Master Sistemi Embedded per Internet of Things

CORSO «ORGANIZZAZIONE, PROCESSI E CERTIFICAZIONI»

SW INTEGRATION AND TEST MANAGEMENT

Maria Teresa Palomba – A-Key S.r.l.

t.palomba@a-key.it

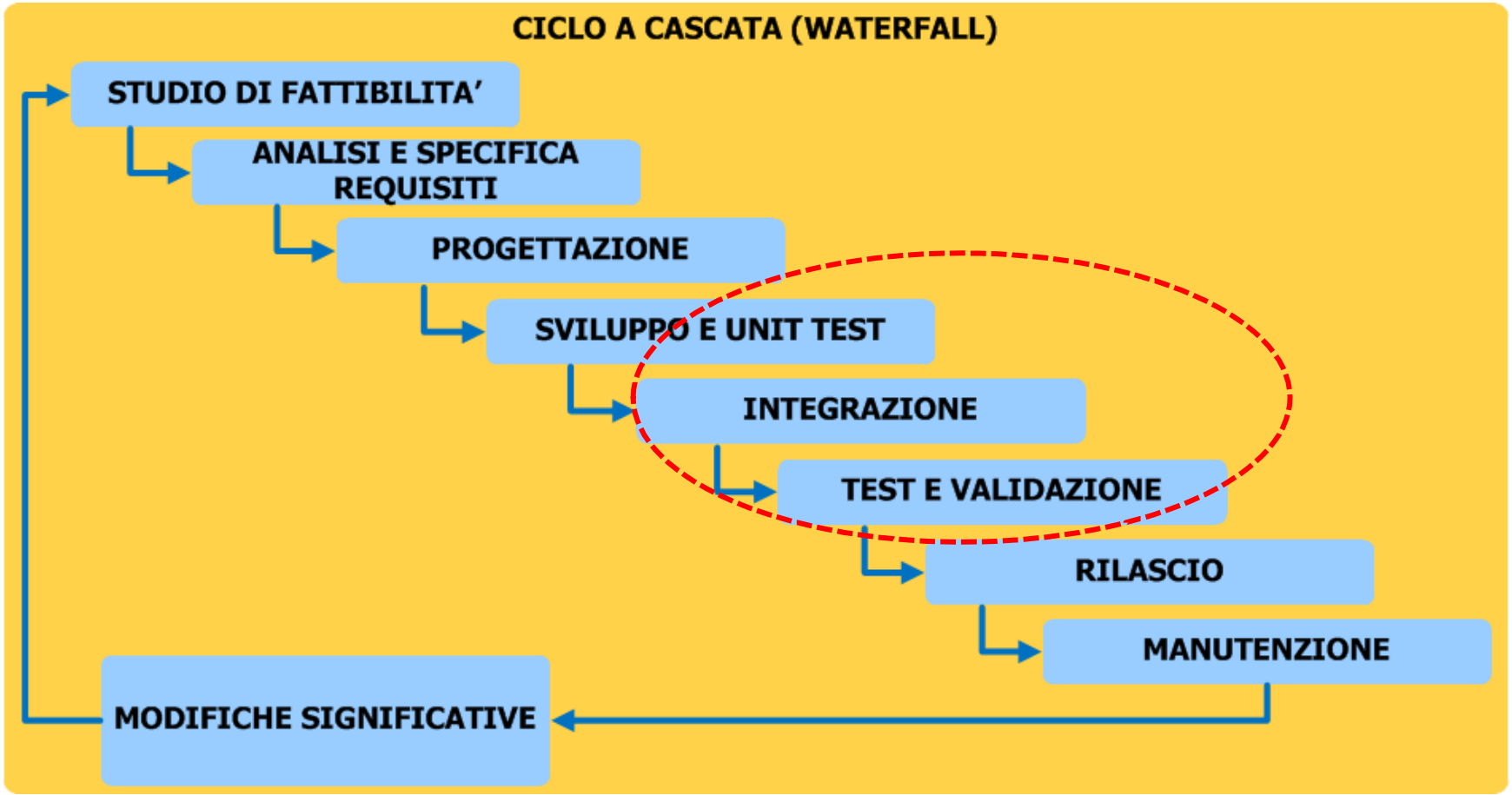
a-key

Riferimenti:

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

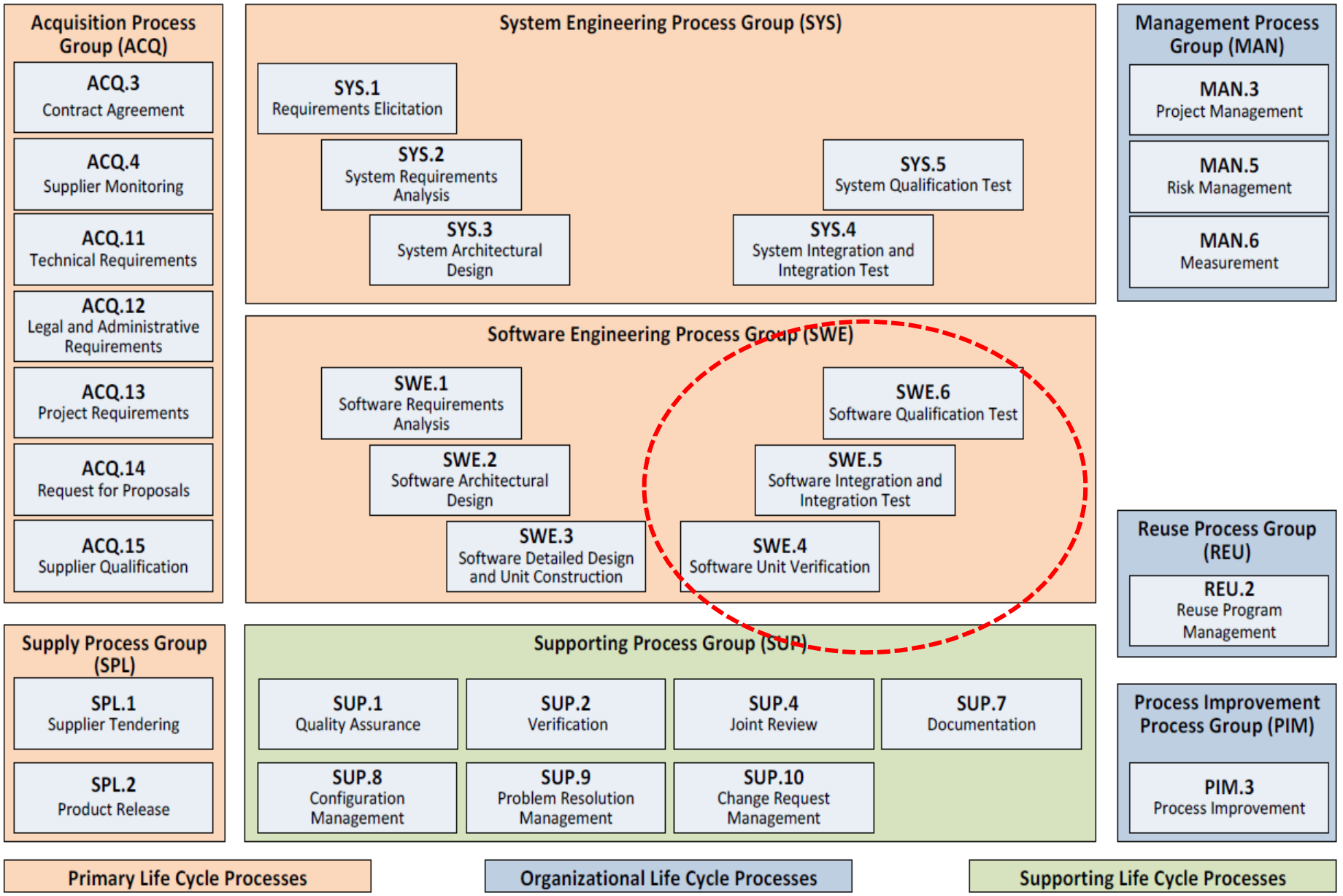
B. http://www.sei.cmu.edu/library/abstracts/reports/10tr033.cfm modello CMMI-DEV v. 1.3

C. http://www.automotivespice.com/fileadmin/software-download/Automotive_SPICE_PAM_30.pdf - Automotive SPICE PAM PRM v.3.0


CICLO A CASCATA (WATERFALL)
STUDIO DI FATTIBILITA'
ANALISI E SPECIFICA REQUISITI
PROGETTAZIONE
SVILUPPO E UNIT TEST
INTEGRAZIONE
TEST E VALIDAZIONE
RILASCIO
MANUTENZIONE
MODIFICHE SIGNIFICATIVE

a-key

Automotive SPICE process reference model - Overview



SWEBOK

SOFTWARE TESTING

Software testing consists of the **dynamic verification** that a **program** provides **expected behaviors** on a **finite set of test cases**, suitably **selected** from the usually infinite execution domain.

- **Dynamic** → Il test implica sempre **l'esecuzione del programma** sugli input selezionati

- **Finite** → il test è condotto su un **sottoinsieme di tutti i possibili test**, determinato sulla base di **criteri di prioritizzazione e di rischio**. Compromesso tra:
  - Limite di risorse e tempo
  - Set esaustivo di test non limitato

- **Selected** → **l'insieme dei test da eseguire viene selezionato** sulla base delle condizioni date dal progetto, tramite tecniche di **analisi del rischio e valutazioni ingegneristiche**

- **Expected** → **deve essere definito il comportamento accettabile** a fronte di un test, deve sempre **essere possibile decidere se l'output di un test è accettabile o no**

Faults vs. Failures

- **Fault** = anomalia del sistema causa di un malfunzionamento
- **Failure** = effetto indesiderato osservato nel servizio fornito dal sistema
- **Defect** = termine generico, può indicare sia fault che failure

I **faults** possono **non manifestarsi**, non determinare mai una **failure**

Il test rivela failures, ma è fondamentale lavorare per eliminare i faults

Non sempre la causa di un failure può essere identificata.

- **Criterio di selezione dei test** → come individuare un set di casi di test sufficiente per lo scopo specificato.

- **Efficacia del test** → l'efficacia è determinata in relazione all'obiettivo del test.

- **Defect Discovery** → Il test riuscito è quello che porta il sistema ad un failure. Diverso obiettivo rispetto al testare che il sistema esegua i comportamenti attesi, in cui il test riuscito riporta assenza di failure

- **Limitazioni del Testing** →
  - **Il Test non può mai dimostrare l'assenza di fault** (Dijkstra aphorism "program testing can be used to show the presence of bugs, but never to show their absence")
  - **Il Test completo (esaustivo) non è fattibile nel software reale**

- **Path non eseguibili** → flussi di controllo non esercitabili da alcun dato di input

- **Testabilità** →
  - Facilità con cui un dato criterio di coverage del test può essere soddisfatto
  - Probabilità che un set di test case espongano un failure se il software ha dei faults

Test target e Test objective sono la base per definire:

- **Criteri di adeguatezza** del test → Quanto test è sufficiente per ottenere l'obiettivo

- **Criteri di selezione** del test → Quali test devono essere selezionati per ottenere l'obiettivo

Software Testing Target

Classificazione sulla base dell' oggetto del test (elemento o aggregato di elementi):

- Unit
- Integration
- System

Unit Test

Processo di verifica del funzionamento in **isolamento** di **unità**

software che siano **testabili separatamente**

Integration Test

Processo di verifica delle interazioni tra componenti software

1) **Selezione di componenti SW che hanno superato lo Unit test**
2) **Integrazione dei componenti SW** per costituire **aggregati** più grandi
3) **Test** del loro funzionamento in interazione
4) Individuazione di eventuali **anomalie nel loro funzionamento integrato**
5) **Registrazione, reporting e tracciatura delle anomalie** fino alla risoluzione

System Test

Processo di verifica del comportamento di un intero sistema.

- Parte da una base di verifica già effettuata da Unit ed Integration Testing (eliminazione di molte anomalie)

Verifica:

- **Requisiti funzionali:** verso le interfacce **esterne** al sistema (altre applicazioni, Hardware, ambiente operativo)
- **Requisiti non funzionali di sistema:** (security, velocità, accuratezza, affidabilità)

Software Testing Objectives

Il test viene condotto in prospettiva di specifici obiettivi.

Es.

- Specifiche funzionali (conformance testing, correctness testing, functional testing)
- Specifiche Non-functional (performance, reliability, usability,)
- Identificazione di vulnerabilità di security
- Accettazione del SW

in generale **gli obiettivi del test variano con il target del test**

Acceptance / Qualification Testing

Determina se un sistema soddisfa i suoi criteri di accettazione

= verifica i comportamenti desiderati rispetto ai requisiti del cliente

Il cliente specifica o effettua direttamente le attività da effettuare

per verificare che i suoi requisiti siano raggiunti

Installation Testing

Test effettuato previa installazione del sistema nell'ambiente target.

E' un tipo di system testing effettuato nell'ambiente operativo e nelle configurazione hardware prevista, e sotto gli altri vincoli previsti.

Alpha and Beta Testing

Il software viene rilasciato ad un insieme di utenti potenziali per un uso in prova prima del rilascio ufficiale:

- **alpha testing**: insieme ristretto di utenti
- **beta testing**: insieme più esteso di utenti rappresentativi

Gli utenti collaborano alla individuazione dei problemi

Reliability

Ulteriori **test di affidabilità si possono derivare tramite generazione casuale di casi di test**, in funzione del profilo operativo del sistema.

(operational testing).

Regression Testing

Riesecuzione selettiva dei test su un software per verificare che modifiche apportate non abbiamo causato effetti indesiderati e che il software è conforme ai requisiti desiderati.

Dimostra se il SW supera i test che ha superato in una precedente sessione di test.

Selezione : compromesso tra estensione dei test e risorse limitate

Può essere effettuato a tutti i livelli e per tutte le tipologie di test

Performance Testing

Verifica che il SW raggiunga i requisiti di performance specificati

Stress Testing

Esercita il SW al massimo carico di progettazione e oltre, per **determinarne i limiti di comportamento** e per **verificare i meccanismo di difesa nei sistemi critici**

Security Testing

Verifica che il SW sia protetto da attacchi esterni.

- Verifica la confidenzialità, l'integrità e la disponibilità del sistema e dei suoi dati.
- Include test negativi (uso errato o abuso).

Back-to-Back Testing

Due o più varianti del programma sono eseguite con gli stessi input ed i conseguenti output ed errori sono confrontati e analizzati

Recovery Testing

Verifica la capacità del SW di ripartire dopo un crash o un disastro

Interface Testing

Verifica se i componenti si interfacciano correttamente

Configuration Testing

Verifica il SW sotto diverse configurazioni

Usability and Human Computer Interaction Testing

Valuta la facilità con cui un utente finale è in grado di apprendere e utilizzare il SW.

Verifica le funzioni utente, la documentazione a supporto dell'utente, la capacità del sistema di recuperare da errori dell'utente

Master Sistemi Embedded per Internet of Things

CORSO «ORGANIZZAZIONE, PROCESSI E CERTIFICAZIONI»

TECNICHE DI TEST

a-key

Techniques Based on the Software Engineer's Intuition and Experience

- **Ad Hoc** → sulla base dell'esperienza del SW engineer
- **Exploratory Testing** → apprendimento, test design e test execution sono simultanei. I test vengono disegnati, eseguiti e modificati dinamicamente (adattiva, sfrutta la esperienza)

Input Domain-Based Techniques

- **Equivalence Partitioning** → suddivisione del dominio di input in un set di sottoinsiemi basati su criteri specificati. Un set rappresentativo di test è selezionato da ciascuna classe di equivalenza

- **Pairwise Testing (o t-wise)** → I test sono derivati combinando valori interessanti di un set di variabili di input invece che considerare tutte le possibili combinazioni

Input Domain-Based Techniques

- **Boundary-Value Analysis** → test selezionati sui limiti o vicino ai limiti del dominio delle variabili di input.
  - Razionale: molti faults tendono a concentrarsi vicino ai valori estremi del dominio

- **Random Testing** → test sono generati in modo casuale (non a partire dall'operational profile)
  - Il dominio di input deve essere noto
  - Relativamente semplice per l'automazione dei test

Code-Based Techniques

- **Control Flow-Based Criteria** → obiettivo di copertura degli statements, blocchi di statementes, o combinazioni di statements. Adeguatezza misurata in percentuale.

- **Data Flow-Based Criteria** → obiettivo di copertura dei segmenti di path di controllo di flusso a partire dalla definizione di una variabile fino al completamento del suo utilizzo. Utilizza l'analisi del grafo di flusso di controllo del programma per individuare quando ciascuna variabile viene definita, usata e dismessa

❑ Fault-Based Techniques

Test cases disegnati per **rivelare la presenza di categorie di anomalie probabili o predefinite**.

Un **fault model** viene spesso definito per classificare le tipologie di faults.

- **Error Guessing** → test disegnati per **anticipare i faults più plausibili in un dato programma**. Utilizza l'esperienza e lo **storico dei faults scoperti in precedenti progetti**

Usage-Based Techniques

- **Operational Profile** → Test di reliability (operational testing). L'ambiente di test riproduce l'ambiente operativo del SW. Obiettivo: inferire la futura reliability nell'ambiente effettivo di utilizzo sulla base dei risultati dei test.
  - Assegna probabilità agli inputs in base alla loro frequenza di occorrenza nell'ambiente finale

- **User Observation Heuristics** → obiettivo: individuare problemi nella progettazione della Interfaccia Utente. Utilizza metodi euristici – usability inspection methods → osservazione sistematica dell'utilizzo del sistema per determinarne il livello di usabilità da parte delle persone. (walkthroughs cognitivi, analisi dei reclami, osservazioni sul campo, thinking aloud, questionari, interviste)

TIPI DI TEST

Types of tests

L'approccio del software testing dovrebbe includere una combinazione delle tipologie di test fondamentali:

- **black box testing**
- **grey box testing**
- **white box testing**

→ ***opacità*** del codice dal punto di vista del tester

Black box Testing (anche behavioral testing)

- **Ignora la struttura interna del sistema o del componente**
- **Si focalizza solo sugli output generati in risposta ad input e condizioni di esecuzione selezionati**
- Il tester non dovrebbe avere visibilità del codice sorgente
- Il codice è visto come una "**scatola nera**" da parte del tester.

Grey box Testing

- **Testa l'applicazione con una conoscenza limitata del funzionamento interno del sistema**

- **Il tester ha accesso alla documentazione di design ed al database** → utilizzati per predisporre i test in fase di pianificazione e per verificare i risultati del test

White box Testing (anche structural testing o glass box testing)

- Considera la struttura interna del codice sorgente

- Il white box tester definisce i test per sollecitare il codice con parametri selezionati
- Il white box tester è solitamente il developer

Utilizzato per verificare le specifiche funzionali e tecniche delle unit SW e del SW integrato

Black box testing:

- È la tecnica più importante per il Software testing

- Verifica che il SW integrato sia conforme ai Software Requirements e alla Software Requirements Specification

- Deve considerare le seguenti categorie di errore potenziale:
  - Incorrect or missing functions
  - Interface errors
  - Errors in data structures or external database access
  - Behavior or performance errors
  - Initialization and termination errors

Grey and White box testing

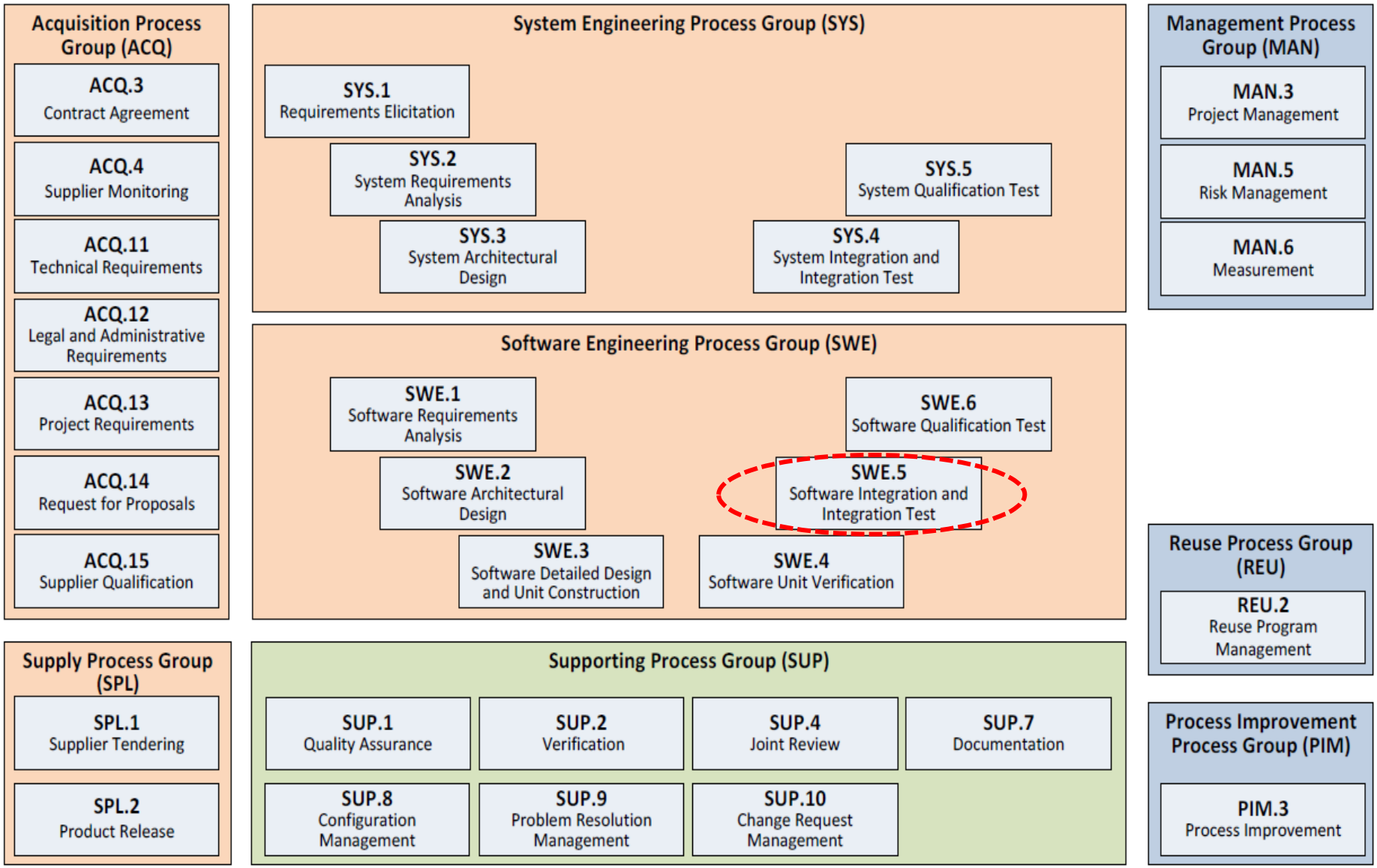
- Da utilizzare al livello di unit and integration test per verificare le specifiche funzionali e tecniche delle unit SW e del SW integrato
- Da utilizzare al livello di software test per verificare l'esecuzione dei "critical paths" tra sottosistemi

INTEGRATION

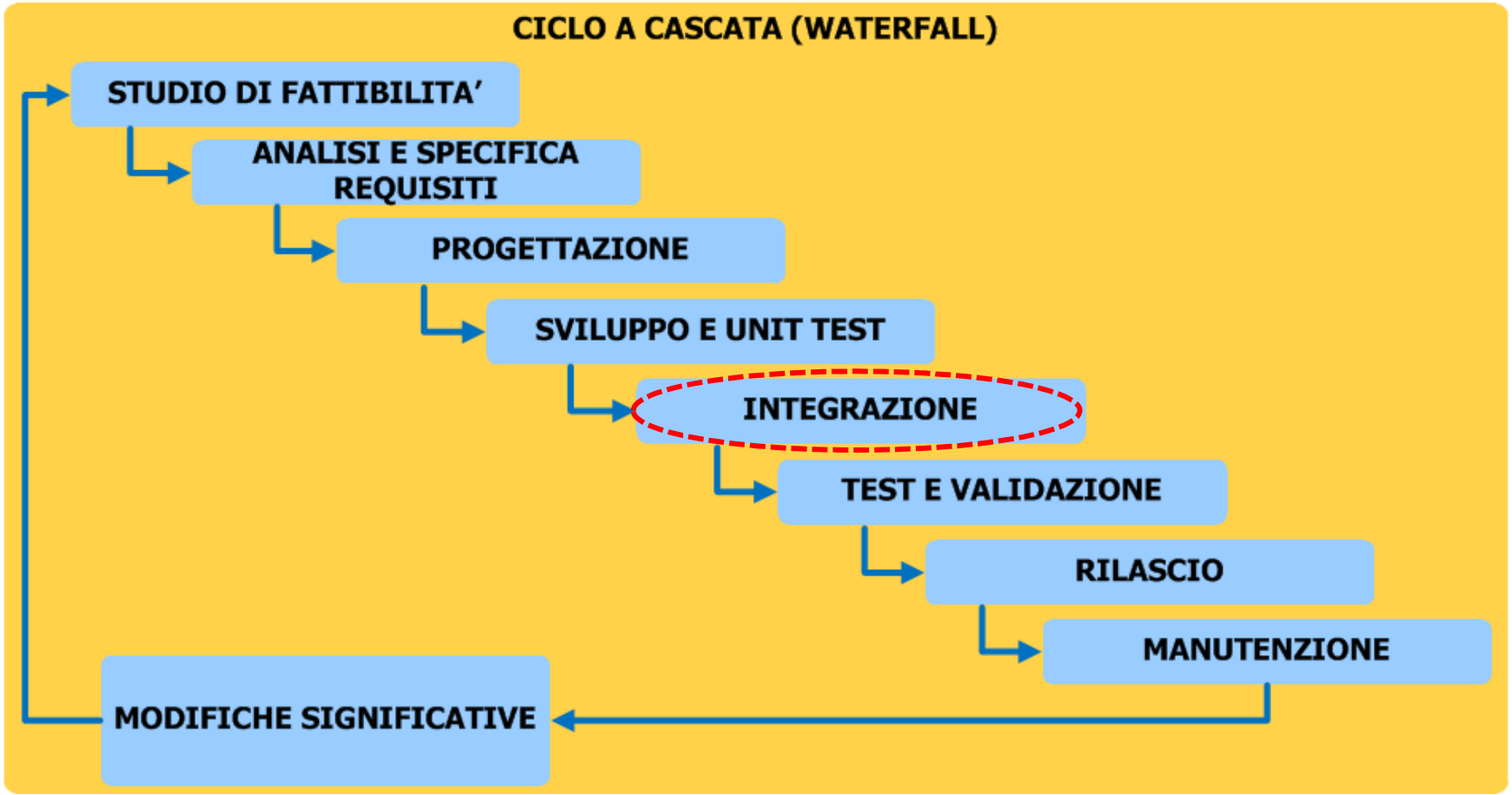
a-key

a-key

Automotive SPICE process reference model - Overview



Ciclo di vita del Software



SOFTWARE INTEGRATION AND INTEGRATION TEST

The purpose of the Software Integration and Integration Test Process is:

- to **integrate the software units** into **larger software items** up to a complete integrated software consistent with the software architectural design
- to **ensure** that the **software items are tested**
- to **provide evidence for compliance of the integrated software items** with the software architectural design, including the interfaces between the software units and between the software items.

Tipici problemi individuati in fase di integrazione

- Interpretazione inconsistente di parametri e valori tra i componenti
- Incompatibilità a livello dinamico
- Comportamenti omessi
- Violazione di vincoli di dimensione, dominio, risorse

Software Integration Strategies

La Software integration può essere condotta secondo le seguenti strategie:

- **"Big bang" integration strategy**
- **Incremental integration strategies**

"Big bang" integration strategy

Testa ogni modulo in isolamento e integra tutti i moduli contemporaneamente

- Tutti o quasi i moduli sviluppati e testati vengono aggregati e poi sottoposti ad integration testing
- Metodo molto efficace per ridurre i tempi del processo di integrazione
- Applicabile a progetti semlici e nelle prime fasi di integrazione
- Non richiede una pianificazione specifica dell'ordine di costruzione dei componenti, che verranno integrati contemporaneamente

“Big bang” integration strategy

Vantaggi: conveniente per sistemi piccoli

Svantaggi :

- Necessità di drivers e stubs per ogni modulo
- Non permette lo sviluppo parallelo
- Difficile localizzazione dei fauls
- Facile mancata individuazione di faults di interfaccia

Incremental integration strategy

Prevede di scrivere e testare il software in piccole parti e poi combinare le parti una alla volta

Steps:

1) Sviluppare uno "skeleton" funzionale del sistema
2) Progettare, codificare, testare e debuggare una piccola parte nuova
3) Integrare la parte nello "skeleton"
4) Testare/debuggare l'insieme prima di aggiungere altre parti

Incremental integration strategies

Vantaggi rispetto alla strategia Big-bang:

A. **E' facile localizzare i faults**. Ogni nuovo problema rilevato è ovviamente correlato al nuovo componente integrato. Il rischio è ridotto. E' chiaro dove cercare il problema, sia in caso di problemi di implementazione del modulo, sia in caso di problemi di interazione col resto del sistema

B. **E' facile monitorare l'avanzamento.** E' chiaro quali e quante features siano presenti nello stato di integrazione, senza attendere il completamentodi tutte

C. **Le unità del sistema sono testate in modo più completo**. L'integrazione inizia presto nel progetto → le unità vengono testate più spesso anche come parte diun sistema integrato

Incremental integration strategies

Vincoli di pianificazione:

- Impone dei vincoli di precedenza nell'ordine di aggregazione dei componenti, a causa delle dipendenze tra essi
- **Richiede una pianificazione accurata anche della fase di costruzione per rispettare l'ordine e la schedulazione di integrazione dei componenti del sistema**

Metodologie di pianificazione:

- **Top Down**
- **Bottom Up**
- **Sandwich**

Top Down Integration strategy

1) Il componente in cima alla gerarchia (top unit) viene scritto e integrato per primo (Es. Una main window, il loop di controllo di una applicazione, etc.)
2) E' necessario realizzare degli stubs per esercitare la top unit
3) Man mano vengono realizzati gli altri componenti in ordine top-down
4) Ciascun componente realizzato viene integrato in ordine top-down, sostituendo lo stub corrispondente
5) E' necessario realizzare degli stubs per esercitare il nuovo componente

Top Down Integration strategy

E' importante avere specificato attentamente le interfacce per ridurre i problemi in fase di integrazione

Vantaggi:

- La logica di controllo del sistema viene testata molto presto
- Tutti i componenti in cima alla gerarchia vengono esercitati molto → i problemi concettuali e di design vengono individuati velocemente
- Permette di completare un sistema parzialmente funzionante presto nel progetto

Svantaggi:

- Necessità di stubs: altro codice che può presentare errori

Bottom-Up Integration

1) I componenti in fondo alla gerarchia sono scritti e testati per primi
2) E' necessario realizzare dei driver per esercitare le low units
3) Man mano vengono realizzati gli altri componenti in ordine bottom-up
4) Ciascun componente realizzato viene integrato in ordine bottom-up, sostituendo il driver corrispondente
5) E' necessario realizzare dei driver per esercitare il nuovo componente

Bottom-Up Integration

Vantaggi:

- Restringe la possibile fonte di errori al solo componente in corso di integrazione, rendendo più facile la localizzazione degli errori
- L'integrazione può iniziare presto nel progetto
- Esercita interfacce di sistema potenzialmente problematiche presto nel progetto

Svantaggi:

- Lascia l'integrazione e il test delle interfacce principali ad alto livello alle fasi finali del progetto → problemicettuali e di design sono individuati solo dopo che tutto il lavoro di costruzione è stato effettuato
- Necessità di un design completo del sistema prima di poter iniziare l'integrazione

- Necessità di stubs: altro codice che può presentare errori

Sandwich Integration

Approccio combinato delle strategie top-down e bottom-up per combinare i vantaggi di entrambe.

Fondamentale → **Identificazione delle priorità di integrazione e attenta pianificazione, in base a criteri selezionati**

Sandwich Integration

Criteri di prioritizzazione (es):

- **Criterio del Rischio associato con ciascun componente**:
  1) Per primi i componenti ad alto livello in cima alla gerarchia
  2) Poi le interfacce di device e le utilities di utilizzo esteso in fondo alla gerarchia
  3) Poi le unità a livelli intermedi della gerarchia che presentano aspetti di complessità particolare e criticità
  4) Poi le altre unità a livelli intermedi

- **Criterio di integrazione per Features**:
  - Viene integrata una Feature alla volta
  - Le SW units vengono aggregate in Feature Trees = collezioni gerarchiche di componenti che realizzano una feature

Master Sistemi Embedded per Internet of Things

CORSO «ORGANIZZAZIONE, PROCESSI E CERTIFICAZIONI»

CMMI- DEV
AUTOMOTIVE SPICE

a-key

Process areas

VERIFICATION - VER

"The purpose of Verification (VER) is to **ensure that selected work products meet their specified requirements**"

Process areas

VERIFICATION

- **SG 1 Prepare for Verification**
  - SP 1.1 Select Work Products for Verification
  - SP 1.2 Establish the Verification Environment
  - SP 1.3 Establish Verification Procedures and Criteria
- **SG 2 Perform Peer Reviews**
  - SP 2.1 Prepare for Peer Reviews
  - SP 2.2 Conduct Peer Reviews
  - SP 2.3 Analyze Peer Review Data
- **SG 3 Verify Selected Work Products**
  - SP 3.1 Perform Verification
  - SP 3.2 Analyze Verification Results

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

Software Integration and Integration Test

"The purpose of the Software Integration and Integration Test Process is to **integrate the software units into larger software items** up to a **complete integrated software consistent** with the **software architectural design** and to **ensure** that the **software items are tested** to provide **evidence for compliance** of the **integrated software items** with the **software architectural design**, including the **interfaces** between the software units and between the software items."

Process outcomes

1) a **software integration strategy** consistent with the project plan, release plan and the software architectural design is developed to integrate the software items;

2) a **software integration test strategy** including the **regression** test strategy is developed to test the software unit and software item interactions;

3) a **specification for software integration test** according to the software integration test strategy is developed that is suitable to provide evidence for compliance of the integrated software items with the software architectural design, including the interfaces between the software units and between the software items;

4) **software units and software items are integrated** up to a complete integrated software according to the integration strategy;

Software Qualification Test

"The purpose of the Software Qualification Test Process is to **ensure that the integrated software** is **tested** to **provide evidence** for **compliance with the software requirements**."

Process outcomes

1) a **software qualification test strategy** including regression test strategy consistent with the project plan and release plan is developed to test the integrated software;
2) a **specification for software qualification** test of the integrated software according to the software qualification test strategy is developed that is suitable to provide evidence for compliance with the software requirements;
3) **test cases** included in the software qualification test specification are **selected** according to the software qualification test strategy and the release plan;
4) **the integrated software is tested** using the selected test cases and the results of software qualification test are recorded;
5) **consistency and bidirectional traceability** are established between software requirements and software qualification test specification including test cases and between test cases and test results; and
6**) results** of the software qualification test are summarized and communicated to all affected parties.