SI = (NIO \cdot level) / M_T$$

M_T nombre total (propis + heredats) de mètodes de la classe
level profunditat en la jerarquia

- SI alt $\Rightarrow$ més probable que la jerarquia no sigui correcte, no s'ajusta a l'abstracció de la classe genèrica

Mètriques - Exemples

- Mètriques orientades a la dificultat del test

  - **PPP** *percentage of public and protected*
    - % d'atributs publics d'una classe respecte privats i protegits, comptant els heredats
    - PPP alt incrementa la probabilitat d'efectes colaterals imprevistos entre classes $\Rightarrow$ dissenyar més proves

  - **APD** *acces to public data members*
    - nombre de mètodes que accedeixen directament (sense cridar mètodes d'accés) a atributs d'alguna altra classe, anant en contra doncs de l'encapsulament de les classes
    - APD alt $\Rightarrow$ alta probabilitat efectes colaterals entre classes

  - **NRC** *number of root classes*
    - nombre de jerarquies d'herència
    - NCR alt $\Rightarrow$ creix l'esforç de prova necessari

Mètriques

- Procés de mesura
  - Ha de ser **automàtic**. Existeixen eines, pero són complexes o cares.
  - Les dades es **recullen durant** el projecte, **no al final**
  - **No** s'han de recollir **més** dades **de les necessàries**