POLITECNICO DI TORINO

Corso di Laurea Magistrale

in Ingegneria Aerospaziale

Tesi di Laurea Magistrale

CERTIFICAZIONE DI UN TOOL PER FAULT TREE ANALYSIS

RELATORI:

Prof. Paolo Maggiore

Ing. Stefano Genta

CANDIDATO

Daniele Giardina

Dicembre 2018

IN COLLABORAZIONE CON:

Il Gruppo Teoresi, nato a Torino nel 1987, è oggi una realtà internazionale che conta oltre 600 dipendenti in Europa e negli Stati Uniti. Opera come partner qualificato supportando i clienti nello sviluppo di prodotto e di processo, attraverso tecnologie innovative e un'esperienza globale in ambito ingegneristico. Il Gruppo Teoresi è attivo in una vasta gamma di settori di mercato e rappresenta una combinazione di competenze tecniche e know-how applicativo in grado di coprire l'intero ciclo di sviluppo del prodotto, dalla fase di progettazione concettuale sino al supporto after-market.

La passione per la ricerca e l'innovazione ha spinto Teoresi negli anni a collaborare con le migliori università, dal 2012 Teoresi collabora con il Politecnico di Torino per lo sviluppo di tesi nel settore aerospaziale.

INDICE

INDICE DELLE FIGURE

CAPITOLO 1. INTRODUZIONE

1.1 OBIETTIVI:

Il presente elaborato si pone come obiettivo quello di descrivere il processo che porta all'ottenimento di una certificazione per i tool utilizzati nel processo di sviluppo di un prodotto, sia esso hardware o software. L'ambito di utilizzo è quello della sicurezza, pertanto è necessario che tutto il materiale utilizzato durante il ciclo di vita del prodotto sia ritenuto idoneo secondo un certo standard, contenuto all'interno delle normative.

Le normative e gli standard a cui si fa riferimento derivano dallo IEC (International Electrotechnical Commission), e sono state adattate ai diversi ambiti di interesse, in particolare al ramo ferroviario, automobilistico e aeronautico.

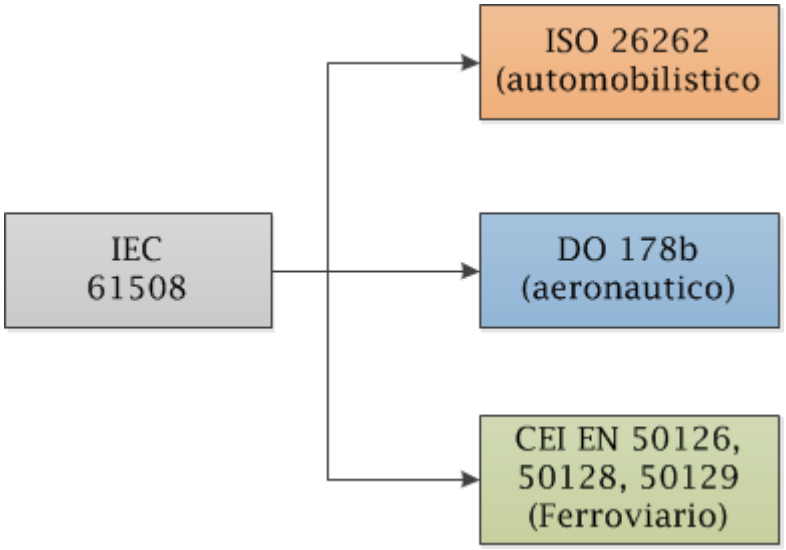

Figura 1.1: Genealogia delle normative

L'obiettivo non è quello di ottenere fisicamente la certificazione del prodotto, processo lungo e dispendioso per l'azienda, ma tracciare una baseline prendendo a riferimento gli standard contenuti nelle normative sopra riportate. Tali standard, al loro interno, riportano le indicazioni da seguire per l'ottenimento della certificazione da parte di enti internazionali come TUV, in maniera da dimostrare che l'utilizzo di tale tool, in applicazioni di tipo safety critical, non incide durante la fase di progetto e sviluppo in maniere ritenute pericolose per la sicurezza.

In tutto il parco normativo sopra citato, le uniche due che richiamano il processo di certificazione del tool sono la ISO 26262-8 e la DO-178B. Le due norme, seppur concettualmente uguali per quanto concerne gli obiettivi finali, sono diverse nelle metodologie per le quali questi vengono raggiunti.

La ISO 26262-8 parla di certificazione del file attraverso l'assegnazione di un parametro chiamato Tool Confidence Level, il quale permette di calcolare il livello di confidenza posseduto nei risultati ottenuti dal tool una volta lanciato il calcolo. Sulla base del TCL si possono determinare le richieste necessarie per ottenere la certificazione del tool. Il significato ed il calcolo di tale parametro saranno approfonditi successivamente.

La DO-178B, invece, intende la certificazione come qualificazione del software. Questa normativa fa un'importante distinzione tra development tool e verification tool, a rimarcare il

ruolo del tool nel processo di sviluppo, nonché per ragioni di tracciabilità. Bisogna ricordare che, essendo una norma di stampo aeronautico, è più anziana e presenta degli standard di sicurezza più elevati rispetto alle sue omologhe in campo automobilistico e ferroviario. Le pratiche da seguire per la qualificazione del tool si basano su un parametro chiamato Control Category, che assume due valori (CC1 o CC2) a seconda del tipo di tool che si intende sviluppare, sia esso development oppure verification.

La differenza tra certificazione e qualificazione è importante, poiché mentre un prodotto certificato ISO 26262, una volta ottenuto il marchio di certificazione, può essere utilizzato in ogni progetto che riguardi l'applicazione di tale norma, la qualifica deve essere ottenuta per ogni tool utilizzato e per ogni ambito diverso nel quale questo viene utilizzato. Ciò vuol dire che, per ogni progetto, è necessario qualificare il tool per quella applicazione. Altra differenza importante, che si evince leggendo e confrontando le due normative, è l'attenzione verso la tracciabilità. La normativa aeronautica spinge tanto su questo spetto, per ragioni di controllo qualità e per la delicatezza che le applicazioni aeronautiche sono chiamate a svolgere, con livelli di sicurezza richiesta più alti. Un'ultima differenza facilmente riscontrabile è la leggibilità delle due norme. Quella automotive risulta essere più chiara e concisa in certi aspetti rispetto a quella aeronautica. Ciò è dovuto, oltre che alla maggior giovinezza della ISO rispetto alla DO, anche alla rigidezza di quest'ultima. Quando si parla di prodotto aeronautico non si può lasciare nulla a libera interpretazione pertanto si vanno ad esaminare tutti gli aspetti.

Altro obiettivo, oltre alla creazione di una baseline per la certificazione, sarà quello di applicare entrambe le norme, in maniera da dimostrare in primis l'applicabilità, ed in secondo luogo se il tool sia certificabile o meno, ponendo l'accento sulle differenze tra le due diverse strategie seguite.

Riassumendo, gli obiettivi di questo elaborato sono:

- Tracciare una baseline per la certificazione/qualificazione di un tool per l'esecuzione di Fault Tree Analysis, seguendo le norme ISO 26262 e DO-178B;
- Confrontare le due strategie di certificazione, valutando quale sia la migliore.
- Valutare la certificabilità del tool, mettendo in evidenza gli aspetti che portino ad un risultato positivo o negativo;

1.2 FAULT TREE ANALYSIS. COS'È E COME È FATTA:

Il tool oggetto del presente elaborato verte sull'esecuzione della fault tree analysis (FTA), un metodo utilizzato nell'ambito delle disciplina RAMS che consente di eseguire delle analisi di affidabilità, disponibilità e sicurezza.

Questa disciplina si cominciò a diffondere, a partire dagli anni '30 del secolo scorso, in campo aeronautico prima, ed aerospaziale poi, quando ci si cominciava a puntare l'attenzione sulle tematiche di affidabilità e sicurezza. Con la nascita dell'aviazione commerciale e con l'uso dell'energia nucleare, negli anni '50, diventò di primaria importanza implementare metodi che salvaguardassero le persone coinvolte nell'uso del mezzo aereo, dall'essere sottoposte a rischi inaccettabili (definizione, questa, comunemente accettata per esprimere il concetto di sicurezza). In particolare si cercava di evidenziare quelli che potessero essere i modi di guasto comuni dei componenti aeronautici, in maniera da prevedere ed intervenire, prima che le

condizioni di rischio introdotte sul sistema velivolo, evolvessero in guasti singoli o multipli dai possibili esiti catastrofici. Un ulteriore impulso derivò dall'aumento delle prestazioni dei velivoli, che introduceva una maggiore complessità nei sistemi.

I campi aeronautico e nucleare sono stati pionieri nell' introduzione di tali tecniche. Le discipline RAMS, sono trasversali, e applicabili a molti aspetti della vita quotidiana. Prendiamo ad esempio le nostre case, le nostre automobili o tutti quei mezzi che utilizziamo per spostarci da casa al lavoro, i luoghi che utilizziamo per lavorare, studiare, divertirci; ciascuno di questi deve soddisfare certi requisiti di sicurezza in maniera da non sottoporre le persone a livelli di rischio elevati. Le discipline RAMS, possono trovare applicazione anche nelle comuni attività quotidiane, nei processi all'interno di un sito produttivo o di un'azienda operante nel settore dell'ICT. La vastità del campo di applicazione, e la natura probabilistica di tale disciplina, hanno fatto sì che tutti questi aspetti fossero ignorati o poco considerati al di fuori del campo aerospaziale per molti anni. Ma con l'avanzamento tecnologico e la maggiore complessità dei sistemi con i quali si ha comunemente a che fare nella vita di tutti i giorni, ha portato alla ribalta tali tecniche, che stanno sempre più prendendo piede in campi nei quali, fino a pochi anni fa, non venivano prese in considerazione.

Sulla base di quanto esposto, risulta necessario affidarsi a delle tecniche che permettano, in maniera più o meno precisa, di identificare quegli elementi che possono generare rischi all'interno del sistema, e che permettano anche di calcolare tali livelli, basandosi su stime di affidabilità e disponibilità di un sistema.

Varie tecniche sono utilizzate a tale scopo: FMEA/FMECA, FTA, HAZOP, Zonal Safety Analysis, RDB. Sono tecniche di tipo qualitativo e/o quantitativo. Qualitativo vuol dire che danno un'interpretazione scollegata da un numero vero e proprio, e soggetta a considerazioni di tipo soggettivo. È necessario che chi esegue analisi di tipo qualitativo abbia una certa esperienza, in maniera da andare ad identificare subito quei fattori di rischio che più di altri possono portare ad effetti gravi. Analisi quantitative, invece, danno delle stime numeriche, basate su tecniche probabilistiche.

1.2.1 FTA:

Fatta una dovuta introduzione al concetto di sicurezza e data una velocissima panoramica delle tecniche utilizzate, ci si concentra su quella implementata dal tool oggetto di questo elaborato. È un tool per l'esecuzione di Fault Tree Aanalysis (FTA), e quindi è necessario approfondire i concetti legati a questo tipo di analisi.

La FTA è in grado di identificare le combinazioni di guasti/rotture che possono portare a uno specifico guasto del sistema, e quindi può evidenziare debolezze del progetto che tecniche più semplici possono non cogliere, soprattutto con molte ridondanze [1].

FTA è impiegata per esaminare problemi dalle conseguenze molto gravi, che possono risultare da guasti la cui eliminazione/miglioramento può comportare un grande impegno di risorse finanziarie [1].

La Fault Tree Analysis è una metodologia di analisi deduttiva di tipo top down, adatta all'analisi di eventi complessi derivanti dal succedersi di più eventi singoli. L'approccio top down prevede che, partendo da un evento di guasto di un componente (definito TOP event), si scenda nell'analisi fino a trovare gli elementi di guasto o gli eventi che portano allo scatenarsi del guasto (BASIC event).

I passi che portano alla creazione di un albero sono:

- Sviluppo del fault tree: si ragiona deduttivamente a ritroso partendo dal top event per arrivare ad individuare quelli che sono i basic events;
- I basic events sono collegati al top event attraverso delle relazioni logiche, che sfruttano l'algebra boolena (concetti di AND, OR, NOT, ecc);
- L'albero viene rivisto in maniera da correggerlo e migliorarlo introducendo anche fattori di natura umana;

Tali operazioni possono essere effettuate anche da tool informatizzati. Tuttavia è bene precisare che tale tecnica è prettamente soggettiva: non esiste un metodo unico per arrivare alla formulazione di un albero dei guasti, e persone diverse possono pervenire a risultati diversi.

Come già riportato, i basic events concorrono al top event collegandosi tra di loro tramite delle porte logiche, che mostrano il propagare della causa del guasto lungo tutto l'albero. Per ogni albero esiste un solo top event. Gli altri obiettivi della FTA sono: ricavare l'equazione che lega tra di loro i basic events e mostrare, nel caso in cui si associno delle tecniche quantitative, come l'affidabilità (espressa come un numero) dei basic events possa determinare quella del top event.

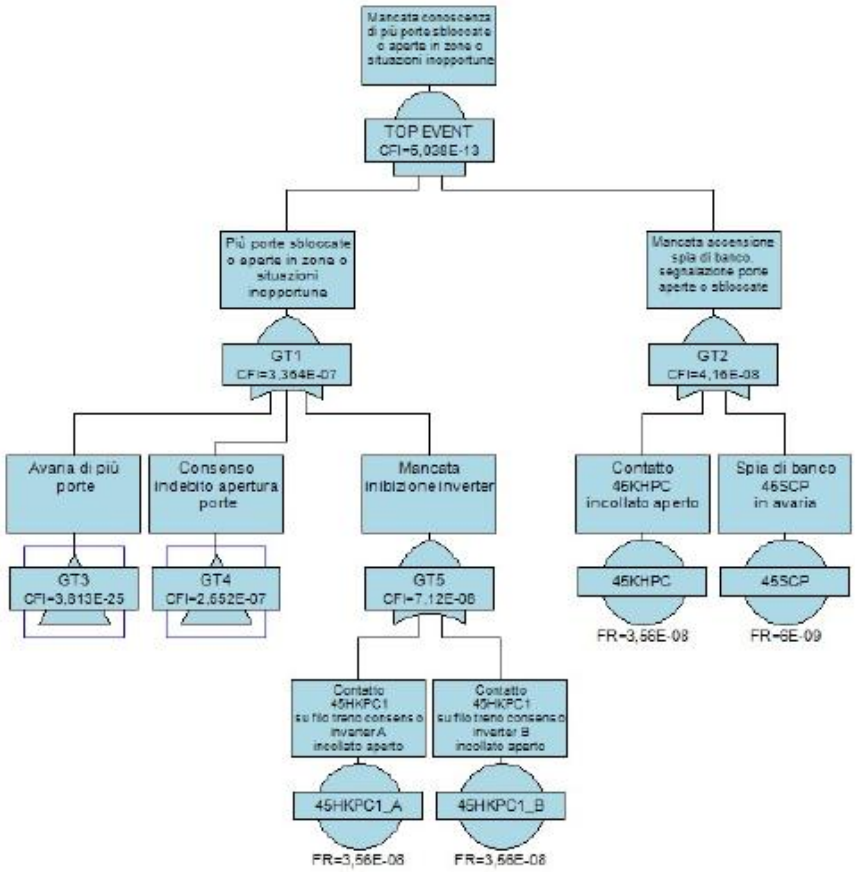

Figura 1.2: Esempio di un albero di guasto ottenuto con un tool commerciale

Presentandosi come una tecnica quantitativa, è bene dare qualche accenno di quella che è la matematica utilizzata per arrivare ai risultati. Si basa sulla teoria delle probabilità e sull'algebra booleana.

1.2.2 TEORIA DELLE PROBABILITÀ

La teoria delle probabilità contiene al suo interno la teoria degli insiemi, l'algebra delle probabilità e il teorema di Bayes.

- La teoria degli eventi è un approccio generale nel quale gli eventi sono visti come degli insiemi, dei quali bisogna calcolare la probabilità di accadimento. Presi tre eventi A, B e C, ognuno con una certa probabilità di accadimento, è possibile determinare come questi sono legati tra loro. Un esempio che classicamente viene utilizzato, è quello dei dadi. Se diciamo che A è l'uscita di un numero compreso tra 2 e 4, B è l'uscita di un numero pari e C l'uscita di un numero dispari, si può dire che A, che contiene due numeri pari e uno dispari è l'intersezione di B e C. La figura seguente rappresenta il concetto espresso attraverso i diagrammi di Venn.

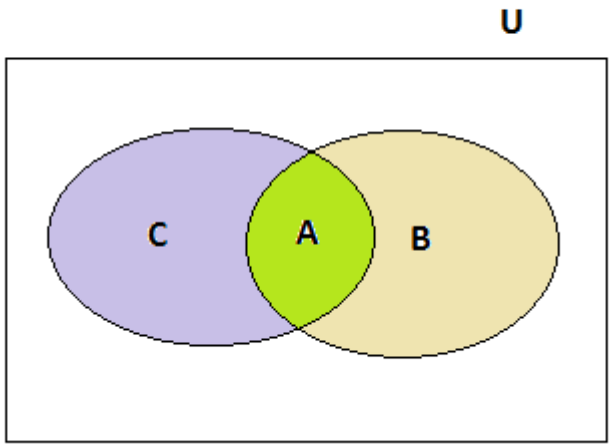

Figura 1.3: Diagramma di Venn

U è l'insieme universo, e contiene tutti gli altri. Le operazioni comuni effettuate sugli insiemi sono l'unione (U) e l'intersezione (∩). L'unione è una somma, l'intersezione è una moltiplicazione mirata ad evidenziare quali elementi hanno in comune gli insiemi.

- L'algebra delle probabilità si basa sulla relazione tra gli eventi. Due eventi che si dicono mutualmente esclusivi se il verificarsi dell'uno esclude l'altro. In questo caso le probabilità di accadimento vengono sommate:

$$P(A) + P(B) = S$$

Graficamente, così come nella FTA, un evento di tal genere si indica con la porta OR.

Due eventi si dicono mutualmente indipendenti se l'accadimento di A non ha influenze su B. In questo caso le probabilità di accadimento vengono moltiplicate:

$$P(A)*P(B) = S$$

Graficamente, così come nella FTA, un evento di tal genere si indica con la porta AND.

- Il teorema di Bayes dice che ogni sottoinsieme A costituisce una partizione dell'insieme universo, e che l'unione di tutti questi sottoinsiemi genera l'insieme universo. Questo teorema, del quale viene qui omessa la dimostrazione, permette di ricavare la probabilità di accadimento per un insieme universo suddiviso in k sottoinsiemi.

1.2.3 ALGEBRA BOOLEANA:

Come si è potuto intuire, un albero dei guasti riporta delle relazioni matematiche per il calcolo delle probabilità. Assegnata una certa probabilità di accadimento al basic event, si vede come questa propaga fino ad ottenere quella del top event. I vari basic events, sono collegati tra di loro attraverso delle porte logiche. Partendo da questi viene fatta propagare l'analisi fino al top event. La propagazione viene studiata tramite l'algebra booleana, che essenzialmente si basa su due valori: 0 e 1. 1 è per un evento positivo, 0 per un evento negativo. Tramite le porte logiche AND e OR, di cui è stato discusso il significato, si può pervenire a delle equazioni logiche, che come risultato restituiranno valore 1 se l'evento si verifica, 0 se non si verifica. Tale algebra, nell'ambito del tool oggetto di questo elaborato, permette di calcolare l'equazione che combina i basic events per arrivare al top event.

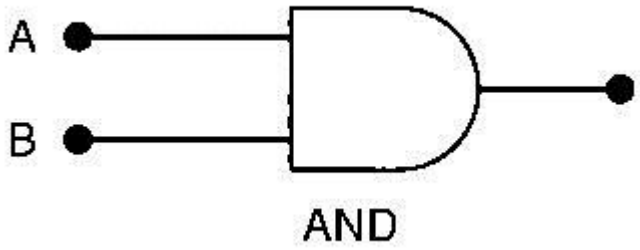

Figura 1.4: Simbolo indicante la porta logica AND

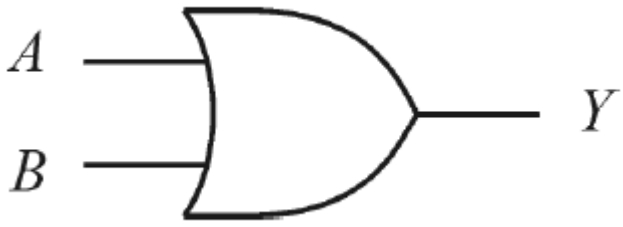

Figura 1.5: Simbolo indicante la porta logica OR

La logica booleana viene implementata tramite le tavole di verità, che mostrano il funzionamento delle porte logiche sopra riportate. Tali tavole sono delle tabelle, che mostrano, data una coppia di input A e B, il risultato dato dalla loro combinazione:

A	B	A ∧ B
0	0	0
0	1	0
1	0	0
1	1	1

Figure 1.6: Tavola di verità relativa alla porta logica AND

1.2.4 CREAZIONE, ANALISI E VALUTAZIONE DELL'ALBERO DEI GUASTI:

Per essere efficacemente eseguita, una Fault Tree Analysis deve seguire i seguenti step:

- Identificazione dell'obiettivo: bisogna avere chiari in mente gli obiettivi dell'analisi, identificando l'adeguato top event;
- Definizione del top event: una volta identificato, bisogna definirlo adeguatamente, essendo il punto di partenza dell'analisi;
- Definizione del metodo di risoluzione: durante questa fase il top event viene scomposto in eventi intermedi, fino ad arrivare ai basic event, identificando quali sono le relazioni logiche che legano quest'ultimi per arrivare al top event. Si vanno allora ad individuare il numero di porte logiche necessarie per la propagazione.
- Costruzione dell'albero: si parte sempre dai basic events individuati, si legano tra di loro tramite le porte logiche e si raggiunge il top event;
- Lancio dell'analisi: se si esegue un'analisi quantitativa, sulla base dell'albero costruito le probabilità vengono combinate fornendo il risultato relativo al top event;
- Presentazione dei risultati: si controlla che tutto sia stato eseguito in maniera adeguata, e che i valori ottenuti siano sensati, per individuare eventuali errori. È possibile allora confrontare i risultati ottenuti con gli obiettivi. Durante questa fase, sulla base dell'albero, si può ottenere l'equazione dei cutset. Viene chiamato cutset l'insieme dei basic events, e l'equazione non è altro che la rappresentazione logica dell'albero funzionale.

Una regola semplice, ma che molto spesso viene ignorata, è associare ad ogni evento dell'albero una descrizione, in maniera da rendere l'albero leggibile a persone terze, e per avere una migliore visualizzazione dei risultati. Il modello rappresentato in Figura 1.2 rappresenta l'albero di guasto relativo al sistema di apertura delle porte con treno in movimento.

Le tipologie di analisi eseguibili, nonché i tipi di output ottenibili dipendono dalle capacità del programma. Questo aspetto incide anche sulla quantità di blocchi e funzioni a disposizione del programma. Un tool commerciale possiede anche porte di tipo NOT, EXCLUSIVE OR, VOTE, PRIORITY AND. Sebbene la descrizione delle funzioni di tali porte logiche esuli dallo scopo del presente elaborato, per completezza vengono citate. Basate sulle due fondamentali AND e OR, introducono delle funzionalità in grado di rendere più complesso (e quindi migliorare il grado di approfondimento dell'analisi) l'albero, esaminando un set più grande di possibilità.

1.3 MATLAB E SIMULINK:

Nel seguente paragrafo sarà analizzato l'ambiente di sviluppo utilizzato per la creazione del tool, basato su software (Matlab) e la suite di modellazione (Simulink).

1.3.1 PANORAMICA SU MATLAB:

Matlab (abbreviazione di MATrix LABoratory) è un programma creato dalla softwarehouse Matworks con lo scopo di fungere da sofisticato strumento per il calcolo scientifico.

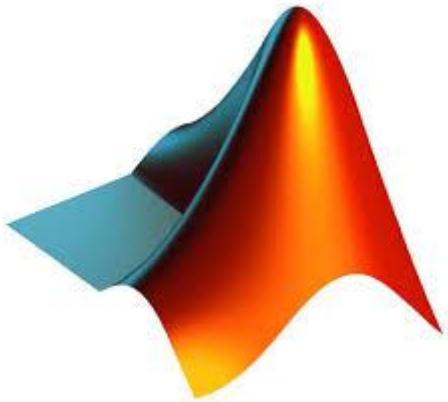

Figura 1.7: Logo della softwarehouse Matworks

Come è intuibile dal nome, è un programma che eccelle nella manipolazione matriciale. Di fatti, qualsiasi numero o variabile di tipo numerico gli si passi, viene memorizzata come una matrice o vettore. Questo metodo di rappresentazione interno, risulta essere molto comodo in tutti quei campi e quelle applicazioni che fanno un uso spinto del calcolo matriciale. Di fatti può essere considerato come una sofisticata calcolatrice, in grado di eseguire non soltanto le basiche operazioni matematiche tra numeri, ma anche le più complesse e lunghe operazioni matriciali. Nel tempo ha anche acquisito una sempre maggiore capacità nell'esecuzione del calcolo simbolico. Ma Matlab è molto di più di un semplice strumento di calcolo: sono tantissime le applicazioni che è possibile realizzare, associando il tutto ad un'interfaccia grafica e alla possibilità di rendere graficamente disponibili, in tempi brevi (relativamente alle capacità di calcolo del calcolatore), i risultati ottenuti. A differenza dei comuni linguaggi di programmazione come C, C++, Fortran, ecc., Matlab ha già internamente tutta una serie di routine e librerie di funzioni che permettono di velocizzare la programmazione e l'implementazione di programmi di calcolo, anche complessi. Anche l'importazione di dati provenienti da programmi esterni è molto agevole: le funzioni che permettono l'importazione e la lettura di files in formato testuale o Excel sono in grado di riconoscere le diverse colonne e immagazzinare i files in vettori o matrici pronte all'uso. Sulla base di questi pochi esempi (ma se ne potrebbero fare molti di più) non stupisce come Matlab si stia sempre più affermando in tutte le applicazioni ingegneristiche e non, laddove siano richiesti software in grado di creare programmi di calcolo complessi, ma di facile lettura.

Matlab, così come gli altri linguaggi di programmazione di alto livello, è predisposto per il calcolo numerico: si presta bene alla ripetizione di operazioni matematiche cicliche in poco tempo, restituendo risultati numerici in continua. Tuttavia, data la sua natura, risulta essere più lento rispetto ad un omologo script scritto in C++ o Fortran, data la necessità di convertire

in matrice qualsiasi numero gli venga passato. Tuttavia, rispetto a questi ultimi, il calcolo matriciale è molto più veloce ed immediato da impostare.

Un vantaggio posseduto da Matlab è la non necessità di dichiarare anticipatamente le variabili ed il loro tipo. A differenza del C++, dove devono essere preventivamente dichiarate prima dell'uso, Matlab è in grado di riconoscere l'uso di una nuova variabile e allocare durante l'elaborazione lo spazio in memoria per utilizzarla. Questo riduce la possibilità di commettere errori sistematici negli script dovuti alla mancata dichiarazione di variabili.

Si è già parlato della sua nutrita libreria di funzioni built-in che consente, con poche righe di codice, di creare programmi che altrimenti richiederebbero un numero di righe di gran lunga superiore. Per esempio è in grado di riconoscere automaticamente il prodotto tra due matrici, eseguendo anche un'attività di verifica sulla consistenza del calcolo. Questa operazione, riprodotta in C, richiede un numero di righe 30 volte superiore. Le funzioni, si prestano bene anche al calcolo simbolico, permettendo di rappresentare funzioni sia in campo reale che complesso, operare il calcolo di autovalori e autovettori delle matrici, risoluzione di sistemi lineari, passare da funzioni nel domino del tempo al dominio della frequenza, l'esecuzione e l'implementazione di metodi matematici.

Pertanto Matlab è visto come l'ambiente di sviluppo del futuro, sempre più usato in ambito industriale, soprattutto aerospaziale.

Come si presenta la schermata di avvio di Matlab:

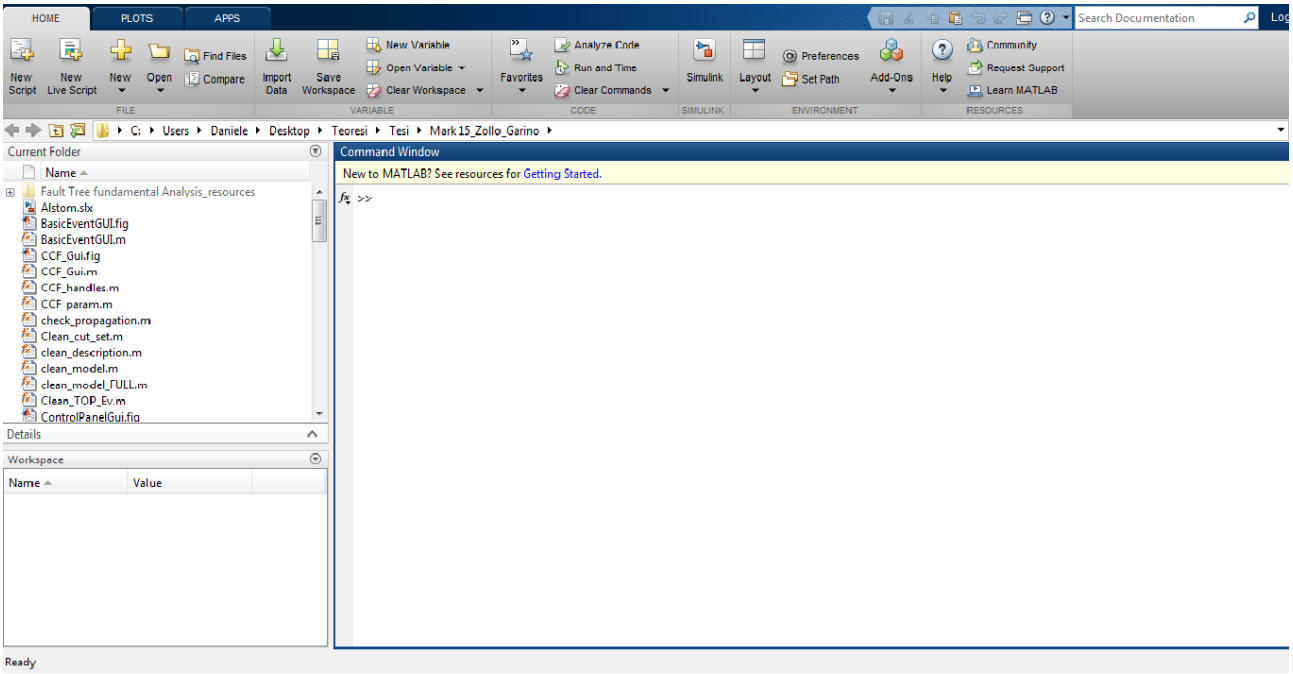

Figura 1.8: Schermata di avvio di Matlab

La schermata presenta una command window, che permette di utilizzare il programma come una calcolatrice, inserendo le variabili ed ottenendo immediatamente, ed in cascata, i risultati. Risultati che vengono memorizzati e resi disponibili nel workspace in basso a sinistra. L'utente può scegliere se visualizzare o meno tali risultati, ma in ogni caso questi restano salvati nel workspace ed è possibile visualizzarli in qualsiasi momento per osservare

l'evoluzione del calcolo. Nella command window è possibile scrivere uno script completo, comprese le iterazioni. L'unico difetto di tale modo di operare è l'impossibilità di eseguire nuovamente il programma una volta terminate le istruzioni. Per questo si rende necessaria la creazione di uno script, utilizzando i tasti nella barra degli strumenti in alto a sinistra. Questo permette di generare script e funzioni eseguibili più volte nel tempo, ed esportabili. L'insieme degli script e delle funzioni che concorrono al funzionamento del programma, sono invece contenuti in un "path", cui fa riferimento Matlab per ricavare tutti gli strumenti necessari all'esecuzione del programma. Il path è indicato in alto, come percorso, ed il contenuto del percorso selezionato è mostrato in alto a sinistra.

Questa introduzione, seppur schematica e poco approfondita, permette di avere un'idea di come si presenta il software, e questo sarà importante quando, più avanti, saranno richiamate le normative di interesse, all'interno delle quali è necessario tener conto anche del software usato per lo sviluppo del tool.

1.3.2 PANORAMICA SU SIMULINK:

Simulink è la suite di modellazione utilizzata da Matlab per modellare e simulare sistemi fisici. È un ambiente che sfrutta una rappresentazione a blocchi per simulare parti di un sistema, e mostrare subito le relazioni che intercorrono tra di essi, rendendo la comprensione del modello estremamente intuitiva. È possibile assegnare ad ogni blocco i parametri necessari del modello, oppure si può interfacciare con Matlab per la creazione di un modello parametrico, dove le variabili vengono dichiarate in uno script apposito e poi passate a simulink, semplicemente passando il loro nome all'interno dei blocchetti. Tutto ciò permette di ottenere il modello fisico del sistema senza dover ricorrere a lunghe sessioni di programmazione in altri linguaggi come C++ o Fortran.

Simulink è inteso come un programma di tipo "Model Based". Le capacità principali sono:

- Modellazione e simulazione del sistema: si ha a disposizione un ampio spazio di progetto dove modellare e testare il sistema e l'impianto fisico che lo caratterizza;
- Esecuzione di test sin dalle fasi iniziali: non è più necessario ricorrere a costosi prototipi per poter eseguire dei test sul modello. Simulink è in grado di offrire una soluzione più economica che simuli realisticamente il comportamento di un sistema reale. Può convalidare il progetto con i test hardware-in-the-loop e la prototipazione rapida. Assicura la tracciabilità, dai requisiti alla progettazione del codice.
- Genera automaticamente il codice, senza dover ricorrere alla scrittura di migliaia di righe a mano: basta inserire le variabili necessarie in un blocchetto. Questo permette di generare automaticamente codice C o HDL con lo stesso comportamento del modello creato.

L'uso combinato di Matlab e Simulink, integra programmazione testuale e grafica creando un ambiente di simulazione semplice ed intuitivo. Simulink può usare un'ampia suite di funzioni già implementate in Matlab, basta aggiungere tale codice ad un blocco Simulink.

Matlab permette anche di ottenere i risultati di simulazione provenienti da Simulink.

Simulink basa la sua rappresentazione grafica sulla teoria di grafi, richiamandone gli elementi fondamentali per una più immediata lettura. Si presta bene ad applicazioni controllistiche negli ambiti dell'elettronica di potenza e non, sistemi di controllo, elaborazione segnali,

robotica e molto altro. Ciò ne sta garantendo una rapida diffusione nel mondo industriale ed accademico.

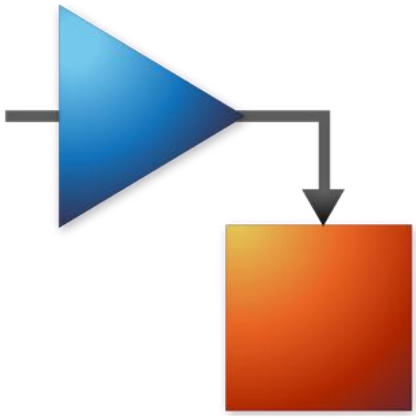

Figura 1.9: Logo Simulink

La figura seguente mostra un modello in Simulink usato per modellare la presenza di fine corsa in un sistema di tipo massa-molla-smorzatore. Simulink implementa blocchi che consentono di inserire nel modello istruzioni di tipo condizionali o scelte multiple.

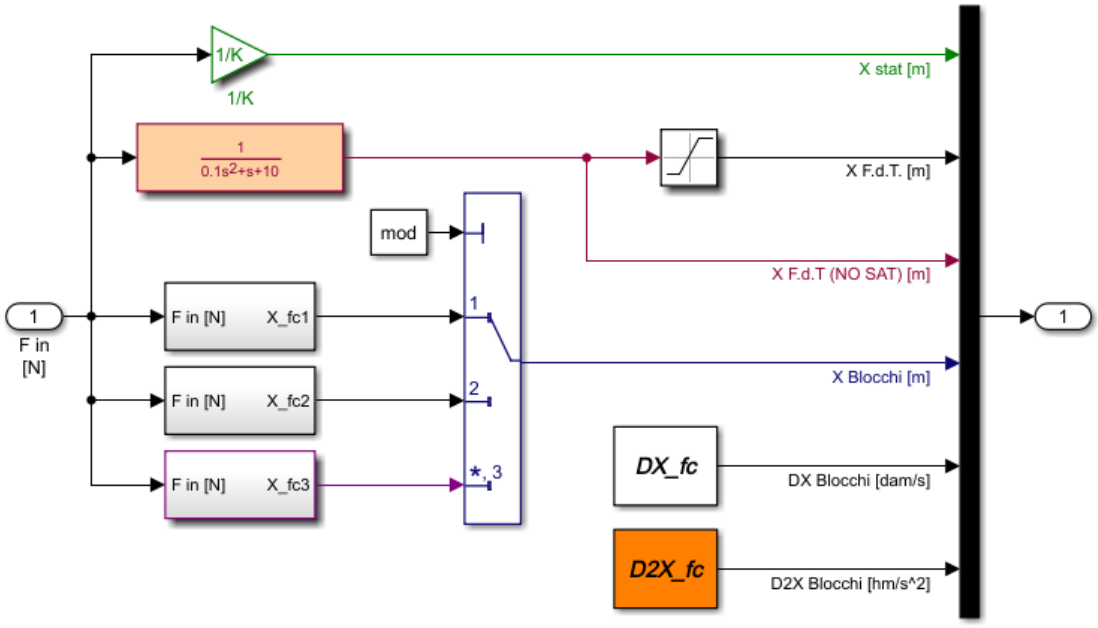

Figura 1.10: Esempio di modello realizzato in Simulink

Il tool oggetto del presente elaborato, dati i vantaggi e le possibilità offerte da Simulink, utilizza questo ambiente per la creazione degli alberi di guasto. È stato possibile tramite Matlab programmare i blocchetti necessari alla creazione dell'albero, inserendoli in una libreria apposita dalla quale è possibile recuperare i blocchi al momento della creazione di un nuovo albero.

CAPITOLO 2. NORMATIVA FUNCTIONAL SAFETY

In questo capitolo si vuole presentare il quadro normativo relativo alla sicurezza funzionale, mettendo in evidenza come questo viene declinato nei diversi ambiti di interesse (aeronautico, ferroviario e automobilistico). Si introdurranno i parametri che servono a calcolare e identificare il valore di affidabilità dei sistemi ritenuti "safety critical" e come sia possibile rispettare i requisiti di sicurezza. Una volta chiarito l'ambito nel quale si inquadrano le normative, verrà attenzionato il ciclo di vita del software/hardware, e come sia necessaria la certificazione di tutti gli strumenti "tools" che vengono sfruttati in tale ambito. In ultima analisi saranno approfondite le normative che richiedono la certificazione di tali tools e si darà uno sguardo alle attività prescritte.

2.1 PRESENTAZIONE QUADRO NORMATIVO SAFETY:

Nella vita di tutti i giorni siamo costantemente sottoposti a dei pericoli "hazard", che minano la nostra sicurezza. Gli effetti degli hazard possono andare dal piccolo incidente con conseguenze trascurabili, a situazioni catastrofiche che mettono in serio pericolo la vita stessa dell'uomo o l'ambiente. Diventa allora necessario stabilire dei gradi di pericolo, ed i criteri che ci permettano di distinguere un livello più elevato da uno meno elevato.

Il livello di rischio non è qualcosa di facilmente quantificabile, dipende da tanti fattori ed è prettamente soggettivo. Un esempio è quello del pedone che attraversa una strada. Un pedone può decidere di attraversare quando è sicuro che non sopraggiunga nessuno, valutando questa situazione come sicura. Un altro pedone, invece, può decidere di attraversare quando le vetture sopraggiungono ma sono lontane. Entrambi hanno valutato il livello di rischio, e lo hanno ritenuto accettabile, anche se l'esperienza e lo stesso istinto ci inducono a pensare che la seconda situazione sia quella più pericolosa.

È inoltre impensabile eliminare totalmente tutti i fattori di rischio presenti attorno alle persone. Il rischio deve essere accettato come un qualcosa di ineliminabile, date le tantissime variabili dalle quali dipende. Si può però agire su di esso limitandolo. Limitare il rischio vuol dire prima di tutto saper riconosce tutte quelle situazioni ritenute pericolose, e quali sono le variabili che portano a tali situazioni. Successivamente si agisce su tali variabili, con tecniche ben precise, in maniera da ridurre il livello di rischio.

Nell'ingegneria, qualora si presentino rischi derivanti da processi e/o prodotti pericolosi, si impongono misure di sicurezza preventive e mitigative. Le potenziali conseguenze di un evento incidentale su persone, comunità, ambiente e la stessa azienda, devono essere valutate appieno, in modo da prendere le necessarie misure di sicurezza per non superare, in qualunque condizione, la soglia di "rischio accettabile" [2].

Sono nate in tal senso delle normative che dettano le linee guida per il riconoscimento delle situazioni pericolose, e per il raggiungimento degli obiettivi di sicurezza richiesti. Tali norme, per citarne alcune, sono IEC 61508, RTCA DO-178B, ARP 4754, Def Stan 00-56, ecc. L'obiettivo di tali normative è quello proteggerci e difenderci nel tempo da tutte le possibili situazioni di pericolo.

Come si decide quale livello di rischio è accettabile? L'"Health and Safety Committee" ne ha dato un'interpretazione su tre livelli:

- **Rischio inaccettabile**: qualunque situazione ricada in questo livello presenta rischio non giustificato;
- **ALARP**: acronimo di As Low As Possible. In questa regione il livello di rischio è accettato purchè si intraprendano misure preventive che portino ad una sua riduzione sino ad un livello accettabile.
- **Rischio accettabile**: l'obiettivo da raggiungere.

Il rischio viene calcolato come il prodotto tra probabilità di accadimento per severità.

$$R = PxS$$

Sulla base di tale prodotto sono state create delle matrici, dette "matrici di rischio" che permettono di identificare quelle zone desiderabili nelle quali il rischio può essere accettato. Nella figura seguente viene riportato un esempio.

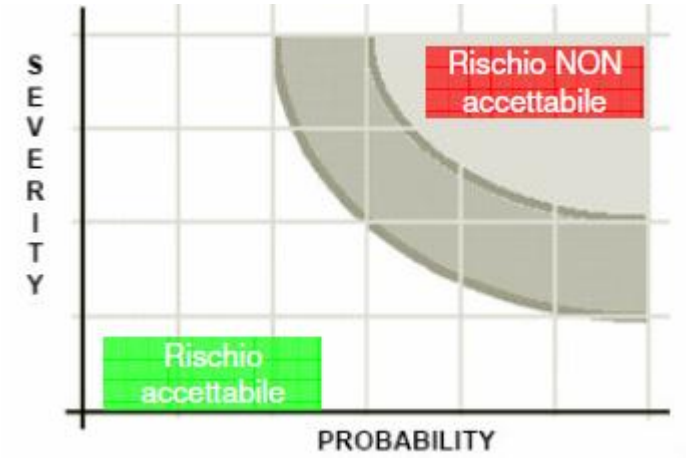

Figura 2.1: Rappresentazione delle zone di rischio

Il quadro normativo presentato in Figura 1.1 illustra la genealogia delle norme per applicazioni safety critical di tipo hardware e software. Tali applicazioni conosceranno nel tempo un incremento del livello di complessità, in accordo con requisiti sempre più complessi e stringenti, dettati da tali normative. Un software complesso non è mai esente da errori, e tali errori, se non controllati, possono propagare portando a condizioni di serio pericolo. Per questa ragione è importante seguire ed implementare un percorso di sviluppo chiaro, in maniera da creare un software sicuro. In questo ambito rientrano le normative citate, introducendo un processo standard di certificazione che tenga conto di tutto il ciclo di vita del prodotto.

Oggi le funzioni di sicurezza sono demandate ad applicazioni di tipo hardware e software, soprattutto su tutti quei sistemi complessi, in particolar modo quelli di trasporto. In tale ambito la normativa che ha dettato inizialmente le linee guida è stata la IEC 61508.

2.1.1 IEC 61508:

Tale norma discende dalla "International Electronic Commission", l'ente internazionale preposto alla standardizzazione in ambito elettrico, elettronico e tecnologie correlate. Molti dei suoi studi sono definiti in collaborazione con l'ISO, tanto che è possibile trovare la definizione di questa norma come ISO IEC 61508.

La norma indica i criteri di realizzazione dei sistemi elettrici, elettronici ed elettronici programmabili per applicazioni di sicurezza, al fine di ottenere un livello di sicurezza funzionale. È una norma di tipo stand alone, in quanto non contiene solo gli aspetti applicativi relativi ai sistemi di misura e controllo dei processi industriali, ma anche tutti gli aspetti di descrizione generale delle metodologie, proponendo un metodo che considera l'affidabilità in termini quantitativi. [2]

La norma si applica alle safety functions, ossia quelle funzioni che sono necessarie per mantenere in sicurezza un certo sistema in relazione ad un particolare evento di rischio. Ad esempio, tali funzioni possono essere svolte dalle valvole pressure relief del sistema idraulico di un aereo, dove i sensori, ed il software che le controlla, sono degli elementi che svolgono la funzione di mantenere attivo e funzionale il sistema, pena la possibilità di incorrere in eventi catastrofici.

La sicurezza può essere intesa dalla norma anche in senso funzionale: ossia può essere lo stesso sistema sotto controllo ad esercitare le funzioni di sicurezza. Prendiamo sempre come esempio le valvole del sistema idraulico del velivolo. Una valvola può agire secondo la sua funzione nominale, ma può anche implementare delle funzioni di sicurezza che prevedano che questa, in determinate situazioni, si comporti come una pressure relief. Questo permette di rendere il sistema più snello e meno complesso, evitando di inserire valvole che nella condizione nominale sarebbero in off. Ovviamente una valvola che deve assolvere tale compito dovrà essere sottoposta agli standard di sicurezza contenuti nella IEC 61508 o suoi derivati.

La normativa introduce il concetto di **SIL (Safety Integrity Level)**, che sarà approfondito più avanti. Il SIL è letteralmente il livello di integrità della sicurezza, inteso come la probabilità che un componente hardware o software di tipo safety critical non svolga la sua funzione. Il SIL allora ha un ruolo importante nell'analisi della sicurezza dei sistemi, tanto da poter determinare, in alcuni casi, anche il numero di ridondanze da adottare. Il SIL stabilisce anche chi deve eseguire tutti i test di validazione di un componente safety critical, demandando, per il livello più alto, tale ruolo a persone differenti dai progettisti.

Il SIL non viene deciso dalla norma, che ha solo funzione indicativa, ma deve essere associato coerentemente con le analisi di sicurezza effettuate sul componente.

La norma copre tutto il ciclo di vita del prodotto, dalla fase di design a quella di disposal. Determina, per ogni fase del ciclo di vita, quali sono gli accorgimenti e le pratiche da implementare per poter considerare il componente sicuro. Fornisce anche una metodologia per stabilire la specifica dei requisiti del prodotto in ordine con i necessari livelli di sicurezza funzionale da raggiungere. Inoltre, la sua ideazione e stesura, è funzionale anche a sviluppi futuri, legati al continuo avanzamento tecnologico di hardware e software.

Un **sistema strumentale di sicurezza (SIS)** è un insieme di: sensori, risolutori logici, elementi finali, dispositivi I/O, interfaccia utenti, alimentatori.

IEC 61508 è uno standard generico. È un documento corposo, che presenta di 7 parti:

1) Requisiti generali;
2) Requisiti per parti elettriche, elettroniche ed elettroniche programmabili con funzione di sicurezza;
3) Requisiti software;
4) Definizioni e abbreviazioni;
5) Esempi di metodi per la determinazione del SIL;
6) Linee guida nell'applicazione delle parti 2 e 3;
7) Overview delle tecniche e misure dello standard IEC 61508.

La parte 1 è contiene le linee generali della normativa, contenendo le attività da portare avanti per ogni fase del ciclo di vita del prodotto. Le parti 2 e 3 forniscono le linee guida per la stesura dei requisiti relativi all'hardware e al software. Queste parti sono indubbiamente le più importanti di tutta la normativa, e sono quelle che sono state riprese dalle altre normative in tale ambito, adattandole allo specifico campo di applicazione. Al punto 5, invece, vengono fornite le linee guida per determinare il SIL.

La figura seguente da un overview della normativa.

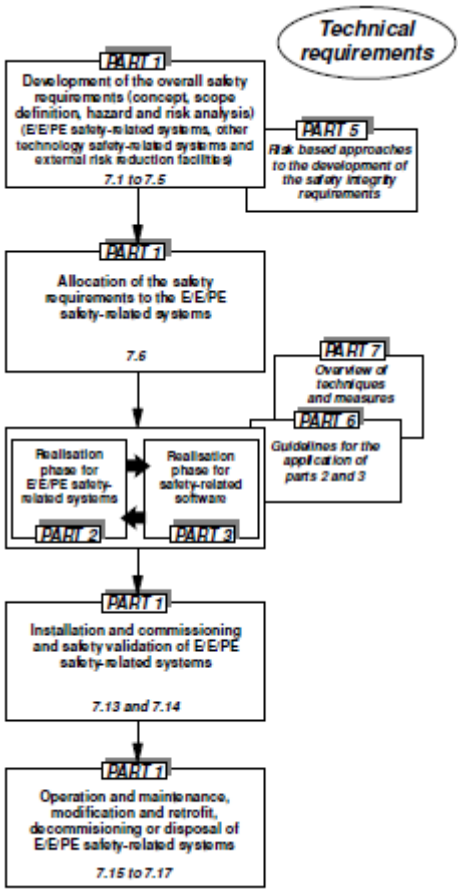

Figure 2.2: Overview della norma IEC 61508

Lo sviluppo di una funzione di sicurezza, in accordo con lo standard, necessita dei seguenti passaggi:

1) Identificare ed analizzare il rischio;
2) Determinare le conseguenze del rischio;
3) Determinare la riduzione del rischio necessaria per ogni rischio ritenuto inaccettabile;
4) Specificare i requisiti di sicurezza per ogni per ogni tecnica di riduzione (includere anche i livelli di SIL assegnati);
5) Progettare le funzioni di sicurezza in accordo con i requisiti assegnati;
6) Implementare le funzioni di sicurezza;
7) Validare le funzioni di sicurezza.

La norma introduce il "Ciclo di vita della sicurezza", come linea guida della sua applicazione. Il ciclo di vita della sicurezza divide il processo di sviluppo in fasi, prescrivendo per ciascuna fase quelle metodologie da applicare per far sì che si possa arrivare ad una certificazione del prodotto. Prescrive quindi, all'azienda che voglia implementare tale processo, di dotarsi di un sistema di gestione della sicurezza ed un sistema di gestione della qualità del processo.

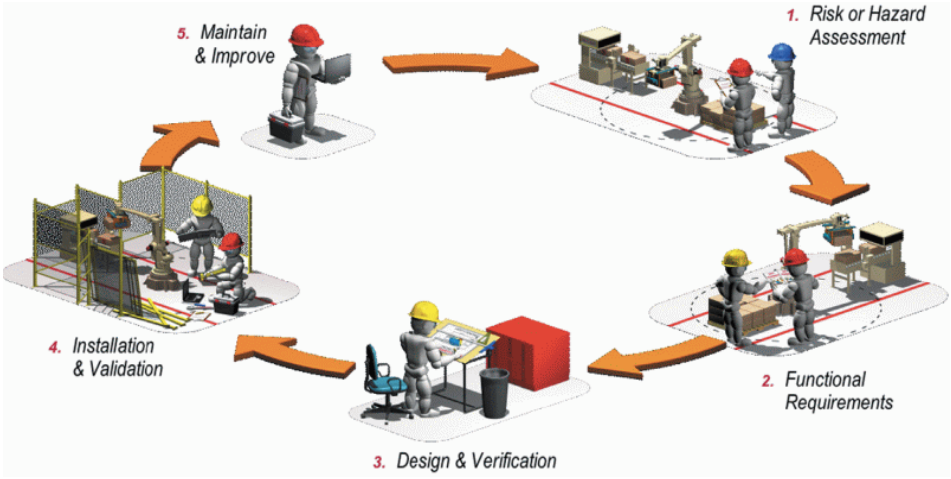

Figura 2.3: Esempio di ciclo di vita della sicurezza

Il Ciclo di vita della sicurezza (Safety life cycle), è così articolato:

1) Concezione generale del progetto della sicurezza funzionale. Gli obiettivi sono sviluppare un livello di comprensione del processo, del sistema di controllo del processo e del relativo ambiente, in maniera da consentire una conduzione soddisfacente delle attività a completamento del ciclo di vita; individuare le fonti di pericolo.
2) Scopo generale del progetto della sicurezza funzionale. Gli obiettivi per questo step sono: individuare le apparecchiature sotto controllo ed il relativo sistema di controllo da sottoporre ad analisi di rischio; definire scopo e tipologia dell'analisi; definire la tipologia dei vari eventi iniziatori, sia interni che esterni, delle sequenze incidentali;
3) Analisi di rischio. Bisogna identificare gli eventi incidentali e le apparecchiature coinvolte con i loro sistemi di controllo, in tutte le condizioni possibili conosciute, come modi operativi, guasto ed uso improprio; determinare la sequenza di eventi incidentali con la loro probabilità di accadimento calcolata in maniera qualitativa e quantitativa; determinare

i rischi associati agli eventi incidentali; valutare le misure da intraprendere per la riduzione del rischio necessaria.

4) Allocazione delle funzioni di sicurezza ai livelli di protezione indipendenti. Si allocano le funzioni di sicurezza per prevenire, mitigare e controllare le situazioni di rischio; si allocano i valori di riduzione del rischio come obiettivi da raggiungere; si analizzano gli errori di causa comune (common cause failures o CCF), cioè quelli che portano ad un annullamento della ridondanza.
5) Specifica dei requisiti di sicurezza. Si sviluppa la specifica dei requisiti di sicurezza e vengono allocati sui vari sottolivelli del sistema.
6) Si progetta e si sviluppa il sistema strumentale di sicurezza;
7) Esecuzione dei test di accettazione;
8) Si installa e si pone in servizio il sistema strumentale di sicurezza;
9) Si eseguono i Site Acceptance Test, ossia i test di accettazione per assicurarsi che effettivamente il sistema strumentale della sicurezza esegua le sue funzioni;
10) Operazione e manutenzione. Il sistema è operativo, e si eseguono su di esso tutti gli interventi manutentivi atti a mantenerlo in tale stato per tutto il periodo di vita del componente stesso.
11) Modifiche. È possibile che durante la vita operativa sia necessario apportare delle modifiche per mantenere il sistema aggiornato. Tali modifiche fanno sì che si ritorni indietro alla fase 1 e 2, e si riesegua tutto il processo integrando i nuovi aggiornamenti.
12) Dismissione. Si esegue un'analisi di dismissione, per assicurarsi che questa sia debitamente autorizzata, in maniera da non pregiudicare il funzionamento del sistema di sicurezza.

Il ciclo di vita qui è visto in forma del tutto generale, ma può essere relegato alla singola parte hardware, software o entrambe una volta avvenuta l'integrazione.

Bisogna porre l'attenzione sul fatto che la norma, tuttavia, non copre alcuni aspetti che, seppure non esattamente legati al progetto, sono di fondamentale importanza nelle pratiche di riduzione del rischio.

Il primo aspetto è il fattore umano. Serve ad indicare l'impatto dell'uomo sulle pratiche di sicurezza. Seppur l'avanzamento tecnologico ha portato ad una maggior autonomia dei sistemi, la supervisione umana è imprescindibile. Tuttavia questo introduce nuove variabili nel sistema di tipo esterno, che possono comportare situazioni di rischio molto diversificate. Quindi, nonostante la normativa non ne parli, è comunque un bene prendere in considerazione tale fattore ed inserirlo all'interno delle analisi di rischio.

I safety case sono il secondo aspetto non tenuto in conto. Un safety case è una dimostrazione logica, supportata da numerose evidenze, per il quale un sistema è sufficientemente sicuro da essere utilizzato in una determinata applicazione.

Riassumendo quanto è stato esposto finora: lo standard IEC 61508 è in primis uno standard di gestione della sicurezza funzionale. I sistemi di sicurezza strumentale vanno progettati modernamente in accordo a tale standard, da personale qualificato. La conformità a tale standard consente nel tempo di raggiungere e mantenere nel tempo gli obiettivi prestazionali prefissati per un sistema di sicurezza, salvaguardando vite umane, ambiente e salute.

2.1.2 La declinazione della IEC 61508 nelle varie discipline tecniche:

Nel precedente paragrafo è stato fatto un overview della normativa IEC 61508. Tale normativa, del tutto generale, detta le pratiche e le metodologie da seguire per ottenere un sistema sicuro. Data la generalità dei propri contenuti, ha fatto sì che gli enti normativi relativi alle diverse discipline tecniche, soprattutto nel campo dei trasporti, l'abbiano adattata alle proprie applicazioni specifiche. Ne sono nate così delle norme che si ispirano e richiamano la IEC 61508, pur declinandola in applicazioni più pratiche. La figura 1.1 esplica questo rapporto.

I concetti che vengono ripresi dalle normative ISO 26262, DO-178B e nella triade CEI EN 50126, 50128, 50129 sono essenzialmente gli stessi indicati all'interno della normativa, pur contenendo delle differenze relative allo specifico campo di applicazione.

La famiglia delle CEI EN si focalizza sull'ambito **ferroviario**, tranviario, filotranviario e metropolitane. La 50126 è una specifica delle applicazioni RAMS a tale ambito, introducendo dapprima tutti quei concetti utili all'analisi di affidabilità, disponibilità, manutenibilità e sicurezza del sistema ferroviario, parlando di tassi di guasto, modi di guasto, manutenzione del sistema e tutti gli altri aspetti prettamente facenti capo alla disciplina RAMS. Importante è l'attenzione che pone sull'identificazione dei requisiti in ambito RAMS, ed i mezzi per raggiungerli. In particolare, bisogna porre l'attenzione sulla classificazione dei guasti, intesa, in ambito ferroviario come:

- IMPORTANTE (o guasto immobilizzante): è un guasto che comporta l'arresto forzato del treno, e ritardi superiori ad un certo livello con conseguenti costi oltre la soglia prevista. Un esempio può essere il guasto che vedere un treno fermarsi in galleria.
- MAGGIORE (o guasto di servizio): tale guasto non per forza causa una immobilizzazione del mezzo, ma può portare a dei disservizi, con conseguenti riduzioni delle prestazioni. Tuttavia non per forza porta a ritardi e/o costi superiori ad un certo livello minimo.
- MINORE: un guasto che non ha grossi effetti sul sistema.

Questa parte della normativa riguarda anche la categorizzazione del rischio. In ambito ferroviario sono previsti 6 livelli di probabilità di accadimento:

- Frequente;
- Probabile;
- Occasionale;
- Remoto;
- Improbabile;
- Inverosimile.

Ci sono 4 livelli di severità del guasto:

- Catastrofico: più vittime in seguito al guasto, molte feriti gravi e/o danni maggiori all'ambiente;
- Critico: una vittima e/o lesioni gravi di una persona. Tale tipo di guasto porta alla perdita di un sistema principale;
- Marginale: ferite leggere e/o danni limitati all'ambiente;
- Insignificante: ferite leggere.

Dall'analisi RAMS si può calcolare la probabilità di accadimento, e legarla ad un adeguato livello di severità, in maniera da ottenere una matrice di rischio sulla riga di quella mostrata in figura 2.1. Partendo da questa matrice, bisogna identificare le pratiche necessarie alla riduzione del rischio o, laddove non si può avere il controllo diretto sugli eventi, accettarlo.

La normativa, fatte queste dovute introduzioni, si sposta adesso sui concetti della IEC 61508, che permettono di ottenere la riduzione del rischio attraverso adeguati sistemi di sicurezza, che guardano alla sicurezza funzionale. La sua applicazione si ritrova nelle CEI EN 50128 e CEI EN 50129, relative rispettivamente a metodi, strumenti e tecniche per lo sviluppo di sistemi software per la sicurezza, e l'approvazione di sistemi hardware e per il segnalamento ferroviario. L'obiettivo è quello di avere un sistema definito "sicuro al guasto". Un sistema sicuro al guasto è un sistema per il quale, in seguito al verificarsi di un evento di guasto, vede intervenire una ridondanza che ristabilisce il livello di sicurezza voluto. Ovviamente viene raccomandato di far svolgere questa funzione a due componenti indipendenti, di cui uno solo funzionante mi permette di rimanere nel livello di sicurezza voluto.

La CEI EN 50126 richiama il concetto di ciclo di vita del sistema, così come trattato nella IEC 61508, descrivendone tutte le varie fasi. La figura seguente mostra il ciclo di vita sottoforma di V-model, altra maniera rappresentativa dello stesso.

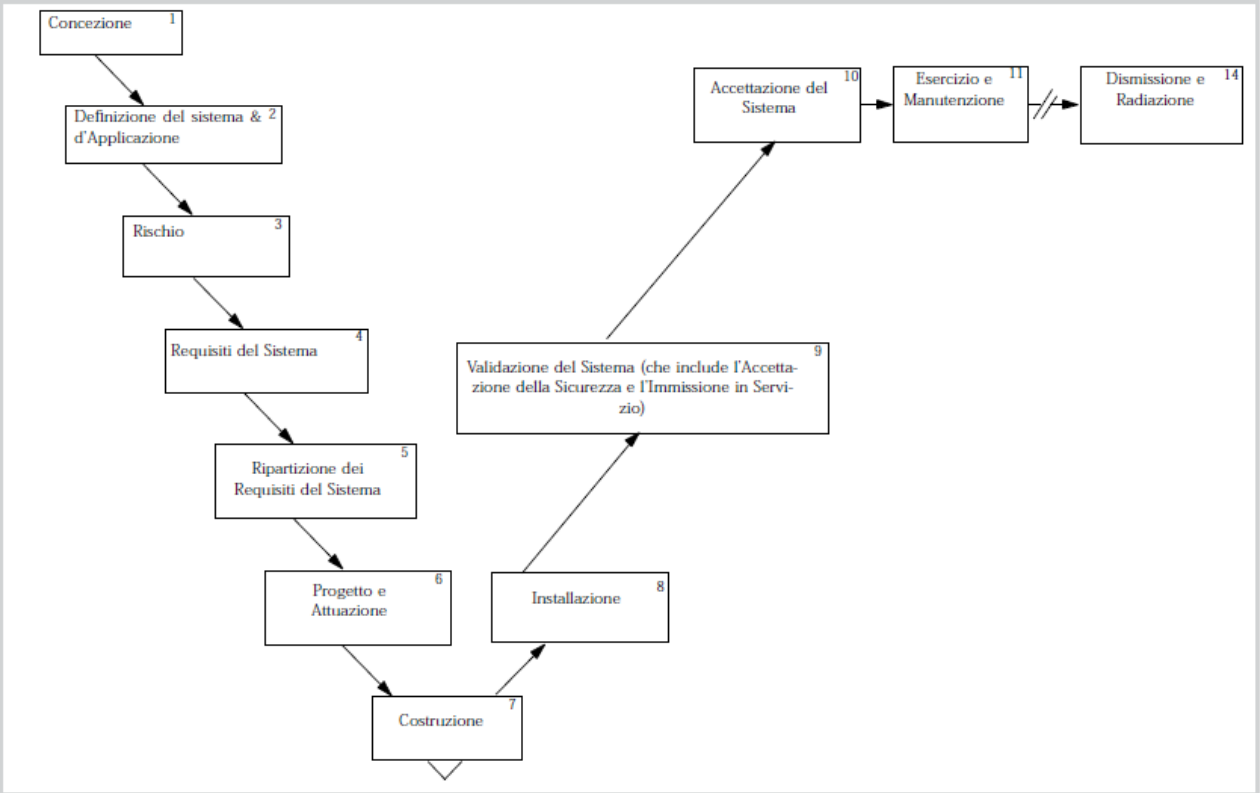

Figura 2.4: V-Model del ciclo di vita della sicurezza in ambito ferroviario [3]

Segue poi tutta la metodologia di specifica di requisiti sui vari aspetti del sistema ferroviario, in accordo con le fasi del ciclo di vita.

Questa norma allora declina, nella forma più generale, le richieste della IEC 61508.

Ricordando che la IEC 61508 riguarda tutti quei sistemi di sicurezza funzionale, la CEI EN 50128 riprende le sue direttive applicandole all'ambito software. Seppur risulti ridondante rispetto alla 50126 in alcune parti, approfondisce quei concetti che, volutamente, la 50126 lascia in sospeso o sui quali non pone maggiormente l'attenzione. Infatti, se quest'ultima rappresenta l'applicazione della IEC 61508 nella forma più generale, la 50128 scende più nel particolare, prendendo i requisiti esposti nella 50126 riportandoli al livello del software. Il ciclo di vita stesso è declinato sugli elementi del software, con un'opportuna specifica dei requisiti, e le metodologie per la definizione dell'architettura, progetto ed implementazione, verifica, integrazione e validazione. In seguito tratta anche le metodologie di manutenzione del software, oltre che la valutazione sulla qualità dello stesso. Si presenta allora come un insieme di pratiche per il progetto e lo sviluppo sicuro del software. Il concetto di SIL, già più volte richiamato, trova piena applicazione in tale norma, specificandone i livelli (5) e il livello di sicurezza associato. È importante che, una volta determinato il livello di SIL tramite analisi RAMS, venga stilata una specifica dei requisiti per ottenere tale livello di integrità.

La CEI EN 50129, riprende tutti i concetti già espressi, ma dal punto di vista hardware e dei sistemi di segnalamento ferroviario. Pone allora l'attenzione sui requisiti tecnici, identificazione dei modi di malfunzionamento, dopo ovviamente aver associato l'adeguato livello di SIL al componente. Una differenza rispetto alla precedente 50128 è la presenza delle condizioni di accettazione e approvazione della sicurezza. Rappresenta una lista di prove da effettuare sul componente in maniera da certificarne la gestione di qualità e sicurezza dal punto di vista funzionale e tecnico.

La IEC 61508, in ambito ferroviario, viene allora declinata in tre diverse norme, riguardanti i tre diversi aspetti del sistema, gestendo la stesura della specifica dei requisiti di sicurezza per tutto il progetto e suggerendo le pratiche per ottenere i livelli di sicurezza voluti. Le pratiche sono relative a tutti i punti del ciclo di vita della sicurezza come riportato nella IEC.

In ambito **automotive** la IEC 61508 viene relegata alla ISO 26262. L'impostazione è molto simile a quanto visto per la normativa ferroviaria, con la differenza che è possibile trovare tutte le pratiche in un'unica norma, divisa però in sezioni, dove ogni sezione riguarda un particolare aspetto del progetto. Si tratta di 10 sezioni, che vanno dalla parte introduttiva nella sezione 1, alla declinazione della IEC 61508, sulla falsariga di quanto visto per la normativa ferroviaria, nelle parti 4, 5 e 6, relative rispettivamente allo sviluppo al livello di sistema, hardware e software. Di particolare interesse, in quanto verrà richiamata in seguito nel presente elaborato, è la sezione 8, dove vengono definite tutte le pratiche accessorie, comprese quelle per la certificazione degli strumenti software/hardware usati nel ciclo di vita della sicurezza per sviluppare i sistemi o procedere alla loro validazione. Una parte apposita è invece riservata al concetto di ASIL, ossia il SIL intercalato in ambito automotive.

In base al livello di ASIL richiesto dall'applicazione, la normativa riporta delle pratiche raccomandate per raggiungerlo e testare l'effettivo raggiungimento del livello. Tali pratiche, che vengono raccomandate o altamente raccomandate.

Il rischio è valutato in questa norma con una scala, che classifica da S0 a S3 la severità dell'evento in base agli effetti sull'uomo, e da una scala della probabilità da E0 a E4 (da incredible a high probability).

Il ciclo di vita della sicurezza anche qui è centrale nello sviluppo del prodotto, e viene utilizzato come base per la stesura dei requisiti, individuandoli per ogni fase del ciclo di vita,

analogamente a quanto fatto per le CEI EN. Viene trattato nella parte 2 della normativa in forma generale. Ovviamente, data la diversità del sistema automotive da quello ferroviario, saranno diversi i requisiti e le tecniche di validazione usate, così come i livelli di sicurezza ricercati. L'impostazione comunque è la stessa, così come i livelli di severità della failure, relativi però al sistema automotive. La pratica per la determinazione del livello di ASIL viene eseguita nella stessa maniera, usando tecniche come FMEA, FMECA e FTA per calcolare la probabilità di guasto. Poi si parte dal sistema, dal ciclo di vita dello stesso per determinare i requisiti per la sicurezza in ogni fase. Tali requisiti vengono intercalati nella parte relativa all'hardware ed al software, le quali possiedono un proprio ciclo di vita separato e seguono una propria specifica di requisiti, stesa in accordo con questa normativa. La figura seguente esplica quanto riportato.

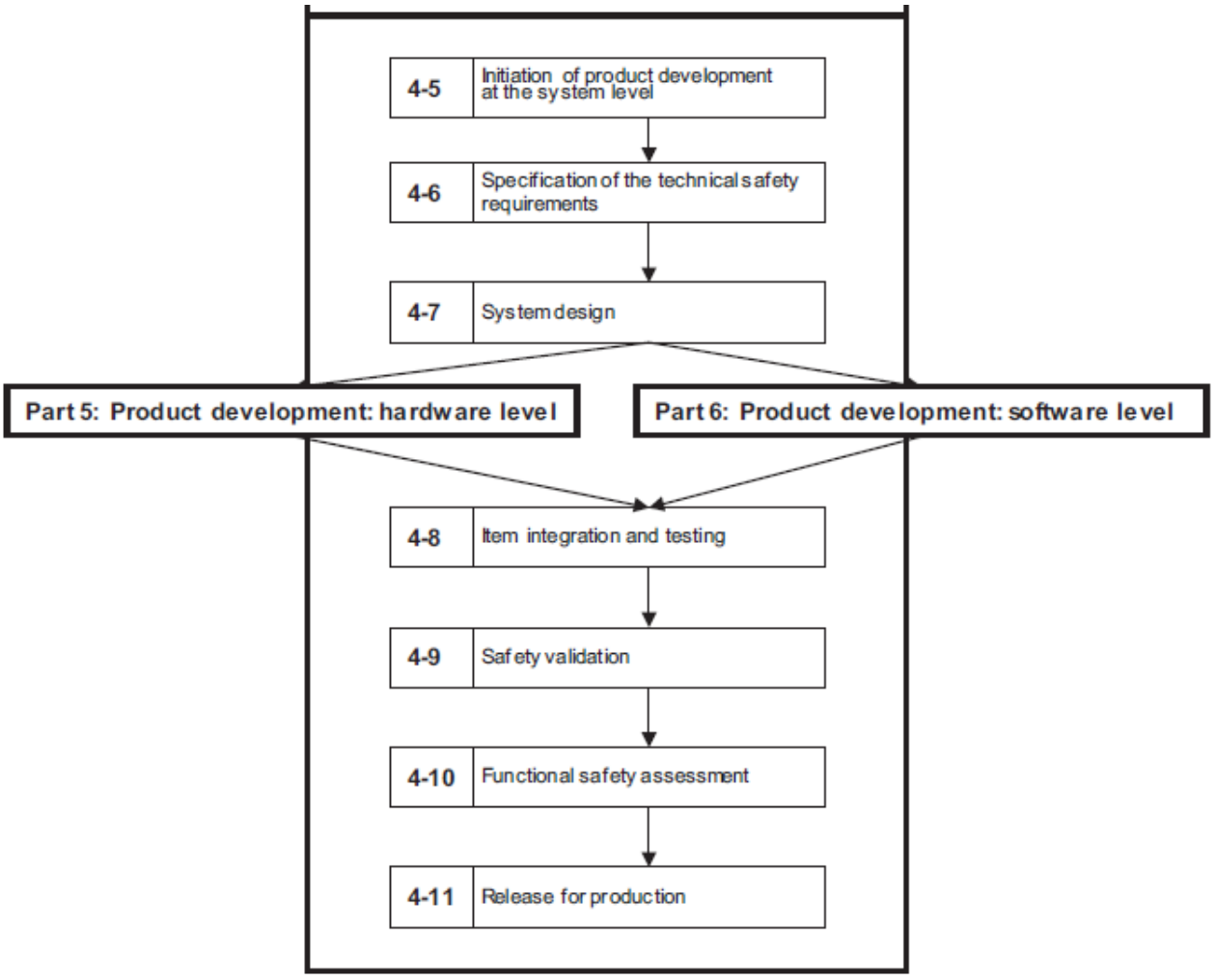

Figura 2.5: Particolare del ciclo di vita della sicurezza secondo ISO 26262 [5]

L'ultima declinazione della IEC 61508 esaminata nel presente elaborato è quella in ambito **aeronautico**, sottoforma della DO-178B. Questo è lo standard più vecchio tra quelli considerati, essendo il problema della sicurezza molto sentito in ambito aeronautico sin dagli albori dell'aviazione commerciale nel secondo dopoguerra. Una differenza che salta subito all'occhio rispetto alle due precedenti normative è l'assenza di una sezione specifica per l'hardware: questa normativa pone principalmente l'attenzione sui software definiti safety

critical, che gestiscono funzioni critiche del velivolo. Queste funzioni possono andare da quelle relative alla diagnostica del sistema, a quelle dei computer deputati alla guida e controllo del velivolo. È uno standard molto stringente che punta molto, rispetto alle norme precedenti, sulla tracciabilità di ogni fase del processo del ciclo di vita del software.

Il ciclo di vita del software è centrale, analogamente alle norme precedenti e come voluto dalla IEC 61508.

Il processo di sviluppo del software contenuto all'interno di questa normativa è diviso in 3 fasi: una prima fase nella quale viene stesa la specifica dei requisiti, una seconda fase riguarda il processo di design del software e la terza fase che riguarda la scrittura del codice e l'integrazione. L'obiettivo è quello di assicurare la correttezza, il controllo e la confidenza del software in tutto il ciclo di vita del prodotto. La normativa si focalizza sui possibili errori che si possono incontrare durante tutta la stesura del codice e nella fase di testing, suggerendo delle pratiche su come evitare tali problemi.

Il rischio, in questa normativa, è valutato in base al livello del software. Viene prima identificata la severità della failure, in accordo a quanto definito dagli enti normativi aeronautici. I livelli sono:

- Catastrophic: perdita totale del velivolo, impossibilità di continuare il volo o atterrare, numerose vittime;
- Hazardous/Severe-Major: failure che comporta la riduzione delle capacità dell'aeromobile, o dell'equipaggio, nel continuare la missione. In particolare può causare un aumento estremo del carico di lavoro per l'equipaggio, riduzione elevata del margine di sicurezza e la possibilità per i passeggeri di riportare ferite gravi o la morte stessa.
- Major: failure che aumenta il carico di lavoro dell'equipaggio, comporta una riduzione del margine di sicurezza, causa una riduzione del comfort dei passeggeri con conseguente pericolo di feriti.
- Minor: failure che non pregiudica significativamente le capacità dell'aeromobile, causa un leggero aumento del carico di lavoro per l'equipaggio e ha impatto solo dal punto di vista economico (ritardi/cancellazione del volo).
- No effect: nessun effetto visibile.

Sulla base di queste informazioni, si stabilisce il livello del software, da A ad E. Analogamente a quanto fatto dalla norma automotive e differentemente rispetto a quella ferroviaria, come per l' ASIL non si associa un vero e proprio valore di probabilità, il livello del software è scelto in base alla probabilità di accadimento ed all'esito della failure. Nel paragrafo precedente questa distinzione sarà approfondita.

La normativa, rispetto alle precedenti, mostra quindi un'impostazione leggermente diversa, anche dovuta alla sua già citata anzianità. Ma non è il solo motivo in quanto l'applicazione delle norme che riguardano la sicurezza funzionale, si sono applicate dopo, e su impulso dell'evoluzione tecnologica a tutti quei sistemi di trasporto in cui prima non erano adeguatamente considerate.

Modernamente, l'uso della IEC 61508 e sue derivate, ha portato quindi al centro la sicurezza dei sistemi. Come si è potuto notare, salvo alcune differenze dovute all'anzianità delle norme

ed allo specifico campo di applicazione, nonché all'ente normativo che le ha emanate, i principi sono tutti gli stessi, e possono essere schematizzati nel seguente modo:

- Ciclo di vita della sicurezza. Che sia relativo al software, all'hardware o al sistema, il principio applicativo è uguale. Dalla fase di concezione, design e sviluppo, sino alla fase di dismissione, bisogna avere una specifica dei requisiti ai quali attenersi.
- Sulla base delle analisi di rischio si determina il SIL, ASIL o Software Level, per stilare quei requisiti di sicurezza che poi dovranno essere validati in accordo con il livello scelto.
- Disposizioni delle pratiche ritenute sicure per il progetto. Si punta in particolar modo sulla metodologia di progetto.
- Produzione di tutta la documentazione che certifica la sicurezza del sistema, con particolare interesse per la tracciabilità in ambito aeronautico.

2.2 I PARAMETRI DI INTEGRITÀ DELLA SICUREZZA:

In questo paragrafo saranno esaminati più nel dettaglio i parametri SIL, ASIL e livello del software (che da adesso, per semplicità, sarà citato come SL, Software Level), per descrivere meglio il loro utilizzo nel progetto e le implicazioni di tale uso. Si effettuerà un confronto tra i tre, mettendo in evidenza come concettualmente siano la stessa cosa, portando però a valutazioni volutamente su scale differenti, in base all'ambito di utilizzo.

2.2.1 SIL:

Il SIL, letteralmente *Software Integrity Level* (livello di integrità della sicurezza), indica la probabilità che un sistema di sicurezza fallisca nell'effettuare correttamente le sue funzioni in tutte le condizioni previste. Il SIL assume quindi il valore di parametro di progetto, in quanto, una volta ottenuto, tutto il progetto deve essere impostato per raggiungere tale valore. Si è già detto come questo influenzi i requisiti specifici per hardware e software, e come anche la validazione di questi ultimi sia legata al livello di SIL.

Il SIL, inteso nella IEC 61508 e nella CEI EN 50126, si divide in 4 livelli, da 4 (il massimo) a 1 (il minimo) (0 è presente solo nella CEI EN 50127), e a ciascuno di questi livelli si associano delle pratiche diverse in maniera da ottenerne il raggiungimento. In particolare, per il livello 0, non si associa nessuna prescrizione particolare, non essendo il sistema considerato critico per la sicurezza. A tal proposito, la normativa CEI EN 50126 riporta:

"*Un SIL deve essere assegnato solo ad un "elemento", chiamato equipaggiamento isolato che svolge una o più semplici funzioni e che può essere sostituito da un altro che svolge la stessa/le stesse funzione/i. Generalmente, tale "elemento" è spesso l'equipaggiamento di più basso livello che può essere sostituito durante una operazione di manutenzione correttiva di primo livello.*" [3].

"*Un SIL riguarda solo un livello atteso di fiducia nella sicurezza per un prodotto. […]. I requisiti di sicurezza e i requisiti di disponibilità sono interdipendenti nel contesto del trasporto ferroviario. Il concetto dei SIL non copre tutti gli aspetti di un sistema e perciò*

considerare solo i SIL può non essere sufficiente (p.e. modi di funzionamento degradato o stati di emergenza con differenti requisiti di sicurezza, etc.)." [3]

Spesso il livello di SIL determina anche la ridondanza da adottare.

La normativa IEC 61508 determina i livelli di SIL in base al tipo di componente, che sia questo Low Demand Mode, cioè che il suo intervento non sia continuo nel tempo, ma solo ad intervalli medio lunghi, e Continuous/High Demand Mode, cioè il suo funzionamento deve essere garantito in continua o con una frequenza elevata.

Safety Integrity Level	Low Demand Mode of Operation
4	$>= 10^{-5}$ to 10^{-4}
3	$>= 10^{-4}$ to 10^{-3}
2	$>= 10^{-3}$ to 10^{-2}
1	$>= 10^{-2}$ to 10^{-1}

Figura 2.6: Livelli di SIL per applicazioni Low Demand [1]

Safety Integrity Level	Continuous/High Demand Mode of Operation
4	$>= 10^{-9}$ to 10^{-8}
3	$>= 10^{-8}$ to 10^{-7}
2	$>= 10^{-7}$ to 10^{-6}
1	$>= 10^{-6}$ to 10^{-5}

Figura 2.7: Livelli di SIL per applicazioni Continuous/High-Demand [1]

Una volta che vengono effettuate le analisi di sicurezza su un particolare componente, in accordo con la CEI EN 50126, è possibile assegnare il corretto livello di SIL, e formulare la specifica dei requisiti di sicurezza per quel determinato livello. L'integrità della sicurezza può essere vista in maniera sia quantitativa, come riportata sopra, che qualitativa (come fatto ad esempio dalla normativa aeronautica che sarà esaminata successivamente). Quantitativamente è possibile analizzare le componenti hardware, affetti da un modello di guasto casuale, mentre qualitativamente è possibile l'analisi del software, a cui si associano guasti ed errori di tipo sistematico, provenienti dalla programmazione.

L'integrità della sicurezza può differire da applicazione ad applicazione, di conseguenza è bene anche tenere in conto questo aspetto quando viene effettuata la stima. È possibile che l'assegnazione di un certo SIL ad una centralina, posta all'interno del sistema con una certa funzione, vari se tale centralina viene presa al di fuori del sistema considerato.

Si è detto in precedenza che il livello di SIL influenza anche il processo di valutazione e validazione dei requisiti. In particolare indica chi deve essere preposto a tale attività, distinguendo tra progettista/implementatore, verificatore e validatore come da CEI EN 50128.

La normativa dice che il valutatore, deve comunque essere riconosciuto dall'ente deputato al controllo della sicurezza, e non deve appartenere al fornitore, per ragioni di indipendenza. Il progettista/implementatore, verificatore e valudatore possono appartenere tutti alla medesima società.

Le regole da seguire in tale fase, in base al livello di SIL assegnato sono le seguenti:

- SIL 0: nessuna prescrizione. Progettista/implementatore, verificatore e validatore possono essere la medesima persona;
- SIL 1 e 2: Progettista/implementatore diverso da verificatore e validatore, che possono essere la medesima persona;
- SIL 3 e 4: Progettista/implementatore diverso da verificatore e validatore, che possono essere la medesima persona. Inoltre il verificatore ed il validatore non devono relazionare mediante il project manager.

La figura seguente mostra i passaggi principali legati all'analisi di rischio e all'assegnazione del SIL.

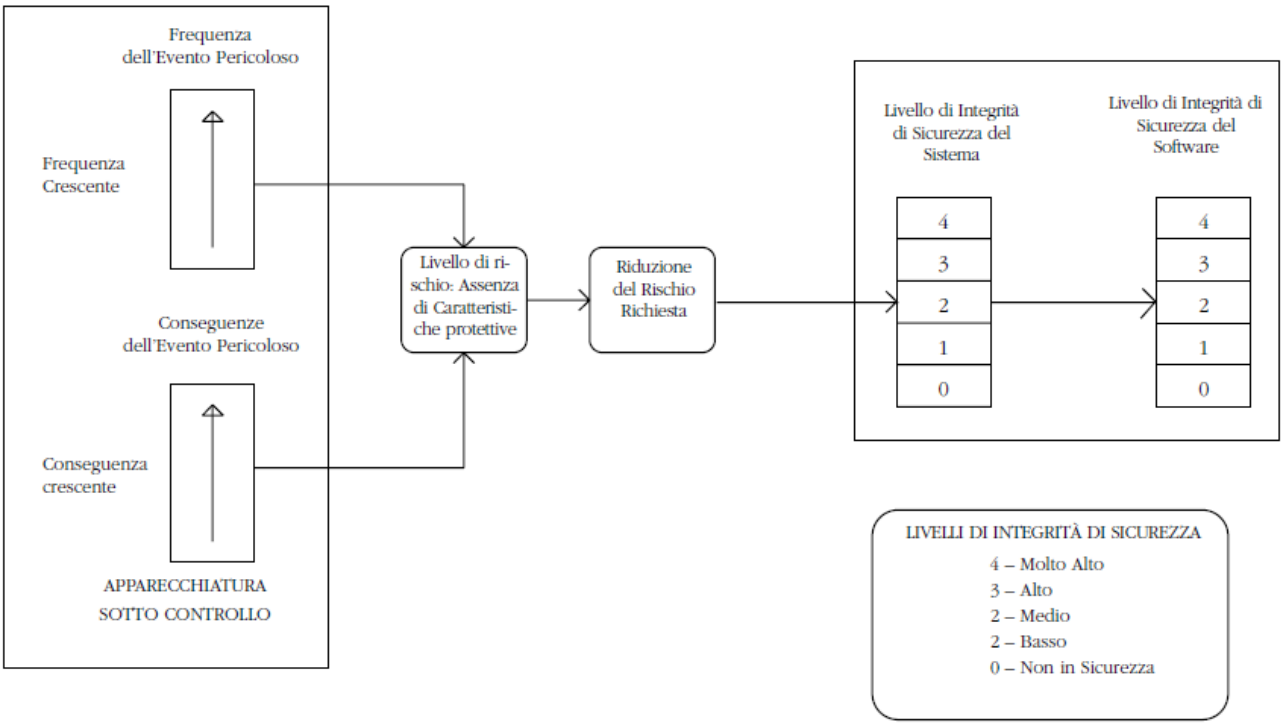

Figura 2.8: Descrizione di come si perviene alla scelta del SIL secondo CEI EN 50128 [4]

2.2.2 ASIL:

L'ASIL, *Automotive Safety Integrity Level,* è il corrispettivo del SIL in ambito automotive, e ne rispecchia le funzioni. Tuttavia, dato l'ambito diverso di applicazione, vi sono delle differenze, sia nel come viene specificato, sia di come influenza la scrittura dei requisiti di sicurezza. La norma ISO 26262 è quella relativa al campo automotive.

La ISO 26262 riporta quattro livelli di ASIL, da A (livello minimo) a D (livello massimo). Come per la norma precedente, l'ASIL interviene nel processo di stesura della specifica dei requisiti, e nelle procedure di valutazione/validazione.

L'ASIL si determina a partire dalle analisi di rischio, su componenti potenzialmente pericolosi per l'incolumità se non dovessero compiere le funzioni di sicurezza alle quali sono deputati. Assegnare un certo ASIL vuol dire stabilire un obiettivo di sicurezza da raggiungere. Il raggiungimento degli obiettivi di sicurezza deve poi essere certificato da una valutazione o validazione a posteriori sul rispetto della specifica dei requisiti.

Richiamando quanto riportato al punto 2.1.2 circa la ISO 26262, sulla classificazione del rischio, la scelta dell'ASIL è guidata nella seguente maniera:

- Determinazione del livello di severità dell'evento di failure del sistema o componente:

	Class			
	S0	**S1**	**S2**	**S3**
Description	No injuries	Light and moderate injuries	Severe and life-threatening injuries (survival probable)	Life-threatening injuries (survival uncertain), fatal injuries

Figura 2.9: Descrizione del livello di severità secondo ISO 26262-3 [6]

- Determinazione della probabilità che tale evento avvenga:

	Class				
	E0	**E1**	**E2**	**E3**	**E4**
Description	Incredible	Very low probability	Low probability	Medium probability	High probability

Figura 2.10: Descrizione della probabilità di accadimento di un evento secondo ISO 26262-3 [6]

- Scelta la classe di controllabilità dell'evento:

	Class			
	C0	**C1**	**C2**	**C3**
Description	Controllable in general	Simply controllable	Normally controllable	Difficult to control or uncontrollable

Figura 2.11: Classificazione della controllabilità dell'evento secondo ISO 26262-3 [6]

La classe di controllabilità dell'evento indica quanto questo possa essere controllato dal guidatore o qualsiasi altra persona potenzialmente a rischio. La classe C0 è assegnata quando la perdita di controllo su quell'elemento non causa rilevanti problemi per la sicurezza, mentre la classe C3 è assegnata quando la perdita di controllo sull'elemento causa seri pericoli per l'incolumità. In particolare, all'elemento che presenta classe C0, non deve essere assegnato nessun livello di ASIL.

Sulla base di quanto riportato nelle figure 2.9, 2.10 e 2.11, si fa una valutazione incrociata dell'impatto sulla sicurezza di quel sistema o componente, e viene assegnato un certo livello di ASIL. Bisogna evidenziare che in ambito automotive il controllo qualità ha maggior peso di quanto lo abbia in ambito ferroviario, circa le normative che si ispirano alla IEC 61508. Di conseguenza, nel caso in cui il rischio associato ad un certo evento è basso, si può evitare l'assegnazione di un livello di ASIL in favore di un semplice controllo di qualità (Quality Management QM).

La valutazione incrociata porta ad una tabella, che permette di scegliere il livello di ASIL corretto per quell'applicazione. Il tutto è riportato nella figura seguente.

Severity class	Probability class	Controllability class		
		C1	C2	C3
S1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	A
	E4	QM	A	B
S2	E1	QM	QM	QM
	E2	QM	QM	A
	E3	QM	A	B
	E4	A	B	C
S3	E1	QM	QM	A
	E2	QM	A	B
	E3	A	B	C
	E4	B	C	D

Figura 2.12: Assegnazione ASIL [6]

A differenza di quanto visto per la normativa CEI EN, l'assegnazione dell'ASIL non si basa su un parametro numerico che indichi una probabilità di accadimento, ma su una scala che studia i rapporti causa-effetto, circa il potenziale evento di failure. Analogamente a quanto avviene sempre per la citata CEI EN, determinato il livello di ASIL, si procede con la stesura della specifica dei requisiti, in accordo con il rispetto di tale livello.

L'ASIL determina anche le pratiche e le persone coinvolte nel processo di valutazione e validazione dei requisiti. La normativa, nella sezione ISO 26262-2, riporta chi sono le entità coinvolte in tale processo, e quali devono essere i loro rapporti. La scelta viene fatta anche in base al particolare tipo di analisi, review, assessment o misura da adottare. Le categorie assegnate dalla simbologia usata sono:

- —: nessun requisito o raccomandazione da usare riguardo la conferma di questa misura;
- I0: Dovrebbe essere intrapresa una misura di conferma. Se si decide si intraprenderla, deve essere effettuata da persone differenti;
- I1: La misura di conferma deve essere intrapresa, ed effettuata da persone differenti;
- I2: La misura di conferma dovrebbe essere intrapresa, ed effettuata da persone di team differenti;
- I3: La misura di conferma dovrebbe essere intrapresa, ed effettuata da persone appartenenti a differenti dipartimenti o organizzazioni.

Questa distinzione, in accordo con il livello di ASIL scelto, consente la massima indipendenza di pensiero in maniera da intraprendere tutte quelle misure che portino alla verifica del rispetto dei requisiti di sicurezza. Per maggiore chiarezza si consiglia di consultare la ISO 26262-3:2011 (E) a pag 15.

2.2.3 SOFTWARE LEVEL:

Il software level si distacca dalla definizione propria di SIL o ASIL, pur riprendendone i concetti che ne stanno alla base.

La normativa aeronautica riporta tale scala di valutazione, in base alla criticità del software, senza assegnare nessuna scala di failure rate a cui rifarsi. Riprende la stessa concezione che era stata data all'ASIL. Si basa allora su delle valutazioni puramente qualitative in base alle funzioni che è chiamato a svolgere il software, e come queste incidono sulla sicurezza del velivolo. In base ai rapporti di causa-effetto, viene data una classificazione al software come software level, e ci si deve assicurare, durante la fase di sviluppo, che il software abbia un grado di maturità tale da assicurare il livello assegnato.

La gravità di un evento di failure del sistema, ha portato a stilare la seguente scala di effetti, in accordo con i regolamenti FAA AC-25-1309-1° oppure JAA AMJ 25-1309:

- CATASTROPHIC: condizione di failure che causa la perdita del velivolo e la morte dei suoi occupanti, oppure l'impossibilità di continuare il volo o atterrare;
- HAZARDOUS/SEVERE-MAJOR: evento di failure che reduce la capacità dell'aeromobile o del suo equipaggio a far fronte a condizioni avverse generate da tale evento. Altre conseguenze riguardano: riduzione del margine di sicurezza o capacità funzionali, stress fisico o aumento del carico di lavoro per l'equipaggio tale da non consentire di portare a termine in maniera completa i task assegnati, condizioni avverse sugli occupanti che possono portare a ferite fatali ad un numero limitato di questi;
- MAJOR: condizione di failure che porta ad un aumento del carico di lavoro per l'equipaggio o ad una riduzione delle capacità dell'aeromobile di far fronte alle condizioni avverse generate da tale evento. Altre conseguenze riguardano: aumento del carico di lavoro per l'equipaggio riducendone l'efficienza, una riduzione del margine di sicurezza, perdita di comfort per gli occupanti compresa la possibilità di ferite;
- MINOR: condizione di failure che non riduce in maniera significativa il margine di sicurezza del velivolo, e che comporta un intervento dell'equipaggio all'interno delle sue capacità. Comporta inoltre un leggero aumento del carico di lavoro, oltre quello normalmente sostenuto, e fastidio per gli occupanti.
- NO EFFECTS: non si hanno effetti sull'aeromobile, sull'equipaggio o i passeggeri.

La definizione del Software Level si basa sul contributo del software al potenziale evento di failure, ed il grado assegnato dipende dagli effetti in accordo con la scala sopra enunciata. Gli effetti che si ottengono in seguito ad un suo malfunzionamento derivano dalle analisi di rischio effettuate. Questo influenza anche i requisiti e la loro allocazione. I livelli del software sono:

- LEVEL A: Software il cui funzionamento anomalo comporta o contribuisce ad una condizione di failure di tipo catastrophic per l'aeromobile;

- LEVEL B: Software il cui funzionamento anomalo comporta o contribuisce ad una condizione di failure di tipo hazardous/severe-Major per l'aeromobile;
- LEVEL C: Software il cui funzionamento anomalo comporta o contribuisce ad una condizione di failure di tipo major per l'aeromobile;
- LEVEL D: Software il cui funzionamento anomalo comporta o contribuisce ad una condizione di failure di tipo minor per l'aeromobile.
- LEVEL E: Software il cui funzionamento anomalo comporta o contribuisce ad una condizione di failure di tipo no effects per l'aeromobile. In questo caso non sono necessarie le linee guida contenute nella normativa.

Inizialmente i livelli vengono associati a parti del software in seguito alle verifiche di rischio effettuate. In genere, nel caso di failure di tipo multiplo, nel caso di implementazione delle misure di sicurezza di tipo parallelo o sequenziale, si associa a tutto il software il livello relativo alla condizione di failure con effetti peggiori.

2.2.4 CONSIDERAZIONI FINALI:

Dal confronto tra le tre diverse normative, e la classificazione del livello di sicurezza che esse danno alle applicazioni di tipo software o hardware, si evidenzia come vi siano delle differenze evidenti tra di queste, pur mantenendo lo stesso principio di fondo: il livello di SIL, ASIL o SL è assegnato in base agli effetti che l'evento di failure ha sul sistema.

L'unica normativa tra le tre, a considerare in maniera trasversale sia software che hardware è la CEI EN 50126-50128-50129, a cui viene associato un livello di SIL basato su un tasso di guasto che, nel caso del livello massimo assegnato (SIL 4), prevede una failure nell'arco di anche migliaia di anni. Questa normativa quindi detta delle precise linee guida, e aiuta nella formulazione dei requisiti di sicurezza e nella loro allocazione in maniera più efficace rispetto ai corrispettivi automotive o aeronautica. Di contro, l'assegnazione del SIL comporta l'esecuzione sin dalle prime fasi del ciclo di vita di analisi quantitative, volte a stabilire il rispetto o l'assegnazione del SIL a quel particolare sistema/sottosistema.

La normativa automotive ISO 26262 dà un approccio qualitativo all'assegnazione dell'ASIL. Prevedendo un procedimento che, tramite l'uso di tabelle e lo studio degli effetti che un malfunzionamento ha sugli occupanti del veicolo, permette di individuare il livello di ASIL da assegnare al sistema. Questo porta alla formulazione dei requisiti di sicurezza sul sistema e alla loro allocazione sui sottosistemi con l'obiettivo di ottenere il livello richiesto.

La normativa aeronautica, invece, non si preoccupa, a differenza delle precedenti, dell'hardware, per il quale però prevede delle valutazioni di rischio e affidabilità al pari delle colleghe, ma si interessa del software. Il software, in base al grado di incidenza che ha sulla sicurezza del velivolo, e a quanto un suo malfunzionamento incide su un evento di failure, si vede assegnato in modo qualitativo un livello, che ne indica la criticità, e che di conseguenza obbliga ad applicare delle tecniche opportune che ne certifichino la sicurezza.

Quindi, dal confronto delle tre emerge che le scale e i metodi di valutazione sono diversi, in base all'ambito di applicazione. Anche la severità delle procedure applicabili in seguito alla determinazione del livello di SIL, ASIL o SL è diversa, pur avendo una base comune che lega un possibile malfunzionamento alle conseguenze che questo ha su sistema ed occupanti del sistema.

CAPITOLO 3. CERTIFICAZIONE DEL TOOL

In questo capitolo ci si concentrerà sull'elemento centrale del presente elaborato. Nei primi due capitoli si è data una breve introduzione a quelli che sono gli argomenti trattati e alla normativa functional safety di riferimento, concentrandoci sulle tre applicazioni tecniche della IEC 61508 che sono state ritenute più importanti, anche se sarebbe riduttivo riferirsi soltanto alle seguenti per l'analisi della sicurezza funzionale.

Una volta che è chiaro cosa richiedono le normative e come vengono classificate le applicazioni hardware e software sensibili per la sicurezza, ci si concentrerà su tutti quegli strumenti che concorrono alla fase di sviluppo dei sistemi di tipo safety critical, sia dal punto di vista progettuale, che della sicurezza. Il motivo è che tali strumenti – o tool come saranno chiamati nel seguito – possono portare all'interno del sistema degli errori di tipo non casuale, ma sistematici, ossia dipendenti da chi esegue la programmazione di un software ad esempio, che introducono quindi dei malfunzionamenti che facilmente, e senza le dovute attenzioni, possono sfuggire anche ad un occhio più esperto, riducendo i margini di sicurezza dei sistemi.

Nonostante sia quindi necessario provvedere ad uno studio di tali tool e ad una loro certificazione/qualifica, la IEC 61508 non dice nulla al riguardo, attenzionando soltanto la possibilità della presenza di tali errori. Così fa anche la CEI EN 50126-50128-50129.

Le uniche due normative dove si parla apertamente di tool utilizzati nel ciclo di vita della sicurezza, e ne dettano le linee guida per una loro certificazione/qualificazione sono la ISO 26262-8 e la DO-178B, con un apposito paragrafo dove sono riportate le linee guida da seguire per avere un tool adatto alle applicazioni di sicurezza che competono a tali normative.

Negli anni alcune software houses si sono adattate a tali normative, producendo dei software tool utili nei più svariati ambiti dello sviluppo e delle analisi dei progetti, ottenendo anche certificazioni da parte di enti autorizzati, come la TUV. Questo dimostra come ci si stia sempre più orientando anche in tal senso, producendo prodotti all'avanguardia che consentano l'esecuzione di analisi col minimo pericolo di errori sistematici.

Nel seguito saranno allora analizzati i capitoli facenti parte dalla ISO 26262-8 e della DO-178B, mettendo anche in evidenza le principali differenze.

3.1 COSA CHIEDE LA ISO 26262:

La prima normativa che sarà analizzata è quella di tipo automotive. Si parla di certificazione del tool nella sezione 8 (Supporting processes) al capitolo 11 (Confidence in the use of software tools).

Detta le linee guida per arrivare ad una certificazione del tool, cioè tutte quelle procedure che certificano che il software può essere usato per analisi durante il processo di sviluppo di un sistema, poiché in accordo con gli standard di tale normativa. Questo fa sì che, una volta certificato, il tool di tipo software, potrà essere utilizzato per tutte le applicazioni simili.

Un esempio di software house dedicatasi alla progettazione ed allo sviluppo di un tool da usare durante le fasi di progetto è quello della Isograph, con il software per fault tree analysis FaultTree+, certificato secondo ISO 26262. All'indirizzo web https://www.isograph.com/software/reliability-workbench/fault-tree-analysis-software/ è possibile trovare il certificato.

La normativa indica due obiettivi da perseguire: il primo obiettivo è determinare il livello di confidenza richiesto nei software tools utilizzati; il secondo obiettivo è trovare una maniera per qualificare il software tool in accordo con le analisi da eseguire secondo normativa ISO 26262.

La normativa dice che un tool usato per lo sviluppo di un sistema software o hardware deve supportare un sufficiente adattamento al safety life cycle ed ai task assegnati dalla ISO 26262. In questi casi la confidenza è orientata sul raggiungere effettivamente gli obiettivi richiesti dalla normativa. In particolare si vuole diminuire il rischio di fault sistematici nel prodotto sviluppato in seguito all'uso di un software, e legati ad output errati. Inoltre il processo di sviluppo viene ritenuto adeguato con rispetto della ISO se le attività e i task richiesti dalla norma si basano sul corretto funzionamento del software.

Per determinare il corretto livello di confidenza del tool usato, devono essere valutati i seguenti criteri:

- La possibilità che un malfunzionamento del tool ed il conseguente output errato possono introdurre fault o sbagliare nel rilevare errori negli elementi di sicurezza relativi ad esso o a quelli in fase di sviluppo;
- La confidenza nella prevenzione o nel rilevare questi errori ed il loro corrispondente output.

Per valutare il livello di confidenza richiesto, misure interne al software come monitoraggio, o esterne (linee guida, test, review), implementate nel processo di sviluppo, devono essere considerate e valutate.

Il capitolo poi riporta dei paragrafi, in particolare relativi ai requisiti che dovrebbero essere presenti nella specifica dei requisiti del tool.

- **11.4.1**: è il requisito generale. Dice che se il software ha un impatto sul progetto dell'hardware o del software, che risponde ai requisiti dettati dalla ISO 26262, e le analisi si basano sui suoi risultati, e gli output non sono stati esaminati o verificati, allora bisogna far sì che il software rispetti le linee guida contenute in tale norma.
- **11.4.2**: Se questo software è stato sviluppato indipendentemente da un utilizzo particolare, il livello di confidenza o qualifica dovrebbe essere confermato con quanto richiesto dalla ISO 26262-2 nella tabella 1, prima che tale tool venga usato.

Confirmation review of the software tool criteria evaluation report and the software tool qualification report[b] (see ISO 26262-8:2011, Clause 11) Independence with regard to the persons performing the qualification of the software tool	—	I0	I1	I1	Applies to the highest ASIL of the requirements that can be violated by the use of the tool

Figura 3.1: estratto dalla Tabella 1 della ISO 26262-2

Confirmation review of the proven in use arguments (analysis, data and credit), of the candidates (see ISO 26262-8:2011, Clause 14) Independence with regard to the author of the argument	I0	I1	I2	I3	Applies to the ASIL of the safety goal or requirement related to the considered behaviour, or function, of the candidate

Figura 3.2: estratto dalla Tabella 1-continua della ISO 26262-2

- **11.4.3**: quando si usa questo tool, bisogna assicurarsi che sia qualificato per l'ambiente di utilizzo e il suo scopo.
- **11.4.4**: l'utilizzo del tool deve essere pianificato, tenendo in conto diverse voci, in particolare quella che fa riferimento al metodo di qualifica adottato dal software, in maniera che sia rispondente all'utilizzo che se ne deve fare.

Per assicurarsi il corretto uso del tool, bisogna avere a disposizione la seguente documentazione:

- Descrizione di features, funzioni e proprietà;
- Manuale di guida all'utilizzo;
- Descrizione dell'ambiente di utilizzo;
- Descrizione del comportamento atteso dal tool sotto certe condizioni anomale;
- Una descrizione dei malfunzionamenti conosciuti e appropriate misure di contrasto;
- Le misure per individuare i malfunzionamenti o gli output errati identificati durante la determinazione del livello di confidenza del software.

Quindi si richiede un manuale del software che riporti tutte le voci sopra riportate.

Ci si concentra ora sulla valutazione del tool, effettuandone l'analisi. Questa analisi si basa sulla scelta di alcuni parametri, analogamente al calcolo dell'ASIL, in maniera da ottenere il livello di confidenza del tool.

- Determinazione del **Tool Impact (TI)**: è la possibilità che il software utilizzato possa introdurre errori oppure fallisca nel rilevarli nel prodotto che si sta sviluppando.
  - TI 1 è usato quando non esiste tale possibilità;
  - TI 2 è usato in tutti gli altri casi.
- Determinazione del **Total error Detection (TD)**: rappresenta la confidenza nella misura che il software, in seguito ad un malfunzionamento, produca un corrispondente output errato. Vi sono i seguenti livelli;
  - TD 1: alta confidenza nel fatto che il malfunzionamento ed il corrispondente output errato siano riscontrati e vi si possa porre rimedio;
  - TD 2: media confidenza nel fatto che il malfunzionamento ed il corrispondente output errato siano riscontrati e vi si possa porre rimedio;
  - TD 3: in tutti gli altri casi. In particolare questo valore viene scelto se non vi sono misure sistematiche nel processo di sviluppo disponibile, ed i malfunzionamenti, con i rispettivi output errati, vengono riscontrati randomicamente. Se il tool è usato per verificare l'output di un altro tool, l'interdipendenza tra questi è usata per scegliere il TD.

Ne caso non fosse chiaro quale TI o TD adottare, si consiglia di mantenersi conservativi.

Dall'unione di TI e TD si sceglie il **Tool Confidence Level (TCL)** come da tabella:

		Totale error Detection		
		TD1	**TD2**	**TD3**
Tool impact	**TI 1**	TCL1	TCL1	TCL1
	TI 2	TCL1	TCL 2	TCL 3

Esistono diversi metodi di certificazione in base al TCL considerato. In particolare, se TCL 1 non è necessario intraprendere alcuna misura.

La tabella presente nell'immagine seguente mostra, nel caso di TCL 3, quali sono le misure da adottare. In realtà non vi è alcuna differenza nelle misure tra TCL 3 e TCL 2, cambiano soltanto le pratiche consigliate e altamente consigliate da applicare. Queste dipendono inoltre dal livello di ASIL per l'applicazione che si intende sviluppare/esaminare con il tool in questione.

Table 4 — Qualification of software tools classified TCL3

Methods		ASIL			
		A	B	C	D
1a	Increased confidence from use in accordance with 11.4.7	++	++	+	+
1b	Evaluation of the tool development process in accordance with 11.4.8	++	++	+	+
1c	Validation of the software tool in accordance with 11.4.9	+	+	++	++
1d	Development in accordance with a safety standard[a]	+	+	++	++

[a] No safety standard is fully applicable to the development of software tools. Instead, a relevant subset of requirements of the safety standard can be selected.

EXAMPLE Development of the software tool in accordance with ISO 26262, IEC 61508 or RTCA DO-178.

Figura 3.3: Pratiche raccomandate a seconda del livello di ASIL al quale ci si vuole conformare [7]

Le voci nella tabella riportano i paragrafi del capitolo ai quali rifarsi per avere rispondenza con la pratica scelta. Dovrebbero essere applicate tutte e 4 le voci richieste, anche se l'attestato rilasciato alla Isograph fa riferimento a solo 2 pratiche. Prima di parlare dei punti 1a, 1b e 1c, è bene approfondire quanto richiesto al punto 1d: dice che lo sviluppo del software tool deve seguire un appropriato standard di sicurezza. Come sarà approfondito più avanti, vi sono molte differenze tra un software tool e un software embedded con funzioni di sicurezza, che invece risponde alle normative esaminate nel Capitolo 2. Pertanto la stessa tabella, nella nota a margine, dice che nessuno standard è completamente applicabile per lo sviluppo di software tools, tuttavia è possibile, nella specifica dei requisiti, selezionarne alcuni di carattere generale, che possono ben associarsi al tool in esame.

- **11.4.6**: risponde al punto "a" della tabella. Dice che la certificazione del software tool dovrebbe essere documentata includendo i seguenti:
  - Identificativo unico e versione del software tool;
  - Il massimo livello di confidenza per il quale il tool è classificato assieme al riferimento alle analisi di valutazione;
  - Il massimo ASIL o requisito di sicurezza che potrebbe essere violato in seguito ad un malfunzionamento del software tool;
  - La configurazione e l'ambiente di sviluppo per il quale il tool è certificato;
  - La persona e l'ente che rilascia la qualifica;
  - I metodi applicati secondo la tabella in figura 3.3;
  - I risultati delle misure effettuate per la qualifica del software tool;
  - L'uso di vincoli e malfunzionamenti identificati durante la qualifica, se applicabile.

- **11.4.7.2**: è un sottoparagrafo del 11.4.7 così come gli altri due punti a seguire. Rappresenta la prima pratica richiesta nelle tabelle per il TCL. Dice che un software tool dovrebbe essere discusso, sulla base dell'incremento della confidenza con l'uso, usando i seguenti punti:
    - Il software tool è stato usato in precedenza per lo stesso scopo, in un ambito comparabile di uso e ambiente, e con gli stessi requisiti funzionali;
    - La giustificazione dell'aumento di confidenza è sostenuta da una buona base di dati;
    - La specifica del tool non cambia;
    - Che tutte le occorrenze riguardanti malfunzionamenti e output errati sono stati accumulati per via sistematica.
- **11.4.7.3**: l'esperienza accumulata nell'uso del software tool dovrebbe essere analizzata e valutata secondo i seguenti punti:
    - Unico identificativo e versione del tool;
    - Configurazione del software;
    - Il dettaglio del periodo di uso e i dati rilevati in tale periodo;
    - La documentazione dei malfunzionamenti monitorati e degli output errati riscontrati, con le condizioni nelle quali ciò è avvenuto;
    - La lista delle precedenti versioni monitorate ed i malfunzionamenti corretti;
    - Le azioni intraprese per il contrasto a tali malfunzionamenti.
- **11.4.7.7**: il livello di confidenza è relativo alla sola versione analizzata.

- **11.4.8.2**: assieme al punto seguente, sono due sottoparagrafi della 11.4.8. dice che il processo di sviluppo del software tool dovrebbe rifarsi ad un appropriato standard.
- **11.4.8.3**: la valutazione del processo di sviluppo del tool dovrebbe essere provata da una valutazione basata su un appropriato standard internazionale, e l'appropriata applicazione della valutazione del processo di sviluppo dovrebbe essere dimostrata. Uno standard in tal senso, fortemente raccomandato è lo SPICE, del quale si parlerà più avanti.

Una nota sul paragrafo 11.4.8. Tale paragrafo guida nella certificazione di qualità del processo del software. È essenziale che sia l'azienda che scrive il codice, che il codice del software tool stesso, siano certificati di qualità. La certificazione di qualità fa sì che sia il codice, che l'azienda che ha scritto tale codice, obbediscano ad uno standard internazionale che permette la prevenzione di tecniche di programmazione ritenute pericolose o errate, in quanto possono portare a codici pesanti non ottimizzati, con elevata presenza di errori sistematici che poi si possono riflettere sugli output. Una certificazione del processo qualità previene, almeno in linea teorica, la presenza di tali errori, e funge da assicurazione per il cliente che acquista il tool. Tuttavia la certificazione di qualità del processo, è condizione necessaria ma non sufficiente alla certificazione del tool.

- **11.4.9.2**: sottoparagrafo della 11.4.9. La validazione del tool deve rifarsi ai seguenti criteri:
    - Le misure di validazione devono dimostrare che il tool risponda ai requisiti. Possono essere usati test per certificare la qualità degli aspetti funzionali e non;

- I malfunzionamenti e i loro corrispettivi output errati occorsi durante il processo di validazione, dovrebbero essere analizzati insieme alle informazioni delle loro possibili conseguenze, e le misure per trovarli ed evitarli;
- La reazione del software all'uso in condizioni anomale (input out-of-range, configurazioni sconsigliate, carichi di lavoro eccessivi, ecc).

Una volta scelte le pratiche da adottare, e messe in campo le misure richieste, devono essere prodotti due documenti:

- Software tool evaluation criteria report (risultante dai punti 11.4.1 a 11.4.5 e 11.4.10 della presente norma);
- Software tool qualification report (risultante dai requisiti da 11.4.1 a 11.4.10);

3.1.1 SPICE:

L'Automotive SPICE (questo è il nome completo), più comunemente noto semplicemente come SPICE, è uno standard usato principalmente in ambito automotive per certificare la qualità di un processo software. È uno standard molto completo, anche se richiede una certa destrezza per essere maneggiato, data la grande mole di informazioni riportate al suo interno. Lo SPICE deriva dalla norma internazionale ISO 15504. Il suo compito è valutare in maniera obiettiva i processi di un'azienda e individuare dei possibili miglioramenti del processo.

Non è una normativa di tipo functional safety, pertanto viene descritto in questo sottoparagrafo, poiché richiesto dalla ISO 26262 per la certificazione qualità del software.

Lo SPICE permette di valutare la capacità che ha un'azienda nei confronti di un certo processo, procedendo a delle valutazioni obiettive dei processi realizzati al suo interno, e consente, sempre mediante valutazioni obiettive, di valutare la rispondenza di un processo allo standard di qualità. Nel seguito se ne fornisce una descrizione.

Dal punto di vista dell'azienda, viene valutata la capacità di gestire un processo. Questa capacità può essere valutata sulla base di un framework bidimensionale, che mette in relazione la capacità raggiunta dall'azienda nel maneggiare il particolare processo. Entrambi questi aspetti sono valutati tramite delle tabelle o dei fogli, in cui sono riportate delle pratiche da mettere in atto per ottenere l'attestazione della capacità tramite lo standard. In figura il framework bidimensionale:

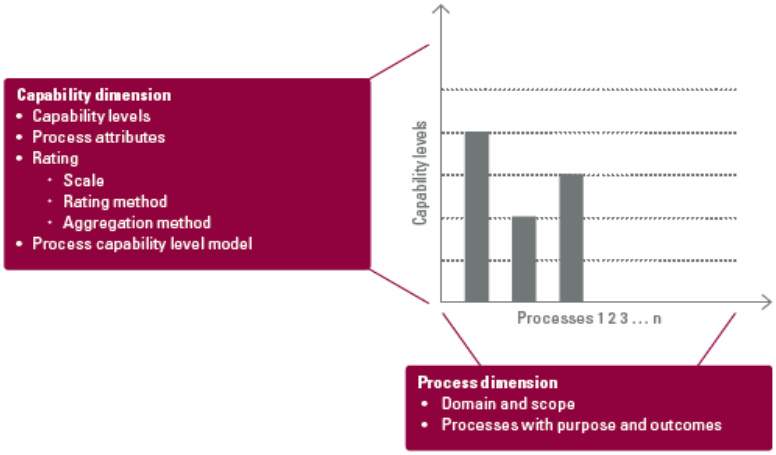

Figura 3.4: Framework 2D alla base della valutazione dello SPICE

La lettura che si deve dare ai risultati ottenuti tramite lo SPICE è il capability level dell'azienda in riferimento a quel dato processo. La lettura allora avviene come per una matrice di dimensione m x n, dove m rappresenta il capability leve dell'azienda in relazione all'n-esimo processo.

Vi sono 5 capability levels:

- Level 0: incomplete process;
- Level 1: Performed process;
- Level 2: Managed process;
- Level 3: Established process;
- Level 4: Predictable process;
- Level 5: Innovating process.

Il capability level si determina attraverso dei **process attributes (PA)**, il cui soddisfacimento, nei confronti di quel determinato processo, permette di valutare il capability level dell'azienda.

Per ogni livello vi sono 2 PA da soddisfare, tranne per il livello 0 che non presenta PA e il livello 1 che ne presenta uno soltanto. Il soddisfacimento di un PA si ha attraverso una scala di valori chiamata NPLF, acronimo derivante dal voto assegnato:

- N: Not Achieved;
- P: Partially Achieved;
- L: Largely Achieved;
- F: Fully Achieved.

La tabella nell'immagine seguente aiuta nell'assegnare la corretta valutazione al PA considerato:

N	*Not achieved*	*0 to ≤ 15% achievement*
P	*Partially achieved*	*> 15% to ≤ 50% achievement*
L	*Largely achieved*	*> 50% to ≤ 85% achievement*
F	*Fully achieved*	*> 85% to ≤ 100% achievement*

Figura 3.5: Scala NPLF

Poi la scala può essere ulteriormente raffinata introducendo i livelli P+, P-, L+ ed L-. La valutazione può essere fatta attraverso tre diversi metodi di raiting, a seconda dell'obiettivo e del tipo di processo in esame.

L'ottenimento del capability level si ha quando i PA relativi al level capability inferiore sono valutati F (Fully) e quelli relativi al level capability in esame sono valutati come L (Largely). Fa eccezione il livello 1 dove basta una L.

Nella figura seguente è riportata una tabella nella quale sono presenti i capability levels ed i relativi PA necessari per l'ottenimento.

Scale	Process attribute	Rating
Level 1	**PA 1.1: Process Performance**	**Largely**
Level 2	PA 1.1: Process Performance **PA 2.1: Performance Management** **PA 2.2: Work Product Management**	Fully **Largely** **Largely**
Level 3	PA 1.1: Process Performance PA 2.1: Performance Management PA 2.2: Work Product Management **PA 3.1: Process Definition** **PA 3.2: Process Deployment**	Fully Fully Fully **Largely** **Largely**
Level 4	PA 1.1: Process Performance PA 2.1: Performance Management PA 2.2: Work Product Management PA 3.1: Process Definition PA 3.2: Process Deployment **PA 4.1: Quantitative Analysis** **PA 4.2: Quantitative Control**	Fully Fully Fully Fully Fully **Largely** **Largely**
Level 5	PA 1.1: Process Performance PA 2.1: Performance Management PA 2.2: Work Product Management PA 3.1: Process Definition PA 3.2: Process Deployment PA 4.1: Quantitative Analysis PA 4.2: Quantitative Control **PA 5.1: Process Innovation** **PA 5.2: Process Innovation Implementation**	Fully Fully Fully Fully Fully Fully Fully **Largely** **Largely**

Figura 3.6: Capability Levels e necessaria valutazione dei PA [8]

Nel momento in cui è impossibile raggiungere la valutazione richiesta per quel PA, è necessario adottare delle misure per ottenere quanto voluto, o abbassare la richiesta del capability Level. In genere le aziende difficilmente arrivano ad un capability level 3.

Nella figura 3.4, l'ascissa si riferisce all'implementazione del processo. In maniera da giudicare l'ottenimento dei risultati o il raggiungimento degli obiettivi, una valutazione permette di ottenere delle evidenze obiettive. Queste evidenze sono mappate tramite gli indicatori del **Process Assessment Model (PAM)**, che permettono di stabilire la corrispondenza con i risultati del processo pertinenti e i risultati degli attributi del processo.

Vi sono due tipi di indicatori:

- **Process performance indicators**: applicabili esclusivamente al capability Level 1. Forniscono un'indicazione del grado di realizzazione dei risultati del processo.
- **Process capability indicators**: applicabili dal capability Level 2 al 5. Forniscono un'indicazione del grado di realizzazione dei risultati degli attributi del processo.

La verifica degli indicatori è usata per confermare che certe pratiche sono state messe in atto, come dimostrato dalle prove messe in atto durante la verifica. Tutte queste prove provengono dall'esame dei prodotti di lavoro dei processi valutati o dalle dichiarazioni fatte dagli esecutori e dai responsabili dei processi.

Tipologie di process performance indicators sono i **Base Practises (BP)** e i **Work Product (WP)**. L'esistenza di BP e WP forniscono evidenza delle prestazioni dei processi ad esse associati. Similmente, la presenza dei process capability indicators, fornisce prove della capacità del processo.

Entrambi BPs e WPs, si riferiscono ad uno o più risultati del processo. BPs rappresentano un indicatore orientato all'attività, mentre i WPs rappresentano degli indicatori orientati al risultato.

Per il process capability indicators, si hanno **Generic Practise (GP)** e **Generic Resource (GR)**. Entrambi sono relativi ad uno o più PA obiettivi. La differenza tra GP e GR è che i primi rappresentano indicatori orientati all'attività mentre i GR rappresentano indicatori orientati all'infrastruttura per giudicare le prove obiettive.

La figura seguente mostra allora come si suddividono questi indicatori per il processo e il capability level dell'azienda:

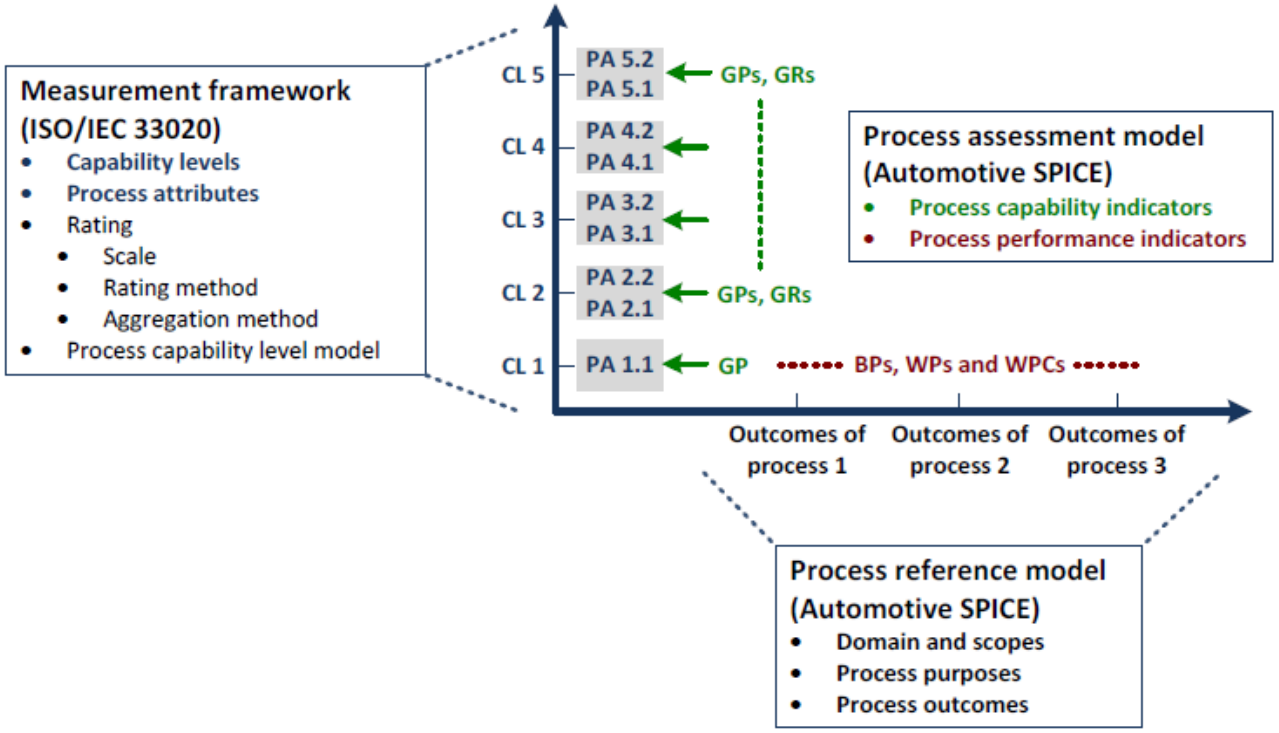

Figura 3.7: Indicatori del processo nella scala di valutazione dei capability levels [8]

Nonostante il fatto che il capability Level 1 di un processo sia caratterizzato solo dalla misura in cui i risultati del processo sono raggiunti, il quadro di misurazione richiede che ciascun livello riveli un attributo di processo e, pertanto, richieda al PAM di introdurre almeno un indicatore di capacità di processo. Pertanto, l'unico attributo di prestazione del processo per capability Level 1 (PA.1.1) ha una singola generic practises (GP 1.1.1) che indica come riferimento editoriale ai rispettivi indicatori di prestazione del processo.

I processi che vengono tenuti in conto nello SPICE riguardano diversi aspetti del life cycle del prodotto, in particolare sono coperte tre macro aree: Primary Life Cycle Processes, Organizational Lyfe Cycle Processes, Supporting Lyfe Cycle Processes. All'interno di queste tre macro aree troviamo un'ulteriore suddivisione che riguarda diversi aspetti delle attività manageriali, organizzative e di progetto. Alla fine, all'interno di questa seconda suddivisione, si trovano le Base Practises da mettere in atto per ottenere l'implementazione del processo.

La figura seguente, tratta direttamente dal manuale dello SPICE, è esplicativa di quanto esposto:

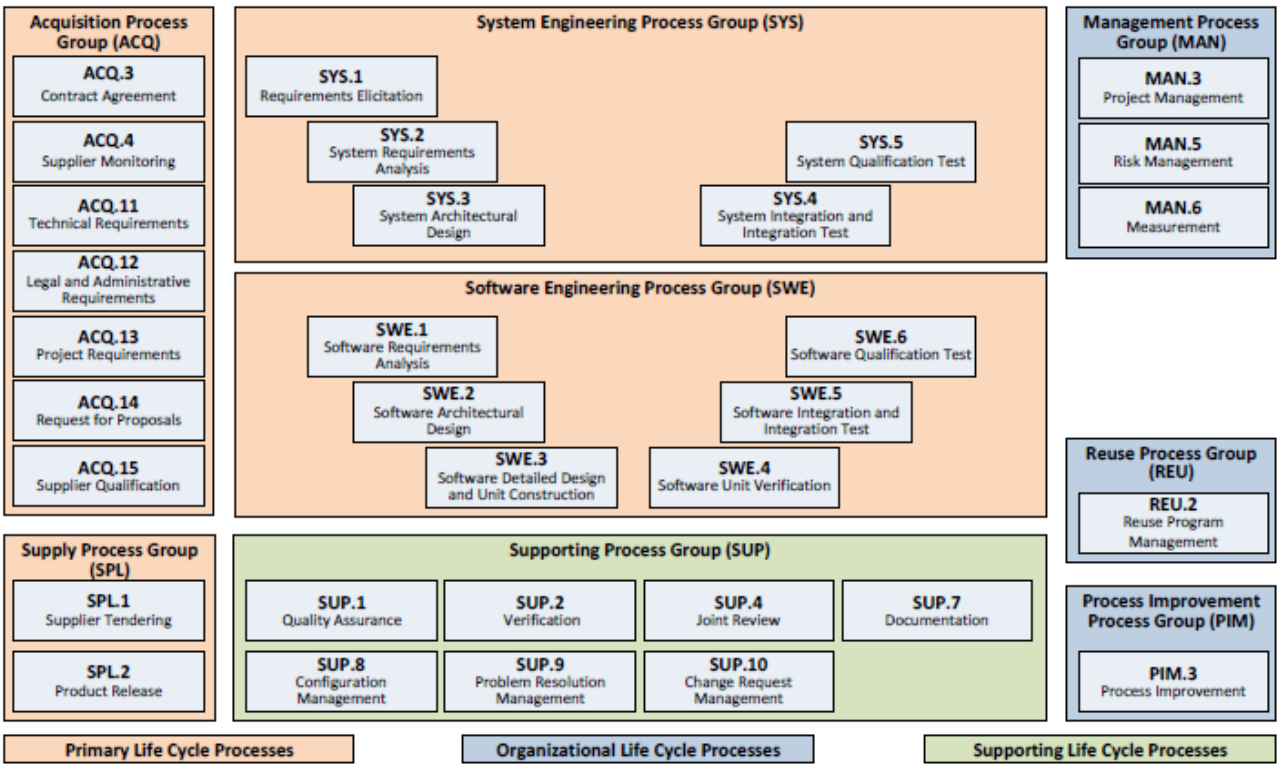

Figura 3.8: Automotive Spice process reference model - Overview [8]

Ogni processo viene descritto attraverso un template ben definito, dove sono riportate tutte le informazioni necessarie per quel processo, relative al Process Reference Model e ai Process Performance Indicators.

PROCESS REFERENCE MODEL		Process ID Process Name Process Purpose Process Outcomes	I singoli processi sono descritti in termini di nome del processo, scopo del processo e risultati del processo per definire il modello di riferimento del processo secondo SPICE. Inoltre viene fornito un identificatore di processo.
PROCESS PERFORMANCE INDICATORS		Base Practises	Una serie di base practises per il processo che forniscono una definizione dei compiti e delle attività necessarie per raggiungere lo scopo del processo e soddisfare i risultati del processo
		Output Work Products	Un numero di work product associati a ciascun processo

Questa descrizione, seppur breve, permette di dare uno sguardo generale allo SPICE, e di comprenderne le sue sezioni principali. Il manuale riporta, per ogni processo, i dati secondo il tamplate illustrato sopra. Di questi, in relazione allo sviluppo del software tool oggetto del presente elaborato, e come richiesto dalla ISO 26262-8, bisogna far riferimento alla sezione di software engineering process, per qualificare la qualità del processo che ha portato alla

scrittura del codice. Ciò implica quindi di seguire le base practises contenute in ogni processo dal SWE 1 al SWE 6. Secondo il modello dello SPICE, poi sarà necessario determinare il capability level dell'azienda relativamente a questi processi.

Come anticipato in precedenza, lo SPICE permette una valutazione del processo sulla base della capacità che ha l'azienda di implementare tale processo. Si viene allora a generare una sorta di matrice, seguendo lo schema riportato in figura 3.4, dove sulle righe vi è il capability level dell'azienda e sulle colonne i processi implementati. Per fare un esempio, l'azienda X ha Livello 2 per il processo SW1, Livello 3 per il processo SW4 e così via. Sono assenti solo quei processi che non riguardano l'azienda o il particolare mercato sul quale si muove l'azienda, che vengono definiti allora non applicabili. Bisogna considerare che comunque i capability level sono abbastanza concentrati: sarà impossibile trovare un'azienda con Livello 1 per un processo e Livello 4 per tutti gli altri. In genere i livelli sono contigui e poco sgranati tra loro, quindi sarà più facile trovare aziende con Livelli 2 e 3 in quasi tutti i processi, salvo quelli di nuova implementazione per i quali potrebbe essere riscontrato un Livello 1, ma che comunque sarà innalzato il prima possibile al Livello 2 o 3. Si ricorda, inoltre, che le aziende difficilmente ottengono un Livello maggiore di 3 nella gestione dei processi, attestandosi su un livello medio pari a 2 e 3 per il main business.

3.2 COSA CHIEDE LA DO-178B:

La seconda normativa nella quale è trattata la certificazione, o meglio qualificazione, di un software tool è la normativa aeronautica DO-178B. In particolare l'aspetto della qualificazione è trattato al capitolo 12 nel paragrafo 12.2.

Un elemento che è bene mettere subito in evidenza è il fatto che, a differenza della precedente normativa automotive, si parla non più di certificazione, ma di qualificazione. Nonostante lo stesso termine sia usato più volte, ed in maniera indistinta anche nella precedente normativa, la differenza qui è sostanziale. La certificazione di un prodotto, implica che una volta che tale prodotto riceve l'attestazione dall'ente certificatore, può essere utilizzato e viene accettato per tutti i progetti che rientrano nell'ambito per il quale è stato certificato. Non a caso il software commerciale FaultTree+ è utilizzato in molte realtà industriali che accettano la certificazione di tipo automotive, comprese realtà aerospaziali. La qualifica del prodotto, invece, ricalca per certi aspetti lo stesso significato, ma in aeronautica, dato che il grado di sicurezza ricercato è più elevato, impone che per ogni progetto debba essere qualificato tutto il materiale utilizzato. Di conseguenza il software tool viene qualificato per quel particolare tipo di progetto, ma necessiterà di un'ulteriore qualifica laddove debba essere utilizzato in altri progetti.

La DO-178B dice che è necessario eseguire la qualifica del tool laddove venga utilizzato per automatizzare processi. In particolare è necessaria la qualifica quando i processi ed i meccanismi che riguardano le verifiche trattate in questa normativa sono fatte da un software esterno.

La normativa fa una distinzione importante tra i software tools utilizzati:

- **Software development tools**: sono tools i cui output fanno parte del software dell'aeromobile, e pertanto possono introdurre errori sistematici. Come dice il nome sono tool utilizzati durante la fase di progetto e/o realizzazione;

- **Software verification tools**: sono tools che non introducono errori sistematici, ma possono fallire nel rilevarli. Un aspetto importante è che l'output di tali tool non viene verificato successivamente.

L'obiettivo è assicurare che il tool abbia un livello di confidenza uguale o superiore ai processi che mira ad automatizzare.

La qualifica del tool include:

- Che venga qualificato in base ad uno dei due tipi citati sopra;
- Software development tools e software verification tools devono essere qualificati entrambi come development tools, a meno che la suddivisione tra le due funzioni possa essere dimostrata;
- Il processo di gestione della configurazione ed il processo di assicurazione della qualità del software per aeromobili devono essere applicati anche per il software tool da qualificare.

Inoltre devono essere adottate le seguenti misure:

- Se il software tool deve essere qualificato, il processo di sviluppo del tool deve avere gli stessi obiettivi del software embedded da installare a bordo velivolo;
- Il **software level (SL)** assegnato al tool dovrebbe essere lo stesso assegnato al software embedded da installare a bordo velivolo. Tuttavia una giustificazione in accordo con l'autorità di certificazione, può portare ad un abbassamento del livello del software;
- L'applicazione deve essere in accordo con i requisiti operazionali del tool. Questo può portare ad un periodo nel quale il tool deve essere testato e validato in maniera da identificare problemi ed eventualmente correggerli;
- I software development tools devono essere verificati per controllare la correttezza, la coerenza e completezza dei requisiti operativi del tool e verificare che questi siano rispettati. Per un tool di validazione gli obiettivi sono diversi rispetto a quelli del software del velivolo: i requisiti di alto livello sono intesi come quelli operazionali. La verifica di tali requisiti può essere ottenuta da:
    - Review dei requisiti operazionali del software come descritti nel sottoparagrafo 6.3.1 della presente normativa ai punti a e b: si tratta di compliance with system requirements, accuratezza e consistenza;
    - Dimostrare che sono soddisfatti i requisiti operazionali sotto le condizioni normali di funzionamento;
    - Dimostrare che il tool rispetta i requisiti operazionali mentre opera in condizioni anormali, inclusi disturbi esterni o apposite failure inserite nel tool o nel suo ambiente;
    - Si possono eseguire test addizionali per confermare il rispetto dei requisiti;
    - Si devono eseguire delle analisi di copertura strutturale appropriate per il software level del tool;
    - Eseguire analisi di robustezza del tool per flussi di dati o di controlli;
    - Bisogna avere un piano di qualifica del tool (o Software Tool Qualification Plan).

Un tool di verifica, a differenza di un development tool, deve avere solo una specifica dei requisiti operazionali.

Le linee guida per la qualifica del tool devono contenere:

- La control category del software. Questa assume due valori: **CC1** e **CC2**, rispettivamente per i development tools ed i verification tools. Queste categorie vengono assegnate al life cycle del software. Queste categorie sono correlate ai controlli di gestione della configurazione basata sui dati. Si ha una tabella nella quale sono presenti le misure da adottare per avere rispondenza ad una control category. L'immagine seguente mostra la tabella così come è presentata nella DO-178B, con le voci che devono essere soddisfatte per ciascuna control category ed il riferimento ai sottoparagrafi della normativa che dettano le linee guida per il loro soddisfacimento:

SCM Process Objective	Reference	CC1	CC2
Configuration Identification	7.2.1	•	•
Baselines	7.2.2a, b, c, d, e	•	
Traceability	7.2.2f, g	•	•
Problem Reporting	7.2.3	•	
Change Control - integrity and identification	7.2.4a, b	•	•
Change Control - tracking	7.2.4c, d, e	•	
Change Review	7.2.5	•	
Configuration Status Accounting	7.2.6	•	
Retrieval	7.2.7a	•	•
Protection against Unauthorized Changes	7.2.5b(1)	•	•
Media Selection, Refreshing, Duplication	7.2.7b(2), (3), (4), c	•	
Release	7.2.7d	•	
Data Retention	7.2.7e	•	•

Figura 3.9: Software configuration management in relazione alla control category e riferimenti nella normativa [9]

Se si sta trattando la qualifica di un development tool, vi sono altre misure da prendere in considerazione, data la maggiore sensibilità degli output ottenuti, specie in applicazioni safety critical. I verification tools, presentano invece una procedura più rilassata, proprio per la loro caratteristica di non introdurre errori nel software del velivolo, tuttavia le loro funzioni sono ugualmente sensibili e pertanto necessitano di essere trattati con attenzione. Tool di questo tipo sono ad esempio dei debugger del codice, oppure dei test tool che vanno a verificare che tutte le righe del codice siano eseguite almeno una volta durante i test per evidenziare che non vi siano conflitti potenzialmente pericolosi per la sicurezza.

L'uso di un tool di questo tipo, infine, deve essere approvato dall'ente certificatore, e ciò avviene in due steps:

- Per i software development tools:
    - viene certificata la rispondenza ai requisiti operazionali;

    - e che questo sia in accordo con la review al termine della fase di realizzazione.
- Per i software verification tools:
    - L'accordo con il piano degli aspetti di certificazione del software del velivolo;
    - e che questo sia d'accordo con la review al termine della fase di realizzazione del software del velivolo.

3.2.1 ASSICURARE LA QUALITÀ:

Analogamente a quanto richiesto dalla ISO 26262, anche la DO-178B richiede un piano che certifichi la qualità del software. A differenza della norma precedente, che per la certificazione di qualità del processo si appoggiava ad uno standard esterno come lo SPICE, la normativa aeronautica ha già al suo interno tutte le indicazioni da seguire. Tali indicazioni possono essere trovate al capitolo 8.

La qualità del processo relativa al software si determina rispettando due punti:

- Assicurando che i componenti del software soddisfino o eccedano nelle misure previste dal life cycle del software per quel determinato livello del software;
- Assicurando che i cambiamenti al life cycle del software siano contenuti nei progetti del software.

Il capitolo 8 descrive obiettivi ed attività legate all'ottenimento della **Software Quality Assurance**. Il processo della SQA valuta il ciclo di vita del software ed i suoi risultati per ottenere la garanzia che gli obiettivi siano soddisfatti, le carenze rilevate, valutate monitorate e risolte e che i dati del ciclo di vita dei prodotti software siano conformi ai requisiti di certificazione.

Gli obiettivi che concorrono alla SQA si ottengono assicurando che:

- I processi di sviluppo del software e i processi integrati siano conformi ai piani software approvati ed agli standard;
- I criteri di transizione per i processi del ciclo di vita del software siano soddisfatti;
- Viene condotta una review della conformità del prodotto software.

I criteri di transizione servono a determinare quando un processo può essere inserito o reinserito nel ciclo di vita del software.

Per soddisfare gli obiettivi della SQA devono essere adottate le seguenti misure, adattabili anche alla certificazione di qualità del processo per i software tools:

- Il processo della SQA dovrebbe assumere un ruolo attivo nel ciclo di vita del software, ed il processo della SQA dovrebbe essere abilitato con autorità, responsabilità ed indipendenza per assicurare che gli obiettivi della SQA siano soddisfatti;
- Il processo della SQA dovrebbe provvedere ad assicurare che i piani del software e gli standard siano sviluppati e rivisti per la consistenza;
- Il processo della SQA dovrebbe provvedere ad assicurare che il processo del life cycle è rispondente ai piani del software ed agli standard;

- Il processo della SQA dovrebbe includere verifiche dello sviluppo del software e dei processi integrali durante il ciclo di vita del software per ottenere che:
    - I piani del software siano disponibili;
    - Le deviazioni da tali piani e dagli standard siano rilevati, registrati, valutati, tracciati e risolti;
    - Le deviazioni approvate siano registrate;
    - L'ambiente di sviluppo del software è adeguato ai piani del software;
    - Report dei problemi, tracciabilità e azioni correttive devono essere conformi al software configuration management plan;
    - Input forniti ai processi del ciclo di vita del software, dalla sicurezza del sistema al processo di valutazione siano stati affrontati.
    - Il processo della SQA dovrebbe garantire che i criteri di transizione per il ciclo di vita del software siano stati soddisfatti in conformità con i piani del software approvati;
    - Il processo della SQA dovrebbe provvedere ad assicurare che i dati del life cycle del software siano controllati in accordo con le control categories;
    - Prima della consegna dei prodotti software presentati come parte di un'applicazione di certificazione, dovrebbe essere condotta una revisione della conformità del software;
    - Il processo dell'SQA dovrebbe produrre dei records con le attività del processo, includendo audit e prove di completamento della software conformity review per ogni prodotto considerato come parte dell'applicazione da certificare.

L'obiettivo della software conformity review è assicurare che il life cycle del software sia stato eseguito interamente e che il codice eseguibile del software sia stato controllato e possa essere rigenerato.

Le attività per ottenere la qualità del software sono raccolte anche nel capitolo 11 paragrafo 11.5 della normativa. Parla del piano della SQA per soddisfare gli obiettivi del processo della SQA.

Il piano della SQA può contenere descrizioni sul miglioramento del processo, metriche e metodi di management. Inoltre dovrebbe includere:

- Ambiente: una sua descrizione includendo scopo, organizzazione delle responsabilità ed interfacce, standards, procedure, strumenti e metodi;
- Autorità: Indicare chi ha l'autorità del piano della SQA, responsabilità, indipendenza, includendo l'autorità che deve approvare il piano della SQA;
- Attività: ossia le attività che vengono svolte durante il ciclo di vita del software includendo: metodi, attività legate al problem reporting e tracciabilità, descrizione della software conformity review;
- Criteri di transizione;
- Timing;
- Definizione dei record del processo della SQA;

Infine, i risultati del processo della SQA devono essere inseriti all'interno di SQA records.

3.3 CONSIDERAZIONI FINALI:

Effettuata una panoramica su quanto richiesto dalle due normative circa la certificazione di un software tool, è necessario riassumere e confrontare quanto esse riportano, mettendo in evidenza le differenze che, come è ovvio pensare, sono presenti date le diverse discipline tecniche alle quali fanno riferimento.

La normativa automotive ISO 26262 è quella che traccia un quadro chiaro sul processo da seguire per la certificazione di un software tool. Permette di fare delle valutazioni sul tipo di software che si sta cercando di certificare e detta delle precise linee da seguire. Non a caso oggi i prodotti commerciali usati in ambito industriale per analisi di sicurezza sono rispondenti a tale standard.

La qualità del software viene raggiunta tramite le linee guida della normativa ISO 15504, che poi sono state tradotte nel manuale dell'Automotive SPICE. Questo manuale contiene, in maniera chiara e semplice tutte le pratiche da seguire per avere un processo di qualità. I processi a cui fa riferimento sono riportati nella figura 3.8, e possono essere collegati alle capacità aziendali.

Se la normativa di stampo automotive si è adattata negli anni alla sempre maggiore presenza e necessità di uso di software esterni per la realizzazione dei processi industriali legati al ciclo di vita del prodotto, la normativa aeronautica, seppur più anziana, tiene in considerazione l'uso di tali software operando una distinzione in più: software usati durante la fase di sviluppo del software embedded o sistema velivolo, e software usati per la verifica/validazione delle misure adottate durante la fase di sviluppo. Questa distinzione è stata fatta in ragione al diverso impatto che tali software tools hanno sugli output della fase di sviluppo, difatti seguono procedure diverse, più stringenti per i development tools più blande per i verification tools.

Il punto di contatto tra le pratiche da seguire è la certificazione della qualità del processo. A differenza della normativa automotive, che si appoggia ad altre normative o standard esterni come lo SPICE, la stessa DO-178B possiede al suo interno le indicazioni e le misure da implementare affinché il software tool possa essere certificato di qualità. Tali misure, seppur legate maggiormente al prodotto software embedded, possono essere applicate al software tool come richiesto dalla normativa al capitolo 12, anche se con un certo grado di adattamento dettato dal fatto che si ha sempre a che fare con software esterni, che spesso rispondono a standard non prettamente aeronautici e che pertanto necessitano solo di una qualifica.

Come accennato in precedenza, nella DO-178B si parla apertamente di qualifica del software tool, in quanto si deve soltanto dimostrare che tale software tool implementa quelle misure che lo rendano un prodotto sicuro da usare nell'ambito di sviluppo e realizzazione del software aeronautico. La qualifica, inoltre, si rende necessaria per ogni applicazione diversa nella quale si fa uso di tale tool. Questa normativa non richiede allora un rilascio di una certificazione del tool come previsto dalla ISO 26262, che invece è alla base dei software commerciali prodotti dalla Isograph usati anche in ambito aerospace. Quindi se ne può trarre la conclusione che la certificazione secondo ISO 26262 rimane quella maggiormente orientata all'ambito commerciale, mentre le linee guida DO-178B sono più usate in ambito interno all'azienda per qualificare la rispondenza del tool all'attività progettuale.

3.4 NOTE SULLA DO-178C:

Nel 2011 la RTCA ha prodotto una nuova versione della normativa relativa alla certificazione del software safety critical. La nuova edizione è dovuta allo sviluppo tecnologico al quale si è assistito nel primo decennio del duemila, che ha introdotto nuovi strumenti, sempre più potenti e pratici nelle mani dei progettisti. Uno di questi strumenti, in precedenza poco considerato, è il model-based design, ossia un metodo che basa la progettazione su modelli e su programmi generatori di codici come Matlab-Simulink.

Questi programmi, inizialmente, erano ritenuti poco affidabili nel mondo aeronautico, poiché fino alla loro introduzione, il codice veniva scritto con procedure rigide e linguaggi di programmazione complessi, come l'ADA ad esempio, i cui costrutti erano molto strutturati e poco leggibili, ma nello stesso tempo, dalla complessità, ne derivava una certa sicurezza sul fatto che non fossero presenti errori. Altri linguaggi comunemente usati erano Fortran, o C.

La DO-178B è una norma risalente al 1992, pertanto è fortemente orientata all'uso di tali linguaggi, ed utilizza un approccio molto conservativo. La versione C, invece, modernizza lo standard, introducendo nuovi criteri di valutazione e degli annessi, nello specifico il DO-330, che ampliano il campo di applicazione della norma anche ai software tool, che invece la 178B mantiene al suo interno.

Le differenze tra le due norme non sono tantissime ad un primo sguardo, i concetti trattati sono sempre gli stessi e lo stesso indice risulta molto simile. Le differenze sono principalmente sul contenuto, dove molti concetti vengono ampliati per far fronte all'introduzione delle nuove tecniche e tecnologie.

Per rimanere all'interno del contesto del presente elaborato, si esaminerà come la 178C si approccia ai tool utilizzati nel processo di sviluppo del software safety critical.

Il riferimento dall'indice è sempre al capitolo 12 paragarafo 2: Tool Qualification. La differenza con la 178B è evidente sin da subito: la normativa non differenzia più i tool usati in development tools e verification tools, con l'obiettivo primario di caratterizzare il tool all'interno di una di queste due categorie. La nuova versione si avvicina molto alla classificazione che fa la ISO, eliminando quella distinzione ed introducendo un parametro di qualifica o **Tool Qualification Level (TQL)**. Il TQL è un parametro che deve essere scelto dal progettista o da chi intende qualificare il proprio tool, sulla base del software level e di tre criteri:

- CRITERIO 1: Tool parte del velivolo e può introdurre errori al suo interno. Sono quei tools che volano con il velivolo, e si occupano di eseguire funzioni sul software o i sistemi del velivolo (i modelli di diagnostica sono un esempio);
- CRITERIO 2: Un tool che automatizza i processi di verifica e può fallire nel rilevare errori. Gli output permettono di eliminare o ridurre:
    - Processi di verifica;
    - Processi di sviluppo.
- CRITERIO 3: Un tool che, nell'ambito della sua destinazione d'uso, potrebbe fallire nel rilevare un errore. Questo criterio è molto generale, e si associa a tutta una famiglia di tools che operano sul software senza avere su di esso effetti critici.

Di conseguenza si può associare il criterio 1 ai tools di diagnostica o i development tools, il criterio 2 ai verification tools ed il criterio 3 a tool che non appartengono a tale famiglia. Quest'ultimo è un criterio molto rilassato, associabile a tutta una serie di tools usati in fase di progetto che, comunque, possono portare un contributo nella rilevazione degli errori.

Si calcola allora il TQL: la scala prevede **5 livelli**: TQL1 è il più stringente, TQL2 il meno stringente. Si crea una tabella dove le righe contengono il software level dell'applicazione, le colonne il criterio di valutazione scelto. L'immagine seguente riporta tale tabella:

Software Level	Criteria		
	1	2	3
A	TQL-1	TQL-4	TQL-5
B	TQL-2	TQL-4	TQL-5
C	TQL-3	TQL-5	TQL-5
D	TQL-4	TQL-5	TQL-5

Figura 3.10: Tool Qualification Level Determination [10]

Per il Tool Qualification Process, la DO-178C rimanda all'annesso DO-330, dove sono descritte le pratiche da applicare per ogni TQL scelto.

3.4.1 DO-330:

L'annesso DO-330 è come una normativa nella normativa: riporta tutti gli aspetti del software life cycle applicati al tool. Espande i concetti esposti nella 178B introducendo aspetti che non vengono contemplati. Si parla infatti di **Tool qualification planning process**, **Tool development life cycle and process**, ossia ciclo di vita dello sviluppo del tool, **Tool verification process**, ossia processo di verifica del tool, ecc. Si presenta come una norma nella norma, prettamente riguardante il tool, trattandolo alla stregua del software embedded.

Per il presente elaborato ci si interessa ai processi integrali relativi alla verifica del tool: verification process, configuration management process, quality assurance process e liaison process.

Sono processi che vanno di pari passo con tutto il ciclo di vita del tool. Dei criteri che vengono influenzati dal TQL, sono i criteri di transizione, i quali determinano quando un processo può essere inserito o reinserito nel ciclo di vita del tool. Dipendono dai processi di sviluppo o dai processi integrali.

Nel capitolo successivo verrà trattata la qualifica del tool FTA secondo DO-178B, pertanto ci si concentrerà su come la 178C tratta invece le pratiche che sono state riassunte nel paragrafo 3.2. In particolare saranno descritti il verification process, il tool operational verification and validation process, il tool configuration management process ed il quality assurance process.

Il verification process riguarda la verifica tecnica degli output dei processi di sviluppo e verifica del tool. Si applica tramite un tool verification plan. L'annesso A della DO-330 riporta le tabelle dove sono inserite le pratiche da seguire in base al TQL scelto per il tool. Lo scopo è rilevare e riportare gli errori introdotti durante la fase di sviluppo. La rimozione di questi, invece, non fa parte del processo. In particolare si va a verificare che:

1) I requisiti operativi siano implementati nella specifica dei requisiti ed il tool li supporti (validazione);
2) I requisiti siano implementati nell'architettura ed allocati ai livelli più bassi;
3) Architettura del tool e requisiti di basso livello siano sviluppati nel codice e soddisfino il design;
4) Il codice eseguibile soddisfi i requisiti.

Gli obiettivi sono soddisfatti tramite una serie di review, analisi, test cases e procedure, con la conseguente messa in atto delle procedure di test. Sono presenti procedure che riguardano la scrittura delle review, il tool testing con gli obiettivi e le attività e i tipi di test da fare.

Per gli operational, validation e verification process, si fa riferimento alla tabella T-0 presente nell'annesso A. Vi sono anche qui degli obiettivi da raggiungere, per entrambe validazione e verifica, e vengono consigliate, nella tabella, le attività da svolgere in relazione al TQL scelto.

Per il Tool Configuration Management Process, vengono considerate, analogamente alla DO-178B le Control Categories. Tutto ciò che è riportato nella tabella all'annesso A risulta perfettamente uguale alle pratiche richieste nella vecchia versione della norma. Pertanto di queste si vedrà l'applicazione pratica nel capitolo successivo. Vi sono comunque delle attività da compiere per questo processo, riportate nella tabella T-8, e anche queste scelte sulla base del TQL. In ogni caso la control category rientra, poiché non soltanto l'attività deve essere svolta, ma anche rispondere ad una control category (principalmente la CC2). In ogni caso, anche se fossero presenti delle differenze, queste sarebbero di piccola entità.

Il Tool Quality Assurance Process aggiunge qualcosa in più rispetto alla 178B. Si introduce un quarto obiettivo, derivante da una scorporazione dell'obiettivo 1 della 178B. Le richieste riguardano i piani di sviluppo del tool e che i processi integrali, di cui si è appena scritto, siano sviluppati e rivisti per assicurarne la consistenza. Le pratiche sono rimandate alla tabella T-9 dell'annesso A, che risulta molto differente rispetto a quella presente nella 178B, pur chiedendo la CC2. Presenta cinque voci e rimanda alle attività necessarie per raggiungere gli obiettivi richiesti.

Si lascia al lettore la facoltà di verificare il contenuto delle tabelle, poiché tale norma non sarà applicata nel successivo capitolo. Si è preferito riportare delle note che illustrino a grandi linee il suo contenuto. La normativa è giovane, e la sua applicazione sta prendendo sempre più piede avendo soppiantato la vecchia DO-178B. Tuttavia gran parte dei velivoli oggi operanti, sia civili che militari, il cui progetto risale a prima del 2011 o che non hanno subito grossi aggiornamenti avionici successivamente a quella data, operano ancora con equipaggiamenti certificati 178B.

CAPITOLO 4. APPLICAZIONE AL CASO DI STUDIO

Questo capitolo rappresenta il fulcro del presente elaborato. Tutte le informazioni ricavate nel precedente capitolo adesso troveranno un'applicazione pratica su un tool FTA creato in progetti di tesi precedenti. L'applicazione è da ritenersi parziale in quanto, come descritto al paragrafo 1.1, l'obiettivo non è arrivare alla certificazione del tool, processo lungo e dispendioso che va oltre gli obiettivi di una tesi, ma dimostrare l'applicabilità delle normative e se è possibile implementare il processo. In tal caso saranno dettate le linee guida che portino ad implementare un processo di certificazione.

Riprendendo le pratiche richieste dalle due normative riportate nel capitolo 3, si darà uno sguardo alla loro implementazione pratica. Tramite degli esempi si potrà determinare quali pratiche trovano una loro implementazione nel tool, e quali invece falliscono. In tal caso verranno evidenziate le misure da mettere in atto per migliorare il tool tale da renderlo certificabile.

Si farà allora una semplice applicazione pratica di quanto riportato nella ISO 26262-8 e nella DO-178B, compresi gli aspetti legati alla qualità del processo.

4.1 PRESENTAZIONE DEL TOOL FTA:

Il tool oggetto di questo capitolo è un software creato utilizzando Matlab e Simulink, che ha lo scopo di eseguire Fault Tree Analysis (maggiori dettagli nel Capitolo 1). Il tool è un primo esempio di programmazione in Matlab e Simulink per questo tipo di applicazioni. Si sfrutta la capacità dei due software nel lavorare sia in maniera simbolica che numerica, creando delle funzioni apposite che eseguano delle stime di probabilistiche, associate a dei blocchi che permettono, tramite Simulink, una combinazione degli stessi per generare l'albero del guasto e la propagazione dell'analisi fino al risultato finale al top event.

L'idea alla base è molto semplice: tramite una libreria contenente tali blocchetti creati ex-novo, si può generare l'albero, collegandoli come in un qualsiasi modello Simulink. In seguito, grazie ad un ulteriore blocchetto di controllo, si possono passare i dati dell'analisi ai basic events inseriti in precedenza, e far propagare quei valori in maniera da avere i risultati al top event.

Tal tool si propone di richiamare il funzionamento del software FaultTree+, che è stato usato come termine di paragone nella realizzazione di tale tool. Tra i numerosi modelli probabilistici implementati in questo software, il tool in esame si basa sulla **Rare Approximation**.

Adesso se ne esamineranno, più nel dettaglio, le funzionalità.

Condizione necessaria affinchè il tool possa funzionare, è avere installato Matlab e Simulink sul proprio pc da lavoro.

Aprendo il file FT_library si ha accesso alla libreria dei blocchetti che implementano la fault tree analysis. Sono stati creati i blocchetti per i basic events, intermediate events e top events (questi tre sono quelli che contengono i dati dell'analisi. In particolare il basic event sta alla base dell'albero, mentre il top event è l'obiettivo dell'analisi. Gli intermediate events sono degli eventi intermedi dei quali potrebbe essere utile la conoscenza). Un'analisi FTA necessita delle relazioni logiche tra i vari eventi, pertanto sono state create le porte AND ed OR, alla base di qualsiasi equazione logica. Un altro blocco particolare, è quello che riguarda i

Common Cause Failures: questi blocchi descrivono eventi di failure che, nei sistemi complessi, possono portare ad una vanificazione delle ridondanze. Pertanto, in molte situazioni, è necessario tenerli in conto nelle analisi di affidabilità dei sistemi, poiché permettono di identificare quegli aspetti che possono portare a vanificare le ridondanze di un sistema rendendolo in sostanza non sicuro.

Di seguito sono presentati tutti i blocchetti ed il loro funzionamento, che è reso più agevole grazie alla GUI implementata con Matlab. Per maggiori informazioni fare riferimento alla voce [1] della bibliografia.

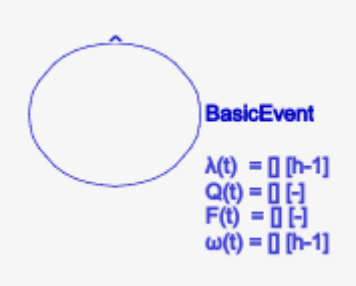

Figura 4.1: Blocchetto del Basic Event

Le voci λ(t), Q(t), F(t) e ω(t) si riferiscono ai valori assegnabili al blocchetto. Riguardano, in ordine, il tasso di guasto, l'indisponibilità, l'inaffidabilità e la frequenza di guasto. Dalla GUI che si attive cliccando due volte su tale blocchetto, è possibile assegnargli un Tag univoco per identificare il guasto durante l'analisi, scegliere il modello di Failure rate (Constant o Fixed), il tipo di componente (Repairable o Unrepaireable), inserire gli input e mostrare gli output in base al tipo di analisi scelta. È presente un pulsante che, richiamando l'apposita funzione Matlab, permette dati gli input, di calcolare le voci λ(t), Q(t), F(t) e ω(t).

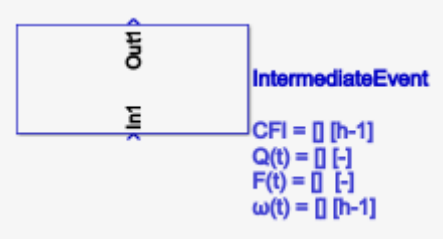

Figura 4.2: Blocchetto dell'Intermediate Event

Il blocchetto dell'Intermediate Event può essere usato per raccogliere le informazioni provenienti da un cutset di basic events. Questo blocchetto interroga le porte AND ed OR in maniera da restituire gli output. Nella GUI associata si può anche analizzare l'equazione del cutset.

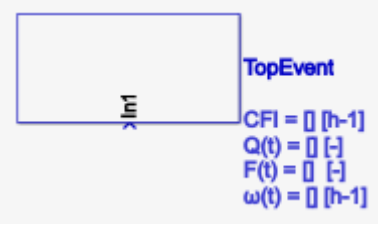

Figura 4.3: Blocchetto del Top Event

Il blocchetto del Top Event rappresenta l'obiettivo dell'analisi e pertanto si trova in cima all'albero (ne da evidenza il solo collegamento sottostante). Il blocchetto presenta i risultati finali dell'analisi, e fornisce la conditional failure intensity del sistema e i parametri Q(t), F(t) e ω(t) complessivi. La GUI permette inoltre di valutare l'equazione del Cutset dell'albero tramite apposito tasto e richiamo alla funzione Matlab.

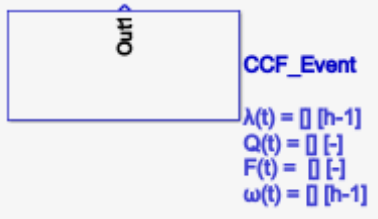

Figura 4.4: Blocchetto del Common Cause Failure

Permette di settare un CCF. La GUI si presenta più complessa, proprio per la natura complessa di tale evento. Si rimanda ad [1] per maggiori dettagli.

Figura 4.5: Blocchetto di descrizione dell'evento

Blocchetto usato per dare una descrizione all'evento. Aiuta a dare una maggiore leggibilità all'albero.

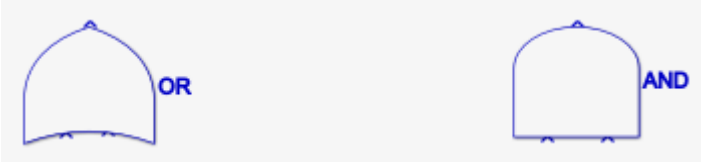

Figura 4.6: Blocchetti rappresentanti le porte logiche AND e OR

Figura 4.7: Blocchetto del Control Panel

Questo blocchetto rappresenta il cervello del modello. Le sue funzioni sono molteplici:

- Permette di settare l'analisi tramite il success criteria da raggiungere;
- Indica il tempo di missione da simulare;
- Permette la valutazione del cutset complessivo;
- Permette l'inizio della simulazione attraverso il pulsante Start Propagation;
- Consente di ripulire il modello dai dati inseriti;
- Consente di creare dei fogli in Excel in cui salvare i dati del modello creato, oppure importarli da un File Excel creato in precedenza o opportunamente formattato.

Una volta creato il nuovo modello in Simulink, e denominato in maniera analoga al modello nel control panel, l'albero può essere costruito creando il cutset di basic events, inserendo i dati appropriati, e combinandoli tramite le porte logiche in maniera da arrivare al Top Event. A questo punto dal control panel si può far partire la propagazione, che porta al risultato cercato. Nel seguito un esempio molto semplice di albero dei guasti creato con il tool.

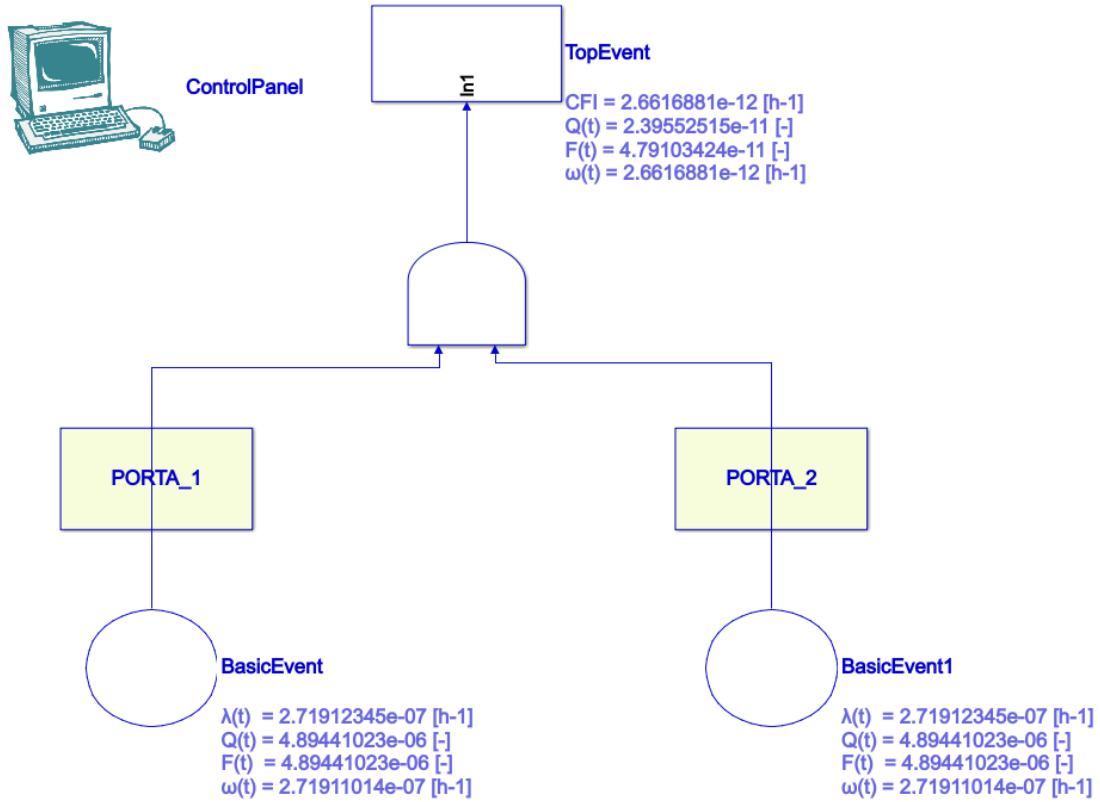

Figura 4.8: Esempio di modello creato con il tool Simulink

Il tool ha già superato una prima fase di validazione, che ha confermato la correttezza delle equazioni presenti al suo interno, e la capacità di eseguire stime di affidabilità corrette fino a 9 cifre significative. I test sono stati eseguiti confrontando gli output con quelli del programma commerciale FaultTree+, mostrando errori minimi se non nulli tra i risultati. Questo permette di considerare il tool funzionante. Per ulteriori approfondimenti sul funzionamento e sulle capacità del tool, nonché sul processo di validazione si rimanda all'elemento [1] della bibliografia. Come si potrà osservare nel seguito, sono però presenti alcune deficienze, che dovranno essere corrette prima della fase di certificazione. Un'altra pecca è la necessità di avere a disposizione Matlab/Simulink per poter utilizzare il tool.

4.2 SPECIFICA DEI REQUISITI:

Prima di iniziare ad esaminare il tool, bisogna porre l'attenzione sul fatto che entrambe le normative richiedono una specifica dei requisiti, che deve essere di tipo funzionale o operativa. Tale specifica non è stata trattata al momento della scrittura del tool, di conseguenza potrebbero essere presenti molte debolezze che portino il tool a non essere certificabile.

Risulta allora necessario procedere alla stesura di questa specifica, ponendo l'attenzione sul seguente aspetto: entrambe le normative chiedono che la specifica dei requisiti venga fatta dal

progettista del software, e che il validatore/verificatore sia una persona diversa. Nel caso in esame il validatore/verificatore dei requisiti diventa anche colui che stende la specifica, e ciò può portare a conflitti o influenze negative sul processo, poiché la scelta dei requisiti da adottare potrebbe essere influenzata a posteriori dai parametri di funzionamento del tool.

Una specifica di requisiti, nel caso più generale, deve contenere le seguenti informazioni:

- Identificativo univoco del requisito;
- Descrizione del requisito;
- Tracciabilità del requisito (intesa come le relazioni padre – figlio nel caso di requisiti allocati su un livello più basso);
- Responsabile del requisito (colui che deve dimostrarne l'implementazione);
- Metodo di verifica del requisito;
- Note sul requisito.

Per il tool in esame è difficile, allo stato attuale, formulare una specifica completa dei requisiti, pertanto ci si concentrerà sulla scrittura di pochi requisiti, sia di tipo funzionale che operazionale, ma che possano essere facilmente dimostrabili e fungano da esempio per la stesura di una specifica più completa e ampia. Inoltre saranno da base per gli esempi successivi quando richiesta la loro verifica dalla normativa.

Per questa breve specifica di altissimo livello, sono stati sviluppati nove requisiti funzionali e 8 requisiti operativi. Bisogna ricordare che non si tratta di software embedded con funzioni di sicurezza, ma di un software esterno progettato per eseguire analisi di affidabilità, e che le normative, pur richiamando il rispetto di tali requisiti, non dicono nulla sulla loro stesura.

REQUISITI FUNZIONALI			
ID	**Requisito**	**Metodo di verifica**	**Note**
F01	Il tool deve poter girare su diversi sistemi operativi	Analisi	/
F02	Il tool deve essere in grado di creare alberi di guasto complessi	Analisi	/
F03	L'utente deve poter inserire i dati attraverso una GUI	Ispezione	/
F04	Il tool deve poter esportare i dati inseriti	Analisi	/
F05	Il tool deve poter importare i dati dall'esterno	Analisi	/
F06	Il tool deve mostrare i risultati dell'analisi e l'equazione logica del cutset	Ispezione	/
F07	L'utente deve poter salvare il lavoro e riprenderlo in qualsiasi momento senza perdere nessun dato	Analisi	/
F08	Il tool deve poter gestire modelli complessi	Analisi	/
F09	L'utente deve poter aprire la GUI cliccando su ogni blocchetto	Analisi	/

REQUISITI OPERATIVI			
ID	**Requisito**	**Metodo di verifica**	**Note**
OP01	Il tool deve poter gestire fino a 20 basic events	Analisi	/
OP02	Il tool deve poter calcolare i parametri necessari all'analisi autonomamente	Analisi	Si intende il corretto funzionamento dei command button inseriti nella GUI
OP03	Il tool deve propagare l'analisi in tempi piccoli	Analisi	Per tempi piccoli si intendono tempi inferiori al minuto.
OP04	Il tool deve poter mostrare i valori ai livelli intermedi, senza errori	Analisi	/
OP05	Il tool deve poter mostrare messaggi di errore nel caso di uso improprio	Analisi	/
OP06	Il tool deve creare, quando richiesto, fogli di calcolo pre-formattati	Ispezione	/
OP07	I risultati non devono differire fino alla quarta cifra significativa da quelli ottenuti con un tool commerciale	Analisi	/
OP08	Le porte logiche devono poter ricevere almeno 5 input	Analisi	/

Le metodologie di verifica che si consiglia di adottare in tal caso sono principalmente:

- **Analisi**: riguarda l'esecuzione delle funzioni del tool tramite esempi pratici;
- **Ispezione**: Comporta una semplice ispezione visiva dello stato del software, degli input o degli output.

È stata volutamente tralasciata la colonna relativa al responsabile del requisito, poiché nelle due normative viene proposto, a seconda del livello di sicurezza richiesto, la figura più adatta a controllare l'applicazione del requisito. Viene anche tralasciata la colonna relativa alla tracciabilità, trattandosi tutti di requisiti di alto livello senza nessuna allocazione su livelli più bassi del software, come invece sarebbe raccomandabile in una specifica più completa.

Una volta completata la specifica dei requisiti, si può passare alle pratiche di certificazione. Verranno esaminate prima quelle relative alla ISO 26262 e successivamente quelle relative alla DO-178B.

4.3 APPLICAZIONE DELLA ISO 26262:

I primi due paragrafi della sezione 11 della ISO 26262-8 servono a determinare se tale normativa è applicabile al caso in esame. In particolare il primo paragrafo dice espressamente che, se il tool ha un qualche impatto sul progetto, allora tale norma va applicata.

Un tool che esegue analisi di tipo Fault Tree, sebbene non abbia un impatto diretto sul design del prodotto finale, viene utilizzato nell'ambito del ciclo di vita nella parte riguardante le analisi di sicurezza. È importante che durante questa fase vengano messe in luce possibili

criticità che possano minare la sicurezza del progetto, e nel contempo si indichino le misure per porvi rimedio. Un output numerico errato da parte di un software di questo tipo può portare a stime affidabilistiche che, nel caso migliore, sottostimano la sicurezza, rendendo l'analisi conservativa, ma in casi di sovrastima si può introdurre un errore sistematico che, sul lungo periodo, può portare a conseguenze gravi. Pertanto, stabilito che il tool FTA in questione impatta su un aspetto particolare, e oggi sempre più sentito del progetto, è necessario introdurre le misure contenute in questa norma.

Il secondo punto dice che se il tool è sviluppato indipendentemente da un utilizzo particolare, bisogna usare delle misure di conferma dello stesso contenute all'interno della normativa in base all'ASIL a cui si aspira arrivare nell'analisi.

Il livello a cui si aspira in tal caso è l'ASIL C/D, i più alti.

Si applica quanto riportato nelle nelle figure 3.1 e 3.2 circa la responsabilità delle review che vengono richieste al termine del processo di certificazione. Per un ASIL D, la confirmation review of the software tool criteria report ed il software tool qualification report possono essere eseguite da persone diverse, pur appartenenti alla stessa organizzazione (grado di indipendenza I 1), mentre la confirmation review che riguarda le prove eseguite su analisi, dati, crediti, ecc. è più stringente, in quanto viene richiesto che tali competenze siano di un ente anche esterno all'azienda ed al team di sviluppo (livello di indipendenza I 3). Entrambi questi aspetti possono essere considerati soddisfatti in quanto non solo chi esegue le analisi è una persona diversa dal programmatore, ma essendo tale elaborato una continuazione ideale della voce [1] della bibliografia, è rispettata anche l'indipendenza esterna.

Fatte quindi le dovute premesse ed evidenziato che l'autore del presente elaborato possiede l'autorità, in accordo a quanto riportato dalla norma, a procedere con l'applicazione della stessa, si vanno a definire i parametri che permettono di caratterizzare il tool.

La caratterizzazione avviene tramite la scelta del Tool Confidence Level (TCL), esaminato nel paragrafo 1 del capitolo 3. Si determinano allora il Tool Impact (TI) ed il Tool Detection (TD) in base alle caratteristiche del tool.

Il TI è la possibilità che il software utilizzato possa introdurre errori oppure fallisca nel rilevarli nel prodotto che si sta sviluppando. Nel caso in esame il tool non partecipa direttamente allo sviluppo del software embedded, ma compie delle analisi di sicurezza le quali, seppur sensibili, non impattano in maniera significativa sul software.

Di conseguenza di sceglie un **TI 2**.

Sulla scelta del TD è necessario fare alcune considerazioni. Rappresenta la confidenza nella misura che il software, in seguito ad un malfunzionamento, produca un corrispondente output errato. Vuol dire conoscere la probabilità con la quale è possibile riscontrare gli errori nel tool. Per questa tipologia di tool, la possibilità di riscontrare errori nel calcolo non è elevata, se non da un utente con una certa esperienza e che abbia una certa manualità nei numeri coinvolti nelle analisi.

Pertanto si potrebbe affermare che il tool sia classificabile **TD 3**.

Dalla tabella presente a pag. 32, unendo i valori di tali parametri calcolati, si determina un **TCL 3**. La normativa riporta allora una tabella con le pratiche, più o meno altamente raccomandate, per tale TCL (in figura 3.3 è presente la tabella riferita direttamente al TCL 3),

sono altamente raccomandate per ASIL C/D la validazione del software e lo sviluppo dello stesso in accordo con uno standard di sicurezza, che può essere lo stesso trattato nella sezione 6 della ISO 26262. Questo porterebbe a scrivere una specifica dei requisiti molto più stringente e ramificata di quella posta come esempio nel paragrafo precedente.

Tuttavia il tool FTA oggetto del presente elaborato è stato fatto prendendo come spunto un software commerciale già certificato ISO 26262, il più volte citato FaultTree+ della Isograph, la quale ha ottenuto dall'ente certificatore TUV la certificazione in accordo ai paragrafi 11.4.8 ed 11.4.9 della seguente normativa.

Questo aspetto porta a considerare il fatto che il TD possa essere stato scelto in maniera troppo conservativa, e che in realtà potrebbe bastare un TD 2 per il tool in esame, ritenendo media la confidenza nel fatto che l'output errato possa essere rilevato e vi si possa porre rimedio. In effetti, riprendendo il concetto espresso in precedenza, e tenendo in conto che le analisi FTA per loro natura generalmente sono svolte da personale esperto, ci si può affidare alla capacità dell'utente per rilevare gli output errati da un semplice confronto output attesi-rilevati. Inoltre, volendo seguire un processo collaudato, che ha portato a certificare un tool di tal tipo, si procederà nel seguito considerando un TCL 2, seguendo quanto fatto dalla Isograph.

Si seguiranno a tal proposito i paragrafi 11.4.8 ed 11.4.9. Il primo riguardante la certificazione di qualità del processo secondo standard Automotive SPICE, il secondo riguarderà una mini validazione con qualche test pratico sul tool per verificare alcuni dei requisiti riportati in precedenza, verifica malfunzionamenti, ed uso in condizioni improprie.

4.3.1 IMPLEMENTAZIONE SPICE:

In accordo con quanto espresso nella 11.4.8 della normativa, verrà effettuata un'applicazione pratica dello standard SPICE per la certificazione della qualità del processo. Le premesse dietro tale applicazione sono le seguenti:

- Non si seguirà tutta la procedura di implementazione dello standard, in ragione della complessità dello stesso, tuttavia saranno esaminate le base practises richieste per l'implementazione del processo e, laddove possibile, se ne darà un riscontro pratico sul tool in esame;
- Essendo un tool programmato come progetto di tesi, seppur con il supporto di una struttura aziendale già certificata SPICE, quest'ultima non rientra nel processo di certificazione, pertanto non vi sarà associato il proprio capability level. Sarà invece calcolato un nuovo capability level basandosi sulle capacità dell'autore del presente elaborato, in maniera da dare una visione quanto più possibile a 360° dello standard.

Lo SPICE è stato descritto nel paragrafo 3.1.1 del presente elaborato. L'obiettivo è quello di dimostrare che è stato implementato un processo, di conseguenza si cercherà di ottenere il Livello 1 per i processi software identificati come applicabili al tool in oggetto.

Il Livello 1 parla semplicemente di processo implementato, e che questo raggiunga i suoi scopi. Il Livello 1 presenta solo una Generic Practise, che chiede soltanto il raggiungimento degli scopi illustrati nelle base practises. La figura seguente mostra come si presenta la generic practise così come riportato nel documento dello standard.

Generic practices	**GP 1.1.1 Achieve the process outcomes** [ACHIEVEMENT a] Achieve the intent of the base practices. Produce work products that evidence the process outcomes.
Generic resources	**Resources are used to achieve the intent of process specific base practices** [ACHIEVEMENT a]

Figura 4.9: Generic Practise 1.1

Il passo successivo è individuare, tra i processi previsti dal Process Reference Model, quelli applicabili al tool in questione. Bisogna ricordare che lo SPICE nasce come uno standard riferito all'implementazione di processi che riguardano il software embedded, quindi non tutti i processi indicati sono applicabili. In riferimento alla figura 3.8 vengono identificati i processi:

- SWE.2;
- SWE.3;
- SWE.4;

Non essendo un software embedded, i processi SW.1, SW.5 e SW.6 non sono applicabili, parlando esplicitamente di integrazione hardware/software.

Per ognuno di tali processi dovrà essere assegnata una valutazione secondo la scala NPLF, con l'obiettivo dell'ottenimento di almeno la valutazione L (Largely), per l'ottenimento del Livello1.

Essendo richiesta soltanto la verifica di una singola Generic Practise, la valutazione sarà data con una scala numerica, in base a quante Base Practises sono soddisfatte, in maniera da ottenere un dato oggettivo che permetta di assegnare un voto NPLF. Ovviamente tale sistema di valutazione non è l'unico possibile, non essendovi comunque una maniera standard di effettuare la valutazione, e considerando che, in ogni caso, tutti gli outcomes della GP devono essere soddisfatti. I livelli superiori, richiedono altri tipi di valutazioni che vanno oltre il pieno rispetto della GP 1.1, e per i quali il criterio di valutazione può essere diverso se non divenire abbastanza soggettivo. Esistono tre metodi raccomandati dallo standard, ma la loro applicazione, anche nel contesto di questo elaborato, risulta complessa e molto soggettiva. Anche per questo motivo, assieme alla sola GP da verificare, si è optato per il metodo numerico.

Verrà valutato adesso il processo SWE.2. Ricordando il layout con il quale si presentano il Process Reference Model ed i Process Performance Indicators, saranno esaminati gli outcomes, dando giustificazione del loro soddisfacimento o meno secondo quanto previsto dalle BP assegnate per ognuno di essi e sulle quali verte la valutazione. Vi sono 6 outcomes e 9 base practices. Vengono presentati di seguito gli outcomes, seguiti dalle considerazioni fatte relativamente alle base practices.

1) **Viene definita un'architettura del software che identifica gli elementi del software.** Ciò vuol dire che il software deve possedere un'architettura alla base, che permetta di individuare le sue parti e le sue funzionalità. Tale architettura, nel tool, seppur senza essere esplicitamente documentata, come in realtà vorrebbe la BP

SWE.2.BP.1 associata, è resa evidente dalle funzioni Matlab create, nonché dai blocchetti della libreria Simulink e dalle funzioni associate alla GUI. Di conseguenza, pur mancando la documentazione, raccogliendo opportunamente le funzioni, si può identificare l'architettura dello stesso. Ovviamente ciò porta il presente outcome a non essere del tutto soddisfatto, di conseguenza, nell'ottica di una vera implementazione del processo, sarebbe necessario documentare tale architettura.

2) **I requisiti del software sono allocati sugli elementi del software**. Il riferimento è la SWE.2.BP2. Come espresso in precedenza, manca una vera e propria specifica dei requisiti del software, motivo per il quale ne è stata formulata una a scopo esemplificativo nel paragrafo precedente. Tale specifica presenta solo requisiti di altissimo livello, che tuttavia, in accordo con le caratteristiche del tool, mostrano una buona allocazione sulle funzionalità del tool. Pertanto tale outcome può considerarsi soddisfatto. (N.B. Tale affermazione è valida solo nell'ambito di questo elaborato).
3) **Le interfacce di ogni elemento software sono definite**. Il riferimento è la SWE.2.BP3. Le interfacce riguardano i modi con cui le varie parti dell'architettura si parlano tra di loro, e si collegano le une tra le altre. Il linguaggio di programmazione adottato (laddove possa essere ritenuto tale), permette un perfetto matching tra le funzioni di calcolo e le funzioni della GUI, facendo in modo che le informazioni inserite nella GUI possano essere trasferite alla funzione di calcolo, e che gli output possano essere trasferiti alla GUI. Inoltre, la simbologia del Simulink, che riguarda l'uso di blocchetti, fa sì che l'utente possa agire sulla GUI, e che le informazioni inserite possano essere elaborate per fornire gli output di ciascun blocco, necessari per mostrare i dati del calcolo, e per la propagazione dello stesso. I blocchi, tramite il collegamento in ambiente Simulink, si interfacciano tra di loro, permettendo la propagazione dei dati in maniera da arrivare al Top Event. Di conseguenza tutte le interfacce tra gli elementi dell'architettura sono collegate tra di loro.
4) **Il comportamento dinamico e gli obiettivi di consumo delle risorse degli elementi software sono definiti**. Il riferimento è la SWE.2.BP.4. In realtà non vi sono obiettivi sul consumo di risorse del software, data la sua natura di programma di calcolo, diversa da quella del software embedded a cui lo standard fa riferimento. Oltretutto non vi sono dei requisiti riguardo a questo aspetto, soltanto un requisito operativo riguarda il timing del programma, che deve fornire risultati in tempi piccoli. Tale requisito può essere confermato analizzando un modello complesso tramite stress test, verificando che comunque i tempi di calcolo risultino ragionevoli, e che il risultato venga fornito in tempi inferiori al minuto. Data la natura del tool, non è necessario provvedere in futuro alla stesura di una specifica di requisiti su consumo risorse, mentre potrebbe essere richiesta, solo per la competitività commerciale del tool, una specifica di requisiti sul timing. Per l'esempio in questione, l'outcome è parzialmente verificato.
5) **La consistenza e la tracciabilità bidirezionale tra requisiti ed architettura sono stabilite**. Il riferimento è la SWE.2.BP.7. Questo outcome richiede quindi un certo grado di tracciabilità sui requisiti ed il fatto che questi siano legati all'architettura. La tracciabilità bidirezionale copre l'allocazione dei requisiti software agli elementi della progettazione architettonica del software. Tramite la specifica dei requisiti è possibile notare come, almeno al livello più alto, vi sia una sorta di allocazione, seppur marginale, sugli elementi dell'architettura del software. Questo fa si che l'outcome sia in parte soddisfatto, non avendo comunque una specifica dei requisiti dettagliata che consenta di poter verificare quanto descritto. Pertanto, qualora si volesse procedere

con un ulteriore sviluppo, a parte esprimere una specifica dei requisiti completa, bisognerà verificare che ogni requisito sia allocato sull'architettura del software.

6) **L'architettura è accettata e comunicata a tutte le parti interessate.** Il riferimento è la SWE.2.BP.9. Questo outcome richiede che, una volta completata e verificata l'architettura, questa venga comunicata a tutte le parti interessate, che dovranno portare avanti il lavoro. La comunicazione dei risultati è essenziale per l'implementazione del processo, ed effettivamente una descrizione dell'architettura del software, così come una sua comunicazione verso terze parti interessate (relatori, referenti aziendali, studenti e chiunque sia interessato alla consultazione di [1]. Oppure si può fare riferimento al paragrafo 4.1 del presente elaborato dove viene descritto il programma). Pertanto la base practice, anche se in parte, mostra l'applicazione parziale dell'outcome.

Vi sono altre due base practices da considerare, che riguardano l'applicazione a diversi outcomes simultaneamente. Tali base practices sono la SWE.2.BP.6 e SWE.2.BP.8. Richiedono, rispettivamente, di definire dei criteri di valutazione per le architetture alternative e valutarle secondo tali criteri. I criteri di valutazione possono includere caratteristiche di qualità (modularità, manutenibilità, espandibilità, scalabilità, affidabilità, realizzazione della sicurezza e usabilità) e risultati dell'analisi del make-buy-reuse. La seconda chiede di assicurare la consistenza tra software e requisiti.

Per quanto riguarda la SWE.2.BP.6 bisogna evidenziare il fatto che il tool rimane comunque aperto a nuove modifiche, essendo disponibile il codice sorgente e non escludendo i necessari irrobustimenti in futuro. Pertanto la BP è confermata.

Per la SWE.2.BP.8 come più volte ripetuto, in assenza di una specifica dei requisiti vera e propria, è difficile poterne confermare l'applicazione totale, tuttavia, tenendo ad esempio la specifica dei requisiti tracciata sopra, si può confermare l'applicazione della BP in questione.

Quando si applica lo SPICE, le valutazioni in genere sono fatte in maniera parzialmente soggettiva, ed i 3 raiting methods presentati procedono nello stesso senso. In genere è necessario che tutti gli outcomes, e relative BP siano applicati, per avere applicazione del processo, di conseguenza, il fatto che il tool sia rispondente in parte ad alcuni outcomes, e che quindi non risulti completamente implementato, potrebbe generare dei problemi nel caso in cui si parli di livelli superiori all'1. Tale livello, si ricordi, chiede la semplice applicazione delle BP del processo, senza ulteriori valutazioni, quindi, in accordo anche con gli obiettivi del presente elaborato, risulta più semplice effettuare delle valutazioni sull'implementazione del processo e valutare il capability level.

La valutazione è eseguita in maniera matematica, considerando un punteggio per ogni BP considerata, da un minimo di 0 ad un massimo di 1. Il totale corrisponde al numero di BP presenti per il processo considerato. Si andrà poi a calcolare la percentuale di applicazione del processo, ed in base a tale percentuale verrà assegnato un voto secondo scala NPLF. Si ricorda che l'obiettivo per il Livello 1 è ottenere L.

Nella tabella sottostante si tiene traccia della valutazione.

	SWE.2.BP1	SWE.2.BP2	SWE.2.BP3	SWE.2.BP4	SWE.2.BP7	SWE.2.BP9
Punti	0,75	1	1	0,5	0,25	0,75

Questi punteggi, seppur associati agli outcome, fanno riferimento alle BP collegate. Bisogna sommare i punteggi relativi alla BP.6 e BP.8.

	SWE.2.BP.6	SWE.2.BP.8
Punti	0,75	1

Si ha allora, dalla somma, un punteggio pari a 5,325. Tale punteggio deve allora essere rapportato al numero di BPs presenti, cioè 9. Pertanto 5,325/9 = 0,592 da cui si assegna una percentuale del 59,2%. Secondo la scala NPLF, per il processo in questione, si ottiene valutazione **Largely achieved – (L-)**, che comunque cade nell'intervallo relativo a **Largely (L)**. Pertanto, per il processo SWE.2 abbiamo un capability level **Livello 1.** Il processo pertanto è implementato.

Tuttavia, dati i punteggi ottenuti secondo le BP legate a tale processo, è necessario fare ulteriore lavoro affinchè si possa ottenere una completa applicazione del processo, con obiettivo Fully achieved (F).

Bisognerebbe avere a disposizione una specifica dei requisiti più ampia e articolata, da allocare ai livelli più bassi. Sarebbe necessario definire meglio l'architettura, in maniera da procedere con un'allocazione migliore dei requisiti, in maniera da garantire anche la tracciabilità bidirezionale. Non è necessario applicare dei requisiti che riguardino il consumo di risorse, non essendo un software embedded per cui, un consumo di risorse eccessivo può portare al blocco del sistema, ma può essere utile dare qualche requisito per l'aspetto prestazionale. Infine dovrebbero essere prodotti dei report che indichino come l'architettura è approvata. Questi accorgimenti sono necessari per un eventuale e reale processo di certificazione futuro.

Si procede adesso alla valutazione del processo SWE.3. Tale processo interessa il design di dettaglio del software e la programmazione delle sue unità. Lo scopo è quindi provvedere ad una valutazione dettagliata del design per le componenti del tool, specificarle e programmarle. Vi sono 6 outcomes e 8 base practices.

1) **Design e sviluppo dettagliato delle unità del software.** Vi si associa la SWE.3.BP1. Chiede lo sviluppo dettagliato delle unità del software. Questo vuol dire, in riferimento al tool creato, che bisogna aver disegnato e programmato nel dettaglio tutte le unità del software. Questa pratica è stata eseguita dal momento che ogni blocchetto che rientra nella FTA effettuata da tool è stato programmato nel dettaglio per svolgere le funzioni necessarie. L'outcome è implementato.
2) **Interfacce tra le unità del software definite.** Vi si associa la SWE.3.BP2. I blocchetti del tool si interfacciano tramite il software Simulink, che prevede il collegamento tramite frecce a delle opportune porte di input ed output del blocchetto. Pertanto è ben definita la relazione tra queste unità, che sono alla base del tool.
3) **Comportamento dinamico del software definito.** Vi si associa la SWE.3.BP3. Il comportamento dinamico delle unità del software descrive come queste sono in

grado di adattarsi, dinamicamente e quindi in continuo, alla variazione degli input. Un'unità dinamica vede allora variare gli input nel tempo, dando allora risposte differenti nel tempo. Il tool in esame ha invece un comportamento completamente statico. Una volta immessi gli input ed avviata l'analisi, esso restituisce dei risultati. Se si volessero cambiare gli input sarebbe necessario fermare l'analisi, inattivando quindi il tool. Pertanto questo outcome non è applicabile al tool in esame, non essendoci nessun comportamento dinamico da descrivere. Pertanto la BP legata alla sua applicazione sarà esclusa dal calcolo finale.

4) **Consistenza e tracciabilità bidirezionale tra requisiti ed unità del software, architettura del software e design del software, tra design del software e unità del software.** Vi si associano la SWE.3.BP5 e SWE.3.BP6. Il tool non ha meccanismi per la tracciabilità, come espresso in precedenza, pertanto la BP5 non risulta essere applicata. La BP6, invece, può essere facilmente verificata tramite i requisiti, se questi risultino allocati alle unità del software oppure se vi sia rispondenza tra queste e l'architettura. Tutto ciò, ad una prima analisi, mostra l'applicazione, in buona parte della BP, pertanto si ha un parziale soddisfacimento dell'outcome.
5) **Il design di dettaglio del software e le relazioni con l'architettura sono approvate e comunicate alle parti interessate.** Vi si associa la SWE.3.BP7. In questo caso non si ha alcuna approvazione circa il design e la sua relazione con l'architettura. Il tool funziona, in accordo con gli scopi per il quale è stato creato, pertanto si ha un riscontro pratico dell'accordo tra design ed architettura, poiché funziona esattamente nella maniera per la quale è stato pensato. Pertanto questo outcome è parzialmente confermato.
6) **Le unità del software sono state prodotte.** Vi si associa la SWE.3.BP6. In accordo con il design di dettaglio del software, le unità sono state prodotte e funzionano secondo quanto programmato. Pertanto l'outcome è implementato.

La SWE.3.BP4, invece abbraccia più outcomes, e riguarda la valutazione del design del software in termini di interoperabilità, interazione, criticità, complessità tecnica, rischio e testabilità. Tutte queste valutazioni non sono state effettuate, in quanto gli unici test effettuati hanno riguardato la rispondenza degli output con quelli del concorrente commerciale FaultTree+. Pertanto le valutazioni sono state fatte solo ad un livello superficiale, quindi questa BP è debolmente implementata.

La valutazione del processo segue gli stessi criteri adottati per il processo precedente. Unica eccezione il 4 che prevede l'applicazione di più BPs, pertanto avrà un punteggio somma delle due. L'outcome 3, invece, viene escluso dal calcolo poiché non applicabile.

	SWE.3.BP1	SWE.3.BP2	SWE.3.BP3	SWE.3.BP5/6	SWE.3.BP5/7	SWE.3.BP8
Punti	1	1	///////////////	0,5	0,5	1

Alla SWE.3.BP4 viene dato punteggio di 0,25.

Pertanto, si ha un punteggio pari a 4,25, e considerate 7 BPs (la BP3 non è applicabile) si ottiene: 4,25/7 = 0,607 che in percentuale porta ad un 60,7%. Pertanto, in analogia al processo precedente, si ottiene un voto pari a **L-**, pertanto il capability level assegnato è **Livello 1**.

Che il tool non abbia un comportamento dinamico è accettabile, poiché rientra nel suo scopo. Tuttavia è comunque necessario produrre una specifica dei requisiti più accurata, una base di documentazione che certifichi la tracciabilità ed uno strumento di comunicazione delle parti del processo migliore. È anche necessario verificare in maniera più approfondita quanto riportato nella BP4, effettuando delle valutazioni accurate.

Sarà esaminato adesso l'ultimo processo applicabile al tool in esame: SWE.4.

Tratta la verifica delle varie unità software, con lo scopo di fornire evidenze di conformità delle unità del software con il design di dettaglio ed i requisiti non funzionali.

Il processo presenta 5 outcomes e 7 BPs.

1) **Deve essere sviluppata una strategia di verifica, inclusa una strategia di regressione, per verificare le unità del software.** Vi si associa la SWE.4.BP1. Un abbozzo di strategia in tal senso venne evidenziata quando si testò il programma per confermarne il corretto funzionamento, confrontandolo con il corrispettivo commerciale. Furono eseguite diverse prove, che puntarono a valutare la correttezza dei risultati con conseguente test su tutte le unità della libreria Simulink inserite in un unico modello [1]. Meno si può dire per la strategia di regressione, la quale, invece, dovrebbe essere implementata. Di conseguenza l'outcome è quasi del tutto implementato.
2) **Sviluppo dei criteri di verifica in accordo con le strategie di verifica delle unità del software che sono adatte a produrre evidenze dell'accordo tra unità del software e il design dello stesso, e accordo con i requisiti non funzionali.** Vi si associa la SWE.4.BP2. I possibili criteri per la verifica dell'unità includono casi di test unitari, test data, verifica statica, obiettivi di copertura e standard di codifica come le regole MISRA. Come specificato in precedenza, è stato effettuato un mini programma di verifica e validazione che accertasse il funzionamento corretto del software. Tuttavia tali test non hanno seguito un piano bene preciso, se non l'assegnazione di alcune task generiche. Pertanto l'outcome può definirsi non completamente applicato. Il programma sembra però rispettare i requisiti operazionali trattati nella sezione 4.2 (che, si ricorda nuovamente, è soltanto una specifica di esempio), pertanto la seconda parte dell'outcome è applicata.
3) **Le unità del software sono verificate in accordo in accordo con la strategia di verifica, ed i risultati sono registrati.** Vi si associano le SWE.4.BP3 e SWE.4.BP4. La prima BP richiederebbe di effettuare delle verifiche statiche delle unità del software. Tali verifiche prevedono analisi statiche, review del codice, check contro coding standards ed altre tecniche. La seconda BP richiede di testare le unità del software, in accordo ad una strategia di verifica, registrare i risultati e generare dei log. Tutto ciò è stato effettuato in buona parte, in accordo con la strategia di verifica decisa in [1]. Ovviamente non è del tutto completa, poiché sarebbe necessario un piano di verifiche più dettagliato ed articolato. Ma nel complesso, rimanendo all'interno dell'esempio, l'outcome è implementato. Si ricorda anche qui che sarà dato punteggio doppio raccogliendo due BPs.
4) **Consistenza e tracciabilità bidirezionale stabilita tra unità del software, criteri per la verifica e risultati di verifica.** Vi si associano le SWE.4.BP5 e SWE.4.BP6. Le due BPs chiedono rispettivamente di assicurare la tracciabilità

bidirezionale e assicurare la consistenza. Come per i processi precedenti, tale outcome non è implementato, se non in minima parte.

5) **Risultati della verifica riassunti e comunicati a tutte le parti interessate.** Vi si associa la SWE.4.BP7. Fornire tutte le informazioni necessarie dall'esecuzione del test case in un riepilogo consente ad altre parti di giudicare le conseguenze. Questo è stato fatto, poiché grazie ai contenuti presenti in [1] è possibile arrivare alle conclusioni che sono state tratte nel seguito. Di conseguenza l'outcome è implementato.

Si passa al calcolo del capability level per questo processo.

	SWE.4.BP1	SWE.4.BP2	SWE.4.BP3/4	SWE.4.BP5/6	SWE.4.BP7
Punti	0,75	0,25	1,75	0,25	1

Si ha un punteggio di 4 su un totale ottenibile di 7. Pertanto si ottiene 4/7 = 0,57 che in percentuale porta ad un 57%, permettendo, anche in questo caso, di ottenere un **L-**, conferendo il **Livello 1** anche per tale processo.

Nonostante l'assenza di una strategia di verifica chiara e ben articolata, la mancanza di tracciabilità, e l'assenza di documentazione adeguata, anche questo processo ha raggiunto il livello di capacità minimo.

Tuttavia, nel caso in cui si decidesse di procedere per una certificazione, si dovrebbe stilare un piano di verifica più articolato e completo, creare dei record e dei log relativi alle verifiche effettuate ed assicurare la tracciabilità del tutto.

I risultati dell'applicazione dello standard, allora, mostrati sulla falsariga della figura 3.7 sono riassumibili, per il Software Engineering Process Group (SWE) nella seguente maniera.

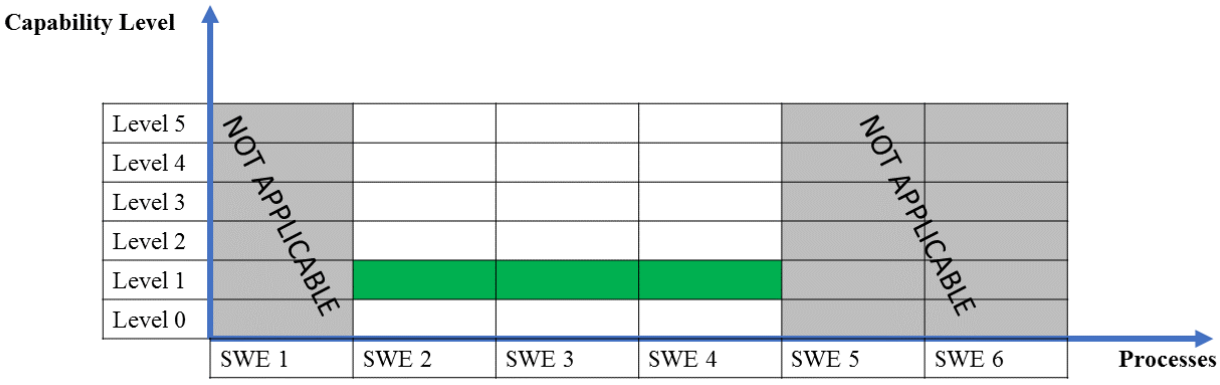

Figura 4.10: Valutazione SPICE

In conclusione, si è dimostrato come, almeno negli scopi di questo elaborato, il processo che ha portato alla scrittura del tool avrebbe una certificazione di qualità. Il Livello 1 è il minimo indispensabile per poter procedere alla certificazione dello stesso, ed è stato ottenuto in maniera risicata secondo una specifica di requisiti che non corrisponde a quella

che ci si aspetterebbe per un programma di tale livello, così come sono assenti documenti che traccino le attività dei processi implementati, e dei piani di verifica articolati, miranti ad identificare e correggere anomalie del tool.

In sostanza, il processo che ha portato al tool è di qualità, ma si consiglia comunque un irrobustimento dello stesso e la stesura di tutta la documentazione necessaria che comprenda: requisiti, architettura, design delle unità del software, tracciabilità bidirezionale, piani di verifica e validazione.

Si ritiene conclusa la parte relativa alla sezione 11.4.8 della ISO 26262-8.

4.3.2 VALIDAZIONE DEL TOOL:

Questo sottoparagrafo tratta il secondo punto della ISO 26262-8 che è stato identificato come necessario per certificare il tool riguarda l'applicazione della sezione 11.4.9.

Ciò che viene richiesto è quello di validare, con opportuni test, il tool, in maniera da verificarne il soddisfacimento dei requisiti, evidenziare malfunzionamenti ed operare delle azioni correttive e verificare la reazione del tool a condizioni di utilizzo improprie (come input out-of-range, configurazioni sconsigliate, carichi di lavoro eccessivi, ecc).

Questo richiede l'effettuazione di test sul tool, riportando i comportamenti dello stesso e confrontando i risultati ottenuti con quelli aspettati in maniera da dimostrare il corretto funzionamento. Si eseguirà a tal proposito, ed in accordo con gli obiettivi del presente elaborato, una serie di pochi test, evidenziandone l'organizzazione, l'esecuzione ed i risultati raggiunti.

A tal proposito può essere utile tracciare un Test plan, che miri ad identificare i difetti ed i malfunzionamenti del tool, suggerendo delle strategie di risoluzione. Un Test Plan ben eseguito può portare ad un miglioramento del tool stesso. In genere, di un test, si riportano: ID, responsabilità, assunzioni, test, comunicazione, analisi del rischio, segnalamento di difetti e ambiente. Un Test Plan, in genere, viene eseguito in maniera schematica, eseguendo prima i test unitari, ossia sulle singole parti del programma, ed infine quelli di sistema, per verificare che le unità siano perfettamente integrate ed il programma funzioni al meglio. Un metodo del genere non è il più realistico, in quanto l'utente potrebbe non usare tutte le funzioni del programma.

Il tool in esame ha già eseguito dei test di tipo unitario che hanno portato a verificare il corretto funzionamento delle operazioni matematiche implementate (vedi [1]), pertanto verranno effettuati test solo a livello di sistema.

Altri tipi di test, oltre ai due sopracitati, riguardano i test di integrazione, che mirino ad identificare che il programma funzioni anche in un ambiente diverso da quello nativo. I test di regressione, invece, sono eseguiti ogni volta che un pezzo di programma viene modificato in seguito all'identificazione di un difetto.

- **Verifica soddisfacimento dei requisiti**:

  All'inizio del paragrafo 4.1 è stata stilata una specifica dei requisiti. Nel precedente paragrafo si è detto che il tool, già sembra soddisfare i requisiti operativi, e questo nasce dal fatto che la specifica è stata scritta a posteriori, conoscendo il comportamento del tool. Adesso si vuole però dare una dimostrazione pratica di quanto affermato.

I test saranno eseguiti al livello di sistema, ed andranno a soddisfare tutti i requisiti espressi nella specifica delle 4.1. In genere è consigliabile eseguire un test per requisito, in maniera da verificarli singolarmente, tuttavia, trattandosi di un mero esempio pratico, saranno svolti pochi test: un validation test, uno stress test ed un interaction test. Questo poiché si tiene in conto la semplicità dei requisiti riportati. Una volta che sarà resa disponibile una specifica più dettagliata, tali test dovranno essere eseguiti in maniera completa e con la giusta documentazione a supporto.

Il primo esempio permette di verificare tutti i requisiti che richiedono un'analisi come metodo di verifica, eccetto F08 ed OP01 che invece possono essere verificati attraverso uno stress test, che verrà eseguito successivamente.

L' esempio è tratto dal mondo ferroviario, in particolare si tratta del sistema freno di stazionamento di un treno, che deve evitare l'avanzamento del treno in deposito. Il validation test è prima eseguito in ambiente FaultTree+ e dopo con il tool in esame, in maniera da avere un parametro di confronto e, nello stesso tempo, soddisfare il requisito OP07.

Di seguito l'analisi effettuata con il tool commerciale:

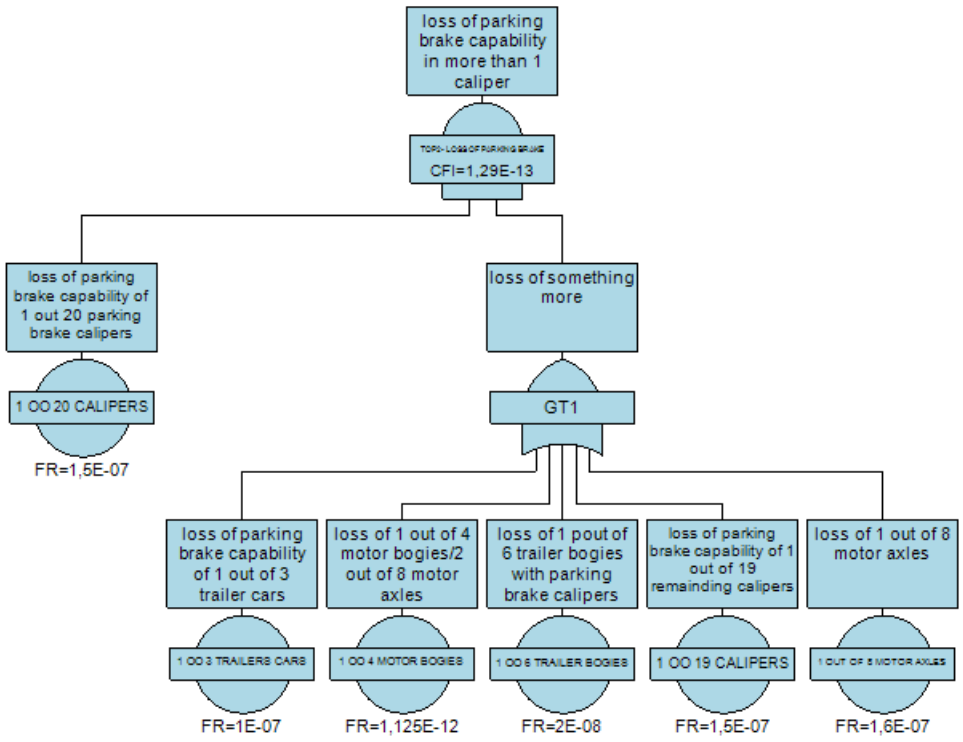

Figura 4.11: FTA su sistema frenante in ambiente FaultTree+

L'albero presenta 5 basic events, di cui sono riportati i failure rate, più un altro basic event il cui effetto viene moltiplicato all'uscita della porta OR.

Il test è un'analisi di indisponibilità del sistema, su 18 ore di funzionamento con MTTR (Mean Time To Repair) di un'ora.

L'obiettivo è ricreare tale albero nel tool, e verificare che, oltre al soddisfacimento dei requisiti, sia verificata anche la correttezza degli output, elemento essenziale per la certificazione dello stesso.

Nel seguito l'esempio costruito con il tool in esame.

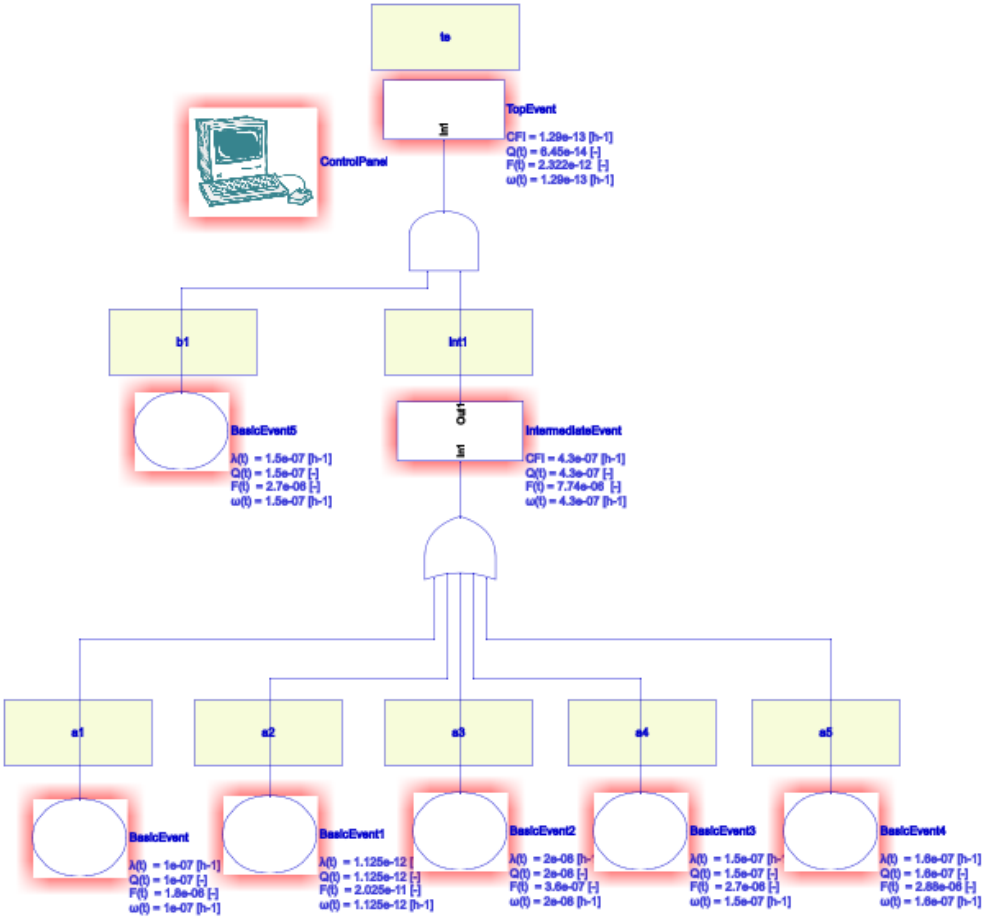

Figura 4.12: FTA su sistema frenante in ambente tool Matlab-Simulink

Il tool ha una rappresentazione grafica diversa rispetto a quella del FaultTree+, anche in termini grafici: nel tool sono presenti per ogni basic event i valori di failure rate, unavailability, unreilability e failure density. Tali valori, a seconda del tipo di modello scelto e del tipo di analisi che si vuole eseguire, possono essere inseriti tramite GUI, come in un normale blocco Simulink. Anche il numero di entrate nelle porte logiche può essere settato nella stessa maniera. Il control panel, invece, permette di scegliere il tipo di analisi, denominare il modello, avviare l'analisi ed esportare/importare i dati formattati in Excel. Le etichette sopra ogni basic events permettono la descrizione dell'evento, mostrando solo il nome associato ad esso. Il top event, oltre a mostrare i risultati, permette anche di visualizzare l'equazione del cutset.

Seppur la verifica dei requisiti richieda un singolo test per requisito, per ragioni di spazio, e poiché si tratta comunque di un esempio pratico, il validation test effettuato ha permesso di verificare i requisiti presenti nella specifica della sezione 4.2, ad eccezione dell'F08 e OP01 che invece saranno dimostrati nello stress test successivo.

Di seguito viene stilato un test plan esemplificativo, riportando le informazioni essenziali del test e l'esito dello stesso. Seguirà poi una tabella dove saranno riportati

gli esiti della verifica dei requisiti, documento che andrebbe allegato al software tool qualification report.

VALIDATION TEST

ID: TEST01

RESPONSABILE:
Validatore/Verificatore del programma;
ASSUNZIONI:
Tool completamente disponibile. Si presenta come una libreria Simulink;
TEST:
Il test mira ad accertare il soddisfacimento dei requisiti verificabili per analisi presenti nella specifica. In particolare sarà eseguito un solo test che però porti al suo interno tutti gli aspetti da dimostrare. Si creerà un modello contenente tutti gli aspetti che si vogliono verificare e si dimostrerà il corretto funzionamento del tool.
COMUNICAZIONE:
Ente certificatore, azienda produttrice del software;
ASPETTI CRITICI:
Corretta propagazione dell'analisi dal cutset al top event. Risultati numerici in accordo con software terzo.
SEGNALAMENTO DIFETTI:

- *Bordatura di colore rosso dei blocchi, dovuto alle impostazioni della libreria, per evitare manipolazioni dei blocchi di default accidentali. Non influisce sul calcolo.*
- *Fallito salvataggio delle etichette riportanti il nome del basic event. È necessario rieditare dalla GUI associata;*
- *Modello non riconosciuto se non precedentemente salvato con il nome associato all'albero;*

AMBIENTE:
Matlab-Simulink ver. 2018a, OS Windows 7 Ultimate 64 bit

Figura 4.13: Validation Test Plan

Il test ha avuto esito positivo, in quanto è stato possibile dimostrare il corretto funzionamento del tool ed il corretto soddisfacimento dei requisiti, oltre che la correttezza degli output al confronto col software commerciale. La criticità non ha dato evidenza durante l'esecuzione del test. I difetti riscontrati sono di lieve entità e non inficiano il corretto funzionamento del tool, anche se si consiglia una correzione se tale tool vuol essere certificato per uso commerciale.

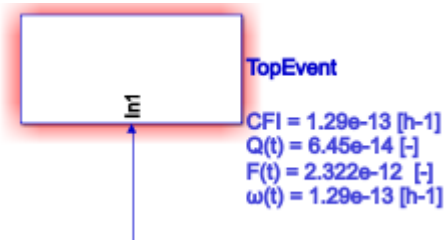

Figura 4.14: Risultato del Validation Test, in accordo col test in figura 4.11

Lo stress test, invece, mira a verificare il comportamento del tool ai limiti delle sue capacità, ed in tal caso ad evidenziare malfunzionamenti o comportamenti anomali che potrebbero essere oggetto di misure correttive. Tale test, a differenza del caso precedente, prevede un albero non realistico contenente 20 basic events collegati ad una porta OR. L'obiettivo è verificare che i risultati ottenuti siano, anche in questo caso, in accordo con il software commerciale usato come elemento di paragone.

Di seguito il Test Plan apposito.

STRESS TEST

ID: TEST02

RESPONSABILE:
Validatore/Verificatore del programma;
ASSUNZIONI:
Tool completamente disponibile. Si presenta come una libreria Simulink;
TEST:
Il test mira a verificare il comportamento del tool sotto stress, usando un cutset numeroso, dal quale si deve ottenere il corretto output. Sono utilizzati 20 basic events. I basic events saranno tutti uguali per ragioni di controllo.
COMUNICAZIONE:
Ente certificatore, azienda produttrice del software;
ASPETTI CRITICI:
Corretta propagazione dell'analisi dal cutset al top event. Risultati numerici in accordo con software terzo.
SEGNALAMENTO DIFETTI:
- *Impossibilità di usare cutsets con più di 20 basic events;*

AMBIENTE:
Matlab-Simulink ver. 2018a, OS Windows 7 Ultimate 64 bit

Figura 4.15: Stress Test Plan

Il test ha dato esito positivo, l'informazione ha propagato in maniera corretto ed il risultato corrisponde a quello ottenuto con in tool commerciale. Il test ha inoltre permesso di identificare il numero massimo di eventi che il tool è in grado di gestire, corrispondente a 20. Al di fuori il tool non propaga mostrando errori sulla command window di Matlab.

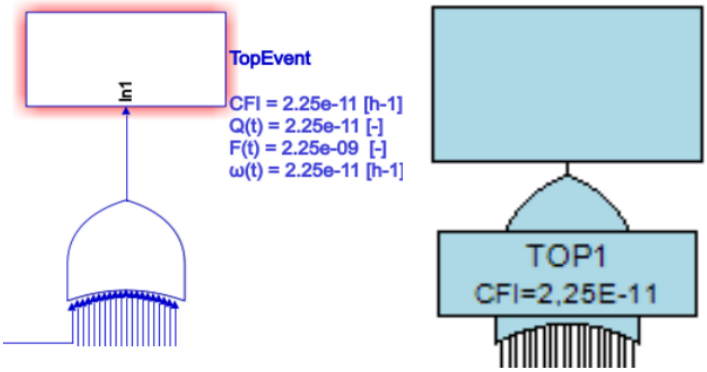

Figura 4.16: Risultato stress test

Il test successivo è l'interaction test, per verificare il soddisfacimento dei requisiti operazionali che riguardano l'interazione con l'esterno (esportazione/importazione dati, formattazione automatica dei fogli Excel, ecc). Tali test possono essere eseguiti sia nell'ambito dell'esempio riguardante il primo test, sia nel secondo esempio. L'obiettivo è validare i requisiti che hanno come metodo l'ispezione. Di seguito il Test Plan complessivo.

INTERACTION TEST

ID: TEST03

RESPONSABILE:
Validatore/Verificatore del programma;
ASSUNZIONI:
Tool completamente disponibile. Si presenta come una libreria Simulink;
TEST:
Il test viene eseguito utilizzando i comandi che permettono di esportare ed importare i dati inseriti/da inserire nel tool per il calcolo. Il tool deve poi permettere di salvare e rendere disponibile il lavoro ogni qualvolta venga richiesto.
COMUNICAZIONE:
Ente certificatore, azienda produttrice del software;
ASPETTI CRITICI:
- *Salavtaggio degli alberi come modelli;*
- *Corretta formattazione dei file in uscita.*

SEGNALAMENTO DIFETTI:

AMBIENTE:
Matlab-Simulink ver. 2018a, OS Windows 7 Ultimate 64 bit

Figura 4.17: Interaction Test Plan

Il test ha avuto esito favorevole, i dati sono stati esportati come formato Excel adeguatamente formattati e disponibili per l'uso. Il modello dell'albero viene salvato e reso disponibile alla riapertura, seppur con la necessità di rieditare le etichette assegnate ai basic events.

	A	B	C	D	E
1	BASIC EVENTS				
2	Basic Event ID	Event Label	Q(t) [-]	F(t) [-]	ω(t) [h^{-1}]
3	BasicEvent	a1	1,00E-07	1.8e-06	1,00E-07
4	BasicEvent1	a2	1,13E-09	2,03E-08	1,13E-09
5	BasicEvent2	a3	2,00E-08	3.6e-07	2,00E-08
6	BasicEvent3	a4	1.5e-07	2.7e-06	1.5e-07
7	BasicEvent4	a5	1.6e-07	2.88e-06	1.6e-07
8	BasicEvent5	b1	1.5e-07	2.7e-06	1.5e-07

Figura 4.18: Esempio di output relativo ai dati inseriti nei basic events dell'albero in figura 4.12

	A	B	C	D	E	F	G	H	I	J	K
1	BASIC EVENTS										
2	Basic Event ID	Event Label	Model Type	Component Type	Failure rate λ [h^{-1}]	MTTR [h]	Mission Time T [h]	Test Interval τ [h]	Q(t) [-]	F(t) [-]	ω(t) [h^{-1}]
3	BasicEvent										
4	BasicEvent1										
5	BasicEvent2										
6	BasicEvent3										
7	BasicEvent4										
8	BasicEvent5										

Figura 4.19: File Excel formattato per essere preso in input dal tool

I tre test come già scritto più volte, miravano al soddisfacimento dei requisiti formulati nella specifica della sezione 4.2. Nel seguito si propone una tabella che presenta gli esiti della verifica dei requisiti, riportando delle note nel caso in cui il tool soddisfa il requisito solo in maniera parziale.

Nel caso in cui un requisito non fosse pienamente soddisfatto o soddisfatto affatto, vi è il rischio di un rifiuto da parte dell'ente certificatore, con conseguente obbligo di modifica in maniera da rendere pienamente soddisfatto il requisito. Il soddisfacimento dei requisiti è fondamentale per la certificazione del tool.

ID	DESCRIZIONE	TEST	ESITO	NOTE
F01	Il tool deve poter girare su diversi sistemi operativi	TEST01	Passato	/
F02	Il tool deve essere in grado di creare alberi di guasto complessi	TEST02	Passato	/

F03	L'utente deve poter inserire i dati attraverso una GUI	TEST01	Passato	/
F04	Il tool deve poter esportare i dati inseriti	TEST01	Passato	/
F04	Il tool deve poter esportare i dati inseriti	TEST03	Passato	/
F05	Il tool deve poter importare i dati dall'esterno	TEST03	Passato	/
F06	Il tool deve poter mostrare i risultati dell'analisi e l'equazione logica del cutset	TEST01	Passato	/
F07	L'utente deve poter salvare il lavoro e riprenderlo in qualsiasi momento senza perdere nessun dato	TEST03	Parzialmente Passato	Le etichette memorizzano ma non visualizzano il tag dell'evento assegnato
F08	Il tool deve poter gestire modelli complessi	TEST02	Parzialmente Passato	Lavora con un numero adeguatamente alto di basic events, tuttavia non supporta alberi molto strutturati
F09	L'utente deve poter aprire la GUI cliccando su ogni blocchetto	TEST01	Passato	/
OP01	Il tool deve poter gestire fino a 20 basic events	TEST02	Passato	Limite massimo accettato dal tool
OP02	Il tool deve poter calcolare i parametri necessari all'analisi autonomamente	TEST01	Passato	Si intende il corretto funzionamento dei command button
OP03	Il tool deve poter propagare l'analisi in tempi piccoli	TEST02	Passato	/
OP04	Il tool deve mostrare i valori ai livelli intermedi senza errori	TEST01	Passato	/
OP05	Il tool deve mostrare messaggi di errore nel caso di uso improprio	TEST02	Passato	/
OP06	Il tool deve creare, quando richiesto, fogli di calcolo pre-formattati	TEST03	Passato	/
OP07	I risultati non devono differire fino alla quarta cifra significativa da	TEST01	Passato	/

	quelli ottenuti con un tool commerciale			
OP08	Le porte logiche devono poter ricevere almeno 5 input	TEST02	Passato	/

Tramite i tre test effettuati è stato possibile dimostrare la rispondenza del tool alla specifica dei requisiti. Ovviamente, ed è bene sottolinearlo ancora, tale specifica è stata creata dall'autore del presente elaborato in violazione a quanto richiesto dalla normativa, pertanto una rispondenza a tali requisiti non comporta la certificabilità del tool. Tuttavia, se fosse pienamente corretto l'approccio seguito, già da adesso si può ritenere il tool certificabile, ma con dei limiti. Il requisito F07, che non viene rispettato appieno, non inficia il corretto funzionamento del tool, pertanto non influenza gli output, ma l'F08 non permette la creazione di alberi molto strutturati, inficiando le capacità del tool. Segnalare tale limite in un manuale, in maniera da evitare errori dell'utente, non influenza gli output.

- **Verifica dei malfunzionamenti**:

Altro aspetto della validazione è l'identificazione dei malfunzionamenti del tool e la loro raccolta, con i conseguenti output errati e le conseguenze degli stessi. Tali malfunzionamenti possono avere delle conseguenze sul calcolo, e se non corretti possono portare ad output errati che, di conseguenza, possono avere degli effetti sul prodotto.

Nel corso della validazione sono stati eseguiti 3 test, su un tool il cui funzionamento era già stato dimostrato in [1]. Tuttavia è stato possibile identificare dei comportamenti non corretti del tool in alcune condizioni di lavoro. Tali comportamenti tendono ad inficiare la propagazione del risultato, avendo l'aspetto di un vero e proprio malfunzionamento.

1) Un primo malfunzionamento si riscontra nel caso in cui il modello non venga salvato prima di lanciare l'analisi. Il tool non riconosce l'albero, di conseguenza non solo il calcolo non propaga, ma si perviene ad un blocco dello stesso con conseguente log dell'errore su command window di Matlab. Questo non genera un output scorretto, poiché il calcolo non riesce a propagare fino al top event.
2) Un secondo malfunzionamento lo si ha con alberi complessi. In tal caso il tool riesce a propagare una parte dell'albero, arrestandosi poco dopo, e mostrando un messaggio di errore. Tale errore blocca il calcolo, non permettendo il propagare dell'informazione fino al Top Event. Purtroppo, al momento della scrittura di questo elaborato, non è stato identificato il motivo di tale malfunzionamento, che ha come conseguenza una forte limitazione sulle capacità del tool, e su un suo possibile uso commerciale. L'indicazione che si dà in tal caso è quella di usare alberi non troppo complessi e poco strutturati, aspetto che dal punto di vista commerciale rappresenta una forte limitazione.
3) Ulteriore malfunzionamento riscontrato riguarda il numero di basic events utilizzabili. Il tool, in teoria, dovrebbe gestire un numero qualsiasi di basic events al di là del requisito. Tuttavia, durante la validazione, è stato riscontrato un malfunzionamento qualora venga inserito un numero maggiore di 20 basic events.

Anche in questo caso il programma arresta la propagazione mostrano un log dell'errore nella command window. Anche per questo tipo di malfunzionamento, al momento della scrittura dell'elaborato, non è stata trovata soluzione. Come per il punto precedente, questo incide sulle prestazioni del tool. Seppur un uso di più di 20 basic events sia raro, un tool commerciale dovrebbe poter gestire tutti i basic events che l'utente desidera inserire. Questo limite, assieme al precedente, dovrebbe essere corretto con un irrobustimento futuro del tool, che miri a correggere ed ampliare le capacità dello stesso.

- **Uso del software in condizioni improprie**:

La 11.4.9 richiede, come ultima analisi, di verificare il comportamento del software se usato in condizioni e maniere improprie. Tali situazioni si possono verificare quando l'utente commette degli errori in maniera accidentale o voluta. Il tool deve reagire a tali situazioni, mostrando messaggi di errori mirati ed evitando la propagazione di tali errori inseriti durante la fase di utilizzo. In altre parole, il tool deve risultare robusto.

Vi sono dei test che puntano a verificare questi tipi di comportamento, iniettando degli errori e verificando che il software riconosca tali errori arginandoli o, eventualmente, bloccando il calcolo e avvertendo l'utente.

A tal proposito può essere eseguito qualche test per osservare il comportamento del tool, e valutare se questo è corretto o meno. La validazione precedente ha accertato che il tool restituisce valori corretti in base ai tipi di input inseriti, tramite confronto con un prodotto commerciale. Il comportamento del tool non è intelligente, come del resto ci si aspetta da un programma di questo tipo: il suo compito non è riconoscere se l'input è corretto o meno (salvo formattazione non idonea), ma far propagare il calcolo per ottenere i valori nel Top Event. Pertanto, a meno che i numeri inseriti non presentino errori di formato, lui è in grado di eseguire il calcolo.

Un test facilmente eseguibile riguarda la creazione di alberi con parti mancanti. Si può verificare come reagisce il tool nel momento in cui l'utente non inserisce degli elementi necessari alla propagazione come gli Intermediate Events. Questi vengono sfruttati come dei contenitori all'uscita da una porta logica, lavorando come un basic event per il livello successivo dell'albero.

Un altro test, invece, mirerà ad identificare il comportamento del software nel momento in cui non sia stato correttamente inizializzato un basic event. Questo, assieme al precedente, è l'errore più comune nel quale può incorrere l'utente.

È necessario comunque anticipare che il tool non prevede la comparsa di messaggi di errore a schermo, pertanto è necessario identificare la tipologia degli stessi dal log nella command window. Gli errori, a differenza dei malfunzionamenti possono essere corretti in qualsiasi momento durante l'uso del tool, pertanto, a differenza dei malfunzionamenti che non dipendono dall'utente, non necessitano di correzioni da parte dell'azienda realizzatrice del software, ma vanno indicati sul manuale d'uso.

Anche per questi test, è bene avere un adeguato Test Plan.

ERROR TEST

ID: ERR01

RESPONSABILE:
Validatore/Verificatore del programma;
ASSUNZIONI:
Tool completamente disponibile. Si presenta come una libreria Simulink;
TEST:
Il test si esegue introducendo un errore nel tool. Viene volutamente omesso il blocco relativo all'intermediate event per osservare il comportamento del tool.
COMUNICAZIONE:
Ente certificatore, azienda produttrice del software;
ASPETTI CRITICI:
- *Risposta ad una configurazione del modello errata.*

SEGNALAMENTO DIFETTI:

AMBIENTE:
Matlab-Simulink ver. 2018a, OS Windows 7 Ultimate 64 bit

Figura 4.20: Error Test Plan (assenza di intermediate events)

Il test ha messo in evidenza come la mancanza di tale elemento in un albero porti all'immediato blocco dell'esecuzione con conseguente log. Pertanto l'utente ha modo di rendersi conto della presenza di un errore. Tuttavia sarebbe consigliabile inserire una finestra che mostri all'utente, soprattutto se inesperto nell'uso del tool, la tipologia di errore.

Le immagini mostrano l'albero e la tipologia di errore.

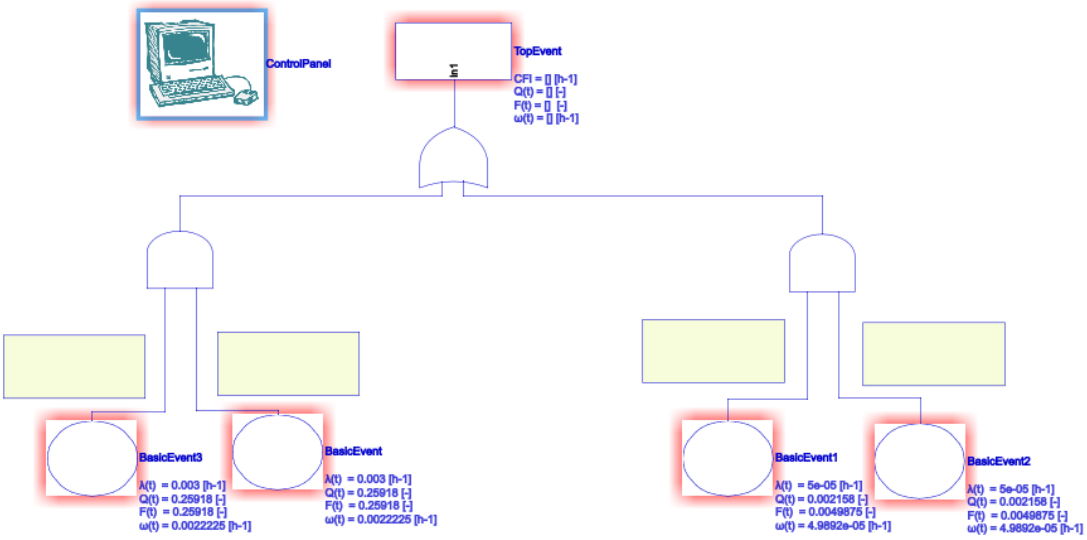

Figura 4.21: Albero senza Intermediate Events

```
Index exceeds array bounds.

Error in ForSym_TopEv (line 61)
     sum=str2sym(Sym_Evs{1});

Error in ControlPanelGui>pushbutton1_Callback (line 265)
TOP_ext=ForSym_TopEv(model_name); % Top di tipo scringa

Error in gui_mainfcn (line 95)
        feval(varargin{:});

Error in ControlPanelGui (line 42)
    gui_mainfcn(gui_State, varargin{:});

Error in
matlab.graphics.internal.figfile.FigFile/read>@(hObject,eventdata)ControlPanelGui('pushbutton1_Callback',hObject,eventdata
Error while evaluating UIControl Callback.
```

Figura 4.22: Log errore dovuto a mancanza Intermediate Events

ERROR TEST

ID: ERR02

RESPONSABILE:
Validatore/Verificatore del programma;
ASSUNZIONI:
Tool completamente disponibile. Si presenta come una libreria Simulink;
TEST:
Il test si esegue introducendo un errore nel tool. Viene volutamente omesso il riempimento di un basic event, per valutare come il tool reagisce a tale mancanza.
COMUNICAZIONE:
Ente certificatore, azienda produttrice del software;
ASPETTI CRITICI:
- *Risposta alla mancanza di dati.*

SEGNALAMENTO DIFETTI:

AMBIENTE:
Matlab-Simulink ver. 2018a, OS Windows 7 Ultimate 64 bit

Figura 4.23: Error Test Plan (Basic Event vuoto)

Un basic event vuoto vede i valori di λ, Q(t), F(t) e ω(t) prendere il valore di NaN (Not a Number). Questo permette sin da subito, con un'ispezione visiva, all'utente di accorgersi dell'errore. Nel momento in cui l'utente fosse disattento, e lanciasse la propagazione, il tool blocca il calcolo e mostra un log di errore nella command window. A differenza del caso precedente, dove il tool sin da subito arresta la propagazione, nella situazione attuale prende i valori immagazzinati nel cutset, e arresta la propagazione al momento della prima porta logica incontrata in cui NaN non riesce ad essere elaborato.

Di seguito il log dell'errore.

```
Index exceeds array bounds.

Error in ForSym_TopEv (line 61)
     sum=str2sym(Sym_Evs{1});

Error in ControlPanelGui>pushbutton1_Callback (line 265)
TOP_ext=ForSym_TopEv(model_name); % Top di tipo scringa

Error in gui_mainfcn (line 95)
        feval(varargin{:});

Error in ControlPanelGui (line 42)
    gui_mainfcn(gui_State, varargin{:});

Error in
matlab.graphics.internal.figfile.FigFile/read>@(hObject,eventdata)ControlPanelGui('pushbutton1_Callback',hObject,eventdata
Error while evaluating UIControl Callback.
```

Figura 4.24: Log errore dovuto a mancato riempimento del basic event

Ovviamente i due test effettuati non sono esaustivi ai fini della certificazione di un tool. Come più volte riportato, la certificazione va oltre i fini del seguente elaborato, e questi due test sono un esempio sulla falsariga della moltitudine di combinazioni che dovrebbero essere testate in maniera da avere un quadro chiaro delle capacità del tool e dei possibili errori riscontrabili.

I dati rilevati ed i risultati dei test effettuati saranno poi inseriti nel manuale utente del programma, da fornire obbligatoriamente con lo stesso.

Con l'applicazione dei due paragrafi della norma ISO 26262-8 si conclude l'iter della procedura di certificazione del tool. I prodotti sono due report:

- **Software tool evaluation criteria report;**
- **Software tool qualification report;**

Contengono al loro interno la specifica dei requisiti, le prove effettuate per dimostrare il soddisfacimento degli stessi, le prove di validazione, i malfunzionamenti riscontrati e la maniera in cui è possibile porre rimedio, le procedure atte ad evitare gli errori degli utenti e le conclusioni del validatore/verificatore. L'applicazione dello standard SPICE certifica che il processo seguito nella produzione del codice e nell'implementazione delle funzionalità del tool è di qualità.

Ai due report fanno riferimento gli stessi punti della normativa, ma vi sono alcune differenze sul contenuto. Il primo report riporta, oltre ai dati identificativi del tool, tutti i criteri usati per effettuarne la valutazione, dalla scelta del Tool Confidence Level ai punti della norma adottati per valutarlo. Si presenta allora come un summary del procedimento adottato, senza descrivere nel dettaglio i test e le misure intraprese. Tali dettagli sono invece parte del secondo documento che, oltre a riprendere le informazioni contenute nel primo, descrive anche le procedure adottate, i test effettuati e le conclusioni.

Entrambi i documenti rappresento il workproduct del procedimento, e devono essere resi disponibili all'ente certificatore che si occuperà di controllare la rispondenza del tool a quanto richiesto dalla normativa, rilasciando l'attestato di certificazione.

Per il tool in oggetto, i paragrafi 4.3.1 e 4.3.2 del presente elaborato, contengono le informazioni che dovrebbero essere inserite nei suddetti report, pertanto si è ritenuto di non procedere con un esempio di stesura degli stessi.

4.4 APPLICAZIONE DELLA DO-178B:

Nel paragrafo 3.2 sono stati trattati gli aspetti centrali della normativa DO-178B relativi alla qualificazione del tool. Tali aspetti sono stati trattati in maniera generale, riportando cosa richiesto dalla normativa, ma non in maniera specifica per il tool in oggetto.

Si ricorda che la normativa tratta due diverse tipologie di tool, che vale la pena richiamare:

- Development tools: software che permettono di automatizzare le procedure in fase di sviluppo del software embedded, e che possono introdurre errori in quest'ultimo. Alcuni esempi possono essere generatori automatici di codice (Simulink è uno di questi), compilatori, linker, ecc. Il prodotto di un development tool è un software che vola col velivolo, e pertanto necessita di grandi attenzioni in fase di realizzazione ai fini della sicurezza.
- Verification tools: software non utilizzati nel processo di sviluppo, e che pertanto non introducono errori, ma possono fallire nell'identificarli. Tali tools si occupano di verificare che gli output di un development tool, o di un software siano corretti, senza che vengano eseguite ulteriori verifiche. Il software non vola assieme al velivolo, ma devono dare evidenza che il software sviluppato con i precedenti sia compliant coi requisiti e standard. Un esempio di verification tool utilizzato nel ciclo di vita del prodotto aeronautico è Reqtify, un tool che si occupa della gestione della specifica dei requisiti, allocandoli, creando alberi e verificando il loro soddisfacimento. Tale tool non necessita di un'ulteriore verifica a valle dei suoi output che pertanto vengono accettati automaticamente.

È naturale allora considerare i development tools come più "pericolosi" ai fini del progetto, e pertanto il loro processo di qualifica risulta essere più lungo e articolato rispetto ad un verification tool.

La normativa, che al paragrafo 12.2 contiene tutte le indicazioni riportate al paragrafo 3.2 del seguente elaborato, e che sono alla base di questo capitolo, chiede che ogni tool usato per automatizzare i processi del ciclo di vita del software, debba essere qualificato secondo una delle due categorie sopra riportate.

L'obiettivo principale di questo elaborato è dimostrare se il tool FTA possa essere certificato o meno. Questo dipende anche dall'applicabilità della norma alla quale si sta facendo riferimento. Sono state già presentate le funzionalità di tale tool, così come nel capitolo 1 è stata introdotta la Fault Tree Analysis. Questa analisi, e di conseguenza il tool, servono a dimostrare la sicurezza di un sistema già dalla fase progettuale: una volta ottenuti gli schemi di un sistema, si esegue questa analisi di tipo probabilistico e si verifica se la probabilità di avere un guasto, oppure i valori di affidabilità, disponibilità, ecc, siano in accordo con quanto prescritto nei requisiti. L'analisi lavora allora più su un'idea che su un oggetto concreto quale può essere anche il software del velivolo. Oltretutto, per quanto possano essere attendibili i risultati dell'analisi, si avrà comunque la necessità di verificare tali output una volta disponibile il sistema.

Il tool quindi non è un development tool, in quanto non interviene nella fase di sviluppo del software embedded, e pertanto non può introdurre, secondo la definizione della normativa, errori sistematici nel codice. Ma non è nemmeno classificabile come verification tool, poiché i suoi output saranno comunque verificati a valle una volta che si rende disponibile il sistema.

Sulla base di quanto affermato, **la normativa DO-178B non è applicabile.**

Questa affermazione è già un risultato, poiché la non rispondenza alle caratteristiche richieste dalla normativa, determina a priorità il fallimento del processo di qualifica, con conseguente rigetto da parte dell'autorità deputata alla qualifica.

Nel seguito si cercherà di dimostrare la veridicità di questa affermazione, e come il tool fallisca nel processo di qualifica. Per fare questo, si assume, per **assurdo**, che il tool abbia le caratteristiche di un verification tool. Questo ci porta ad applicare allora le misure previste dalla normativa per questa tipologia.

La normativa chiede che, se le funzioni sono eseguite in contemporanea (sviluppo e verifica come in un compilatore dotato di debugger), allora bisogna trattare il tool come un development tool e applicare quanto richiesto per questa categoria. Tuttavia, se le funzioni sono perfettamente separate, allora si può operare usando le due strategie. Il tool in esame lavora sin dalle prime fasi di sviluppo, eseguendo analisi di tipo RAMS sul prodotto. È già stato descritto come ciò non rientri nei parametri che portano a ritenerlo un verification tool. Per ritenerlo tale i suoi output dovrebbero avere una confidenza tale da non implicare verifiche successive. Questo non è vero, poiché seppur le analisi sono condotte su un modello preciso, l'utente potrebbe introdurre degli errori nell'albero che darebbero evidenza solo nella fase successiva di prototipazione del sistema, verificando a sua volta l'output. Tuttavia tale funzione è separata da quella di un development tool, ed interpretando letteralmente la norma è possibile applicare la più snella procedura per i verification tools.

Il primo passo da compiere è verificare l'implementazione del software quality process, successivamente si vedrà l'applicazione del software management process.

4.4.1 SOFTWARE QUALITY ASSURANCE PROCESS:

A differenza della normativa ISO 26262, che per la certificazione della qualità si appoggia ad uno standard esterno come lo SPICE, la normativa aeronautica possiede al suo interno tutte le indicazioni necessarie per qualificare il tool.

La normativa, su questo aspetto, considera il software come embedded, di conseguenza, e analogamente a quanto fatto con lo SPICE, è necessario estrapolare tutte quelle richieste applicabili al tool in esame, e classificando come non applicabili le richieste specifiche per il software embedded.

Gli obiettivi per la SQA (Software Quality Assurance) prevedono:

- Il processo di sviluppo del software ed i processi integrali sono rispondenti a progetti e standards approvati;
- I criteri di transizione per il processo del ciclo di vita del software sono soddisfatti;
- Che sia condotta una review di conformità.

Appare subito evidente che il tool non possiede né progetti né è stato creato con standard approvati. Non trattandosi di software che ha applicazione diretta sulla sicurezza, non se ne è

sentito il bisogno al momento della creazione, seppur sia stato comunque necessario eseguire una pianificazione dello stesso (funzioni, interfaccia grafica, integrazione con simulink, ecc). È pur vero che anche in aeronautica vi sono degli standards che indichino in che maniera deve essere scritto un codice. A riferimento esistono la ISO/IEC 12207 per i processi del ciclo di vita, ISO 9126 per gli aspetti qualitativi oltre i vari derivati di tipo MIL o ECSS. Al momento della produzione del tool, non è stato identificato nessuno standard applicabile a questa tipologia, tale per cui anche le sopracitate normative non sono state applicate.

Pertanto non si ha alcuno standard o riferimento normativo al quale il codice si ispira.

Questa rappresenta già una grossa limitazione circa il primo obiettivo da perseguire per la SQA. Non a caso, già con l'applicazione dello SPICE si erano evidenziate talune criticità, aggirate in maniera fortuita nella fase di valutazione, prendendo ad esempio la tesi al riferimento [1]. Sarebbe allora necessario, come era anche stato richiesto al termine del paragrafo relativo allo SPICE, rivedere la scrittura del software e renderla in accordo con uno standard, sia esso lo stesso SPICE.

Si può fare riferimento alla conformità evidenziata in precedenza allo standard SPICE. È comunque uno standard approvato, seppur di ramo automotive, quindi non si ha la certezza che l'autorità deputata alla valutazione del tool lo accetti. Questo determina il fallimento del primo obiettivo.

Il secondo obiettivo richiede di stabilire i criteri di transizione relativi al ciclo di vita del software. La normativa in esame riguarda il ciclo di vita del software embedded e come questo debba essere prodotto per integrarsi nel sistema velivolo. Tuttavia esiste la possibilità che anche per un tool come quello in esame si segua un ciclo di vita, come descritto dalla ISO 12207. La non applicazione di tale normativa, rende il tool non conforme al secondo obiettivo.

Anche il terzo obiettivo può ritenersi fallito, non essendo stata fatta alcuna review di conformità.

76

Table A-9
Software Quality Assurance Process

	Objective		Applicability by SW Level				Output		Control Category by SW level			
	Description	Ref.	A	B	C	D	Description	Ref.	A	B	C	D
1	Assurance is obtained that software development and integral processes comply with approved software plans and standards.	8.1a	●	●	●	●	Software Quality Assurance (SQA) Records	11.19	②	②	②	②
2	Assurance is obtained that transition criteria for the software life cycle processes are satisfied.	8.1b	●	●			SQA Records	11.19	②	②		
3	Software conformity review is conducted.	8.1c 8.3	●	●	●	●	SQA Records	11.19	②	②	②	②

LEGEND:		
	●	The objective should be satisfied with independence.
	○	The objective should be satisfied.
	Blank	Satisfaction of objective is at applicant's discretion.
	①	Data satisfies the objectives of Control Category 1 (CC1).
	②	Data satisfies the objectives of Control Category 2 (CC2).

Figura 4.25: Annesso A DO-178B applicabilità obiettivi in riferimento al software level [9]

Il mancato soddisfacimento dei tre obiettivi richiesti per l'ottenimento della SQA fa si che questa non possa essere ottenuta.

I riferimenti alla Control Category saranno trattati nel seguito.

Se il tool fosse stato progettato come un verification tool, si sarebbe dovuto fare subito riferimento a queste linee guida, si sarebbe dovuto implementare un ciclo di vita del prodotto software e generare una conformity review al termine.

Accertato che la SQA non può essere ottenuta, saranno comunque esaminate le attività da portare a termine per ottenere il soddisfacimento degli obiettivi, con applicazione al tool per evidenziare la presenza di ulteriori deficienze. Tali attività sono funzionali per il raggiungimento degli obiettivi, e la loro messa in conto durante il processo di qualificazione comporta una maggiore possibilità di portare a termine con successo il processo. Ovviamente, data la tipologia di software oggetto delle analisi (software embedded e non verification tool), risulta necessario estrapolare i concetti ed adattarli allo scopo.

- La prima attività è quella di tenere in conto il processo per la SQA in tutto il ciclo di vita del software, con un continuo confronto con l'autorità competente. Quando il tool è stato creato, non è stato pensato per essere qualificato sulla base di questa norma, pertanto non solo risulta assente qualsiasi riferimento al ciclo di vita del software, ma non vi è stata nessuna interazione con delle autorità competenti. Pertanto tale attività risulta non eseguita. È ovvio che, se fosse un verification tool, e se in uno sviluppo futuro si volesse certificare realmente questo tool, sarebbe necessario non soltanto implementare tutte le misure previste, ma anche essere a stretto contatto con l'autorità per verificare la correttezza delle azioni eseguite.
- Il processo serve a dimostrare che i progetti e gli standard sono sviluppati e ne è verificata la consistenza con i requisiti. A tal proposito si può nuovamente fare riferimento alla specifica del paragrafo 4.2 e le attività del paragrafo 4.3.2. Una pianificazione del tool è stata necessaria per arrivare alla sua creazione, e anche se non risulta applicato nessuno standard, si è dimostrato che si ha rispondenza con i requisiti, grazie alla validazione effettuata per la normativa automotive. Manca una consistency review che riporti i risultati della validazione. Tale attività, cade in difetto per l'assenza di una specifica reale dei requisiti, ma se viene fatto riferimento alla specifica creata appositamente per i fini del presente elaborato, risulta parzialmente eseguita. Si raccomanda, per una completa rispondenza, il confronto ed eventualmente una riscrittura del codice in accordo con uno standard.
- Si chiede la rispondenza tra i progetti del tool e il ciclo di vita del software. Ovviamente tale attività non risulta performata, di fatti il tool cade in difetto sul secondo obiettivo.
- La SQA dovrebbe includere audit dello sviluppo del software e dei processi integrali per assicurare i seguenti aspetti
    - Progetti disponibili. Questi sono effettivamente tali, in quanto sono disponibili le funzioni necessarie al funzionamento del tool. Manca tuttavia una guida a tale materiale;
    - Le deviazioni del progetto dallo standard devono essere rilevate, registrate, valutate, tracciate e risolte. Si è già parlato dell'assenza di uno standard

specifico, a meno di riferirsi allo SPICE. Nel paragrafo 4.3.1 sono state rilevate tutte le deviazioni da tale standard che non portano ad una perfetta applicazione delle basic practices, con conseguente indicazione di tutti gli aspetti da migliorare per renderlo rispondente;
    - L'ambiente di sviluppo deve rispondere a quanto riportato nei progetti, e questo è vero, poiché il tool è stato progettato per operare in Matlab-Simulink, di conseguenza ha comportato una scrittura dello stesso in questo ambiente. Questo ambiente tuttavia non è ben visto dal punto di vista della DO-178B (bisogna infatti arrivare alla 178C affinchè venga accettato come un ambiente di sviluppo approvato, date le sue caratteristiche di generatore di codice), quindi seppur si abbia una rispondenza in tal senso a quanto richiesto dall'attività, è la normativa stessa che ne blocca l'applicazione.
- Il life cycle non trova applicazione;
- Bisogna assicurarsi che il ciclo di vita sia controllato in accordo con la control category. Tale attività è fondamentale per la SQA, pertanto nel seguito si verificherà, comunque, la rispondenza a quanto richiesto dalla control category;
- Serve la conformity review;
- Il processo della SQA si conclude con un record dell'esecuzione delle attività riportate fino a questo punto. Vanno inseriti anche gli audit della conformity review.

L'obiettivo della conformity review del software è ottenere garanzie, per un prodotto software presentato come parte di un'applicazione di certificazione, sul fatto che i processi del ciclo di vita del software siano completi, i dati del ciclo di vita del software siano completi e il codice dell'oggetto eseguibile sia controllato e possa essere rigenerato. Questi sono gli aspetti della conformity review attesa al termine del processo della SQA. È necessario tenere presenti questi aspetti laddove si voglia procedere con una reale certificazione di un vero verification tool, e produrre la documentazione adeguata.

Finora è stato esaminato quanto richiesto per ottenere il conseguimento degli obiettivi richiesti dalla Software Quality Assurance. Ma la SQA deve comunque seguire un metodo per poter essere applicata e raggiungere gli obiettivi richiesti. Questi sono descritti nel Software Quality Assurance Plan (SQAP). Deve contenere la descrizione del processo, la metrica adottata e i metodi di gestione. Oltretutto dovrebbero essere inclusi:

- Ambiente: descrizione del contesto nel quale vengono eseguite le attività della SQA;
- Autorità: chi ne ha competenza e chi deve approvare;
- Un resoconto delle attività effettuate;
- I criteri di transizione: cioè quei criteri che confermano il passaggio da uno stadio all'altro del ciclo di vita del software;
- Il timing delle attività;
- I record delle attività seguite durante il processo della SQA;

Tutti questi elementi formano il piano della SQA, pertanto prima ancora di eseguire le attività legate all'ottenimento degli obiettivi, è necessario dotarsi di tale piano, con all'interno riportate le voci scritte sopra, ed assicurarsi che tutte le attività siano riportate in accordo con tale piano.

Con il SQAP terminano le indicazioni della normativa circa la dimostrazione della qualità del software.

La SQA richiede il soddisfacimento di tre obiettivi. Nessuno di questi tre è stato soddisfatto, poiché il tool già negli scopi non può essere considerato un verification tool, pertanto il fallimento nell'ottenimento della SQA era atteso, ed è stato comunque dimostrato dall'immaturità del tool stesso. Il lavoro precedentemente eseguito sullo standard SPICE fa si che, seppur in maniera parziale, il primo obiettivo possa pensarsi in parte soddisfatto (in realtà, non lo è: la normativa non lavora sulle mezze misure, pertanto associare lo SPICE non è comunque una pratica corretta). Nonostante non sia stato tenuto in conto nessuno standard (con particolare riferimento alla ISO 12207 o ISO 9126), il fatto che questo procedimento segua l'applicazione della ISO 26262 ci aiuta ad identificare uno standard effettivamente applicato, che ha permesso di qualificare il processo che ha portato alla produzione del tool. Ovviamente è una considerazione valida all'interno del presente elaborato poiché esistono degli standard che permettono di qualificare il processo più vicini al campo aerospaziale (MIL o ECSS). Il secondo obiettivo è stato visto non raggiunto, mancando del tutto i riferimenti al ciclo di vita del software. Il terzo obiettivo è anch'esso non raggiunto poiché, in assenza dei documenti relativi agli obiettivi sopra, è impossibile stendere una conformity review.

In ogni caso dovrebbero essere eseguite le attività che portino all'ottenimento della SQA, secondo anche un prestabilito SQA Plan. Si è ragionato per assurdo sul fatto che fosse un verification tool, e che quindi il tool FTA fosse stato creato per essere tale. Comunque le attività riguardano più la fase progettuale del tool, che il fatto che il tool sia esso o meno un verification tool. Infatti tutte le considerazioni scritte in precedenza derivano da una valutazione sul livello di maturità e completezza del tool, rispetto alla sua reale natura.

Sulla base delle conclusioni raggiunte, **è possibile stabilire il non raggiungimento della SQA**, cadendo in difetto sia sugli obiettivi, che sulle attività che portano all'ottenimento di tali obiettivi. Tutto questo è però indipendente dall'avere a che fare con un vero verification tool.

4.4.2 QUALIFICA DEL TOOL:

Nei paragrafi precedenti è stato dimostrato che non si tratta di un development tool, e che non può nemmeno essere considerato un verification tool. Procedendo per assurdo si è assunto che fosse in realtà un verification tool ed è stato applicato il processo della SQA. Si è dimostrato come, indipendentemente dal fatto che si stesse ragionando per assurdo, il tool è troppo immaturo per ottenere rispondenza a tale processo. Secondo normativa ISO 26262 è stato dimostrato che il processo è comunque implementato, seppur con molti limiti, secondo la normativa aeronautica invece non vi è rispondenza a nessuno degli obiettivi richiesti. Bisogna comunque tenere in conto come si tratti sempre di un esempio pratico di applicazione di una norma, e di come comunque si vuole dimostrare, indipendentemente dall'esito, l'applicazione della stessa. Le due bocciature ricevute già sarebbero sufficienti per arrestare il processo, ma per avere un quadro completo delle richieste della normativa, e dell'applicazione della stessa, si procede comunque con la seconda del processo di qualifica:

- Determinazione dei criteri di qualifica;
- Applicazione della Control Category nell'ambito del processo di gestione della configurazione del software.

Sulla determinazione dei criteri di qualifica, il fatto che venga trattato come un verification tool porta ad una procedura più rilassata di quanto non si avrebbe con un development tool. L'unica richiesta è infatti il soddisfacimento dei requisiti operazionali nelle condizioni normali di funzionamento.

La richiesta è molto blanda, e semplice da rispettare in riferimento alla specifica dei requisiti trattata nel paragrafo 4.2. Per semplicità si riporta la tabella relativa ai soli requisiti operazionali.

REQUISITI OPERATIVI			
ID	**Requisito**	**Metodo di verifica**	**Note**
OP01	Il tool deve poter gestire fino a 20 basic events	Analisi	/
OP02	Il tool deve poter calcolare i parametri necessari all'analisi autonomamente	Analisi	Si intende il corretto funzionamento dei command button inseriti nella GUI
OP03	Il tool deve propagare l'analisi in tempi piccoli	Analisi	Per tempi piccoli si intendono tempi inferiori al minuto.
OP04	Il tool deve poter mostrare i valori ai livelli intermedi, senza errori	Analisi	/
OP05	Il tool deve poter mostrare messaggi di errore nel caso di uso improprio	Analisi	/
OP06	Il tool deve creare, quando richiesto, fogli di calcolo pre-formattati	Ispezione	/
OP07	I risultati non devono differire fino alla quarta cifra significativa da quelli ottenuti con un tool commerciale	Analisi	/
OP08	Le porte logiche devono poter ricevere almeno 5 input	Analisi	/

A differenza della ISO, non è specificato nulla circa la relazione tra autore e validatore/verificatore. Questo rende la specifica scritta sopra più adatta rispetto a quanto fatto con la ISO. Tuttavia, è questa volta la stessa DO-178B a dettare delle linee guida per la specifica dei requisiti operativi (o operazionali).

La normativa dice che tale specifica dovrebbe includere dei requisiti che:

- Riguardino le funzionalità e le features tecniche;
- Le informazioni per l'utente come una guida all'installazione ed un manuale utente;
- La descrizione dell'ambiente operativo del tool.

Pertanto potrebbe essere una buona idea ampliare tale specifica con dei nuovi requisiti che contengano all'interno le informazioni consigliate dalla norma. Per quanto riguarda le features tecniche, tutti i requisiti presenti nella specifica vi fanno riferimento, per le altre due voci invece possono essere aggiunti:

ID	Requisito	Metodo di verifica	Note
OP09	Il tool deve possedere una guida tecnica all'installazione	Ispezione	/
OP10	Il tool deve possedere un manuale utente	Ispezione	/
OP11	Il tool deve poter operare su versioni di Matlab successive alla 2017b	Analisi	È richiesta una funzione di salvataggio in retrocompatibilità per i nuovi alberi

Questi nuovi requisiti sono facilmente dimostrabili per ispezione, in quanto la presenza di una guida tecnica e di un manuale utente sono disponibili e direttamente visualizzate dall'utente. Per il tool in esame, come installazione si fa fede al manuale di Matlab, facilmente recuperabile in rete sul sito della Mathworks, mentre come manuale utente può benissimo essere fatto riferimento al punto [1] della bibliografia. Per l'OP11 è necessaria l'esecuzione di un test.

Come effettuato per la precedente dimostrazione dei requisiti, anche per la verifica di OP11 è necessario avere a disposizione un test plan. Il metodo di verifica scelto è volutamente analisi, in quanto è necessario dimostrare che il programma possa salvare in modalità per retrocompatibilità gli alberi, in accordo con le note, e che questi possano essere aperti in versioni precedenti. Una delle pecche di Simulink, è il fatto che non possiede, di per sé retrocompatibilità. Ogni anno vengono rilasciate due versioni: "a" nel primo semestre e "b" nel secondo. Tra le due versioni è possibile creare con la "a" un albero che sarà letto dalla "b", ma non vale il viceversa. Aggiornare la licenza del programma è costosa, pertanto eventuali clienti potrebbero preferire una versione più vecchia, con l'handicap di non poter aprire modelli creati con la versione più recente. Da qui discende la necessità di poter salvare in modalità di retrocompatibilità.

La figura seguente mostra il Test Plan per OP11, usando lo stesso formato dei precedenti.

COMPATIBILITY TEST

ID: TEST04

RESPONSABILE:
Validatore/Verificatore del programma;
ASSUNZIONI:
Tool completamente disponibile. Si presenta come una libreria Simulink;
TEST:
Il test mira a verificare la presenza di una modalità di salvataggio in retrocompatibilità dei modelli creati in versioni successive di Simulink. In particolare si dimostrerà come un modello creato con la versione 2018a possa essere aperto con una versione precedente.
COMUNICAZIONE:
Ente certificatore, azienda produttrice del software;
ASPETTI CRITICI:
Salvataggio dell'albero in retrocompatibilità.
SEGNALAMENTO DIFETTI:
----------;
AMBIENTE:
Matlab-Simulink ver. 2018a, OS Windows 7 Ultimate 64 bit

Figura 4.26: Compatibility Test Plan

Il test viene eseguito sull'esempio usato per il validation test del precedente paragrafo (figura 4.12). Si va a salvare il tool in modalità per retrocompatibilità per la versione 2017b. È possibile verificare questo requisito dal percorso File/Export Model to…/Previous version. L'esito del test è positivo, in quanto Simulink permette questa funzione. Ovviamente può essere fastidioso e fonte di errori e ritardi il fatto che tale funzione non si abbia in automatico. Tuttavia, il requisito viene ugualmente soddisfatto.

Nella figura, seguente viene mostrata l'ultima fase dell'operazione di salvataggio in modalità di retrocompatibilità. Si può notare come sia possibile salvare fino alla versione 2011a. Inoltre Simulink permette il salvataggio sia come modello, con intestazione .mdl o come simulazione con intestazione .slx.

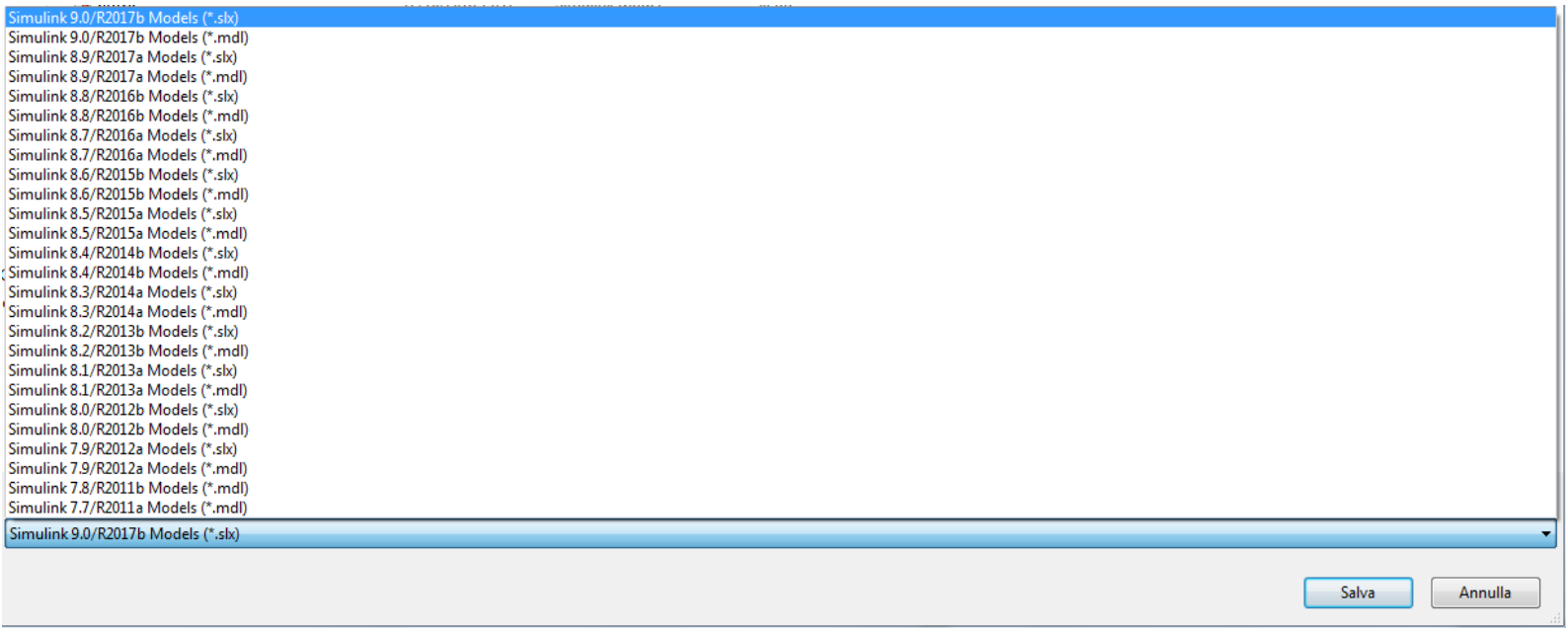

Figura 4.27: Finestra di salvataggio in modalità per retrocompatibilità

I restanti requisiti operazionali sono stati dimostrati nel paragrafo 4.3.2, e se ne è dimostrato il rispetto.

Pertanto si può ritenere che la richiesta fatta dalla normativa, circa i verification tools, sia stata rispettata.

ID	DESCRIZIONE	TEST	ESITO	NOTE
OP01	Il tool deve poter gestire fino a 20 basic events	TEST02	Passato	Limite massimo accettato dal tool
OP02	Il tool deve poter calcolare i parametri necessari all'analisi autonomamente	TEST01	Passato	Si intende il corretto funzionamento dei command button
OP03	Il tool deve poter propagare l'analisi in tempi piccoli	TEST02	Passato	/
OP04	Il tool deve mostrare i valori ai livelli intermedi senza errori	TEST01	Passato	/
OP05	Il tool deve mostrare messaggi di errore nel caso di uso improprio	TEST02	Passato	/
OP06	Il tool deve creare, quando richiesto, fogli di calcolo pre-formattati	TEST03	Passato	/
OP07	I risultati non devono differire fino alla quarta cifra significativa da quelli ottenuti con un tool commerciale	TEST01	Passato	/
OP08	Le porte logiche devono poter ricevere almeno 5 input	TEST02	Passato	/
OP09	Il tool deve possedere una guida tecnica all'installazione	/	Passato	Riferimento a quella fornita da Mathworks

OP10	Il tool deve possedere un manuale utente	/	Passato	Riferimento ad [1]
OP11	Il tool deve poter operare su versioni di Matlab successive alla 2017b	TEST04	Passato	/

È comunque necessario ripetere che tale specifica è stata scritta dall'autore del presente elaborato, in quanto non è stata fornita assieme al tool. La validità della conclusione appena raggiunta è limitata all'esempio pratico di questo elaborato ed alle assunzioni fatte all'inizio del paragrafo. Una specifica reale è molto più ampia e articolata di quella presentata nel paragrafo 4.2, e le modalità di test eseguite non sono quelle che dovrebbero essere eseguite in realtà (il metodo corretto è per ogni requisito un test).

4.4.3 CONFIGURATION MANAGEMENT:

L'ultimo aspetto che la normativa richiede di considerare è il processo di configuration management, presente al capitolo 7 della normativa.

Il Software Configuration Management process (SCM) include le attività di identificazione della configurazione, controllo delle modifiche, baselines per la creazione e l'archiviazione del prodotto software, inclusi i relativi dati del ciclo di vita del software.

Questo processo è richiesto in riferimento all'assegnazione di una Control Category al tool, la quale dipende dalla tipologia di tool che si sta analizzando.

Non trattandosi di un verification tool, anche questa sezione non sarebbe applicabile, ma avendo assunto tale il tool, la normativa richiede di usare le misure contenute nella tabella riportata in figura 3.9, in riferimento alla **Control Category 2 (CC2)**. La control category stabilisce delle procedure e dei requisiti che devono essere rispettati per ottenere la conformità della configurazione del tool. L'applicazione della CC2, in particolare, risulta essere più snella rispetto alla CC1, prevista invece per i development tools.

Nel seguito sono elencate le richieste della control category e si verificherà se il tool in oggetto risponde alla richieste ed, in caso contrario, cosa è necessario implementare affinché tali richieste siano soddisfatte.

- **Identificazione della configurazione**. Il riferimento è al paragrafo 7.2.1 della DO-178B. L'obiettivo dell'attività di identificazione della configurazione è di etichettare in modo univoco ciascun elemento di configurazione in modo che venga stabilita una base per il controllo e riferimento per gli elementi di configurazione. La linea guida richiede:
    - Che la configurazione del tool sia stabilita per il ciclo di vita del software. Ovviamente, mancando i riferimenti al ciclo di vita, questo punto non è rispettato;
    - L'identificazione della configurazione dovrebbe essere stabilità per ogni elemento della configurazione, per le componenti di controllo di tali elementi, e per tutti gli elementi che compromettono il prodotto software. Gli elementi di configurazione sono tutti gli oggetti su cui opera un tool di configurazione, ossia un programma che non appartiene a nessuna delle due classi di tool, ma che aiuta a verificare la bontà del processo. Alla famiglia degli elementi di

configurazione si associano codici sorgente, eseguibili, procedure di test, documenti, ecc. Possiamo pensare che le funzioni e la GUI associata ad ogni blocchetto siano esse gli elementi di configurazione. Un configuration tool dovrebbe avere nel suo database le informazioni che mostrano l'evoluzione di tali funzioni. Tool del genere sono CSV, SVN, Git, ecc, ma non Matlab. Essendo queste funzioni legate ai blocchi, elementi che possono compromettere il funzionamento del software, per bachi al loro interno, allo stato attuale appaiono come "candidati" per essere considerati tali. L'identificazione dovrebbe essere fatta con un configuration tool, anche se tale elemento, per i fini dell'elaborato non è un problema in quanto non viene presentato nulla all'ente certificatore e si può pensare di aver effettivamente usato un tool di configurazione che non dà evidenze poi nel progetto. Però dovrebbero essere trattati tutti questi aspetti identificandoli e comunicandoli nella maniera adeguata.

  - L'identificazione degli elementi di configurazione dovrebbe essere effettuata prima dell'implementazione di qualsiasi cambiamento (correzioni delle funzioni in questo caso) e il tutto reso tracciabile;
  - Un elemento della configurazione dovrebbe essere identificato prima che questo intervenga in un altro ciclo di vita, referenziato da altri cicli di vita oppure usato per la creazione o caricamento di altro software. Il fatto che l'elemento venga identificato è necessario specie se si prevede un'applicazione attiva del tool su un progetto reale, per il quale esiste un ciclo di vita separato.
  - Se il prodotto non può essere identificato fisicamente, allora è necessario che vi siano dei rimandi alla configurazione nel codice eseguibile. Questo punto tuttavia non riguarda il tool in oggetto in quanto fa riferimento ad altre applicazioni, quelle relative al software detto field-loadable, come ad esempio il software di missione di un velivolo (il Tornado setta il computer per la missione tramite schede o memorie esterne contenenti il software necessario).

Questo punto presenta notevoli difficoltà nella sua applicazione, poiché, come più volte evidenziato, mancano i riferimenti al ciclo di vita del software, e non è stata prodotta documentazione che permetta di identificare la configurazione del tool e dei suoi componenti, se non una descrizione dello stesso, dei suoi aspetti funzionali, ed il riferimento al workspace di Matlab come tool di configurazione. Sarebbe necessario allora interessarsi alla creazione di tale documentazione, nel rispetto delle linee guida della presente norma e di quella riguardante il ciclo di vita del software.

- **È richiesta la tracciabilità del software**. Il riferimento è al paragrafo 7.2.2 punti f e g della DO-178B.
  - Un elemento di configurazione dovrebbe essere rintracciabile all'elemento di configurazione dal quale era derivato. Di questo se ne occupa sempre il configuration tool. Questo è necessario dal momento in cui viene richiesta la certificazione per attività o dati relativi al ciclo di vita del software associati all'elemento precedente. La tracciabilità degli elementi è necessaria in ogni caso, tuttavia questo punto non è necessariamente applicabile poiché si presta più all'applicazione sui prodotti del tool piuttosto che sul tool stesso. Oltretutto, se si assume che è stato usato un configuration tool (caso non realistico), si può confermare l'esecuzione dell'attività.

  - Un elemento di base o di configurazione dovrebbe essere rintracciabile o sull'output che identifica o su il processo al quale è associato. Il fatto che l'output del tool sia basato su una combinazione dei blocchetti, che abbiamo già detto essere gli elementi dello stesso, comporta che comunque vi sia un certo grado di tracciabilità tra questi e il risultato dell'analisi eseguita. È infatti possibile identificare eventuali errori o malfunzionamenti verificando l'intero blocco. Questo aspetto è il punto di forza del tool: seppure non si abbia documentazione legata alla tracciabilità degli elementi, tuttavia gli output sono facilmente legabili a questi per la struttura stessa del tool, che permette così di identificare i responsabili di eventuali output errati.
- **Change Control: integrità ed identificazione**. Il riferimento è al paragrafo 7.2.4 punti a e b della DO-178B. L'obiettivo dell'attività di change control è provvedere a registrare, valutare, risolvere ed approvare cambiamenti durante il ciclo di vita del software.
  - I cambiamenti dovrebbero preservare l'integrità degli elementi della configurazione e le loro baselines, provvedendo alla protezione contro cambiamenti non autorizzati. Questo aspetto è effettivamente implementato sul tool, in quanto è programmato per evitare che gli elementi base della libreria acquisiscano gli input dell'utente nell'albero nel quale sono utilizzati Questo aspetto è positivo, poiché evita all'utente di dover svuotare di volta in volta i vari blocchi, evitando anche possibili errori sistematici nell'albero. Tuttavia tale aspetto non esiste per le funzioni alla base del tool, che possono essere modificate liberamente da qualsiasi utente. È pertanto consigliato implementare questo aspetto.
  - Il controllo sui cambiamenti dovrebbe assicurare che qualsiasi variazione ad un elemento della configurazione richieda un cambiamento all'identificazione della configurazione. Cioè si richiede che, in caso si voglia modificare la configurazione di un blocchetto, sia necessario modificare le voci che identificano la configurazione di questo blocchetto.
- **Recupero**. Il riferimento è al paragrafo 7.2.7 punto a della DO-178B. L'obiettivo è quello di assicurare che tutti gli elementi del ciclo di vita del software associati al prodotto, possano essere recuperati nel momento in cui questo volesse essere duplicato. Di fatti l'unico punto a cui fa riferimento la CC2 dice che i dati del ciclo di vita del software associati al prodotto dovrebbero essere recuperati da una fonte approvata, come ad esempio la stessa azienda che lo ha prodotto. Nel caso del tool in esame, seppure derivante da un progetto di tesi, la proprietà del tool è dell'azienda all'interno della quale è stato creato, e questa, in possesso di certificati di qualità approvati, diventa essa stessa una fonte approvata.
- **Protezione contro cambiamenti non autorizzati**. Il riferimento è al paragrafo 7.2.5 al punto b(1) della DO-178B. Chiede che l'attività di review sui cambiamenti includa una verifica di impatto sui requisiti di tipo safety-related, con feedback dal processo di verifica della sicurezza del sistema. Questi requisiti non sono stati considerati al momento della creazione del tool, né sono stati tenuti in conto nella stesura della specifica del paragrafo 4.2. Pertanto si richiede di includere tale tipologia di requisiti, ed effettuare una verifica di impatto dei cambiamenti apportati al tool su tale specifica. Questa attività allora non soltanto comporta la necessità di estendere la specifica dei requisiti, ma anche di modificare opportunamente alcune funzioni del tool per valutare

quanto tali modifiche incidano sulla sicurezza dello stesso. Sono tutte attività che rientro nel planning process del tool.

- **Conservazione dei dati**. Il riferimento è al paragrafo 7.2.7 al punto "e" della DO-178B. Dice che la procedura per la conservazione dei dati dovrebbe essere stabilità per soddisfare le direttive di aeronavigabilità ed abilitare modifiche del software. Per quanto concerne il tool in oggetto, questo non deve necessariamente rispondere ad alcune direttiva di aeronavigabilità, non essendo un software che vola fisicamente assieme al velivolo (e non è nemmeno un tool al quale applicare questa normativa). È comunque bene che i dati vengano salvati ed immagazzinati nel computer sui quali si installa tale software in maniera chiara e corretta, e che gli output del tool, gli alberi per la precisione, siano sempre disponibili per modifiche qualora queste vengano richieste. Poi si possono anche aggiungere considerazioni di tipo economiche sull'uso del tool oppure che anticipino future review dagli enti certificatori. Questo è bene se il software voglia essere considerato commerciale, ma come è facile evidenziare, a differenza della ISO, questa non è una normativa che ha come fine la commercializzazione di un tool, ma solo la sua qualifica per l'inclusione all'interno di un progetto, e pertanto questi aspetti, seppur interessanti per il tool, vanno oltre i suoi scopi.

La CC2 trova anche dei rimandi alla SQA, ed in particolare alla tabella in figura 4.25. Quella tabella, oltre a presentare gli obiettivi necessari per la SQA in base al software level obiettivo, chiede che i dati siano in accordo con quanto richiesto dalla control category. Pertanto l'ottenimento della SQA non è solo legata al soddisfacimento degli obiettivi, ma anche al rispetto della CC di riferimento.

Il tool, sulla CC, si mostra poco strutturato e con evidenti mancanze di tipo documentale piuttosto che strutturali o progettuali. Una eventuale qualifica del tool è comunque esclusa a priori, ma se fosse possibile, verrebbe chiesto di portare a termine con successo le attività richieste dalla control category di riferimento che, in ogni caso, fanno sempre riferimento ad un ciclo di vita che non è stato implementato.

Con l'analisi delle attività della control category, termina il processo di qualificazione del tool secondo la normativa DO-178B.

I risultati ottenuti vengono poi presentati all'autorità di certificazione per il prodotto aeronautico (EASA per Europa o FAA per Stati Uniti), all'interno della documentazione che accompagna il software embedded sul quale il tool ha lavorato. L'autorità determina l'accettazione dell'uso del tool in due steps, dipendenti anche dal tipo di tool presentato (development tool o verification tool). Nel caso in esame ci si aspetta:

- Un agreement al piano per gli aspetti di certificazione del software integrato a bordo velivolo;
- Un agreement al sommario di realizzazione del software per il software installato a bordo velivolo.

Questi due agreement sono il risultato della procedura di qualifica del tool, e se ottenuti vuol dire che la procedura è andata a buon fine e che il tool risulta essere qualificato per l'uso in tale contesto.

Gli obiettivi del presente elaborato, anche in questo caso, non mirano ad ottenere nessun risultato da parte di un'autorità legislativa, mancando in primis l'applicabilità della norma, e per via dell'immaturità riscontrata in molti aspetti dal tool stesso.

Di conseguenza si determina che **il tool non può essere qualificato** secondo DO-178B. Tuttavia rimangono una serie di aspetti, e delle linee guida che potrebbero essere richiamate qualora si decidesse di creare un vero verification tool.

CAPITOLO 5. CONCLUSIONI

Nell'introduzione di questo elaborato si è parlato degli obiettivi che ci si è posti circa il tool e lo scopo stesso dell'elaborato: non è possibile seguire appieno il processo di certificazione poiché ciò richiederebbe un pesante investimento in termini di risorse umane ed economiche da parte dell'azienda. Sfruttando il tool FTA è stato possibile delineare delle linee guida che possono, se mai si decidesse di intraprendere tale strada, portare alla certificazione del tool o di qualsiasi altro tool.

5.1 VALUTAZIONE SULLA CERTIFICABILITÀ DEL TOOL:

Nel capitolo 4 è stato affrontato il processo di certificazione del tool usando le due norme prese a riferimento. Si è osservato come il tool si adattasse, in maniera più o meno rispondente alle richieste della normativa, mettendo in evidenza i punti nei quali il suo livello di maturità garantisce, in parte, il successo dell'applicazione, oppure dove tale livello crea delle discrepanze dalle richieste della normativa. Laddove possibile sono state evidenziate le correzioni da apportare, per aumentare la maturità del tool e per irrobustirlo, con l'obiettivo di renderlo certificabile un domani.

Adesso si presenteranno i risultati ottenuti in maniera critica, sia in riferimento alla ISO 26262, sia alla DO-178B.

Il tool, come era anche lecito pensare, gode di un livello di maturità molto basso, poiché tante delle misure che entrambe le due norme richiedono, non sono state applicate in fase di programmazione, dove ci si è concentrati più sull'aspetto funzionale. Tali problemi nascono non tanto da un mancato funzionamento dello stesso, aspetto che è stato dimostrando effettuando dei test comparativi con un prodotto commerciale oggi molto usato sul mercato per analisi di tipo Fault Tree, ma da una mancanza di documentazione ed attività precedenti alla fase di produzione del tool.

Si chiede, nella DO-178B, un rimando al ciclo di vita del software, presente in standard come ISO 12207, il quale non è stato tenuto in conto durante la fase progettuale. Seguire il ciclo di vita, e produrre la documentazione necessaria, è importante ai fini della tracciabilità del lavoro e per monitorare le caratteristiche del tool, sia durante la fase progettuale/produttiva dello stesso, sia nel tempo, permettendo così di inserire modifiche che ne permettono la sua evoluzione. Questa è una mancanza grande che, seppur maggiormente richiesta nella normativa aeronautica, è comunque citata nella norma automotive.

In riguardo alla DO-178B, si è arrivati a dimostrare che al tool in esame non può neanche essere applicata la normativa, poiché non risponde a nessuna delle due categorie prese in considerazione (development tools o verification tools). Si è dovuta fare un'assunzione che considerasse, per assurdo, il tool come un verification tool, e che permettesse di applicare lo standard mostrando comunque tutte le sue debolezze nei confronti di una norma abbastanza rigida.

Un'altra grande mancanza evidenziata durante la fase di valutazione, è la mancanza di una specifica dei requisiti. Tale specifica è necessaria ai fini di qualsiasi attività di certificazione. I requisiti, ed una loro allocazione sui livelli più bassi dell'architettura, consentono di definire gli obiettivi e le funzionalità del tool stesso, e verificare che le attività progettuali e produttive abbiano prodotto un software rispondente alle aspettative. La validazione dei requisiti serve

proprio a verificare questo aspetto, permettendo di evidenziare le criticità presenti nel tool da una parte, e fornire indicazioni sulle correzioni da apportare dall'altra. Una specifica di tal genere è completamente assente, e già sarebbe bastato questo per bloccare l'intero processo di certificazione. Tuttavia, volendo proseguire nello studio, ed in accordo con gli obiettivi prefissati, è stata creata dall'autore una specifica di alto livello, inserendo quei requisiti che più permettono di mettere in evidenza alcune delle macro-caratteristiche del tool. Questo approccio è ovviamente erroneo, anche in riferimento alle richieste della ISO 26262, che impone la massima distanza tra chi stende la specifica/progetta il tool e chi invece deve occuparsi della sua valutazione (ASIL C/D). Nel caso in esame, invece, le due persone coincidono, andando contro le richieste della normativa, generando una non conformità. Seguire questa linea, inoltre, porta comunque ad una incongruenza: la specifica non è indipendente dai risultati attesi. Anche se in maniera non del tutto voluta, si è comunque arrivati a dei requisiti che mettono esattamente in luce cosa già si conosce o aspetta dal tool. Per questa ragione il tool, preso al di fuori del presente elaborato, non è certificabile/qualificabile in partenza.

Anche i metodi usati per la validazione dello stesso non sono esattamente conformi alle regole che in genere si adottano per questo tipo di compito. Normalmente si dovrebbe prevedere un test per ogni requisito. Tale test dovrebbe essere documentato in maniera apposita, sia nella fase di progettazione, sia nella fase di post-processing dei risultati. Sulla fase di progettazione si è voluto realizzare dei test plan che potessero essere più o meno fedeli a quanto richiesto dagli standard relativi a tale parte. Il post-processing, invece, è stato lasciato sottoforma di commento generale al di sotto del test plan. I test effettuati sono stati scelti per essere rappresentativi delle caratteristiche reali del tool, e per mettere, in alcuni casi, in luce anche eventuali debolezze. Pochi test per dimostrare tutta la specifica. Nonostante ciò alcuni requisiti sono stati soddisfatti solo parzialmente, e questo rappresenta un'altra fonte di insuccesso se la specifica fosse mantenuta tale e non ulteriormente rilassata. Il fatto poi che l'insuccesso abbia riguardato i requisiti operazionali, fa sì che anche la DO-178B veda una non rispondenza e porti ad una bocciatura dello stesso.

Il processo di validazione ha mostrato anche un aspetto funzionale del tool che non era stato osservato nel testo [1] della bibliografia. I test effettuati in quell'ambito miravano a dimostrare che la matematica dietro la creazione dell'albero e la propagazione dei dati numerici fosse corretta. Infatti, seppur questo tipo di validazione ha avuto esito positivo, e i risultati sono perfettamente in accordo con il tool commerciale, vi è un problema legato al numero di basic events inseribili. Più che un problema del tool, è un problema dell'ambiente di sviluppo e calcolo usato dal tool. Per un qualche motivo Matlab, superati i 20 basic events, ragiona in maniera diversa da quanto atteso. Il baco riguarda una qualche funzione della libreria relativa al calcolo simbolico. Il programma dovrebbe solo limitarsi a spostare un'informazione di tipo stringa da un contenitore ad un altro, e questo succede fino a 20 BE. Superato tale numero, viene modificata la stringa, rendendola non leggibile alle altre funzioni, e bloccando la propagazione. Il motivo non è stato ancora scoperto.

Il baco non lavora soltanto in relazione al numero di BE usati, ma anche sulla maggiore strutturazione degli alberi. Anche in tal caso ha mostrato lo stesso problema, pertanto rimane oscuro anche il perché di quest'altro comportamento.

Così come ancora non è stato scoperto il motivo per cui, se il modello non è precedentemente salvato con il nome assegnato all'albero, questo non è riconosciuto da Simulink.

Questi tre problemi riscontrati, non sono un problema dal punto di vista finale dell'utente, poiché, se venisse limitata l'autonomia dello stesso e rilassata la specifica dei requisiti, non sarebbero nemmeno contemplati. Però dal punto di vista commerciale sarebbe una limitazione molto forte.

Il tool ha comunque avuto una serie di successi, soprattutto nei confronti della norma ISO 26262. Il processo che ha portato alla valutazione della qualità del processo di sviluppo ha dato esito positivo. Per i fini del presente elaborato, le mire erano orientate a dimostrare l'implementazione del processo, e così è avvenuto, anche se sono state individuate diverse mancanze alle quali, sfruttando la poca documentazione disponibile, si è comunque riusciti a sopperire. È pure vero che lo standard usato, l'Automotive SPICE, è più orientato al software di tipo embedded da installare a bordo veicolo, e che gli altri processi considerati non riguardano l'ambito del tool. Solo tre processi sono stati ritenuti applicabili, e su di questi si è concentrata l'analisi e la valutazione.

Sulla base delle osservazioni, degli studi fatti sul tool in oggetto e sui test eseguiti, si può affermare che il tool **non è certificabile/qualificabile**. Questo risultato è valido per il **caso reale**, nel quale si dovrebbe muovere un'azienda che fisicamente vuole intraprendere il processo.

Se si rimane **nell'ambito del presente elaborato** qualcosa cambia. Sulla base della specifica, delle assunzioni fatte, del materiale di supporto fornito, il tool **può essere ritenuto certificabile secondo ISO 26262**.

Sempre nell'ambito del presente elaborato, il tool fallisce nelle pratiche richieste per la normativa aeronautica, sia per l'aspetto funzionale (non è un verification tool), sia per le mancanze riscontrate. Pertanto **non è qualificabile DO-178B**.

L'obiettivo di valutare il tool FTA e i suoi aspetti funzionali e non, applicati a due diverse normative, ha portato comunque a due risultati:

- Seppur si è stabilito che il tool non è certificabile, risultato che appariva piuttosto probabile, si è comunque riusciti a seguire un processo logico, complicato sotto molti aspetti e che, si ribadisce, coinvolge un numero elevato di risorse non soltanto economiche;
- Rimane comunque traccia del processo: della documentazione richiesta, delle debolezze del tool e delle relative modifiche da apportare per soddisfare le attese dell'ente certificatore. Questo aspetto è molto utile per l'azienda, o per chi in futuro, vorrà cimentarsi nell'irrobustimento del tool e nella sua reale certificazione.

L'obiettivo prefissato inizialmente è stato allora pienamente raggiunto, mostrando come un compito di tal genere, seppur agli occhi di molti possa risultare noioso, in realtà riserva molte sorprese ed è fonte di soddisfazione laddove si riesce ad evidenziare un successo. Oltretutto avere comunque una guida di riferimento è un risultato spendibile laddove questa possa servire per eventuali sviluppi o ricerche sul tema trattato.

5.2 SVILUPPI FUTURI:

Alla luce delle attività eseguite, l'elaborato si pone allora come un apripista per future attività di certificazione. Le regole seguite valgono per ogni tipo di tool che si intenda certificare in tale senso, compreso il tool FTA trattato.

Sono state evidenziate parecchie mancanze o debolezze di questo tool, che comunque si dimostra, all'interno dei suoi limiti, funzionante.

In futuro si potrebbe pensare di irrobustirlo per adattarlo alla normativa ISO 26262 che, come abbiamo visto, è l'unica in grado di certificare un software di questo tipo. A parte la più volte citata specifica dei requisiti, bisognerà riempire i buchi sulla documentazione e sui processi inerenti. Bisognerà anche risolvere le problematiche relative al Matlab ed agli errori presentati. Sarebbe anche opportuno provvedere ad un'espansione dello stesso, introducendo nuove funzionalità che lo rendano più appetibile sia commercialmente che internamente all'azienda.

A tal proposito potrebbe essere utile prendere spunto da prodotti già esistenti e certificati come il più volte citato FaultTree+.

Il tool è stato creato nell'ottica di una riduzione delle spese di acquisizione delle licenze di software commerciali. L'idea potrebbe essere quella di limitare l'uso di queste licenze a favore di un prodotto sviluppato in casa, certificato per venire incontro alle necessità dei clienti che richiedono l'esecuzione di analisi FTA. Il tool, come architettura, si presta bene a questo compito, seppur si presenti una nuova limitazione di tipo economico: il ricorso al software Matlab-Simulink come piattaforma sulla quale far girare il tool.

Tornando all'elaborato, sono presenti anche delle linee guide utili a chi vorrà in seguito eseguire la certificazione di processi tramite standard SPICE. Questo standard, seppur nato in ambito automotive, possiede al suo interno tante linee guida per la certificazione di processi di tipo aziendale, organizzativo e progettuale. L'applicazione al processo di creazione del tool ha mostrato appunto come recepire lo standard al livello più basso per dimostrare l'implementazione di un processo.

Sono tanti gli sviluppi a cui si presta il lavoro svolto per il presente elaborato, e pertanto si spera che in futuro possa risultare utile a chi vorrà proseguire sulla strada intrapresa.

Il discorso sulla certificazione, tuttavia, non si esaurisce con la rispondenza ad una o più norme. Esiste un'altra via, più lunga nel tempo, che possa permettere all'azienda di ottenere vantaggio dal tool homemade: la certificazione di tipo **proven-in-use**. Tale certificazione, come dice il nome, si ottiene testando nel tempo un componente o tool, raccogliendo i dati in uscita relativi a tassi di guasto e failure intensity per hardware, oppure monitorando gli output di un software e dimostrando che questi sono sempre in accordo con la realtà che vanno a descrivere. Un metodo di tal tipo richiede molto tempo e numerose prove da effettuarsi nei confronti del tool in oggetto. Bisogna dimostrare che, nelle applicazioni nelle quali viene usato, inizialmente come solo strumento di backup, fornisca i risultati attesi. Nel tempo si può creare un database di alberi e modelli i cui output sono noti, dimostrando che il tool fornisce come risultato gli output attesi. In questa maniera si è sicuri che il tool è funzionante e può essere accettato come elemento di progetto.

La normativa ISO 26262 nella sezione 8 trattata nel presente elaborato, al capitolo 14, riporta le pratiche per l'ottenimento della certificazione prove-in-use. Un dato che salta all'occhio è il numero di ore senza incidenti o, nel caso in esame, output errati, per il quale è necessario testare il tool. La tabella seguente riporta, a seconda del livello di ASIL, il requisito della normativa:

Table 8 — Targets for minimum service period of candidate

ASIL	Minimum service period without observable incident
D	$1{,}2 \times 10^9$ h
C	$1{,}2 \times 10^8$ h
B	$1{,}2 \times 10^8$ h
A	$1{,}2 \times 10^7$ h

Figura 5.1: Ore di test necessarie per ottenere la certificazione proven-in-use secondo ISO 26262 [7]

Per ottenere la certificazione per il massimo livello di ASIL (D), è necessario testare il tool per più di 10^9 ore, corrispondenti a circa 14000 anni. Ciò vuol dire che è necessaria un'attività di test su 14000 macchine per un anno, senza ottenere output errati. Come evidente, anche questa si presenta come una via difficilmente percorribile per un'azienda. Questo è dovuto al carattere molto restrittivo dei nuovi standard.

Un processo simile è quello che riguarda la certificazione dei modelli creati in Simulink per ambienti ad alto contenuto tecnologico e, per tradizione, molto restrittivi in termini di sicurezza come quello aerospaziale. I modelli in principio erano prodotti in Fortran, C, C++, ecc, linguaggi di programmazione ritenuti sicuri per via della complessità dei costrutti alla base dei modelli. Tale complessità, seppure deleteria da un lato, faceva sì che il funzionamento del modello fosse accertato.

Oggi la Mathworks mette a disposizione il **Tool Qualification Kit**. Si tratta di un database contenente una serie di documenti preimpostati da compilare, un parco modelli che funge da base per il test ed un PDF riportante i risultati dell'esecuzione di tali modelli oppure i dettagli delle failure degli stessi. Il TQK funziona nel seguente modo: provo un certo modello, in riferimento al test n° x e mi aspetto un successo con un certo risultato o una failure nel punto y con un certo errore. Il kit esegue, basandosi sul modello che voglio qualificare, tutti i test, creando un nuovo report. Si confronta allora il risultato dei nuovi test con quello dei vecchi modelli ai quali viene associato, e si verifica se i risultati sono coerenti: successo o failure attesa. Questo metodo di qualifica potrebbe essere adottato per il tool in esame: si potrebbero creare 100 casi tipici da testare e confrontare con il documento riportante gli esiti voluti. Tuttavia è presente un problema: questo metodo non fornisce nessuna indicazione sul tool qualification plan secondo DO-178C, alla quale si rivolge il TQK, che allora dovrà essere creato in partenza, rimanendo sempre valide le considerazioni fatte sull'applicabilità della norma. Per il mondo automotive, invece, viene già fornito un certification plan, già approvato dall'ente certificatore TUV. Questa procedura si presenta però costosa e non applicabile a tool in evoluzione, in quanto viene rilasciata la certificazione per quella particolare versione. Successive evoluzioni necessiterebbero di una nuova esecuzione della procedura.

Che sia una certificazione basata su normative applicabili, oppure di tipo proven-in-use, la certificazione di un tool di questo tipo è possibile. È un aspetto che coinvolge numerose risorse, soprattutto economiche, per le aziende che si lanciano in questa attività. Ottenere un prodotto home-made funzionante e di qualità, è un valore aggiunto per le aziende, sia per il risparmio economico che ne consegue, sia per questioni di prestigio e visibilità verso i clienti che si appoggiano alle soluzioni proposte dall'azienda, ed ai risultati delle analisi eseguite.

L’ingegnere, seppur abbia al centro delle proprie competenze l’attività progettuale, questa è solo una parte delle numerose attività richiestegli nell’ambito industriale. Un progetto non è solo calcoli e disegni, è fatto anche di documentazione, che l’ingegnere sarà chiamato a produrre per dimostrare che tutto sia coerente con gli standard accettati e ritenuto sicuro. L’attività documentale, per quanto sia trasversale alle sue competenze, è quella che oggi occupa la maggior parte del tempo impiegato su un progetto. Saper dimostrare la creazione di un prodotto sicuro tramite la produzione di documentazione apposita è un aspetto sempre più richiesto in ambito aziendale per via della burocrazia legata al raggiungimento del massimo grado di affidabilità possibile. In tale ambito si pone anche la creazione del tool FTA esaminato. Seppur si sia dimostrato solo in parte la sua certificabilità, dovuta a diverse mancanze in fase progettuale ed immaturità dello stesso, rimangono comunque aperte le possibilità per un suo sviluppo futuro nel quale la documentazione sarà il primo “componente” sul quale concentrare l’attività.

BIBLIOGRAFIA

[1] Andrea Garino, Sviluppo e testing di tool per analisi di sicurezza /affidabilità (Fault Tree Analysis), Tesi di laurea magistrale.

[2] Paolo Maggiore, Master universitario in Affidabilità, Sicurezza e Manutenzione: Metodi di analisi e gestione.

[3] Comitato elettrotecnico italiano, Norma tecnica CEI EN 50126.

[4] Comitato elettrotecnico italiano, Norma tecnica CEI EN 50128.

[5] ISO 26262-4:2011 (E).

[6] ISO 26262-3:2011 (E).

[7] ISO 26262-8:2011 (E).

[8] VDA QMC, Automotive SPICE, Version 3.1

[9] RTCA, Inc. 1992, RTCA/DO-178B, December 1, 1992

[10] RTCA, Inc. 2011, RTCA DO-178C, December 13, 2011