Master Sistemi Embedded per Internet of Things

CORSO «ORGANIZZAZIONE, PROCESSI E CERTIFICAZIONI»

SW UNIT VERIFICATION MANAGEMENT

Maria Teresa Palomba – A-Key S.r.l.

t.palomba@a-key.it

a-key

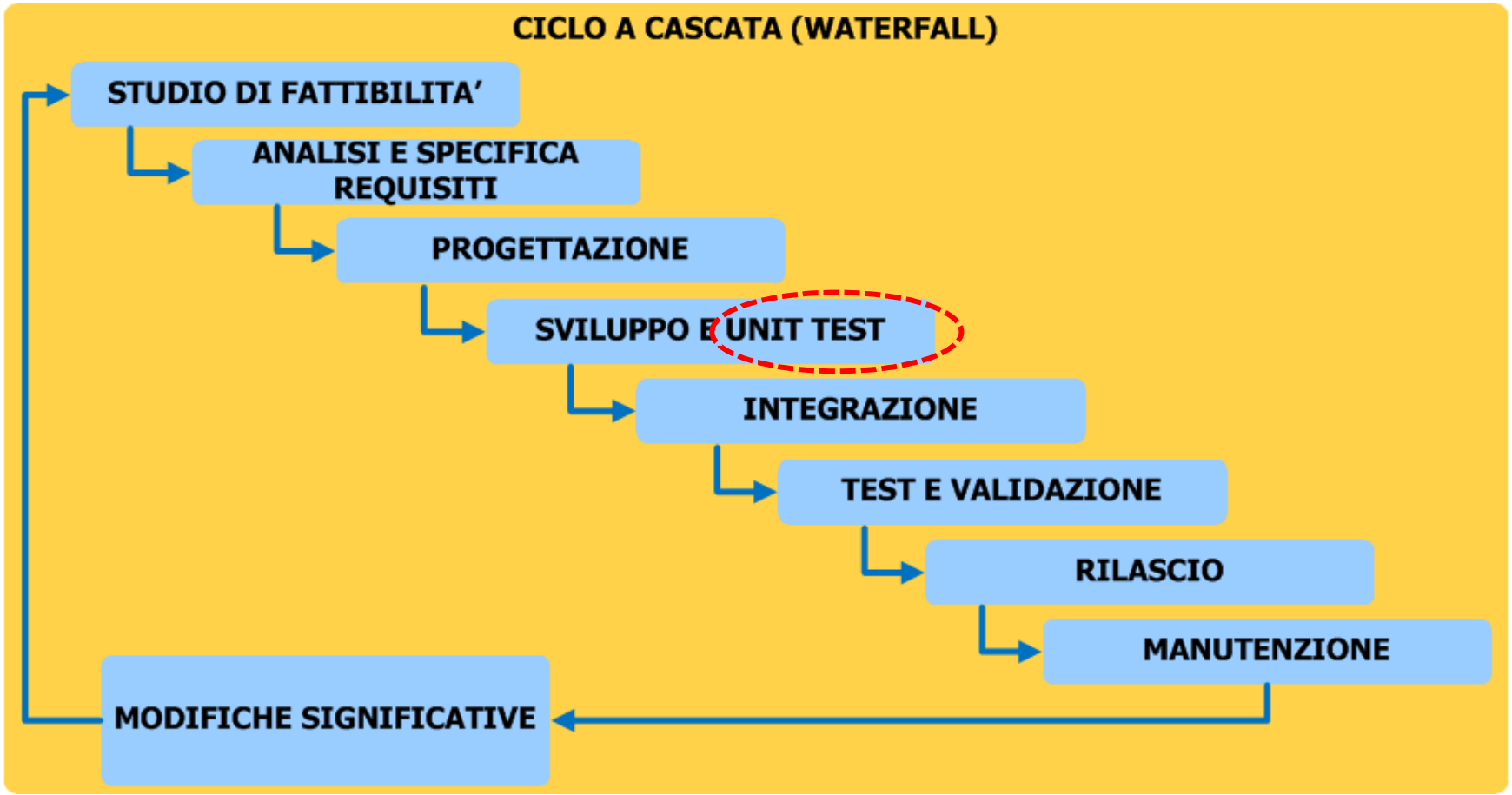
Riferimenti:

A. http://www.computer.org/web/swebok - Software Engineering Body of Knowledge (SWEBOK), version 3.0

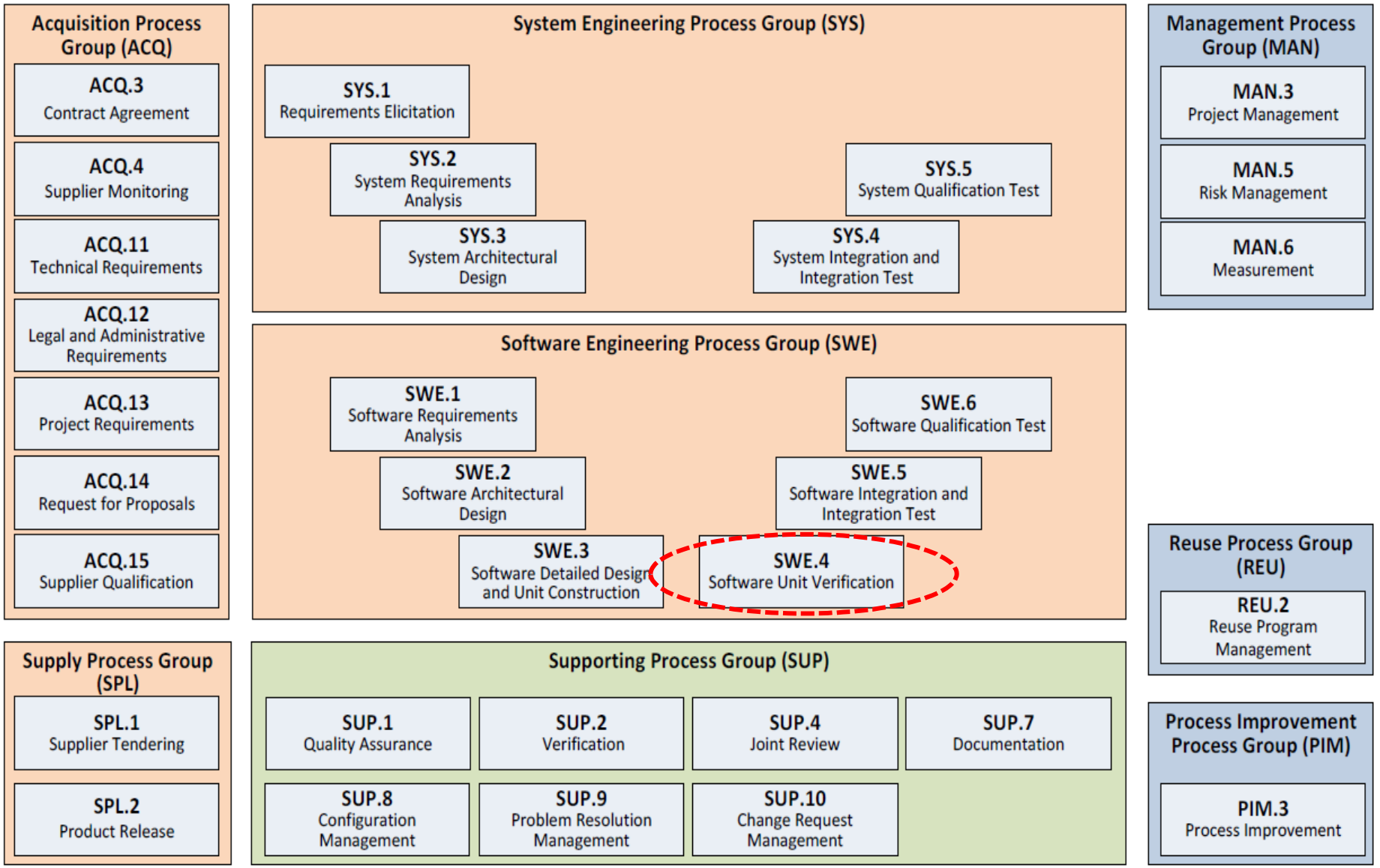
B. http://www.sei.cmu.edu/library/abstracts/reports/10tr033.cfm modello CMMI-DEV v. 1.3

C. http://www.automotivespice.com/fileadmin/software-download/Automotive_SPICE_PAM_30.pdf

Ciclo di vita del Software



Automotive SPICE process reference model - Overview



SOFTWARE UNIT VERIFICATION

The purpose of the Software Unit Verification Process is

- to **verify software units** →
  - to **provide evidence** for **compliance** of the software units →
    - **with the software detailed design** and
    - **with the non-functional software requirements.**

Unit Testing → fase in cui viene testata la **software unit**, **la più piccola parte testabile della applicazione SW**

1) **Prende la più piccola parte di SW testabile nella applicazione**
2) **La isola dal resto del codice**
3) **Verifica se il suo comportamento è esattamente quello atteso**

Lo Unit Test è effettuato dal developer ed è sua responsabilità

- Ogni SW unit deve essere testata individualmente in modo da individuare e correggere immediatamente le anomalie di ciascun modulo

- Solo quando le SW Unit funzionano correttamente e senza anomalie, il componente può passare alla fase successiva (integrazione)

Unit testing → verifica che le funzioni lavorino come atteso

- **Per ogni funzione deve determinare**:
  - se restituisce output appropriati per ogni dato set di valori di input
  - se è in grado di gestire adeguatamente failures se vengono forniti input invalidi

Il set di valori di input deve garantire un Code Coverage elevato

→ **percentuale del codice sorgente che viene effettivamente testato dagli unit test**

Unit Test Scenarios

- Normal Scenario
- Unexpected scenario
  - Bad input value
  - Boundary condition
  - Boundary neighbourhood condition

Unit Test Best Practices

- Assicurarsi che ciascun Unit Test case sia **indipendente** dagli altri
- Testare **solo una SW Unit alla volta**
- **Denominare** gli Unit Test chiaramente e in modo consistente
- **Prima di modificare un modulo** (interfaccia o implementazione), **assicurarsi che il modulo abbia Unit Test case e che superi i Test**
- Assicurarsi che **tutti i bug identificati** durante lo Unit Testing **siano risolti prima di passare alla fase successiva**

Unit Test Benefits

- Trovare i problemi presto nel ciclo di vita (development phase)
- Facilitare le modifiche
- Semplificare il Test di Integrazione
- Mantenere una documentazione d'uso aggiornata

Unit Testing nell'Embedded Systems Software

- **Domanda crescente di qualità** nella delivery di Embedded SW
- **Standard internazionali per la Safety** (es. IEC-61508, ISO-26262) richiedono l'effettuazione del module testing per raggiungere i functional safety level
- **Il costo di un richiamo del prodotto è elevato** (costi e perdita di credibilità) → mitigato da costi e tempi aggiuntivi per testing che assicuri un SW meno difettoso

Unit Testing on Embedded Systems Software

- **Il SW per sistemi Embedded è sviluppato su piattaforme diverse da quella su cui il SW dovrà essere eseguito nell'ambiente operativo finale:**
  - **Difficoltà** nella esecuzione corretta degli Unit Test
  - **Non disponibilità del target HW** finale fino a fasi avanzate del progetto

Unit Testing on Embedded Systems Software

Anticipare il Test su Host o simulatore:

- **Utilizzo dello stubbing** per **eseguire una buona parte del test in ambiente host o su simulatore**
- **Inizio della verifica del codice appena viene completato**, anche se il target HW non è ancora disponibile

Unit Testing on Embedded Systems Software

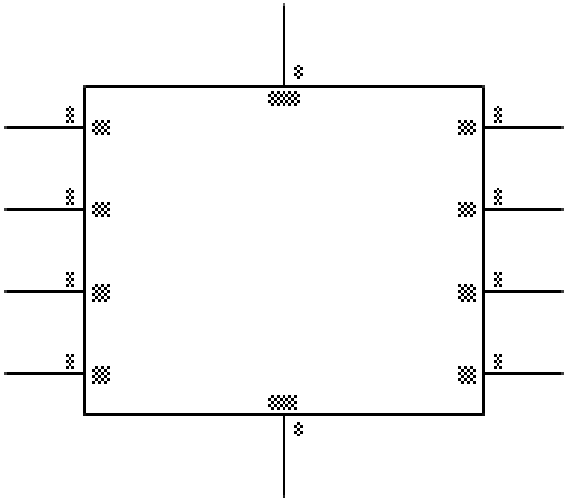
Test su target o simulatore:

→ gli Unit Test cases devono essere installati sulla target board o sul simulatore

Codice da testare

Unit Test case and data

Risultati dello Unit Test



Automazione dello Unit Testing

L'implementazione manuale dello Unit Testing è possibile ma non risolutiva

- **Es. Quanti test effettuare al minimo per verificare l'API di una semplice funzione**:

```
sint8 SimpleFunc(sint32 a, sint16 b, sint8 c)
```

  - Per ciascun parametro definiamo almeno 5 test points (max, min, -1, 0, 1)
  - Si ottengono **125 combinazioni possibili**

Automazione dello Unit Testing

- **Il numero di test necessari per ottenere il coverage della funzione cresce notevolemente** quando si voglia verificare non solo l'API ma anche la sua implementazione

- La determinazione "manuale" dei test rischia di non garantire una copertura adeguata di tutti i casi possibili

Utilizzo di tool automatici per:

- **Supporto nella definizione e preparazione dei test cases**
- **Supporto nella copertura dei casi possibili**

Si tratta sempre di SUPPORTO – l'attività è del developer

Master Sistemi Embedded per Internet of Things

CORSO «ORGANIZZAZIONE, PROCESSI E CERTIFICAZIONI»

UNIT TESTING METHODOLOGIES

a-key

Test suite

- Una collezione di inputs e dei corrispondenti outputs

Unit Testing Approach

Definisce:

- le **tipologie di Unit tests** che saranno effettuate nel progetto
- Le **regole per condurre il test di regressione**
- Le linee guida per la definizione e l'esecuzione degli Unit Test

Tipologie di Unit Test

Il processo di Unit Testing dovrebbe includere una combinazione di due tipologie di test:

- "**Test Strutturale**"
- "**Test Funzionale**"

Test Strutturale

Il Test Strutturale (anche White box Testing o Internal Testing) è **basato sulla conoscenza della struttura del codice**.

Ha per obiettivo **assicurare che tutte le parti del codice siano state eseguite durante il processo di test.**

- Il tester **costruisce una test suite** che **dimostra** che **tutti i rami di decisione e di loop del programma** (es. if, while, switch, etc.) **vengono eseguiti** durante il test

Test Strutturale

- La Test Suite deve includere **sufficienti insiemi di dati di input** da garantire che almeno:
  - **Tutti i metodi** siano stati chiamati
  - Entrambi i **rami** "**true**" **e** "**false**" negli statements "**if**" siano stati eseguiti
  - **Ogni loop** sia stato eseguito **zero, una, e più volte**
  - **Tutti i rami di ogni statement "switch"** siano stati eseguiti

Test Strutturale

Il Test Strutturale **non può dimostrare che il programma funziona in tutti i casi**:

- È un modo efficiente, efficace, sistematico di scoprire errori nel programma
- È efficace nello scoprire errori in in caso di programmi dalla logica complessa
- È efficace nello scoprire variabili inizializzate con valori errati

Test Strutturale – obiettivi e metriche di coverage

- **Statement/Code Coverage**: ogni linea di codice deve essere eseguita da almeno un test
- **Branch Coverage**: ogni branch del codice è eseguito da almeno un test. 100% Branch Coverage assicura 100% Statement Coverage
- **Condition Coverage**: ogni condizione in ogni espressione predicativa è valutata in tutte le condizioni modi possibili
- **Condition-Decision Coverage**: ogni operando booleano è esercitato in modo da poter indipendentemente condizionare l'output di una decisione

Test Funzionale

Il Test Funzionale a livello di Unit (Black box Testing, External Testing) è **focalizzato sul comportamento atteso per il programma**

Ha per obiettivo **verificare che ogni unità sia implementata in accordo con le specifiche**

Il tester deve conoscere il problema che la unit deve risolvere

Il tester **costruisce un set di dati di test (input e corrispondenti output attesi)** che **include dati di input sia "tipici", che "estremi"**

La Test Suite **deve includere input eccezionali o erronei rispetto al programma**.

Regression testing

Il Test di Regressione **permette una validazione consistente e ripetibile di ogni nuova release del prodotto.**

Assicura

- **che i difetti individuati siano stati corretti per ogni release**
- **che non siano stati introdotti nuovi problemi durante la manutenzione**

Il Test di Regressione è spesso supportato da **una test Suite automatizzata** per ridurre tempo e risorse necessarie

Master Sistemi Embedded per Internet of Things

CORSO «ORGANIZZAZIONE, PROCESSI E CERTIFICAZIONI»

CMMI- DEV
AUTOMOTIVE SPICE

a-key

Process areas

TECHNICAL SOLUTION - TS

"The purpose of Technical Solution (TS) is to select, design, and implement solutions to requirements. Solutions, designs, and implementations encompass products, product components, and product related lifecycle processes either singly or in combination as appropriate."

TECHNICAL SOLUTION - TS

Specific Goal and Practice Summary

- **SG 1 Select Product Component Solutions**
  - SP 1.1 Develop Alternative Solutions and Selection Criteria
  - SP 1.2 Select Product Component Solutions
- **SG 2 Develop the Design**
  - SP 2.1 Design the Product or Product Component
  - SP 2.2 Establish a Technical Data Package
  - SP 2.3 Design Interfaces Using Criteria
  - SP 2.4 Perform Make, Buy, or Reuse Analyses
- **SG 3 Implement the Product Design**
  - SP 3.1 Implement the Design
  - SP 3.2 Develop Product Support Documentation

Process purpose

"The purpose of the Software Unit Verification Process is to **verify software units** to provide **evidence** for **compliance** of the software units with the **software detailed design** and with the **non-functional software requirements**"

Process outcomes

1) a software unit verification strategy including regression strategy is developed to verify the software units;

2) criteria for software unit verification are developed according to the software unit verification strategy that are **suitable to provide evidence** for compliance of the software units with the software detailed design and with the non-functional software requirements;

3) software units are verified according to the software unit verification strategy and the defined criteria for software unit verification and the **results are recorded**;

4) consistency and bidirectional traceability are established between **software units, criteria for verification and verification results**; and

5) results of the unit verification are **summarized and communicated** to all affected parties.

Base Practices

- SWE.4.BP1: Develop software unit verification strategy including regression strategy.
- SWE.4.BP2: Develop criteria for unit verification
- SWE.4.BP3: Perform static verification of software units.
- SWE.4.BP4: Test software units.
- SWE.4.BP5: Establish bidirectional traceability.
- SWE.4.BP6: Ensure consistency.
- SWE.4.BP7: Summarize and communicate results.