

CENELEC EN 50716: PRIMO PASSO VERSO NUOVE TECNOLOGIE PER LO SVILUPPO SOFTWARE IN AMBITO FERROVIARIO.

Introduzione

Nel mondo dello sviluppo software per applicazioni ferroviarie, l'adesione agli standard di riferimento è un punto cruciale per garantire sistemi sicuri e affidabili. L'ultima normativa in materia, legata allo sviluppo di software per sistemi elettronici programmabili, è la **CENELEC EN 50716:2023** [1], emessa dal Comitato Europeo di Normazione Elettrotecnica. Tale norma integra, modifica e sostituisce le due normative vigenti (compresi gli amendments) in merito allo sviluppo software in ambito ferroviario, ovvero la **CENELEC EN 50128:2011** (A1/A2:2020) [2] e la **CENELEC EN 50657:2017** (A1:2023) [3]. La nuova norma introduce, altresì, nell'Appendice C, importanti novità sullo sviluppo software, approfondendo tematiche emergenti come l'impiego dell'intelligenza artificiale nel settore ferroviario e lo sviluppo 'Model-Based'.

Le norme EN 50128/50657 rimarranno applicabili fino ad **ottobre 2026**.

Nel febbraio 2024, Il Comitato Elettrotecnico Italiano (CEI) ha pubblicato la CEI EN 50716 [4] senza apportare alcuna modifica/integrazione rispetto alla versione emessa dal Comitato Europeo.

Attualmente, la norma CENELEC EN 50716 non è stata ancora pubblicata nella Gazzetta Ufficiale dell'Unione Europea e, di conseguenza, non è armonizzata per la Direttiva 2016/797/UE relativa all'interoperabilità del sistema ferroviario dell'Unione europea. L'utilizzo di norme non armonizzate deve essere sempre giustificato ai fini della conformità ai requisiti essenziali della Direttiva.

Nei capitoli successivi verranno esaminate in dettaglio tutte le modifiche più rilevanti introdotte dalla EN 50716. Nell'ultimo capitolo, invece, saranno riassunte le modifiche di minore importanza, che riguardano principalmente una riorganizzazione formale della documentazione e delle tecniche/misure richieste, senza comportare cambiamenti sostanziali nello sviluppo del software. L'analisi è stata condotta confrontando la CENELEC EN 50716 con le norme CENELEC EN 50128 e EN 50657, compresi gli amendments.

CENELEC EN 50128 e CENELEC EN 50657: cosa definiscono e perché sono importanti

Prima di analizzare le modifiche introdotte dalla norma CENELEC EN 50716 in merito allo sviluppo software, è utile fare una panoramica sulle normative CENELEC EN 50128 e CENELEC EN 50657, evidenziando i loro ambiti di applicazione e gli aspetti che definiscono.

1. **CENELEC EN 50128:** definisce le procedure e i requisiti tecnici per tutte le fasi del ciclo di vita di sviluppo del software (Fig. 1), dalla progettazione alla manutenzione, destinato ai sistemi di controllo e protezione ferroviaria. La norma si applica al software e all'interazione tra quest'ultimo e il sistema di cui fa parte e si concentra principalmente su tutti i software identificati come "safety critical", per cui possibili guasti potrebbero comprometterne le funzioni di sicurezza. Per ciascuna fase di sviluppo, la norma specifica: l'obiettivo, la documentazione di input e di output, i ruoli, le responsabilità e i requisiti tecnici da seguire, i quali variano in base al livello di integrità della sicurezza (SIL)¹ del software. Il SIL è uno dei concetti chiave della EN 50128 in quanto, più è elevato il suo valore, più pericolose sono le conseguenze di un malfunzionamento del software e più

rigorose devono essere le misure e le tecniche da adottare durante le fasi di sviluppo. Inoltre, per ciascun livello del SIL, vengono fornite indicazioni chiare anche sull'organizzazione e sull'indipendenza dei team coinvolti nel processo di sviluppo. La norma CENELEC EN 50128 non fornisce una guida su come determinare il livello di integrità del software o le funzioni di sicurezza, in quanto tali aspetti sono trattati dalla norma CENELEC EN 50126-1/ CENELEC EN 50126-2:2017 [5], ma identifica le procedure e le attività necessarie per lo sviluppo di un prodotto sicuro, affidabile ed efficiente.

2. **CENELEC EN 50657:** definisce le procedure e i requisiti tecnici per tutte le fasi del ciclo di vita dello sviluppo software (Fig.1), dalla progettazione alla manutenzione, destinato ai sistemi elettronici programmabili a bordo del materiale rotabile. Non rientra nello scopo della norma il software installato a bordo treno che non contribuisce o è separato dalle funzioni operative del materiale rotabile. La CENELEC EN 50657 nasce dalla CENELEC EN 50128 e, come quest'ultima, si applica esclusivamente al software (safety critical e non) e all'interazione tra quest'ultimo e il sistema di cui fa parte. La struttura del documento è identica a quella della norma CENELEC EN 50128, così come la tipologia di informazioni in essa contenute, poiché rappresenta un adattamento della CENELEC EN 50128 per lo sviluppo software destinato alle applicazioni su materiale rotabile.

In entrambe le norme ci sono forti richiami e collegamenti alla norma CENELEC EN 50126-1 e CENELEC EN 50126-2, la quale definisce il processo per l'identificazione delle funzioni di sicurezza da allocare al software: dall'individuazione degli hazards e del livello di rischio, alla definizione dei requisiti di sicurezza. Inoltre, entrambe le normative si applicano sia al software inteso come prodotto generico², sia come applicazione generica³ o specifica⁴, e forniscono procedure e requisiti tecnici per la validazione dei tool di supporto alle attività di sviluppo.

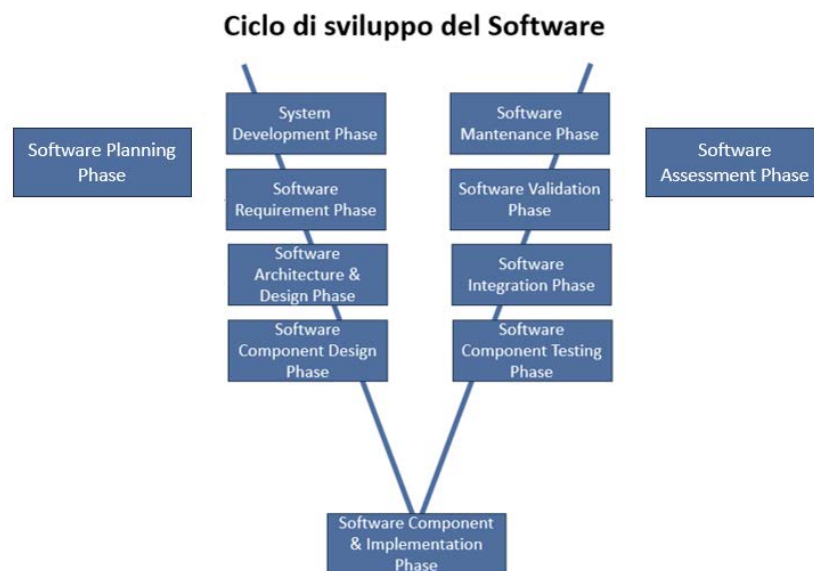


Fig. 1. Fasi del ciclo di vita dello sviluppo software

SIL¹: Safety Integrity Level. Rappresenta il grado di affidabilità che un sistema deve raggiungere per ridurre il rischio di incidenti durante il suo funzionamento. Con il termine Basic Integrity si intende il software o le funzionalità che non hanno nessun impatto sulla sicurezza.

Prodotto Generico²: componente (software e hardware) che offre funzionalità specifiche in contesti operativi e ambientali conformi alle specifiche di riferimento, senza essere specializzato in un contesto particolare.

Applicazione Generica³: prodotto generico adattato funzionalmente ad un contesto operativo e ambientale ben definito. Ad esempio, un interlocking installato in Italia con schemi di principio adattati al contesto italiano.

Applicazione Specifica⁴: applicazione generica configurata per uno specifico impianto tramite i dati applicativi. Ad esempio, un interlocking configurato con i dati di impianto della stazione di Bologna.

CENELEC EN 50716: novità introdotte

La nuova normativa stabilisce i processi e i requisiti tecnici per lo sviluppo software di sistemi programmabili, destinato sia ad applicazioni per il controllo e la protezione ferroviaria che del materiale rotabile. Si applica sia al **software critico** che al **software non critico**, mantenendo la stessa struttura e le medesime tipologie di informazioni degli standard precedenti.

I software precedentemente sviluppati seguendo le normative CENELEC EN 50128 e CENELEC EN 50657 non sono vincolati ai requisiti relativi al software preesistente e sono considerati compliant alla nuova norma. Ciò consente il riutilizzo in nuovi progetti, di software progettati secondo gli standard precedenti, senza la necessità di apportare modifiche.

Relativamente alla conformità dei livelli di integrità del software, viene specificato che per lo sviluppo di funzionalità non legate alla sicurezza (**Basic Integrity**), è necessario seguire un processo che garantisca la qualità del software. Tale processo può essere realizzato applicando i requisiti tecnici previsti dalla norma CENELEC EN 50716, dalla ISO/IEC/IEEE 90003 [7] o da altri standard applicabili. Per i livelli SIL1-SIL4 invece, la norma non impone più la conformità alla EN ISO 9001 [8].

Evoluzione dei ruoli e delle responsabilità nella gestione del software

A seguito di un significativo miglioramento della clausola 5, i requisiti relativi all'organizzazione e ai ruoli nella gestione dello sviluppo software sono diventati più concisi, sintetici e più flessibili. La Figura 2 illustra la nuova struttura organizzativa per i vari livelli del SIL e il Basic Integrity. Di seguito, sono elencate le principali modifiche introdotte dalla nuova norma. Laddove non esplicitamente indicato, le modifiche si applicano a tutti i livelli SIL1-SIL4 e al Basic Integrity.

1. **l'eliminazione del ruolo dell'Integrator:** tutte le responsabilità precedentemente attribuite all'Integrator sono state trasferite ai ruoli dell' *Implementer* e del *Tester*, con un conseguente *aumento delle competenze chiave* richieste per entrambi i ruoli. In particolare, l'*Implementer* deve possedere una solida conoscenza dei metodi di integrazione software, mentre il *Tester* deve avere familiarità con i linguaggi di programmazione, le interfacce software e i sistemi operativi, con una visione più ampia del sistema nel suo complesso. Con l'introduzione della nuova norma, l'*Implementer* deve gestire il processo di integrazione software, mentre il *Tester* deve occuparsi della redazione delle specifiche di test per l'integrazione software (Software Integration Test e Software/Hardware Integration Test);
2. **Indipendenza dei ruoli:** non è più obbligatorio per il Requirement Manager, il Designer e l'*Implementer* rispondere direttamente al Project Manager. La nuova norma sottolinea anche l'indipendenza del ruolo del Validatore, che non deve né supervisionare, né gestire il Requirement Manager, il Designer e l'*Implementer* (e viceversa) e né ricoprire il ruolo di Configuration Manager.
3. **Flessibilità nei cambi di ruolo:** sebbene il Verificatore e il Validatore debbano essere assegnati a livello di progetto e rimanere invariati durante l'intero ciclo di sviluppo, la EN 50716 consente cambi di assegnazione, a condizione che non compromettano le attività e che sia fornita una giustificazione valida;

4. **Possibilità per il Validatore di svolgere le attività del Tester anche per i livelli SIL4 e SIL3:** Il Validatore può svolgere le attività del Tester, purché il lavoro venga successivamente verificato da un'altra persona designata come Validatore;
5. **Possibilità per il Verificatore di svolgere le attività del Tester anche per i livelli SIL1-SIL4:** Il Verificatore può svolgere le attività del Tester, purché il lavoro svolto venga successivamente verificato da un'altra persona incaricata come Verificatore o Validatore. Le norme precedenti prevedevano già questa possibilità ma facevano riferimento solo al Validatore come persona designata alla verifica.
6. **Maggiore flessibilità nelle attività di testing anche per i livelli SIL1-SIL4:** nelle normative precedenti, veniva consentito al Requirement Manager, al Designer e all'Implementer di un determinato componente software, di svolgere attività di test su componenti software differenti. Nella EN 50716, questa possibilità è stata estesa a tutte le attività di test, non solo a livello di componente (come Software Integration Testing, Software/Hardware Integration Testing e Overall Software Testing), a condizione che venga dimostrata **l'indipendenza tra l'implementazione e l'attività di testing**.
7. **Maggiore flessibilità per il ruolo di Assessor:** la nuova norma ha definitivamente stabilito che l'Assessor (figura prevista soltanto per i livelli SIL1-4) deve appartenere a qualsiasi organizzazione di parte interessata, senza necessità dell'approvazione della Safety Authority. L'Assessor deve essere indipendente dal team di progetto e appartenere a un'entità organizzativa distinta da quella dei ruoli coinvolti nel progetto. Inoltre, non è più prevista la comunicazione del piano di assessment alla Safety Authority e agli altri membri del progetto. Questa maggiore flessibilità era già prevista nella norma CENELEC EN 50657, con l'emissione dell'amedment A1:2023. Con la CENELEC EN 50716, si estende anche ai sistemi di controllo e protezione ferroviaria.

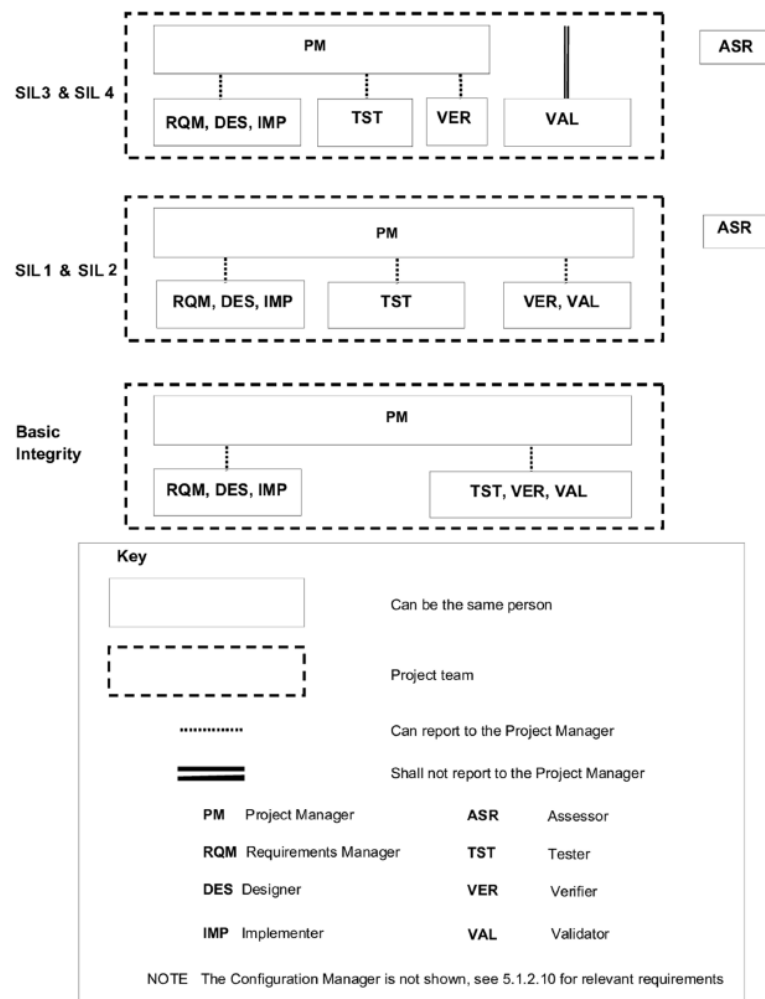


Figura 2: EN 50716: la nuova struttura organizzativa [1].

Nella sottoclausola 5.3.2.13, la norma CENELEC EN 50716 stabilisce che i documenti possono essere creati direttamente da database o strumenti di modellazione. È essenziale, tuttavia, garantire che la tracciabilità tra la versione del documento e quella del database o dello strumento di modellazione venga sempre mantenuta. Questa disposizione era già presente nella CENELEC EN 50657; con la nuova norma, è stata estesa anche allo sviluppo di software destinato ai sistemi di controllo e protezione ferroviaria.

Validazione dei Tool di Supporto

I requisiti tecnici e le procedure per la validazione dei tool di supporto non hanno subito modifiche sostanziali nella nuova norma. L'unica modifica rilevante riguarda la gestione dei tools utilizzati come parte del sistema (cioè che contribuiscono alle sue funzioni), i quali, dopo il deployment nell'ambiente finale di applicazione, devono essere considerati come software e quindi non più soggetti ai requisiti della sottoclausola 6.7, relativa alla validazione dei tools di supporto, ma a quelli delle clausole 4,5,6,7 e 9, relative alla validazione del software.

Coinvolgimento di più organizzazioni e controllo della non interferenza tra componenti

Il coinvolgimento di più organizzazioni nello sviluppo software è un aspetto cruciale, in quanto è fondamentale che i compiti di ciascuna organizzazione siano chiaramente definiti. Inoltre, la documentazione prodotta da ogni organizzazione deve fare riferimento a quella delle altre, al fine di garantire una visione complessiva del progetto. Ogni ente coinvolto è responsabile del rispetto delle normative applicabili e, soprattutto, la complessità del progetto non deve nascondere le responsabilità individuali. In questo contesto, nel *Software Quality Assurance Plan* devono essere descritte tutte le interfacce tra le organizzazioni coinvolte. Questo principio, già introdotto dalla norma CENELEC EN 50657, è ora esteso anche ai sistemi di controllo e protezione ferroviaria.

Un'altra importante novità riguarda il controllo della non interferenza tra componenti con diversi livelli di integrità. Secondo la CENELEC EN 50716, il software responsabile del controllo della segregazione tra i vari componenti deve appartenere al livello di integrità più alto. La norma chiarisce i principali aspetti da considerare per garantire la "non interferenza" tra i componenti, distinguendo tra l'aspetto spaziale e quello temporale:

- 1) spaziale: i dati di sicurezza di un componente non devono essere modificati da altri componenti che operano con livelli di integrità inferiori;
- 2) temporale: un componente non deve interferire con l'esecuzione di un altro componente, sia come tempo di elaborazione del processore e sia nell'utilizzo delle risorse condivise.

Evoluzione del concetto di linguaggio di programmazione: dalla selezione specifica ai requisiti generali

Nella norma CENELEC EN 50716, il concetto di linguaggio di programmazione è stato ampliato e modificato rispetto alle versioni precedenti. In particolare, il focus si è spostato da un elenco di linguaggi di programmazione specifici (come ADA, C, BASIC, ecc.) verso una descrizione dei requisiti del linguaggio, indipendentemente dal linguaggio stesso, riportati nella Tabella A.15 della norma (Figura 3).

In questo contesto, la norma pone maggiore attenzione a requisiti come la programmazione modulare, il supporto per i commenti, la tipizzazione rigorosa e la testabilità. Questo permette una maggiore flessibilità, poiché non è più necessario scegliere solo tra linguaggi specifici, ma si può valutare l'adeguatezza di un linguaggio in base a questi requisiti, qualificando anche linguaggi più recenti come Rust.

Inoltre, la norma estende la sua concezione di linguaggio di programmazione, includendo non solo i linguaggi di programmazione testuali tradizionali, ma anche linguaggi diagrammatici e di modellazione. Viene data priorità a soluzioni che assicurano una maggiore tracciabilità, verifica formale e sicurezza, mentre vengono svantaggiati linguaggi meno verificabili o tracciabili, come quelli basati sulla programmazione orientata agli oggetti. In questo modo, anche linguaggi utilizzati per la creazione di diagrammi o modelli, come quelli per la programmazione visiva o la modellazione dei sistemi, sono ora parte integrante del concetto di "linguaggio di programmazione".

Table A.15 — Suitable Programming Languages

TECHNIQUE/MEASURE	Ref	Basic Integrity	SIL 1	SIL 2	SIL 3	SIL 4
1. Documented and defined operational semantics	D.54	R	HR	HR	M	M
2. Supports modular approaches	D.54, D.38	R	HR	HR	HR	HR
3. Supports commenting	D.54	R	HR	HR	HR	HR
4. Strong type system and checking	D.54	R	HR	HR	HR	HR
5. Supports static analysis	D.54, Table A.19	R	HR	HR	HR	HR
6. Supports Testing	D.54	R	HR	HR	HR	HR
7. Supports Test Coverage Metrics	D.54, Table A.21	R	HR	HR	HR	HR
Requirements: 1) The selection of the languages shall be based on the requirements given in 7.3.						
NOTE For information on assessing the suitability of a programming language see entry in D.54 'Suitable Programming Languages'						

Figura 3: Requisiti essenziali dei linguaggi di programmazione.

Gestione degli Application Data

La CENELEC EN 50716 introduce le seguenti modifiche sui dati applicativi:

1. **separazione tra software generico e dati applicativi:** la norma stabilisce una chiara separazione tra il software generico e i dati applicativi. Questo significa che i dati applicativi devono essere trattati in modo distinto dal software, consentendo di aggiornare e ricompilare separatamente il software e i dati applicativi, a meno che non ci siano modifiche nelle interfacce che li collegano. Qualora sia possibile, il software generico deve essere progettato per rilevare eventuali dati applicativi corrotti;
2. **configurabilità tramite dati applicativi:** la norma stabilisce che, quando un software è configurabile tramite i dati applicativi, deve essere specificato come questi dati interagiscono con il software. Le interfacce tra il software e i dati applicativi devono essere ben definite e documentate, garantendo la possibilità di modificare o aggiornare i dati senza influire sul software, se non tramite modifiche alle interfacce stesse;
3. **introduzione dell'application release note:** la redazione del documento è sotto la responsabilità del Designer e deve contenere:
 - 1) tutte le condizioni applicative da rispettare,
 - 2) le informazioni sulla compatibilità tra i componenti software (esclusi i dati applicativi) e tra il software e l'hardware,
 - 3) le restrizioni sull'uso del software;
4. **importanza dei test di integrazione:** è stato specificato che i test di integrazione sono fortemente raccomandati per il **Basic Integrity** e i livelli **SIL 1-SIL4** e che devono essere eseguiti in ambiente reale.

Modelli di ciclo di vita del software

La norma CENELEC EN 50716 fornisce nella sezione C.1 dell'Appendice C una guida e degli esempi di modelli di ciclo di vita del software, analizzando le caratteristiche principali di due tipi di modelli: quello **lineare** e quello **iterativo**. Il primo organizza le attività in base alle loro dipendenze in modo sequenziale, dove ogni fase successiva dipende dall'output della fase precedente. Il modello lineare prevede dei loop di feedback, poiché ogni cambiamento, dovuto a degli eventi improvvisi durante l'esecuzione del progetto, può comportare il ritorno ad una fase precedente. Anche la CENELEC EN 50716, come esempio di modelli lineari, cita il modello "Waterfall" e il modello a "V", sottolineandone la differenza principale: il primo evidenzia in maniera esplicita solo la dipendenza dalla fase immediatamente precedente, mentre il secondo evidenzia in modo più esplicito la dipendenza di una fase da tutte le fasi precedenti. Nei modelli iterativi le fasi e le attività sono le stesse dei modelli lineari. Il progetto viene suddiviso in pacchetti più piccoli, e sequenze di una o più fasi vengono ripetute all'interno delle singole suddivisioni, fino al raggiungimento dell'obiettivo finale che risulta essere lo stesso dei modelli lineari e che è costituito dal merge di tutti i risultati ottenuti nelle varie parti suddivise. Quando i risultati (o i risultati parziali) di un'iterazione precedente vengono riutilizzati in un'iterazione successiva, è necessario assicurarsi che non siano stati invalidati da modifiche aggiunte nel frattempo. A differenza degli approcci lineari, le ripetizioni delle fasi e delle attività non sono dovute a eventi imprevisi, ma sono pianificate e previste fin dall'inizio nel piano di progetto. La Figura 4 mostra un modello di ciclo di vita esemplificativo in cui le fasi di architettura e Design, test, implementazione e integrazione software vengono ripetute tre volte. Il modello di ciclo di vita iterativo prevede che le attività vengano ripetute su più iterazioni, le quali possono essere anche parallelizzate, con ogni iterazione che estende o modifica parti dei documenti di Design, architettura e requisiti software. Ogni iterazione produce un report di verifica aggiornato, considerando sia i nuovi risultati che quelli già analizzati con le precedenti iterazioni (Figura 5). Gli output finali, dopo l'ultima iterazione, sono equivalenti a quelli prodotti in un modello lineare.

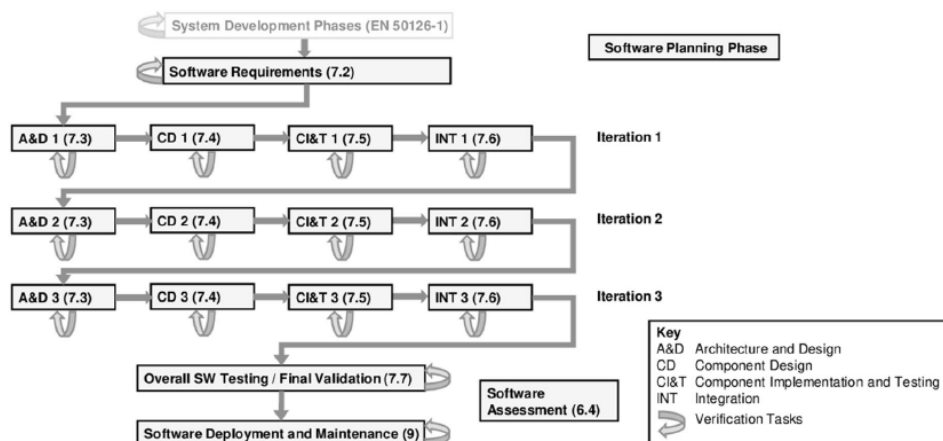


Figura 4: Esempio di modello iterativo [1].

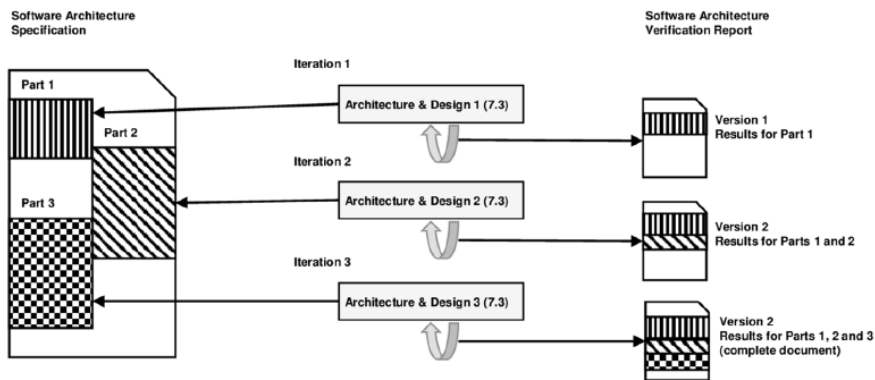


Figura 5: Esempio di modello iterativo per le fasi di verifica e stesura del Verification Report.

Sviluppo “Model-Based”

Nella sezione C.2 dell'Appendice C vengono esplorati i vantaggi di uno sviluppo “Model-Based” e fornisce delle linee guida su come applicarlo nel dominio ferroviario. Il modello viene definito come una rappresentazione logica finalizzata allo sviluppo, alla comprensione, alla spiegazione di aspetti di un sistema, entità o processo. I modelli vengono creati usando notazioni grafiche e/o testuali, che possono essere formali (con definizione matematica) o semi-formali (con semantiche incomplete). Le notazioni formali includono metodi specifici, mentre quelle semi-formali includono linguaggi come UML, SysML e diagrammi a stati. La modellazione offre numerosi vantaggi, come la semplificazione della valutazione di scenari alternativi e la rilevazione anticipata di difetti. Permette inoltre la generazione automatica di codice e documentazione, facilitando la verifica attraverso strumenti avanzati come analizzatori statici e model checker. I controlli di coerenza del modello riducono significativamente l'impegno richiesto per le attività di test, mentre la possibilità di avere simulazioni integrate consente l'analisi dinamica del sistema. Inoltre, la modellazione si integra con altri strumenti di supporto, come quelli per la tracciabilità, e consente il riutilizzo di progetti preesistenti in contesti diversi, ottimizzando l'efficienza e la manutenibilità del software. I modelli possono essere utilizzati in diverse fasi dello sviluppo del software, dalla specifica, progettazione, implementazione, verifica, integrazione e validazione.

A tale scopo, la CENELEC EN 50716, nella sezione C.2.6 dell'Appendice C, fornisce alcune indicazioni su come adattare le tecniche e le misure descritte nell'Appendice A, ad uno sviluppo “Model-Based”. L'uso delle tecniche di modellazione non dovrebbe influire sulla scelta del modello di ciclo di vita o sulla struttura della documentazione per lo sviluppo del software. Tuttavia, i modelli possono migliorare l'integrazione delle fasi e ridurre le sovrapposizioni, poiché un modello può contenere i deliverable di più fasi, riducendone la duplicazione. La modellazione migliora la qualità del software grazie al formalismo dei suoi linguaggi e supporta attività di testing e verifica, come la generazione di casi di test e la simulazione dell'ambiente software.

Tuttavia, i modelli non sono facilmente visualizzabili, modificabili o stampabili senza strumenti dedicati, e la generazione di documenti potrebbe comportare la perdita di informazioni essenziali. È quindi importante garantire che i documenti generati contengano il contenuto appropriato per le attività correlate, come la tracciabilità e l'analisi e che vengano verificati e validati per garantirne completezza, coerenza e conformità ai requisiti di tracciabilità. Gli strumenti di modellazione, come editor e generatori di codice, devono rispettare i requisiti specifici previsti per i tools di supporto (sottoclausola 6.7) e, come per quest'ultimi, l'uso di tecniche ridondanti può ridurre la necessità di evidenze. I modelli archiviati in basi

di dati sono considerati documenti a tutti gli effetti e, pertanto, seguono la norma EN ISO 9000 [9] per la gestione della qualità.

Utilizzo dell'Intelligenza Artificiale e del Machine Learning nel mondo ferroviario

L'uso dell'intelligenza artificiale (AI) e del machine learning (ML) nel settore ferroviario costituisce, ad oggi, un vero e proprio challenge. Il Machine Learning (ML) implementa una struttura software che "impara" il suo comportamento tramite i dati di addestramento. A differenza dei metodi di sviluppo tradizionale, i requisiti di sistema non vengono decomposti per essere tracciabili nell'architettura software e nei componenti, e i dati di addestramento sono probabilmente una rappresentazione incompleta di tutti gli stati di input possibili. Di conseguenza, non è facile dimostrare la completa copertura raggiunta dalle attività di test o da altri metodi di verifica. Le applicazioni di AI/ML sono spesso utilizzate per identificare pattern in grandi quantità di dati, ma l'integrazione dell'IA/ML in settori con rigorosi requisiti di verifica software, come quello ferroviario, è ancora limitata. Le principali sfide per l'uso del ML nell'ambito della CENELEC EN 50716 includono:

- 1 **Completezza e accuratezza dei dati di addestramento:** i dati devono essere sufficientemente completi per rappresentare accuratamente tutte le possibili situazioni;
- 2 **Verifica del software addestrato:** la struttura del software addestrato non è visibile e quindi non è possibile valutare la copertura dei test o utilizzare altre forme di verifica;
- 3 **Validazione della funzionalità approssimata:** il principale vantaggio della maggior parte dei metodi di Machine Learning (ML) risiede nella generalizzazione delle loro funzioni previste in aree applicative non specificamente definite in anticipo. Proprio questo vantaggio si trasforma in una sfida quando si tratta di validazione, ossia nel verificare se la funzionalità finale, che è stata approssimata statisticamente dall'algoritmo di Machine Learning, "ha appreso la funzione corretta";
- 4 **Adversarial attacks e mancanza di causalità:** gli adversarial attacks sono delle tecniche utilizzate per ingannare o compromettere i modelli di AI, in particolare di ML, in modo che commettano errori o producano risultati errati. Questi attacchi vengono realizzati creando input appositamente modificati o ingannevoli, che sembrano essere normali ai modelli, ma che sono progettati per causare una risposta sbagliata da parte dell'algoritmo. Ciò è dovuto al fatto che un algoritmo ML si basa solo su correlazioni statistiche tra input e output e non su una relazione causale.

Queste sfide devono essere superate per rendere le tecniche AI/ML applicabili allo sviluppo software come richiesto da CENELEC EN 50716. Attualmente sono in corso ricerche sulla verifica e validazione di applicazioni AI/ML, ma non esistono ancora pratiche generalmente riconosciute.

Modifiche minori

La norma presenta una serie di modifiche di minore importanza, le quali possono essere così riassunte:

1. **Riorganizzazione formale delle tecniche e delle misure nell'Appendice A:** ad esempio, la tabella A.14 è stata eliminata e le tecniche sono state incorporate nella tabella A.13, con i test funzionali/black box integrati nell'analisi e testing dinamico;

2. **Modifiche all'applicabilità delle tecniche di modellazione:** molte delle tecniche e misure indicate nella Tabella A.17 sono state ora classificate come fortemente raccomandate anche per i livelli SIL1 e SIL2. Questa modifica è considerata di minore rilevanza, in quanto i requisiti richiesti non sono cambiati e i metodi precedentemente definiti come fortemente raccomandati rimangono invariati;
3. **Riorganizzazione formale della documentazione:** ad esempio, il documento di sistema dei requisiti di sicurezza non viene più citato nella nuova norma, in quanto i requisiti sono inclusi nel documento di sistema;
4. **Modifiche al ruolo del configuration manager:** è stato specificato che il configuration manager deve solo assicurarsi (e non produrre) che la release note venga rilasciata includendo anche versioni incompatibili dei componenti software (se presenti). Tale modifica si considera di minore importanza, in quanto era un errore presente nelle precedenti norme. La redazione delle release note è sempre stata in carico al team di Design;
5. **Rimozione definitiva degli application algorithm nella clausola 8:** i titoli dei paragrafi sono stati aggiornati per rimuovere gli application algorithm, in quanto, come era già precedentemente specificato nel paragrafo, i software applicativi utilizzati per la configurazione del software generico, seguono i requisiti delle clausole 4,6,7;
6. **integrazione degli amendments nella EN 50716:** il contenuto degli amendments A1/A2:2020 della CENELEC EN 50128 e A1:2023 della CENELEC EN 50657, ancora applicabile, è stato integrato nella nuova norma;
7. **Modifica alla clausola 3:** sono state apportate modifiche formali al capitolo relativo alle definizioni dei vari termini, inclusi miglioramenti nell'allineamento con le norme CENELEC EN 50126-1 e CENELEC EN 50126-2.

Bibliografia

- [1] CENELEC EN 50716, "Railway applications - Requirements for software development", 2023.
- [2] CENELEC EN 50128, "Railway applications - Communication, Signaling and processing systems-software for railway control and protection systems", 2011 (A1:2020 and A2:2020).
- [3] CENELEC EN 50657, "Railways Applications - Rolling stock applications - Software on Board Rolling Stock", 2017 (A1:2023).
- [4] CEI EN 50716, "Railway applications - Requirements for software development", 2024.
- [5] CENELEC, EN 50126, "Railway Applications - The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS)", 2017.
- [6] EC 61508, "Functional Safety of Electrical/Electronic/Programmable Electric Safety-related Systems", 2010.
- [7] ISO/IEC/IEEE 90003, "Software engineering – Guidelines for the application of ISO 9001:2015 to computer software", 2018.
- [8] EN ISO 9001:2015, *Quality management systems - Requirements* (ISO 9001:2015).
- [9] EN ISO 9000, "Quality management systems - Fundamentals and vocabulary" (ISO 9000:2015).